

Question 1)

What will happen when you attempt to compile and run this code?

```
abstract class Base{
    abstract public void myfunc();
    public void another(){
        System.out.println("Another method");
    }
}

public class Abs extends Base{
    public static void main(String argv[]){
        Abs a = new Abs();
        a.amethod();
    }
    public void myfunc(){
        System.out.println("My Func");
    }
    public void amethod(){
        myfunc();
    }
}
```

- 1) The code will compile and run, printing out the words "My Func"
 - 2) The compiler will complain that the Base class has non abstract methods
 - 3) The code will compile but complain at run time that the Base class has non abstract methods
 - 4) The compiler will complain that the method myfunc in the base class has no body, nobody at all to loooove it
-

Question 2)

What will happen when you attempt to compile and run this code?

```
public class MyMain{
public static void main(String argv){
    System.out.println("Hello cruel world");
}
}
```

- 1) The compiler will complain that main is a reserved word and cannot be used for a class
 - 2) The code will compile and when run will print out "Hello cruel world"
 - 3) The code will compile but will complain at run time that no constructor is defined
 - 4) The code will compile but will complain at run time that main is not correctly defined
-

Question 3)

Which of the following are Java modifiers?

- 1) public
 - 2) private
 - 3) friendly
 - 4) transient
 - 5) vagrant
-

Question 4)

What will happen when you attempt to compile and run this code?

```
class Base{  
    abstract public void myfunc();  
    public void another(){  
        System.out.println("Another method");  
    }  
}  
  
public class Abs extends Base{  
    public static void main(String argv[]){  
        Abs a = new Abs();  
        a.amethod();  
    }  
  
    public void myfunc(){  
        System.out.println("My func");  
    }  
  
    public void amethod(){  
        myfunc();  
    }  
}
```

- 1) The code will compile and run, printing out the words "My Func"
 - 2) The compiler will complain that the Base class is not declared as abstract.
 - 3) The code will compile but complain at run time that the Base class has non abstract methods
 - 4) The compiler will complain that the method myfunc in the base class has no body, nobody at all to loooove it
-

Question 5)

Why might you define a method as native?

-
- 1) To get to access hardware that Java does not know about
 - 2) To define a new data type such as an unsigned integer
 - 3) To write optimised code for performance in a language such as C/C++
 - 4) To overcome the limitation of the private scope of a method
-

Question 6)

What will happen when you attempt to compile and run this code?

```
class Base{  
    public final void amethod(){  
        System.out.println("amethod");  
    }  
}  
  
public class Fin extends Base{  
    public static void main(String argv[]){  
        Base b = new Base();  
        b.amethod();  
    }  
}
```

- 1) Compile time error indicating that a class with any final methods must be declared final itself
 - 2) Compile time error indicating that you cannot inherit from a class with final methods
 - 3) Run time error indicating that Base is not defined as final
 - 4) Success in compilation and output of "amethod" at run time.
-

Question 7)

What will happen when you attempt to compile and run this code?

```
public class Mod{  
    public static void main(String argv[]){  
    }  
    public static native void amethod();  
}
```

- 1) Error at compilation: native method cannot be static
 - 2) Error at compilation native method must return value
 - 3) Compilation but error at run time unless you have made code containing native amethod available
 - 4) Compilation and execution without error
-

Question 8)

What will happen when you attempt to compile and run this code?

```
private class Base{}  
public class Vis{  
    transient int iVal;  
    public static void main(String elephant[]){  
    }  
}
```

- 1)Compile time error: Base cannot be private
 - 2)Compile time error indicating that an integer cannot be transient
 - 3)Compile time error transient not a data type
 - 4)Compile time error malformed main method
-

Question 9)

What happens when you attempt to compile and run these two files in the same directory?

```
//File P1.java  
package MyPackage;  
class P1{  
void afancymethod(){  
    System.out.println("What a fancy method");  
}  
}  
//File P2.java  
public class P2 extends P1{  
    public static void main(String argv[]){  
        P2 p2 = new P2();  
        p2.afancymethod();  
    }  
}
```

- 1) Both compile and P2 outputs "What a fancy method" when run
 - 2) Neither will compile
 - 3) Both compile but P2 has an error at run time
 - 4) P1 compiles cleanly but P2 has an error at compile time
-

Question 10)

You want to find out the value of the last element of an array. You write the following code. What will happen when you compile and run it.?

```
public class MyAr{  
    public static void main(String argv[ ]){  
        int[] i = new int[5];  
        System.out.println(i[5]);  
    }  
}
```

- 1) An error at compile time
 - 2) An error at run time
 - 3) The value 0 will be output
 - 4) The string "null" will be output
-

Question 11)

You want to loop through an array and stop when you come to the last element. Being a good java programmer and forgetting everything you ever knew about C/C++ you know that arrays contain information about their size. Which of the following can you use?

- 1)myarray.length();
 - 2)myarray.length;
 - 3)myarray.size
 - 4)myarray.size();
-

Question 12)

What best describes the appearance of an application with the following code?

```
import java.awt.*;  
public class FlowAp extends Frame{  
    public static void main(String argv[ ]){  
        FlowAp fa=new FlowAp();  
        fa.setSize(400,300);  
        fa.setVisible(true);  
    }  
  
    FlowAp(){  
        add(new Button("One"));  
        add(new Button("Two"));  
        add(new Button("Three"));  
        add(new Button("Four"));  
    }//End of constructor  
}//End of Application
```

- 1) A Frame with buttons marked One to Four placed on each edge.

-
- 2) A Frame with buttons marked One to four running from the top to bottom
 - 3) A Frame with one large button marked Four in the Centre
 - 4) An Error at run time indicating you have not set a LayoutManager
-

Question 13)

How do you indicate where a component will be positioned using Flowlayout?

- 1) North, South, East, West
 - 2) Assign a row/column grid reference
 - 3) Pass a X/Y percentage parameter to the add method
 - 4) Do nothing, the FlowLayout will position the component
-

Question 14)

How do you change the current layout manager for a container

- 1) Use the setLayout method
 - 2) Once created you cannot change the current layout manager of a component
 - 3) Use the setLayoutManager method
 - 4) Use the updateLayout method
-

Question 15)

Which of the following are fields of the GridBagConstraints class?

- 1) ipadx
 - 2) fill
 - 3) insets
 - 4) width
-

Question 16)

What most closely matches the appearance when this code runs?

```
import java.awt.*;
public class CompLay extends Frame{
public static void main(String argv[]){
    CompLay cl = new CompLay();
```

```

    }

CompLay(){
    Panel p = new Panel();
    p.setBackground(Color.pink);
    p.add(new Button("One"));
    p.add(new Button("Two"));
    p.add(new Button("Three"));
    add("South",p);
    setLayout(new FlowLayout());
    setSize(300,300);
    setVisible(true);
}

```

-
- 1) The buttons will run from left to right along the bottom of the Frame
 - 2) The buttons will run from left to right along the top of the frame
 - 3) The buttons will not be displayed
 - 4) Only button three will show occupying all of the frame

Question 17)

Which statements are correct about the *anchor* field?

- 1) It is a field of the GridBagLayout manager for controlling component placement
 - 2) It is a field of the GridBagConstraints class for controlling component placement
 - 3) A valid setting for the anchor field is GridBagConstraints.NORTH
 - 4) The anchor field controls the height of components added to a container
-

Question 18)

What will happen when you attempt to compile and run the following code?

```

public class Bground extends Thread{
public static void main(String argv[]){
    Bground b = new Bground();
    b.run();
}
public void start(){
    for (int i = 0; i <10; i++){
        System.out.println("Value of i = " + i);
    }
}

```

- 1) A compile time error indicating that no run method is defined for the Thread class
- 2) A run time error indicating that no run method is defined for the Thread class

-
- 3) Clean compile and at run time the values 0 to 9 are printed out
 - 4) Clean compile but no output at runtime
-

Question 19)

Is the following statement true or false?

When using the GridBagLayout manager, each new component requires a new instance of the GridBagConstraints class.

- 1) true
 - 2) false
-

Question 20)

Which most closely matches a description of a Java Map?

- 1) A vector of arrays for a 2D geographic representation
- 2) A class for containing unique array elements
- 3) A class for containing unique vector elements
- 4) An interface that ensures that implementing classes cannot contain duplicate keys

SCWCD MOCK QUESTIONS & ANSWERS

1) A software company wants to develop a component that does many of the pre-processing stuffs like whether the requests are coming from authentic sources, filtering unwanted data in the request object. What type of pattern will be the best match for developing such a component?

- a. Front Controller
- b. Business Controller
- c. Business Delegate
- d. Intercepting Filter

2) A class wants to keep track of its number of objects being added to HttpSession objects. Which will be the ideal way to achieve this?

- a. Define a listener class implementing HttpSessionListener and whenever an object is added to HttpSession object, keep track of the number of instances.
- b. Define a listener class implementing HttpSessionBindingListener and in the valueBound() method, keep track of the number of instances being added in a static variable.
- c. It is highly impossible to keep track the number of objects being added to a HttpSession in a Web Application.
- d. The Servlet API provides direct support for keeping track the object instances.

3) What will be the output of the following JSP code snippet at run-time when it is accessed the third time?

```
<% int test = 0; %>
<% ++test; %>
The value of test is
<%= test %>
```

- a. The value of the 'test' variable will be 2.
- b. The value of the 'test' variable will be 0.
- c. The value of the 'test' variable will be 1.
- d. The variable 'test' must be declared at global scope, else a run-time error will occur.

4) What method can be used to retrieve all the parameter names being sent as part of the request by the client?

- a. Use the method 'HttpServletRequest.getParameterNames()' which will return an enumeration of parameter names.
- b. Use the method 'HttpServletResponse.getParameterNames()' which will return an enumeration of parameter names.
- c. Use the method 'HttpServletRequest.getAllParameters()' which will return an enumeration of parameter names.
- d. There is no direct support in the Servlet API to retrieve the name of all the parameters sent by the client.

5) Given the deployment descriptor for a Web Application is

```
<web-app version="2.4">

    <servlet>
        <servlet-name>InitParamsServlet</servlet-name>
        <servlet-class>scwcd14.chap02.InitParamsServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>InitParamsServlet</servlet-name>
        <url-pattern>/InitParamsServlet</url-pattern>

        <init-param>
            <param-name>name</param-name>
            <param-value>javabeat</param-value>
        </init-param>

    </servlet-mapping>

</web-app>
```

What will be the output of the following Servlet?

```
package scwcd14.chap02;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class InitParamsServlet extends HttpServlet {

    public void init(){

        ServletConfig config = getServletConfig();
        System.out.println(config.getInitParameter("name"));
        System.out.println(config.getInitParameter("no-name"));
    }
    ...
}
```

- a. The Servlet will output 'javabeat null' in the Server Console.
- b. The Servlet will fail to load as the init param property 'no-name' is not mentioned in the deployment descriptor.
- c. A NullPointerException will be raised as calling getServletConfig() in the init() method will return a null reference.

6) Following is an excerpt taken from a Deployment Descriptor of some Web Application.

```
<web-app version="2.4">

    <context-param>
        <param-name>app-name</param-name>
```

```

<param-value>MockQuestions</param-value>
</context-param>

<servlet>
    <servlet-name>ContextInfoServlet</servlet-name>
    <servlet-class>scwcd14.chap03.ContextInfoServlet</servlet-class>

    <init-param>
        <param-name>app-name</param-name>
        <param-value>ContextInfoServlet</param-value>
    </init-param>
</servlet>

</web-app>

```

7) Which of the following code will output the message 'MockQuestions ContextInfoServlet' to the client browser (assuming that 'out' is a valid instance of type 'PrintWriter' class)?

```

out.println(config.getServletContext().getParameter( "app-name" ) );
out.println(config.getInitParameter( "app-name" ) );

out.println(config.getInitParameter( "app-name" ));
out.println(config.getServletContext().getInitParameter( "app-name" ));

out.println(config.getServletContext().getInitParameter( "app-name" ) );
out.println(config.getInitParameter( "app-name" ));

out.println(config.getServletContext().getInitParameter( "app-name" ) );
out.println(config.getParameter( "app-name" ) );

```

7) Which of the following tag library definition is valid?

- a. <tag>
- b. <name>hello</name>
- c. <tag-class>test.HelloTag</tag-class >
- d. </tag>
- e. <tag>
- f. <name>anotherHello</name>
- g. <tag-class>test.AnotherHelloTag</tag-class>
- h. </tag>
- i.
- j.
- k.
- l. <tag>
- m. <name>hello</name>
- n. <tag-class>test.HelloTag</tag-class >
- o. </tag>
- p. <tag>
- q. <name>hello</name>
- r. <tag-class> test.HelloTag</tag-class>
- s. </tag>

```
t.  
u.  
v.  
w. <tag>  
x. <name>root</name>  
y. <tag-class>test.RootTag</tag-class >  
z. <sub-tag>  
aa. <name>child</name>  
bb. <tag-class>test.ChildTag</tag-class>  
cc. </sub-tag>  
dd. </tag>  
ee.
```

8) What is the equivalent action code for the following piece of JSP code snippet?

```
EmployeeBean harry = (EmployeeBean)request.getAttribute("harry");  
if (harry == null)  
{  
    harry = new EmployeeBean();  
    request.setAttribute("harry", harry);  
}
```

- a. <jsp:declareBean id = "harry" class = "EmployeeBean" scope = "request" />
- b. <jsp:createBean id = "harry" class = "EmployeeBean" />
- c. <jsp:useBean id = "harry" class = "EmployeeBean" scope = "request" />
- d. <jsp:useBean id = "harry" class = "EmployeeBean" />

9) Which of the following is a valid declaration of a Tag Library in a Deployment Descriptor?

- a. <taglib>
- b. <uri>http://www.javabeat.net/studyKit</uri>
- c. <location>/studyKit.tld</location>
- d. </taglib>
- e.
- f.
- g.
- h.
- i. <taglib>
- j. <uri>http://www.javabeat.net/studyKit</uri>
- k. <location>/studyKit.tld</location>
- l. </taglib>
- m.
- n.
- o.
- p. <taglib>
- q. <taglib-uri>http://www.javabeat.net/studyKit</taglib-uri>
- r. <taglib-location>/studyKit.tld</taglib-location>
- s. </taglib>
- t.
- u.
- v.
- w. <taglib>

```
x. <uri>http://www.javabeat.net/studyKit</uri>
y. <path>/studyKit.tld</path>
z. </taglib>
aa.
```

10) Which of the following EL expression retrieves the value of the attribute by name 'attribute' found in HTTP's request object

- a. \${request.getAttribute()}
- b. \${request.attribute}
- c. \${pageContext.request.getAttribute()}
- d. \${pageContext.request.attribute}

Answers

1) d.

Since all of the operations corresponds to some pre-processing stuff, an Intercepting filter will be the ideal pattern to achieve this.

2) b.

Option b is correct as a custom attribute, if want to be notified when objects of the class are being added to Http Session, should implement the HttpSessionBindingListener interface.

3) c.

The variable 'test' is declared with local scope as it is declared within the doGet() method. So, the number of times, this page is invoked doesn't really matter and the value of the variable 'test' at any point of time will be '1'.

4) a.

Using HttpServletRequest and by calling the method getParamterNames(), all the parameter names can be retrieved.

5) a.

Option a is correct. It will print the value of the init parameters in the Server console.

6) c.

Option c is correct as to retrieve data declared in the <context-param> tag, we need a reference to Servlet Context object and then to call the getInitParameter() method. The data in the init-param tag declared for a Servlet can be obtained directly by calling the getInitParamter() defined on the Servlet Config object.

7) a.

Option b is incorrect as the tag name must be unique with the tag definition file. Option c is incorrect as there is no such tag called 'sub-tag'.

8) c.

An employee object by name 'harry' is taken from the request object which is of type 'EmployeeBean'. Option c is the right one which is defining this.

9) c.

Option c is correct as valid elements for 'taglib' are 'taglib-uri' and 'taglib-location'.

10) d.

A request object can be obtained by calling \${pageContext.request}. Since java methods calls cannot be used (and shouldn't be), option c is incorrect as it uses the java method call syntax (getMethod()).

SCWCD 1.4 Mock Questions - 1

1. A web.xml for a web application contains the following:

```
<login-config>
    <auth-method>FORM</auth-method>
    <realm-name>sales</realm-name>
    <form-login-config>
        <form-login-page>/formlogin.html</form-login-page>
        <form-error-page>/formerror.html</form-error-page>
    </form-login-config>
</login-config>
```

What should formlogin.html contain?

Select 1 correct option.

- A. A base 64 encoded username and password
- B. A header that prompts the browser to pop up the username/password dialog
- C. A form that POSTs to j_security_check url
- D. Any html page that does not require the user to login
- E. Code to redirect the user to the login page

2. Which of the following are valid values for the <auth-method> element of the deployment descriptor?

Select 2 correct options.

- A. DIGEST
- B. CLIENT-CERT

C. SECURE

D. NONE

3. You want to add third party classes bundled as a JAR file and a couple of your own classes to a web application. Which directories would you place them in?

Select 1 correct option.

- A. AR file in WEB-INF/lib and class files in WEB-INF/classes
- B. JAR file in WEB-INF/classes/lib and class files in WEB-INF/classes
- C. both in WEB-INF/classes
- D. both in WEB-INF
- E. JAR file in WEB-INF/jars and class files in WEB-INF/classes

4. Write the name of the deployment descriptor tag that allows you to write a description for a <context-param> element. Please do not add angle brackets.

5. Which of the following statements are correct?

Select 3 correct options.

- A. Authorization means determining whether one has access to a particular resource or not
- B. Authentication means determining whether one has access to a particular resource or not
- C. Authentication means proving whether one is what one claims to be
- D. Data Integrity means that the data is not modified in transit between the sender and the receiver
- E. Data Integrity means that the data cannot be viewed by anybody other than its intended recipient

6. Your web application named "FWorks" uses SpecialMath.class. This is an unbundled class and is not contained in any jar file. Where will you keep this class file?

Select 1 correct option.

- A. FWorks/WEB-INF
- B. FWorks/WEB-INF/classes
- C. FWorks/WEB-INF/lib/classes
- D. FWorks/classes
- E. FWorks/WEB-INF/lib

7. Which jsp tag can be used to retrieve a property of a bean?

Select 1 correct option.

- A. jsp:useBean
- B. jsp:useBean.property
- C. jsp:useBean.getProperty
- D. jsp:getProperty
- E. jsp:property

8. Associate the events with appropriate listener interface:

Drag and drop the matching listener.

- i. session is activated or passivated
- ii. session is timed out
- iii. an attribute is replaced in the session
- iv. a session is created

Select items

- A.HttpSessionListener
- B.HttpSessionBindingListener
- C.HttpSessionActivationListener
- D.None of these

9. Given the code of doGet() method of a servlet (see exhibit). The data should be sent to the client only if loginUser() returns a non null userid. Otherwise a status of SC_FORBIDDEN should be sent. What can be placed at //1 to fulfill this requirement?

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException

{
    String userId = loginUser(req); //this method takes the credentials
                                    from the request and logs in the user.

    if(userId == null)

    {
        // 1 Should send SC_FORBIDDEN
    }

    else

    {
        PrintWriter out = response.getWriter();

        generateAndPublishData(out); //this method writes appropriate date to
out.

    }
}
```

Select 3 correct options

- A. req.getRequestDispatcher("errorpage.jsp").dispatch(req, res, HttpServletResponse.SC_FORBIDDEN);
- B. throw new ServletException(HttpServletResponse.SC_FORBIDDEN);
- C. res.setStatus(HttpServletResponse.SC_FORBIDDEN);
- D. res.sendError(HttpServletResponse.SC_FORBIDDEN, "You are not authorized.");
- E. res.sendError(HttpServletResponse.SC_FORBIDDEN);

10. Which of the following elements of web.xml affect the whole web application instead of a specific servlet?

Select 1 correct option

- A.content-type
- B.init-param
- C.listener

D.application
E.app-config

11.Which of the following statements regarding <jsp:useBean> action are correct?

Select 2 correct options.

- A.It must have an 'id' attribute.
- B.If 'beanName' attribute is present, 'type' must also be present.
- C.It must have a 'scope' attribute.
- D.If 'class' attribute is present, 'type' must also be present.

12. Identify the implicit objects available to EL expressions.

Select 4 correct options

- A.requestScope
- B.application
- C.header
- D.page
- E.pageScope
- F.pageContext

13. Populate the blanks with appropriate examples given in the options.

comment

directive

declaration

scriptlet

custom tag

expression

select the items

- A. <tags:simple name='bob' />
- B.<%=request.getParameter("name")%>
- C.<%request.getParameter("name");%>
- D.<jsp:directive.include file='hello.jsp' />
- E.<%-- String x = "123" --%>
- F.<%!String x = "123"; %>

14. Which of the following XML fragments correctly defines a role named "manager" in web.xml?

- 1.<security-role>manager</security-role>
- 2.<security-role rolename=manager></security-role>
- 3.<security>
 <role-name>manager</role-name>
</security>
- 4.
 <security-role>
 <role-name>manager</role-name>
</security-role>

A.1

B.2

C.3

D.4

15. Which of the following statements is correct regarding HttpSessionBindingListener interface?

Select 1 correct option.

- A.The valueBound() method is called BEFORE the object becomes accessible through HttpSession.getAttribute()
- B.The valueUnbound() method is called BEFORE the object is removed from the HttpSession
- C.The valueReplaced() method is called BEFORE the object is replaced with another object in the session
- D.None of these

16. Consider the following HTML code. Which method of MyFirstServlet will be invoked when you click on the url shown by the page?

```
<html>
<body>
  <a href="/myapp/servlet/MyFirstServlet">Make me say Hello World!</a>
</body>
<html>
```

Select 1 correct option.

- A.doGet
- B.doGET
- C.post
- D.doPost
- E.doPOST

17. Your jsp page connects to the database and retrieves the data. It also formats the data and displays it to the client. Any of these operations can throw exceptions but you do not want to catch them all in this page and so you have written another jsp page that is meant to handle any kind of exceptions. How would you associate that error page named "error.jsp" with this page?

Select 2 correct options.

- A.Add <%@errorPage="error.jsp"%> in this page
- B.Add <%@page errorPage="error.jsp"%> in this page
- C.Add <%@page isErrorPage="true"%> in error.jsp
- D.Add <%@isErrorPage="true"%> in error.jsp

18. You have to send a gif image to the client as a response to a request. How will you acquire the 'out' variable to do this?

Select 1 correct option.

- A.PrintWriter out = response.getWriter();
- B.PrintWriter out = response.getWriter();
- C.FileOutputStream out = response.getOutputStream();
- D.ServletOutputStream out = response.getOutputStream();
- F.ServletOutputStream out = response.getServletOutputStream("image/gif");

19. Which of the given statements are correct regarding the following JSP page code?

```
<jsp:useBean id="mystring" class="java.lang.String" />  
<jsp:setProperty name="mystring" property="*" />  
<%=mystring%>
```

Assume that the request for this page contains a parameter mystring=hello.

Select 1 correct option.

- A. It will print ""
- B. It will print "hello"
- C. It will not compile
- D. It will throw exception at runtime

20. Select the correct sequence of actions that a servlet container performs before servicing any request.

- A. Instantiate listeners defined in the deployment descriptor
- B. Initialize filters defined in the deployment descriptor
- C. Initialize servlets that are set to load on startup
- D. Call the contextInitialized method on the listeners implementing ServletContextListener interface

21. Which of the following is a valid life cycle event listener interface but is NOT configured in the web.xml?

Select 1 correct option.

- A. HttpSessionListener
- B. SessionActivationListener
- C. HttpSessionBindingListener
- D. ContextAttributeListener
- E. SessionAttributeListener

22. How can you retrieve the data sent by the FORM displayed by following HTML page code?

```
<html>  
<body>  
  <form action="/myapp/SaveServlet" method="POST">  
    <input type="file" name="name">  
    <input type="submit" value="POST">  
  </form>  
</body>  
</html>
```

Select 2 correct options.

- A. request.getParameter("name");
- B. request.getAttribute("name");

C.request.getInputStream();
D.request.getReader();
E.request.getFileInputStream();

SCWCD 1.4 Mock Questions - 1 (Answers)

- 1)C
- 2)A,B
- 3)A
- 4)description or <description>
- 5)A,C,D
- 6)B
- 7)D
- 8) i. session is activated or passivated -C
ii. session is timed out -A,B
iii. an attribute is replaced in the session - D
iv. a session is created - A
- 9)C,D,E
- 10)C
- 11)A,B
- 12)A,C,E,F
- 13) comment -E
directive - D
declaration -F
scriptlet -C
custom tag -A
expression - B
- 14)D
- 15)A
- 16)A
- 17)B,C

18)D

19)A

20)A,D,B,C

21)C

22)C,D

SCWCD 1.4 Mock Questions - 2

1.you are using a tag library with prefix "generator", which supports a tag named "random". This tag generates a random number and sets it to a variable named "value". Which of the following will output this value in the page?

Select 1 correct option.

- A.<generator:random>value</generator:random>
- B.<generator:random><%=value%></generator:random>
- C.<generator:random><% int value; %> <%=value%></generator:random>
- D.<generator:random><%getParameter("value")%></generator:random>
- E.None of the above

2. Which of the following pairs of HTTP method and HttpServlet class method are a valid combination for a request and the request handler method?

Select 2 correct options.

- A.GET - service()
- B.POST - doPost()
- C.GET - doPost()
- D.GET - doGet()
- E.POST - service()

3.You want to get notified whenever there is a change in the attribute list of the ServletContext of your web application. Which listener interface would you implement?

Select 1 correct option

- A.ServletListener
- B.ServletContextListener
- C.ServletContextAttributeListener
- D.HttpServletContextListener
- E.HttpServletListener

4.A JSP page myerror.jsp has been invoked as a result of an exception from another JSP page. How can you access the Throwable object that refers to the exception in myerror.jsp?

Select 1 correct option.

- A.Using the implicit variable error
- B.Using the implicit variable request.error
- C.Using the implicit variable exception
- D.Using the implicit variable throwable
- E.None of these because the class of the implicit variable is not java.lang.Throwable

5.Which of the following are valid return values for doStartTag() method?

Select 3 correct options.

- A.BodyTag.SKIP_BODY
- B.Tag.SKIP
- C.Tag.EVAL_BODY_INCLUDE
- D.Tag.EVAL_BODY_AGAIN
- E.BodyTag.EVAL_BODY_BUFFERED

6.Which of the following statements are correct for a custom tag that can take any number of arbitrary attributes?

Select 2 correct options.

- A.The body-content element for the tag in the TLD file must have a value of JSP.
- B.The tag handler must implement the method setAttribute(String key, String value).
- C.The tag element in the TLD file for the tag must have <dynamic-attributes>true</dynamic-attributes>.
- D.The class implementing the tag must implement javax.servlet.jsp.tagext.DynamicAttributes interface.
- E.Dynamic attributes cannot have request time expression values
- F.A JSP page sets a dynamic attribute using <jsp:setDynamicAttribute> action

7.Assuming that <%@ taglib uri="/utils" prefix="util" %> occurs before the use of the custom tags of the tag library named utils, identify the possibly valid empty custom tag declarations. (Assume that transpose is a valid tag in the given tag library.)

Select 2 correct options.

- A.<util:transpose/>
- B.<util:transpose></util:transpose>
- C.<util:transpose>200</util:transpose>
- D.<taglib:util:transpose />
- E.None of the above is correct as every tag has to have a body

8.Which of the following defines the class name of a tag in a TLD?

Select 1 correct option.

- A.tag-class-name
- B.tag-class
- C.class-name
- D.class

9.What should be the value of <body-content> subelement of element <tag> in a TLD file if the tag should not have any contents as its body?

Select 1 correct option.

- A.blank
- B.empty
- C.null
- D.false
- E.The <body-content> subelement itself should be absent

10.Which interface and method should be used to retrieve a servlet initialization parameter value?

Select 1 correct option.

- A.ServletConfig : getParameter(String name)
- B.ServletConfig : getInitParameter(String name)
- C.ServletContext : getInitParameter(String name))
- D.ServletConfig : getInitParameters(String name)
- E.ServletConfig : getInitParameterNames(String name)

11. You need to put a com.enthu.User bean referred by 'userBean' variable in request scope with an ID of "user" from within a servlet. Which of the following statements accomplishes this task?

Select 1 correct option.

- A.request.put("user", userBean);
- B.request.add(userBean, "user");
- C.request.putAttribute("user", userBean);
- D.request.setAttribute("user", userBean);
- E.request.setParameter(userBean, "user");
- F.request.put(userBean, "user");

12.Consider the following code:

```
public class MyTagHandler extends TagSupport
{
    public int doStartTag() throws JspException
    {
        try
        {
            //insert code here
        }
        catch(Exception e){ }

        return super.doStartTag();
    }
}
```

Which of the following options, when inserted in the above code causes the value "hello" to be output?

Select 1 correct option.

- A.JspWriter out = pageContext.getOut();
out.print("hello");
- B.JspWriter out = pageContext.getWriter();
out.print("hello");
- C.JspWriter out = getPageContext().getWriter();
out.print("hello");

```
D.JspWriter out = new JspWriter(pageContext.getWriter());
out.print("hello");
E.JspWriter out = getPageContext().getOut();
out.print("hello");
```

13.Which of the following is a correct JSP declaration for a variable of class java.util.Date?

Select 1 correct option.

- A.<%! Date d = new Date() %>
- B.<%@ Date d = new Date() %>
- C.<%! Date d = new Date(); %>
- D.<%\$ Date d = new Date() %>

14.Which method can be invoked on a session object so that it is never invalidated by the servlet container automatically?

Select 1 correct option.

- A.setTimeOut(-1)
- B.setTimeOut(Integer.MAX_INT)
- C.setTimeOut(0)
- D.setMaxInactiveInterval(-1)
- E.setMaxInactiveInterval(Integer.MAX_INT)

15.Which of the following elements are mandatory under the <web-app> element of a deployment descriptor?

Select 1 correct option.

- A.<doctype>
- B.<app-name>
- C.<servlet>
- D.<doc-root>
- E.None of these.

16.Which of the following implicit variables should be used by a jsp page to access a page initialization parameter?

Select 1 correct option.

- A.pageContext
- B.application
- C.config
- D.context
- E.page

17.You are given a tag library that has:

1. A tag named getMenu that takes an attribute 'subject' which can be a dynamic value.
2. A tag named getHeading that takes an attribute 'report'.

Which of the following are correct uses of this library?

Select 3 correct options.

- A.<myTL:getMenu subject="Finance"/>
- B.<myTL:getMenu subject=<myTL:getHeading report=1/>/>
- C.<myTL:getMenu subject='<myTL:getHeading report="1"/>'>

D.<% String subject="HR"; %> <myTL:getMenu subject="<%="subject%>"/>
E.<myTL:getHeading report="2"/>

18.You are working with a tag library which is packaged in a jar file named htmlutil.jar. This jar file also contains a META-INF/htmlutil.tld file which has a uri element as follows:

<uri>http://www.xycorp.com/htmlLib</uri> What can you do to access this library from your JSP pages

Select 1 correct option.

- A.You must define the <taglib> element in the web.xml to specify the mapping for <taglib-uri> to the location of this jar file.
- B.There is no need for the <taglib> element in the web.xml, however, you need the taglib directive in the JSP pages.
- C.You can directly access the tags of this library from the JSP pages without any taglib directive.
- D.You do not need the taglib directive, but you do need to specify the <taglib> element in the web.xml.
- E.None of these.

19.Which of the following are valid iteration mechanisms in jsp?

1.

```
<% int i = 0;
while(i<5)
{
    "Hello World"
    i++;
} %>
```
2.

```
<jsp:for loop='5'>
    "Hello World"
</jsp:for>
```
3.

```
<% int i = 0;
for(;i<5; i++)
{
    "Hello World";
<% i++;
}
%>
```
4.

```
<table>
<% Iterator it = aSet.iterator();
int i = 0;
while(it.hasNext())
{
    out.println("<tr><td>" + (++i) + "</td>");
    out.println("    <td>" + it.next() + "</td></tr>");
}
%>
</table>
```

```
5.
<jsp:scriptlet>
    for(int i=0; i<5; i++)
    {
</jsp:scriptlet>
<jsp:text>"Hello World!"</jsp:text>
<jsp:scriptlet>
    }
</jsp:scriptlet>
```

Select 3 correct option.

- A.1
- B.2
- C.3
- D.4
- E.5

20. Your web application wants to make use of a role named 'manager', which is defined in the servlet container. Which of the following XML fragments must occur in the deployment descriptor of your web application?

```
1. <role name='manager' />
2. <role>
   <role-name>manager</role-name>
</role>
3. <role>manager</role>
4. <security-role>
   <role-name>manager</role-name>
</security-role>
```

Select 1 correct option.

- A.1
- B.2
- C.3
- D.4

21. Consider the class shown in exhibit. Which of the following statements are correct?

```
public class MyHSAListener implements HttpSessionAttributeListener
{
    public void attributeAdded(HttpSessionBindingEvent e){ }
    public void attributeRemoved(HttpSessionBindingEvent e){ }
}
```

Select 1 correct option.

- A. public void attributeReplaced(...){ } must be added.
- B. public void attributeChanged(...){ } must be added.
- C. The parameter class should be HttpSessionEvent.
- D. The parameter class should be HttpSessionAttributeEvent.
- E. It will compile as it is.

22.Identify the implicit objects available to EL expressions

Select 4 correct options.

- A.request
- B.sessionScope
- C.paramValues
- D.params
- E.cookie
- F.initParam

23.Which of the following statements are correct regarding tag libraries?

Select 1 correct option.

- A.The tag library descriptor for a tag library must be kept in META-INF/taglib.tld, if the tag library is packaged in a jar file.
- B.The tag library descriptor for a tag library may be kept in WEB-INF/taglib.tld, if the tag library is packaged in a jar file.
- C.A JSP 2.0 compliant container is guaranteed to generate implicating mapping for JSTL tag libraries.
- D.A JSP 2.0 compliant container will automatically generate an implicit tag library for a set of tag files.
- E.The tag library descriptor for a tag library not packaged as a jar file may be kept anywhere in /tld directory of the web application's document root

24.Consider the following usage of a custom tag in a JSP page:

```
<jsp:useBean id="student" scope = "session" class="com.xyz.Student" />
<mytaglib:studentTag student='student' />
```

Which of the following statements are correct?

Select 1 correct option.

- A.Application objects such as com.xyz.Student, cannot be passed as attributes to custom tags.
- B.The Student object will be passed to the studentTag tag handler.
- C.The Student object will NOT be passed because no variable named student is defined.
- D.A Student object will not be created if it is not available in the session.
- E.None of these.

25.Assuming that the Servlet Container has just called the destroy() method of a servlet instance, which of the following statements are correct?

Select 2 correct options.

- A.Any resources that this servlet might hold have been released.
- B.The servlet container time out has exceeded for this servlet instance.
- C.The init() method has been called on this instance.
- D.None of the requests can EVER be serviced by this instance.
- E.All threads created by this servlet are done.

26.For a tag to accept any valid jsp code as its body, what should be the value of <body-content> for this tag's taglib descriptor?

Select 1 correct option.

- A.JSP

B.jsp

C.any

D.text

E.The <body-content> subelement itself may be absent.

27.Which of the given options can be used in a servlet code that needs to access a binary file kept in WEB-INF/data.zip while servicing a request? Assume that config refers to the ServletConfig object of the servlet and context refers to the ServletContext object of the servlet.

Select 1 correct option.

A.InputStream is = config.getInputStream("/WEB-INF/data.zip");

B.InputStream is = context.getInputStream("data.zip");

C.InputStream is = context.getResourceAsStream("/WEB-INF/data.zip");

D.InputStream is = context.getResourceAsStream("WEB-INF/data.zip");

E.InputStream is = config.getResourceAsStream("WEB-INF/data.zip");

28.Servlet Container calls the init method on a servlet instance ...

Select 1 correct option.

A.For each request to the servlet.

B.For each request to the servlet that causes a new session to be created.

C.For each request to the servlet that causes a new thread to be created.

D.Only once in the life time of the servlet instance.

E.If the request is from the user whose session has expired.

F.Initialy when the servlet instance is create and then at request time if the request is from the user whose session has expired.

29.Identify the implicit objects accessible to a jsp page that can store objects accessible across multiple requests.

Select 2 correct options.

A.page

B.request

C.session

D.application

E.pageContext

30.You are developing a jsp page named stockindices.jsp. This jsp page needs to use a HTML page named nasdaq.html in the middle of the page, which is updated every ten minutes by some other process. Which of the following lines, when added to stockindices.jsp, ensures that stockindices.jsp uses the latest nasdaq.html?

Select 1 correct option.

A.<%@include page='nasdaq.html' %>

B.<%@include file='nasdaq.html' %>

C.<jsp:include page='nasdaq.html' />

D.<jsp:include file='nasdaq.html' />

E.<jsp:forward page='nasdaq.html' />

- 1)B
- 2)B,D
- 3)C
- 4)C
- 5)A,C,E
- 6)C,D
- 7)A,B
- 8)B
- 9)B
- 10)B
- 11)D
- 12)A
- 13)C
- 14)D
- 15)E
- 16)C
- 17)A,D,E
- 18)B
- 19)C,D,E
- 20)D
- 21)A
- 22)B,C,E,F
- 23)D
- 24)E
- 25)C,D

26)E

27)C

28)D

29)C,D

30)C

SCWCD 1.4 Mock Questions - 3

1. Regarding the processing of a BodyTag handler, in which of the following cases a BodyContent object will be "pushed" into the pageContext?

Select 1 correct option.

- A.If the doStartTag() returns EVAL_BODY_INCLUDE
- B.If the doStartTag() returns EVAL_BODY_BUFFERED
- C.If the doStartTag() returns SKIP_BODY
- D.If the doStartTag() DOES NOT return SKIP_BODY

E.A BodyContent object it is always created and pushed no matter what doStartTag() returns

2.Which of the following apply to Transfer Object design pattern?

Select 2 correct options.

- A.It increases complexity by increasing the number of remote interfaces.
- B. It increases network performance by introducing one coarse grained remote call for multiple finer grained network calls
- C.It reduces network traffic by introducing one coarse grained remote call for multiple finer grained network calls
- D.It increase server throughput by utilizing the CPU better
- E.It increases design overhead due to versioning issues

3.Which of the following deployment descriptor snippets would you use to declare the use of a tag library?

1.

```
<tag-lib>  
  <uri>http://abc.net/ourlib.tld</uri>  
  <location>/WEB-INF/ourlib.tld</location>  
</tag-lib>
```

2.

```
<taglib>  
  <uri>http://abc.net/ourlib.tld</uri>
```

```
<location>/WEB-INF/ourlib.tld</location>
</taglib>

3.

<taglib>
  <taglib-uri>http://abc.net/ourlib.tld</taglib-uri>
  <taglib-location>/WEB-INF/ourlib.tld</taglib-location>
</taglib>

4.

<taglib>
  <tagliburi>http://abc.net/ourlib.tld</uri>
  <tagliblocation>/WEB-INF/ourlib.tld</location>
</taglib>

5.

<taglibmap>
  <uri>http://abc.net/ourlib.tld</uri>
  <location>/WEB-INF/ourlib.tld</location>
</taglibmap>
```

Select 1 correct option.

- A.1
- B.2
- C.3
- D.4
- E.5

4.Which of the following design patterns is used to separate the task of writing the GUI screens and business logic?

Select 1 correct option.

- A.View Logic
- B.Front Controller
- C.Model View Controller

D.Business View

E.Business Delegate

5. For this jsp code to compile and run which of the given options should be true? <jsp:useBean class="com.bookstore.Book" type="java.lang.Object" id="book" />

Select 1 correct option.

A.This statement is wrong as type attribute is invalid.

B.Book must have a public no args constructor

C.Book must have a public constructor but there is no requirement on arguments.

D.Book must have a public getInstance() method.

E.This statement will always throw an exception at runtime no matter what you do to Book class.

6.Your servlet may throw IOException while processing a request. You want to define an error page in your deployment descriptor so that whenever IOException is thrown, this page is serviced to the browser. Which of the following XML fragments correctly specify the mapping:

1.

```
<error-page>
  <exception>java.io.IOException</exception>
  <location>/html/Test.html</location>
</error-page>
```

2.

```
<error-page>
  <exception-class>java.io.IOException</exception-class>
  <location>/html/Test.html</location>
</error-page>
```

3.

```
<error-page>
  <exception-type>java.io.IOException</exception-type>
  <page-location>/html/Test.html</page-location>
</error-page>
```

4.

```
<error-page>
    <exception-type>java.io.IOException</exception-type>
    <location>/Enthuse/html/Test.html</location>
</error-page>
```

5.

```
<exception>
    <exception-type>java.io.IOException</exception-type>
    <location>/Enthuse/html/Test.html</location>
</exception>
```

Select 1 correct option.

- A.1
- B.2
- C.3
- D.4
- E.5

7. Select the tag that comes directly under the <web-app> tag of a web.xml and that is used to specify a class whose object will be sent notifications when changes occur to the ServletContext?

Select 1 correct option.

- A.servlet-context-listener
- B.listener
- C.context-listener-class
- D.listener-class
- E.context-listener

8. Consider the following JSP code (See exhibit).

What will it print for the very first request to this page as well as the web application that contains this page?

```
<html><body>
<%
    Integer count = (Integer)
request.getSession(false).getAttribute("count");

    if(count != null )
    {
        out.println(count);
    }
}
```

```
    else request.getSession(false).setAttribute("count", new Integer(1));  
%>  
  
Hello!  
</body></html>
```

Select 1 correct option.

- A. It will print Hello!
- B. It will print Hello and will set the count attribute in the session.
- C. It will throw a NullPointerException at request time.
- D. It will not compile.

9. The following line of code exists in the doGet method of Servlet:

```
String sid = request.getParameter("jsessionid");
```

Which of the options will retrieve the HttpSession associated with the request? (Assume that the session has already been created.)

Select 3 correct options.

- A. HttpSession session = request.getSession();
- B. HttpSession session = HttpSession.getSession(sid);
- C. HttpSession session = request.getSession(sid);
- D. HttpSession session = request.getSession(true);
- E. HttpSession session = request.getSession(false);

SCWCD 1.4 Mock Questions - 3 (Answers)

1)C

2)B,C

3)C

4)B

5)B

6)D

7)B

8)B

9)A,D,E

SCWCD 1.4 Mock Questions - 4

1. Consider the following web.xml code snippet:

```
<servlet>

    <servlet-name>BankServlet</servlet-name>

    <servlet-class>com.abc.bankapp.BankServlet</servlet-class>

    <security-role-ref>

        <role-name>manager</role-name>

        <role-link>supervisor</role-link>

    </security-role-ref>

</servlet>
```

Which of the following statements are correct?

Select 1 correct option.

- A.The servlet code should use "manager" as a parameter in request.isUserInRole() method.
- B.The servlet code can use "manager" or "supervisor" as a parameter in request.isUserInRole() method.
- C.The servlet code should use"supervisor" as a parameter in request.isUserInRole() method.
- D.The role of "manager" must be defined in the servlet container.
- E.None of these.

2.You are designing a complex webapp that uses multi tier architecture. The application must provide interfaces for HTML as well as XML and should be maintainable.Which design pattern would you use?

Select 1 correct option.

- A.Data Access Object
- B.Business Deligate
- C.MVC
- D.Remote Method Invocation
- E.Transfer Object

3.Which of the following directives are applicable ONLY for tag files?

Select 3 correct options.

- A.attribute
- B.variable
- C.page
- D.include
- E.import
- F.tag

4.Which of the following are correct about FORM based authentication mechanism?

Select 3 correct options.

- A.HTML FORM is used to capture the username and password of the user.
- B.Password is transmitted as plain text.
- C.Password is transmitted in an encrypted form.
- D.Password is transmitted either in encrypted text or in plain text depending on the

browser.

E. This mechanism can be used over HTTPS.

5. Which pattern allows you to replace the presentation logic without much impact on the data representation?

Select 1 correct option.

- A. Model View Controller
- B. Business Delegate
- C. Transfer Object
- D. Data Access Object
- E. Bimodal DataAccess

6. Identify the elements that help describe the attribute characteristics of a JSP custom tag in a TLD file.

Select 3 correct options.

- A. value
- B. name
- C. description
- D. rtexprvalue
- E. class

7. Select the correct return types for ServletContext.getResource() and ServletContext.getResourceAsStream() methods.

Select 1 correct option.

- A. java.io.Resource and java.io.InputStream
- B. java.io.Resource and java.io.BufferedInputStream
- C. java.net.URL and java.io.InputStream
- D. java.io.File and java.io.InputStream
- E. java.net.URL and java.io.FileInputStream

8. Consider the following jsp code:

```
,code>
<html>

<body>

<% String a = "aaa"; %>

<%! String a = "AAA"; %>

<% String b = "bbb"; %>

<%! String b = "BBB"; %>

<% out.println(a+b); %>

</body>

</html>
```

What will be the output?

Select 1 correct option.

- A.aaabbb
- B.aaaBBB
- C.AAAbbb
- D.AAABBB
- E.Compilation error!

9.Which of the following are valid values for the <transport-guarantee> element?

Select 3 correct options.

- A.CONFIDENTIAL
- B.INTEGRAL
- C.SECURE
- D.ENCRYPTED
- E.NONE

10.Write the parent element of <session-timeout> element.

11.Consider the tag handler class shown in exhibit.

What will be printed when the above tag is used as follows in a jsp page:

```
Hello <mylib:mytag> World!</mylib:mytag>

public class MyTag extends TagSupport
{
    public int doAfterBody()
    {
        try
        {
            pageContext.getOut().println("In doAfterBody()");

        }
        catch(Exception e)
        {
        }

        return SKIP_BODY;
    }
}
```

Select 1 correct option.

- A.Hello
- B.Hello World!
- C.Hello In doAfterBody() World!
- D.Hello In doAfterBody()
- E.None of the above.

12.Which of the following HTTP protocol methods is eligible to produce unintended side effects upon multiple identical invocations beyond those caused by single invocation?

Select 1 correct option.

- A.GET
- B.POST
- C.HEAD
- D.PUT
- E.OPTIONS

13.Which method of ServletResponse would you use to set its content type?

Select 1 correct option.

- A.setParameter
- B.setHeader
- C.setAttribute
- D.setContentType
- E.None of the above.

14.<jsp:useBean id="mybean" beanName="my.app.MyBean" class="my.app.MyBean" /> is a valid useBean declaration.

Options.

- A.True
- B.False

15.Which of the following lines can be used to retrieve a servlet initialization parameter "dbname" from the init() method of a servlet?

```
public void init()  
{  
    String dbname = //1 : Insert line here  
}
```

Select 2 correct options.

- A.getServletConfig().getParameter("dbname");
- B.getServletConfig().getInitParameter("dbname");
- C.getServletContext().getInitParameter("dbname");
- D.getInitParameter("dbname");
- E.getInitParameterValue("dbname");

16.Consider the following description of a tag in a TLD:

```
<tag>  
    <name>SmilyTag</name>  
    <tag-class>com.enthware.ctags.SmilyTag</tag-class>  
    <description>  
        Replaces emoticons such as :), :D, and :( with images.  
</description>
```

```
</description>

<body-content>tagdependent</body-content>

<attribute>

<name>name</name>

<required>false</required>

<rtexprvalue>true</rtexprvalue>

</attribute>

</tag>
```

Which of the following statements regarding the above tag are correct?

Select 2 correct options.

- A. It is an empty tag.
- B. It may be used as an empty tag.
- C. It must have a body
- D. It must implement BodyTag interface.
- E. It may take an attribute named 'name'. But if present, its value must be dynamic.

17. Which of the following jsp fragments will print all the parameters and their values present in a request?

1.

```
<% Enumeration enum = request.getParameterNames();>

while(enum.hasMoreElements()) {
    Object obj = enum.nextElement();
    out.println(request.getParameter(obj));
}>
```

2.

```
<% Enumeration enum = request.getParameters();>

while(enum.hasMoreElements()) {
    String obj = (String) enum.nextElement();
    out.println(request.getParameter(obj));
}>
```

3.

```
<% Enumeration enum = request.getParameterNames();  
while(enum.hasMoreElements()) {  
    String obj = (String) enum.nextElement();  
    out.println(request.getParameter(obj));  
} %>
```

4.

```
<% Enumeration enum = request.getParameterNames();  
while(enum.hasMoreElements()) {  
    Object obj = enum.nextElement(); %>  
    <%=request.getParameter(obj); %>  
} %>
```

5.

```
<% Enumeration enum = request.getParameterNames();  
while(enum.hasMoreElements()) {  
    String obj = (String) enum.nextElement(); %>  
    <%=request.getParameter(obj)%>  
} %>  
Select 2 correct options.
```

- A.1
- B.2
- C.3
- D.4
- E.5

18.Which of the following statements are valid JSP directive?

Select 2 correct options.

- A.<%! int k = 10 %>
- B.<% int k = 10; %>
- C.<%=somevariable%>
- D.<%@ taglib uri="http://www.abc.com/tags/util" prefix="util" %>
- E.<%@ page language="java" import="com.abc.*" %>

19. How can you ensure the continuity of the session while using `HttpServletResponse.sendRedirect()` method when cookies are not supported by the client?

Select 1 correct option.

A.By using hidden parameters.
B.By encoding the redirect path with `HttpServletResponse.encodeRedirectURL()` method.
C.By using `HttpSession.encodeURL()` method.
D.By using `HttpServletRequest.encodeURL()` method.
E.By using `HttpServletResponse.encodeURL()` method.

20.Which of the following are valid implicit variables in a JSP page?

Select 2 correct options.

- A.error
B.page
C.this
D.root
E.context

21.A Tag Handler implements BodyTag interface. How many times its `doAfterBody` method may be called?

Select 1 correct option.

- A.BodyTag does not support `doAfterBody`.
B.0
C.1
D.0 or 1
E.Any number of times.

22.Given: You have configured a listener class (see exhibit) in `web.xml` of a web application. Now, consider the following code for the `doGet()` method of a servlet for the same web application.

```
,code>
public void doGet(HttpServletRequest req, HttpServletResponse res)
{
    System.out.println(this.getServletContext().getAttribute("key")); //2
}
```

Which option can be inserted at //1 in the listener code so that servlet code at //2 prints 100?

```
import javax.servlet.*;

public class MyListener implements ServletContextListener
{
    public void contextInitialized(ServletContextEvent sce)
    {
        Integer key = new Integer(100);
```

```
// 1 Insert code here.  
}  
  
public void contextDestroyed(ServletContextEvent sce)  
{  
}  
}  
}
```

Select 1 correct option.

- A.this.setAttribute("key", key);
- B.this.getServletContext().setAttribute("key", key);
- C.this.getContext().setAttribute("key", key);
- D.sce.getContext().setAttribute("key", key);
- E.sce.getServletContext().setAttribute("key", key);

23.Which of the following interfaces declares the methods jsplInit() and jsplDestroy()?

Select 1 correct option.

- A(javax.servlet.jsp.JSP
- B(javax.servlet.jsp.JspServlet
- C(javax.servlet.jsp.JspPage
- D(javax.servlet.jsp.HttpJspPage
- E(javax.servlet.jsp.HttpJspServlet

24.Which of the following statements are correct JSP directives?

Select 2 correct options.

- A.<%@ page %>
- B.<%! taglib uri="http://www.abc.com/tags/util" prefix="util" %>
- C.<% include file="/copyright.html"%>
- D.<%@ taglib uri="http://www.abc.com/tags/util" prefix="util" %>
- E.<%\$ page language="java" import="com.abc.*"%>

25.Which of the following classes hides the implementation details and provides a standard API to the services provided by the servlet container to a jsp page?

Select 1 correct option.

- A.HttpSession
- B.Servlet
- C.JspPage
- D.ServletContext
- E.PageContext

26.Which of the following are true regarding the parameters defined using the <context-param> element of a deployment descriptor?

Select 2 correct options.

- A.They are thread safe.
- B.They are accessible from multiple threads simultaneously and from any servlet of the web application.
- C.They can be modified using the setAttribute() method.

- D.They can be modified using the setParameter() method.
- E.They can be modified using the setInitParameter() method.

27.How can you explicitly expunge the session object?

Select 1 correct option.

- A.You cannot. It can only be expunged automatically after session timeout expires.
- B.By calling invalidate() on session object.
- C.By calling expunge() on session object.
- D.By calling delete() on session object
- E.By calling finalize() on session object.

28.You have declared a useBean tag as:

```
<jsp:useBean id="man" class="animal.Human" scope="application"/>
```

In which type of object will this bean be kept?

Select 1 correct option.

- A.HttpServlet
- B.HttpSession
- C.ServletContext
- D.ServletConfig
- E.ApplicationContext

29.Which of the following is a sensible way of sending an error page to the client in case of a business exception that extends from java.lang.Exception?

Select 2 correct options.

- A.Catch the exception and use RequestDispatcher to forward the request to the error page.
- B.Don't catch the exception and define the 'exception to error-page' mapping in web.xml
- C.Catch the exception, wrap it into ServletException and define the 'business exception to error-page' mapping in web.xml
- D.Catch the exception, wrap it into ServletException, and define the 'ServletException to error-page' mapping in web.xml
- E.Don't do anything, the servlet container will automatically send a default error page

30.Business delegate pattern should be used to enable communication between the JSP code and the enterprise javabeans.

Options.

- A.True
- B.False

SCWCD 1.4 Mock Questions - 4 (Answers)

- 1)A
- 2)C
- 3)A,B,F
- 4)A,B,E

5)A

6)B,C,D

7)C

8)A

9)A,B,E

10)session-config

11)A

12)B

13)D

14)B

15)B,D

16)B,D

17)C,E

18)D,E

19)B

20)B,C

21)E

22)E

23)C

24)A,D

25)E

26)A,B

27)B

28)C

29)A,C

30)A

SCWCD 1.4 Mock Questions - 5

1. Assume that the following header is present in a request sent to a servlet:

Accept: image/gif, image/jpeg, image/bmp

What will be returned when the servlet code calls `request.getHeader("Accept")`?

Select 1 correct option.

A. A Header object containing, name as "Accept" and value as "image/gif".

B. A Header object containing, name as "Accept" and value as "image/gif, image/jpeg, image/bmp".

C. A String array containing "image/gif""

D. A String containing "image/gif, image/jpeg, image/bmp".

E. A String array containing "image/gif", "image/jpeg", image/bmp"

2. You need to send large amount of binary data from the browser to a servlet to be processed. (Say, you want to attach a file while sending email through a web based system). What HTTP method would you use?

Select 1 correct option.

A. GET

B. POST

C. HEAD

D. HIDDEN

E. PUT

3. Which of the following is a possible way to configure an HttpSessionAttributeListener?

Select 1 correct option.

A. By calling `HttpSession.addAttributeListener(...)`

B. By calling `HttpSession.addHttpSessionAttributeListener(...)`

C. An object of a class implementing this interface is automatically configured when it is added to the session.

D. None of these.

4. Servlet Container calls the init method on a servlet instance ...

Select 1 correct option.

A. For each request to the servlet

B. For each request to the servlet that causes a new session to be created.

C. For each request to the servlet that causes a new thread to be created.

D. Only once in the life time of the servlet instance

E. If the request is from the user whose session has expired.

F. Initially when the servlet instance is created and then at request time if the request is from the user whose session has expired.

5. Which of the following methods may be called on a custom tag handler that implements IterationTag interface?

Select 2 correct options.

A. doStartTag

B. doBodyTag

C. doAfterBody

D.doInitBody
E.doEvalBody

6.Which of the following elements of web.xml defines a mapping between a servlet and a URL pattern?
Select 1 correct option.

- A.mapping
- B.servlet-url
- C.url_mapping
- D.url_pattern
- E.servlet-mapping

7.Following is the code for doGet() method of TestServlet. Which of the given statements about it are correct?

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
{
    try
    {
        RequestDispatcher rd =
this.getServletContext().getRequestDispatcher("Login.jsp"); // 1

        rd.forward(req, res); // 2
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

Select 2 correct options.

- A.This will not compile.
- B.This will compile but will not work as expected.
- C.This code will work just fine.
- D.It will compile but not work properly if //1 is replaced with: RequestDispatcher rd = req.getRequestDispatcher("Login.jsp");
- E.It will compile and will work properly if //1 is replaced with: RequestDispatcher rd = req.getRequestDispatcher("Login.jsp");

8.Consider the web.xml snippet shown in the exhibit.

Now consider the code for a jsp file named unprotected.jsp:

```
<html>
<body>
```

```
<jsp:include page="/jsp/protected.jsp" />  
</body>  
</html>
```

Which of the following statements hold true when unprotected.jsp is requested by an unauthorized user?

```
<web-app>  
    ...  
    <security-constraint>  
        <web-resource-collection>  
            <web-resource-name>test</web-resource-name>  
            <url-pattern>/jsp/protected.jsp</url-pattern>  
        </web-resource-collection>  
        <auth-constraint>  
            <role-name>manager</role-name>  
        </auth-constraint>  
    </security-constraint>  
    ...  
</web-app>
```

Select 1 correct option.

- A.The user will be prompted to enter user name and password
- B.An exception will be thrown
- C.protected.jsp will be executed but it's output will not be included in the response
- D.The call to include will be ignored
- E.None of these

9.Which of the following JSP elements can have a <jsp:param ...> element in its body?

Select 1 correct option.

- A.<jsp:include ...>
- B.<%@ include ...>
- C.<jsp:directive.include .../>
- D.<%@ forward ...>
- E.<jsp:action ...>

10.Which of the following implicit variables should be used by a jsp page to access a resource and to forward a request to another jsp page?

Select 1 correct option.

- A.pageContext and config
- B.application and config
- C.config and pageContext
- D.application for both
- E.config for both

11.In which of the following situations will a session be definitely invalidated?

Select 3 correct options.

- A.The container is shutdown and brought up again
- B.No request comes from the client for more than "session timeout" period.
- C.A servlet explicitly calls invalidate() on a session object.
- D.A servlet explicitly calls invalidate() on a session object.

12.Your jsp page uses classes from java.util package. Which of the following statement would allow you to import the package?

Select 1 correct option

- A.<%@import java.util.* %>
- B.<%import="java.util.*"@%>
- C.<%@ page import="java.util.*"%>
- D.<%@ page java="java.util.*"@%>
- E.<%@ page import="java.util.*"@%>

13.Consider the following contents for two JSP files:

In file companyhome.jsp:

```
<html><body>

Welcome to ABC Corp!

<%@ page errorPage="simpleerrorhandler.jsp" %>

<%@ include file="companynews.jsp" %>

</body></html>
```

In file companynews.jsp:

```
<%@ page errorPage="advancederrorhandler.jsp" %>

<h3>Todays News</h3>
```

Which of the following statements are correct?

Select 1 correct option

- A.When companyhome.jsp is requested, the output will contain "welcome..." as well as "Todays News"

- B.companyhome.jsp will not compile
- C.companynews.jsp will not compile
- D.Both the files will compile but will throw an exception at runtime.
- E.None of these

14.Which method of RegisterServlet will be called when the user clicks on "Submit" button for the following form. Assume that RegisterServlet

```
<html>

<body>

<form action="/myapp/RegisterServlet">

    <input type="text" name="method" value="POST">

    <input type="text" name="name">

    <input type="password" name="password">

    <input type="submit" value="POST">

</form>

</body>

</html>
```

Select 1 correct option.

- A.servicePost(HttpServletRequest, HttpServletResponse);
- B.doPOST(HttpServletRequest, HttpServletResponse);
- C.post(HttpServletRequest, HttpServletResponse);
- D.doPost(HttpServletRequest, HttpServletResponse);
- E.None of the above.

15.You are building the server side of an application and you are finalizing the interfaces that you will provide to the client side. But you have not yet decided whether the business rules will be fully implemented as stored procedures or in the java code. Which design pattern you should use to mitigate this concern?

Select 1 correct option.

- A.Model View Controller
- B.Data Access Object
- C.Business Delegate
- D.Facade
- E.Transfer Object

16.Which of the following XML frgments correctly define the <login-config> element of web.xml?

(See Exhibit.)

1.

```
<login-config>
```

```
<auth-method>CLIENT-CERT</auth-method>
<realm-name>test</realm-name>
</login-config>

2.

<login-config>
  <auth-method>CLIENT-CERT</auth-method>
  <realm-name>test</realm-name>
  <form-login-config>
    <form-login-page>/jsp/login.jsp</form-login-page>
    <form-error-page>/jsp/error.jsp</form-error-page>
  </form-login-config>
</login-config>

3.

<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>test</realm-name>
  <form-login-config>
    <form-login-page>/jsp/login.jsp</form-login-page>
    <form-error-page>/jsp/error.jsp</form-error-page>
  </form-login-config>
</login-config>

4.

<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>test</realm-name>
</login-config>
```

5.

```
<login-config>  
    <auth-method>SECURE</auth-method>  
    <realm-name>test</realm-name>  
</login-config>
```

Select 3 correct options.

- A.1
- B.2
- C.3
- D.4
- E.5

17.What are the following deployment descriptor elements used for?

<login-config> <security-constraint> <security-role> Select 1 correct option.

- A.Authorization
- B.Authentication
- C.Privacy
- D.Authentication and Authorization
- E.Data integrity.

18.You want to do some calculations within the object whenever it is added to the session. What would you do to accomplish this?

Select 1 correct option

- A.Make the class of the object implement HttpSessionBindingListener
- B.Configure a HttpSessionAttributeListener in deployment descriptor
- C.Make the class of the object implement HttpSessionListener
- D.Configure a HttpSessionActivationListener in deployment descriptor
- E.Only way is to configure a HttpSessionAttributeListener in the deployment descriptor

19.GET method is not suitable for which of the following operations?

Select 2 correct option.

- A.Retrieving an image
- B.Retrieving a zip file
- C.Submitting a form not containing login or other critical information
- D.Submitting a login form
- E.Updating a database

20.In the case of JSP pages, what is the type of the implicit variable 'out'?

Select 1 correct option.

- A.OutputStream
- B.PrintStream
- C.PrintWriter

D.JspWriter

E.DataOutputStream

21. Match the following.

Comment

directive

declaration

scriptlet

Custom tag

expression

<tags:simple name='bob' />

<%=request.getParameter("name")%>

<%request.getParameter("name"); %>

<jsp:directive.include file='hello.jsp' />

<%-- String x = "123" --%>

<%!String x = "123"; %>

22. Consider the following tag occurring in a JSP page:

<%@page import="java.util.*"%> Which of the following is the XML equivalent of the above tag?

Select 1 correct option.

A.<directive.page import="java.util.*"/>

B.<page import="java.util.*"/>

C.<%jsp:directive.page import="java.util.*"%>

D.<jsp:page import="java.util.*"/>

E.<jsp:directive.page import="java.util.*"/>

23. Your servlet may throw IOException while processing a request. You want to define an error page in your deployment descriptor so that whenever IOException is thrown, this page is serviced to the browser. Which of the following XML fragments correctly specify the mapping:

1.

```
<error-page>
    <exception>java.io.IOException</exception>
    <location>/html/Test.html</location>
</error-page>
```

2.

```
<error-page>
    <exception-class>java.io.IOException</exception-class>
    <location>/html/Test.html</location>
</error-page>
```

3.

```
<error-page>
    <exception-type>java.io.IOException</exception-type>
    <page-location>/html/Test.html</page-location>
</error-page>
```

4.

```
<error-page>
    <exception-type>java.io.IOException</exception-type>
    <location>/Enthuse/html/Test.html</location>
</error-page>
```

5.

```
<exception>
    <exception-type>java.io.IOException</exception-type>
    <location>/Enthuse/html/Test.html</location>
</exception>
```

- A.1
- B.2
- C.3
- D.4
- E.5

24. How can you ensure the continuity of the session while using `HttpServletResponse.sendRedirect()` method when cookies are not supported by the client?

Select 1 correct option

- A.By using hidden parameters.
- B.By encoding the redirect path with `HttpServletResponse.encodeRedirectURL()` method.
- C.By using `HttpSession.encodeURL()` method.
- D.By using `HttpServletRequest.encodeURL()` method.
- E.By using `HttpServletResponse.encodeURL()` method

25. Which of the following XML fragments correctly defines a role named "manager" in web.xml?

1.

```

<security-role>manager</security-role>
2.

<security-role rolename=manager></security-role>
3.

<security>
    <role-name>manager</role-name>
</security>
4.

<security-role>
    <role-name>manager</role-name>
</security-role>

```

Select 1 correct option.

- A.1
- B.2
- C.3
- D.4

26.Which of the following methods of HttpServletRequest can be used to retrieve the parameter values sent from the browser?

Select 2 correct options

- A.getParameter(String name);
- B.getParameter(String name, String defaultValue);
- C.getParameterNames();
- D.getParameterValues(String name);
- E.getParameters(String name);

27.Consider the code for the web.xml for a web application (See exhibit).

Assume that a request: http://localhost:8080/test/aaa/abc.a is sent to this web application named test.

Which of the following statements are correct?

```

<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Assume that DOCTYPE is valid -->

<!DOCTYPE web-app

    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.4//EN"
    "http://java.sun.com/dtd/web-app_2_4.dtd">

<web-app>
    <servlet>

```

```

<servlet-name>TestServlet</servlet-name>

<jsp-file>/requestinfo.jsp</jsp-file>

</servlet>

<servlet-mapping>

    <servlet-name>TestServlet</servlet-name>

    <url-pattern>*.a</url-pattern>

</servlet-mapping>

</web-app>

```

Select 1 correct option.

- A.Path Info of the request will be /aaa/abc.a
- B.RequestPath of this request will be /test//aaa/abc.a
- C.ContextPath of this request will be /test/aaa
- D.This request will be serviced by requestinfo.jsp
- E.None of these.

28.You are using a tag library with prefix "sequenceengine" which supports a tag named "fib". This tag expects a parameter named "limit" of type int. Which of the following is a correct use of this tag?

Select 1 correct option.

- A.<sequenceengine:fib>20</sequenceengine:fib>
- B.<fib:sequenceengine>20</fib:sequenceengine>
- C.<sequenceengine:fib attribute-name="limit" attribute-value="20"></sequenceengine:fib>
- D.<sequenceengine:fib limit="20"></sequenceengine:fib>
- E.<fib:sequenceengine limit="20"></fib:sequenceengine>

29.How can you redirect the request from a servlet to another resource if the servlet encounters an exception?

Select 2 correct options

- A.This cannot be done unless the exception is caught in the servlet.
- B.By specifying a mapping between exception class and the resource in web.xml
- C.This can be done only if the exception is a subclass of javax.servlet.ServletException
- D.This can be done even if the exact class of the exception is not known at compile time

30.Which of the given options correctly declare a useBean tag?

Select 3 correct options

- A.<jsp:useBean id="user" class="myco.util.User" />
- B.<jsp:useBean id="user" type="myco.interfaces.IUser" />
- C.<jsp:useBean name="user" class="myco.util.User" />
- D.<jsp:useBean id="user" beanName="myco.User" class="myco.util.User" />
- E.<jsp:useBean id="user" beanName="myco.User" type="myco.interfaces.IUser" />

SCWCD 1.4 Mock Questions - 5 (Answers)

- 1)D
 - 2)B
 - 3)D
 - 4)D
 - 5)A,C
 - 6)E
 - 7)C,E
 - 8)E
 - 9)A
 - 10)D
 - 11)B,C,D
 - 12)C
 - 13)B
 - 14)E
 - 15)C
 - 16)A,B,C
 - 17)D
 - 18)A
 - 19)D,E
 - 20)D
- 21)comment <%-- String x = "123" --%>
directive <jsp:directive.include file='hello.jsp' />
declaration <%!String x = "123"; %>
scriptlet <%request.getParameter("name");%>
custom tag <tags:simple name='bob' />
expression <%=request.getParameter("name")%>
- 22)E

23)D

24)B

25)D

26)A,D

27)D

28)D

29)B,D

30)A,B,E

SCWCD 1.4 Mock Questions - 6

1. Which constant is used to notify the container to reevaluate the custom tag's body?

Please select one correct answer.

- A. EVAL_BODY
- B. EVAL_BODY_TAG
- C. EVAL_BODY AGAIN
- D. EVAL_BODY_INCLUDE

2. Which of the following statements regarding the JSP action tag are TRUE?

Please select three correct answers.

- A. Provides translation-time instructions to the JSP engine.
- B. Can be used to declare a JavaBean instance in a JSP page
- C. Can be used to generate HTML code to embed an applet on a web page
- D. User-defined actions can be created
- E. language is a standard JSP action.

3. Which of the following methods can be used to pass request to another servlet to handle by using the RequestDispatcher?

Please select two correct answers.

- A. request(ServletRequest req, ServletResponse res)
- B. include(ServletRequest req, ServletResponse res)
- C. dispatch(ServletRequest req, ServletResponse res)
- D. forward(ServletRequest req, ServletResponse res)
- E. process(ServletRequest req, ServletResponse res)

4. Given the following code snippet, what would be the output you can expect to see on the web page?

Please select one correct answer.

```
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet extends GenericServlet {
```

```

public void init() {                                //1
    // Do something
}

public void init(ServletConfig config)           //2
throws ServletException{
    // Do something
    super.init(config);                         //3
}

public void destroy() {                           //4
    // Do something
}

public void service(ServletRequest req,        //5
ServletResponse res) {
    // Do something
}
}

```

- A. Method //1 is necessary for this code to compile
B. Method //2 is necessary for this code to compile
C. Line //3 is necessary for this code to run
D. Method //4 is necessary for this code to compile
E. Method //5 is necessary for this code to compile
5. Which pattern is normally used to encapsulate database SQL statement?
Please select one correct answer.
- A. Value Object
B. Data Value Object
C. Data Access Object
D. Business Object
E. Business Delegate
6. For special character used in request URI, which of the following statements are correct?
Please select two correct answers.
- A. The ampersand (&) is used to separate name/value pairs.
B. The plus sign (+) is used to fill blank space
C. The dash sign (-) is used to separate the name and value.
D. The percent sign (%) is used to start a query string.
7. Which statements are TRUE regarding the HttpServlet method doPost(HttpServletRequest req, HttpServletResponse res)?
Please select two correct answers.
- A. The method is called by the server (via the service method) to allow a servlet to handle a POST or PUT request
B. The method is called by the server (via the service method) to allow a servlet to handle a POST or GET request.
C. The method is called by the server (via the service method) to allow a servlet to handle a POST request
D. The method is called by the server (via the service method) to allow a servlet to handle a

GET request

E. The method allows the client to send data of unlimited length to the Web server a single time.

F. Operations requested through this method will NEVER have side effects.

8. Which of the following design pattern is used to reduce the amount of network traffic when transferring data?

Please select one correct answer

- A. Model View Controller
- B. Data Access Object
- C. Business Delegate
- D. Value Object

9. What is the authentication type which uses digital certificates as a security mechanism for a web based application?

10. Which of the following method is used to retrieve the value associated to the init parameter defined in the init-param tag?

Please select one correct answer.

- A. getParameter(String name)
- B. getInitParameter(String name)
- C. getParameters()
- D. getInitParameterValue(String name)

11. Which of the following statements are TRUE for the code given below?

Please select three correct answers.

```
int MAX_AGE;
Cookie cookie = new Cookie("user", user);
cookie.setMaxAge(MAX_AGE);
response.addCookie(cookie);
```

- A. If MAX_AGE = 10 the cookie will expire after 10 seconds.
- B. If MAX_AGE = 10 the cookie will expire after 600 seconds.
- C. If MAX_AGE = 0 the cookie will be deleted.
- D. If MAX_AGE = -1 the cookie is not stored persistently
- E. If MAX_AGE = -1; the code will generate run-time error.

12. In which directory you will most likely find the file myBaseUtil.jar?

Please select two correct answers.

- A. example/WEB-INF
- B. example/lib
- C. example/WEB-INF/lib
- D. example/WEB-INF/classes
- E. example/META-INF/lib

13. Which of the following requirements are needed for FORM based authentication in a web based application?

Please select four correct answers.

- A. The session attribute j_sessionid must be set.
- B. The action or url must be j_security_check.

- C. The name attribute for the username must be j_username
- D. The form method must be POST.
- E. The name attribute for the password must be j_password.
- F. Client side cookie must be enabled.

14. Which of the following deployment descriptor snippet will map the following request URI:

/tech/hello/index.jsp for web application with context path as "tech"?

Please select one correct answer.

```
A. <servlet-mapping>
    <servlet-name>HelloWorldServlet</servlet-name>
    <url-pattern>/hello/*</url-pattern>
</servlet-mapping>
B. <servlet-mapping>
    <servlet-name>HelloWorldServlet</servlet-name>
    <url-pattern>/hello/*.jsp</url-pattern>
</servlet-mapping>
C. <servlet-mapping>
    <servlet-name>HelloWorldServlet</servlet-name>
    <url-pattern>/hello/index.jsp</url-pattern>
</servlet-mapping>
D. <servlet-mapping>
    <servlet-name>HelloWorldServlet</servlet-name>
    <url-pattern>hello/*</url-pattern>
</servlet-mapping>
```

15. Which of the following methods will enable you to get one or more values from a request object?

Please select two correct answers.

- A. getParameter(String name)
- B. getParameters(String name)
- C. getAllParameters()
- D. getParameterValues(String name)
- E. getAllAttributes()

16. Is the following statement TRUE or FALSE?

Please select one correct answer.

The four methods for session management in the context of web-based application are:

Cookie, HttpSession object, URL rewriting and Hidden value.

- A. True
- B. False

17. Which of the following packages are implicitly imported in the JSP page?

Please select three correct answers.

- A. java.lang.*
- B. java.util.*
- C. javax.servlet.*
- D. javax.servlet.jsp.*
- E. javax.servlet.jsp.tagext.*

18. Which of the following is NOT an authentication method used by a web container?

Please select one correct answer.

- A. BASIC
- B. DIGEST
- C. SSL
- D. FORM

19. Which methods can be used for writing logging message to servlet log file?

Please select two correct answers.

- A. log(String msg)
- B. log(int code, String msg)
- C. log(String msg, Throwable t)
- D. log(int code, String msg, Throwable t)

20. Which tag is used in web.xml to mark a web application to be suitable for running between multiple systems.

Please select one correct answer.

- A. multiple
- B. distributable
- C. resource-ref
- D. transferrable
- E. splitable

21. Which of the following are VALID servlet error attributes?

Please select three correct answers.

- A. javax.servlet.error.status_code
- B. javax.servlet.error.exception
- C. javax.servlet.error.uri
- D. javax.servlet.error.message
- E. javax.servlet.error.query

22. Which statement is TRUE about the following jsp code snippet?

Please select one correct answer.

```
<%
    String theKey = "key";
    String theValue = "value";
    session.removeAttribute(theKey);           //1
%>

    session.setAttribute("<%= theKey %>" ,
                         "<%= theValue %>"); //2

    session.getAttribute("<%= theKey %>");   //3

<%= session.getAttribute(theKey) %>          //4
```

A. The code compiles but might have runtime NullPointerException at //1

B. There will have compilation error at //2 and //3.

C. There will have output as null at //4.

D. There will have output as theValue at //4

23. Which of the following are VALID taglib configuration?

Please select three correct answers.

A.

```
<taglib>
    ...
    <tag>
        <name>myTag</name>
        <tag-class>MyTag</tag-class>
    </tag>
    ...
</taglib>
```

B.

```
<taglib>
    ...
    <tag>
        <name>myTag</name>
        <tag-class>MyTag</tag-class>
        <body-content>SERVLET</body-content>
    </tag>
    ...
</taglib>
```

C.

```
<taglib>
    ...
    <tag>
        <name>myTag</name>
        <tag-class>MyTag</tag-class>
        <attribute>
            <name>name</name>
        </attribute>
    </tag>
    ...
</taglib>
```

D.

```
taglib>
    ...
    <tag>
        <name>myTag</name>
        <tei-class>MyTagInfo</tei-class>
        <body-content>JSP</body-content>
    </tag>
    ...
</taglib>
```

E.

```
<taglib>
    ...
    <tag>
        <name>myTag</name>
        <tag-class>MyTag</tag-class>
        <tei-class>MyTagInfo</tei-class>
        <body-content>JSP</body-content>
        <attribute>
            <name>name</name>
        </attribute>
    </tag>
    ...
</taglib>
```

```

<required>true</required>
<rteprvalue>true</rteprvalue>
<type>java.lang.String</type>
</attribute>
</tag>
...
</taglib>

```

24. Which of the following statement is FALSE regarding JSP page directive attributes default value?
Please select one correct answer..

- A. The session attribute has default value as true
- B. The buffer attribute has default value as 8kb
- C. The autoflush attribute has default value as false
- D. The isThreadSafe attribute has default value as true
- E. The isErrorPage attribute has default value as false
- F. The pageEncoding attribute has default value as ISO-8859-1

25. Which of the following deployment descriptor tags are used for context level parameter initialization?

Please select three correct answers.

- A. param-name
- B. context-name
- C. context-param
- D. param-value
- E. context-value
- F. context-attrib

26. Given the following code snippet, what would be the output you can expect to see on the web page?

Please select one correct answer.

```

// Calling servlet:

public void doGet(HttpServletRequest req,
HttpServletResponse res) throws ServletException, IOException
{
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML><BODY>");
    out.println("I am calling others!");
    RequestDispatcher rd= req.getRequestDispatcher( "/MyServlet" );
    rd.include(req, res);
    out.println("</BODY></HTML>" );
    out.close();
}

// Target servlet:

protected void doGet(HttpServletRequest req,
HttpServletResponse res) throws ServletException, IOException
{
    PrintWriter out = res.getWriter();

```

```
        out.println("I am called by others!");  
A."I am calling others!"  
B."I am called by others!"  
C.Both "I am calling others!" and "I am called by others!"  
D.An IllegalStateException is thrown  
E.An IOException is thrown
```

27.Which statement is NOT true about the SingleThreadModel interface?

Please select one correct answer

- A.By implementing this interface it ensures that servlets handle only one
- B.If a servlet implements this interface, no two threads will execute concurrently in the servlet's service method
- C.This interface has no methods
- D.The servlet container will ensure there will be only one instance of the servlet at a time if the servlet implements this interface
- E.Class variables are not protected by this interface, but instance variables are protected.

28.Which of the following method is called upon the initialization of a servlet context?

Please select one correct answer

- A.contextInitializing(ServletContextEvent e)
- B.contextInitial(ServletContext e)
- C.contextInitialize(ServletContext e)
- D.contextInitialize(ServletContextEvent e)
- E.contextInitialized(ServletContextEvent e)

29. Which of the following statements are TRUE?

Please select three correct answers

- A.XML equivalent for JSP expression <%= expression %> is
<jsp:expression>expression</jsp:expression>
- B.XML equivalent for JSP scriptlet <% scriptlet %> is
<jsp:scriptlet>scriptlet</jsp:scriptlet>
- C.XML equivalent for JSP declaration <%! declaration %> is
<jsp:declaration>declaration</jsp:declaration>.
- D. XML equivalent for JSP include directive <%@ include file="url" %> is <jsp:include file="url"/>, where url must be relative
- E. XML equivalent for JSP page directive <%@ page buffer="16kb" %> is <jsp:page buffer="16kb"/>

30. Please select CORRECT JSP useBean declaration methods

Please select three correct answers

- A.<jsp:useBean id="user" beanName="TestUser" type="com.test.model.User" />
- B.<jsp:useBean id="user" beanName="TestUser" class="com.test.model.User" />
- C. <jsp:useBean beanName="TestUser" class="com.test.model.User" />
- D. <jsp:useBean beanName="TestUser" class="com.test.model.User" />
- E. <jsp:useBean id="user" type="com.test.model.User" />

31. Which statement is TRUE regarding the following code?

Please select one correct answer.

```

import javax.servlet.*;
import javax.servlet.http.*;

public class MyHttpServlet extends HttpServlet
implements SingleThreadModel {

    StringBuffer bufferOne = new StringBuffer(); //1

    static StringBuffer bufferTwo = new StringBuffer(); //2

    protected void doGet(HttpServletRequest req, //3
    HttpServletResponse res) throws java.io.IOException{

        HttpSession session = req.getSession(); //4

        res.setContentType("text/html");
        java.io.PrintWriter out = res.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>This is my servlet!</title> ");
        out.println("</head>");
        out.println("<body>");
        out.println("</body> ");
        out.println("</html> ");
        out.close();
    }
}

```

- A. Variable bufferOne at //1 is NOT thread-safe.
 B. Variable bufferTwo at //2 is NOT thread-safe
 C. Both A and B
 D. Variable req at //3 is NOT thread-safe
 E. Variable session at //4 is NOT thread-safe
 F. Both D and E

32. Select sample web application file listing with appropriate directory structure
 Please select two correct answers.

- A.
 index.html
 /login.jsp
 /images/logo.gif
 /WEB-INF/web.xml
 /WEB-INF/lib/basic.jar
 /WEB-INF/classes/Test.class
 B.
 /index.html
 /login.jsp
 /images/logo.gif
 /WEB-INF/web.xml
 /WEB-INF/jar/basic.jar
 /WEB-INF/classes/Test.class
 C.
 /index.html
 /login.jsp

```
/images/logo.gif  
/WEB-INF/web.xml  
/WEB-INF/classes/basic.jar  
/WEB-INF/classes/Test.class  
D.  
/index.html  
/login.jsp  
/images/logo.gif  
/META-INF/web.xml  
/WEB-INF/jar/basic.jar  
/WEB-INF/classes/Test.class  
E.  
/index.html  
/login.jsp  
/images/logo.gif  
/META-INF/web.xml  
/WEB-INF/lib/basic.jar  
/WEB-INF/classes/Test.class  
F.  
index.html  
/images/logo.gif  
/WEB-INF/web.xml  
/WEB-INF/jsp/login.jsp  
/WEB-INF/lib/basic.jar  
/WEB-INF/classes/Test.class
```

33. Which of the following data element will definitely be thread-safe?

Please select one correct answer

- A. Local variables
- B. Instance variables
- C. Static variables
- D. Class variables
- E. Context attributes

34. Select the correct order that JSP methods are invoked by servlet container

Please select one correct answer.

- A. jsplInit(), jspService(), jspDestroy()
- B. jsplInit(), _jspService(), jspDestroy()
- C. _jsplInit(), jspService(), _jspDestroy()
- D. _jsplInit(), _jspService(), _jspDestroy()

35. Which of the following element is not included in a URL?

Please select one correct answer

- A. Client ip
- B. Protocol
- C. Server Name
- D. Query string
- E. Port name

36. Which of the following listeners is notified when a session is initialized?

Please select one correct answer.

- A. HttpSessionBindingListener
- B. SessionBindingListener

- C. HttpSessionListener
- D. HttpSessionListener
- E. HttpSessionChangedListener

37. Which of the following best describes the life cycle of a JSP?

Please select one correct answer.

- A.
JSP page is translated into a servlet code
Servlet code is compiled
Servlet is loaded into memory
Servlet instance is created
- B.
JSP page is translated into a servlet code
Servlet is loaded into memory
Servlet code is compiled
Servlet instance is created
- C.
JSP is compiled
JSP is translated into a servlet code
Servlet is loaded into memory
Servlet instance is created
- D.
JSP is loaded into memory
Servlet code is compiled
Servlet instance is created
Servlet is loaded into memory
- E.
JSP page is translated into a servlet code
Servlet code is compiled
Servlet instance is created
Servlet is loaded into memory

38. Please identify the three methods declared in javax.servlet.Filter

Please select three correct answers.

- A.service
- B.init
- C.destroy
- D.filter
- E.doFilter

39.Which of the following deployment descriptor segments are VALID for security-related configuration of a web application?

Please select two correct answers.

- A.

```
<login-config>
    <auth-method>FORM</auth-method>
    <login-config>
        <form-login-page>/login.jsp</form-login-page>
        <form-error-page>/error.jsp</form-error-page>
    </login-config>
```

```

</login-config>
B.
<security-role>
    <description>My description.</description>
    <role-name>Manager</role-name>
</security-role>
C.
<security-constraint>
    <web-resource-collection>
        <web-resource-name>SecureStuff</web-resource-name>
        <url-mapping>/servlet/secure</url-mapping>
        <http-method>POST</http-method>
    </web-resource-collection>
</security-constraint>
D.
<security-constraint>
    <auth-constraint>
        <role-name>Broker</role-name>
    </auth-constraint>
</security-constraint>
E.
<security-constraint>
    <web-resource-collection>
        <web-resource-name>SecureStuff</web-resource-name>
    </web-resource-collection>
    <auth-constraint>
        <role-name>Broker</role-name>
    </auth-constraint>
</security-constraint>

```

40. Based on the following information, please construct the full path for the servlet.

Please select one correct answer

```

docbase = c:/temp/
context path = /test
alias name = MyMail
servlet-name = com.jiris.common.util.MailServlet
url-pattern = /mail/*

```

- A.c:/temp/mail/com/jiris/common/util/MailServlet.class
- B.c:/temp/test/com/jiris/common/util/MailServlet.class
- C.c:/temp/mail/test/com/jiris/common/util/MailServlet.class
- D.c:/temp/test/mail/com/jiris/common/util/MailServlet.class

41. The ServletContext object are accessible from which of the following objects?

Please select three correct answers.

- A.HttpServlet
- B.GenericServlet
- C.HttpSession
- D.ServletConfig
- E.ServletResponse

42. Which of the following method is used to store object into a request object?

Please select one correct answer

- A. addAttribute(String name, String obj)

- B. putAttribute(String name, Object obj)
- C. setAttribute(String name, String obj)
- D. setAttribute(String name, Object obj)
- E. addObject(String name, Object obj)

43. Which request method will be invoked for the following code?

Please select one correct answer.

```
,code>
<html>
    <body>
        <form action='/servlet/comment'>
            <p>Please provide your comment here:</p>
            <input type='text' size='40' name='Comment'>
            <input type='submit' value='Submit'>
        </form>
    </body>
<html>
```

- A.GET
- B.POST
- C.HEAD
- D TRACE
- E.PUT

44.Which of the following method might be invoked more than one time?

Please select one correct answer

- A.doStartTag()
- B.doInitBody()
- C.doAfterBody()
- D.doEndTag()

45.Which of the following methods are used to send an error page to the client?

Please select two correct answers

- A.log(String msg)
- B.log(String msg, Throwable t)
- C.sendError(int code)
- D.sendError(int code, String msg)
- E.sendError(int code, String msg, Throwable t)

46.Which of the following requests should be performed by using a POST method?

Please select two corretion.

- A.Inserting a record into a database
- B.Accessing a static page
- C.Retrieving an image
- D.Sending credit card number
- E.Searching record in a database

47.Which of the following are CORRECT ways to define inactive period of 5 minutes of a session before the server invalidates it?

Please select two correct answers

- A.<session-timeout>5</session-timeout>
- B.<session-timeout>300</session-timeout>
- C.session.setMaxInactiveInterval(5);
- D.session.setMaxInactiveInterval(300);
- E.session.invalidate(5);

48.Which are the two mandatory attributes for JSP taglib directive?

Please select two correct answers.

- A.uri
- B.id
- C.name
- D.prefix
- E.value
- F.location

49.A session can be invalidated by which of the following:

Please select three correct answers

- A.After a default period of inactivity, say 30 minutes
- B.Client side user closes the browser
- C.After a specified period of inactivity, say 10 minutes
- D.Client side user machine crashes
- E.Explicitly invalidate a session through method calls

50.What is the method declaration for the method used in the HttpServlet class that handles the HTTP GET request?

Please select one correct answer

- A.doGet(ServletRequest req, ServletResponse res)
- B.getPage(ServletRequest req, ServletResponse res)
- C.doGet(HttpServletRequest req, HttpServletResponse res)
- D.service(HttpServletRequest req, HttpServletResponse res)

SCWCD 1.4 Mock Questions - 6 (Answers)

1)C

To notify the container to reevaluate the custom tag's body, you must return a value of IterationTag.EVAL_BODY_AGAIN in the doAfterBody() method.

2)B,C,D

jsp:bean declares the use of a JavaBean instance in a JSP page.

jsp:plugin instructs the JSP engine to generate appropriate HTML code for embedding applets on a web page.

Custom tags (taglibs) allow user-defined actions to be created.

Answer A is incorrect because the action JSP tag provides request-time instructions to the JSP engine.

Standard action types : jsp:include, jsp:forward, jsp:useBean, jsp:setProperty, jsp:getProperty, jsp:plugin

3)B,D

You can use either forward(...) or include(...) method to pass the request to another servlet to handle. While for forward(...), the control is passing to target servlet, for include(...), the control is still with the current servlet.

4)C,E

You need to override the service() method when you extends GenericServlet.
You need to call super.init(config) if you override this method.

5)B

A Data Access Object pattern is used to encapsulate database access functions. By putting database-specific SQL code into a separate layer as DAO layer, it is easy to modify it without affecting business logic layer, thus increase code manageability.

6)A,B

7)C,E

8)D

9)Client-Cert

The correct answer is CLIENT-CERT which stands for client certificate. It requires the client to provide a digitalcertificate containing information about the issuer, signature, serial number, key type, etc

10)B

11)A,C,D

12)C

13)B,C,D,E

14)C

15)A,D

16)A

17)A,C,D

18)C

19)A,C

20)B

21)A,B,D

22)C

23)A,C,E

24)C

25)A,C,D

26)C

27)D

28)E

29)A,B,C

30)A,D,E

31)B

32)A,F

33)A

34)B

35)A

36)C

37)A

38)B,C,E

39)B,E

40)B

41)A,C,D

42)D

43)A

44)C

45)C,D

46)A,D

47)A,D

48)A,D

49)A,C,E

50)C

1. Consider the following code:

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
{
    PrintWriter out = res.getWriter();
    out.println("Unable to find resource.");
    //1
    response.sendError(404);
}
```

Which of the following lines, when inserted at //1, will ensure that an IllegalStateException is NOT thrown?

Select 1 correct option.

- A.res.clear();
- B.res.empty();.
- C.res.remove();
- D.if(!res.isCommitted())
- E.if(res.getStatus() != res.COMMITTED)

2.Which of the following statements are correct JSP directives?

Select 2 correct options.

- A.<%@ page %>
- B.<%! taglib uri="http://www.abc.com/tags/util" prefix="util" %>
- C.<% include file="/copyright.html"%>
- D.<%@ taglib uri="http://www.abc.com/tags/util" prefix="util" %>
- E.<%\$ page language="java" import="com.abc.*"%>

3.Consider the following contents for two JSP files:

1

In file companyhome.jsp:

```
<html><body>

Welcome to ABC Corp!

<jsp:include page="companynews.jsp" />

</body></html>

<%@ page errorPage="simpleerrorhandler.jsp" %>

In file companynews.jsp:

<%@ page errorPage="advancederrorhandler.jsp" %>

<h3>Today's News</h3>
```

Select 1 correct option.

- A.When companyhome.jsp is requested, the output will contain "welcome..." as well as "Todays News".
- B.companyhome.jsp will not compile
- C.companynews.jsp will not compile
- D.Both the files will compile but will throw an exception at runtime
- E.None of these

4.Which HTTP method is used in FORM based Authentication?

Select 1 correct option.

- A.POST
- B.GET
- C.FORM
- D.HEAD

5.Consider the following code snippets. What will be displayed on the browser when a GET request is sent to FirstServlet assuming that the buffer is large enough to hold all the data before sending the data to the client?

In the doGet() of FirstServlet:

```
PrintWriter out = response.getWriter();
out.println("<html><body>Page 1");
RequestDispatcher rd = response.getRequestDispatcher("SecondServlet");
rd.forward(request, response);
```

In the doGet() of SecondServlet:

```
PrintWriter out = response.getWriter();
out.println("<br>Page 2</body></html>");
```

Select 1 correct option.

- A.Only Page1
- B.Only Page2
- C.Page1 and Page2
- D.IllegalStateException at Runtime.

6.Which of the given options are equivalent?

Select 2 correct options.

- A.<% Hashtable ht = new Hashtable(); %>
- B.<%= Hashtable ht = new Hashtable() %>
- C.<jsp:scriptlet>Hashtable ht = new Hashtable();</jsp:scriptlet>
- D.<jsp:code>Hashtable ht = new Hashtable();</jsp:code>
- E.<jsp:scriptlet>Hashtable ht = new Hashtable()</jsp:scriptlet>

7. When a session becomes invalid, which method is invoked on a session attribute implementing an appropriate interface?

Select 1 correct option.

- A.sessionInvalidated() of HttpSessionListener
- B.sessionDestroyed() of HttpSessionListener
- C.valueUnbound() of HttpSessionAttributeListener
- D.valueUnbound() of HttpSessionBindingListener
- E.valueUnbound() of HttpSessionListener

8. Your jsp page connects to the database and retrieves the data. It also formats the data and displays it to the client. Any of these operations can throw exceptions but you do not want to catch them all in this page and so you have written another jsp page that is meant to handle any kind of exceptions.

How would you associate that error page named "error.jsp" with this page?

Select 2 correct options.

- A.Add <%@errorPage="error.jsp"%> in this page.
- B.Add <%@page errorPage="error.jsp"%> in this page.
- C.Add <%@page isErrorPage="true"%> in error.jsp.
- D.Add <%@isErrorPage="true"%> in error.jsp.

9. Consider the HTML code shown in the exhibit. Which of the following method calls can retrieve the "email" value sent from the browser?

```
<html><body>

    <form action="/myapp/servlet/EmailCatcherServlet">

        Please enter your email: <input type="text" name="email">

        <input type="submit">

    </form>

</body></html>
```

Select 2 correct options.

- A.getParameter("email") of ServletRequest
- B.getParameterValues("email") of ServletRequest
- C.getField("email") of HttpServletRequest
- D.getFormValue("email") of HttpServletRequest
- E.getParameters("email") of HttpServlet

10. What will the following JSP page print?

```
<% { %>

    <jsp:useBean id="sb" class="java.lang.StringBuffer" />
    sb.append( "Hello" );

<% } %>

<%=sb%>
```

Select 1 correct option.

- A. It will print null
- B. It will print "Hello"
- C. It will not compile because <jsp:useBean> cannot be used inside a block
- D. It will not compile because StringBuffer is not a bean
- E. None of these

11. Which HTTP method would you use to test the validity, accessibility, or modification time of a hyperlink?

Select 1 correct option.

- A. GET
- B. POST
- C. HEAD
- D. OPTIONS
- E. PUT

12. A function has been defined in a tag library descriptor as follows:

```
<taglib>
    ...
    <function>
        <name>transform</name>
        <function-class>com.enthu.Functions</function-class>
        <function-signature>java.lang.String transformString(String)</function-signature>
    </function>
    ...
</taglib>
```

Which of the following statements are correct?

Select 1 correct option.

- A. The Function class must have a method with the signature: public String transform(String arg);
- B. The Function class must have a function with the signature: public String transformString(String arg);
- C. The Function class may have any method of the type: public static XXX(String s); but the method name (ie. XXX) must be mapped to "transform" in web.xml.
- D. The Function class may have any method of the type: public static XXX(String s); but the method name (ie. XXX) must be mapped to "transformString" in web.xml.
- E. None of these

13.In your report.jsp page, you want to include the output of "customer.jsp" page. But this page requires an additional parameter "custid", which is not present in the request. Which of the following code snippets does this.

```
1.  
  
<jsp:include page="customer.jsp" custid="1234" />  
2.  
  
<jsp:include page="customer.jsp">  
  
<jsp:param>  
  
<name>custid</name>  
  
<value>1234</value>  
  
</jsp:param>  
  
</jsp:include>  
  
3.  
  
<jsp:include page="customer.jsp" param-name="custid" param-value="1234" />  
4.  
  
<jsp:include page="customer.jsp">  
  
<jsp:param name="custid" value="1234" />  
  
</jsp:include>
```

Select 1 correct option.

- A.1
- B.2
- C.3
- D.4

14.Which of the following statements are correct regarding the import tag of JSTL?

Select 2 correct options.

- A.The String value of the imported content can be made available for use outside of the tag in 'var' variable.
- B.If the url contains a relative path, then the resource must exist in the same webapp.
- C.import tag is more efficient than jsp:include action in the case when large amount of data is imported only to be given to another tag.
- D.import tag is useful to convert URL when cookies are not supported by the client.
- E.The content imported by the import tag can be made available only through a String object.

15.What are the implications of using the HTTP GET method for a form submission?

Select 3 correct options.

- A.You cannot pass binary data to the server

- B.You cannot send unlimited (or a lot of) data to the server
- C.You cannot send multiple values for one parameter to the server
- D.You can only reply with the HEADER information in the response.
- E.The parameters will be appended to the URL as a query string

16. It is important to note that responses of a POST request are never cached.

```
1.  
<taglib>  
  
  <taglib-uri>/binomial</taglib-uri>  
  
  <taglib-location>/WEB-INF/MathLib.tld</taglib-location>  
  
</taglib>  
  
2.  
  
<taglib>  
  
  <taglib-uri>/binomial</taglib-uri>  
  
  <taglib-location>/WEB-INF/MathLib.jar</taglib-location>  
  
</taglib>  
  
3.  
  
<taglib id="ABC_MATH_LIB">  
  
  <taglib-uri>/binomial</taglib-uri>  
  
  <taglib-location>/WEB-INF/MathLib.tld</taglib-location>  
  
</taglib>  
  
4.  
  
<taglib name="ABC_MATH_LIB">  
  
  <taglib-uri>/binomial</taglib-uri>  
  
  <taglib-location>/WEB-INF/MathLib.jar</taglib-location>  
  
</taglib>  
  
5.  
  
<taglib author="ABCINC">  
  
  <taglib-uri>/binomial</taglib-uri>  
  
  <taglib-location>/WEB-INF/MathLib.tld</taglib-location>  
  
</taglib>
```

Select 3 correct options.

- A.1
- B.2
- C.3
- D.4
- E.5

17. Consider the JSP code:

```
<html>
  <head>
    <% int k = 0; %>
  </head>

<body>
</body>
</html>
```

In which method of the generated servlet will the declaration for 'k' be placed?

Select 1 correct option

- A.This will not compile as you cannot put JSP code in the <head> element
- B.init()
- C.doGet()
- D._jspService()
- E.constructor of the servlet.

18.Which event is received by a registered listener when an attribute is added to HttpSession?

Select 1 correct option

- A.HttpSessionChangeEvent
- B.HttpSessionEvent
- C.HttpAttributeChangeEvent
- D.HttpSessionBindingEvent
- E.HttpAttributeEvent

19.You are using a tag library with prefix "generator", which supports a tag named "random". This tag generates a random number and sets it to a variable named "value". Which of the following will output this value in the page?

Select 1 correct option

- A.<generator:random>value</generator:random>
- B.<generator:random><%=value%></generator:random>
- C.<generator:random><% int value; %> <%=value%></generator:random>
- D.<generator:random><%getParameter("value")%></generator:random>
- E.None of the above.

20. Identify the techniques that can be used to implement 'sessions' if the client browser does not support cookies.

Select 3 correct options

- A. Using Http headers
- B. Using https protocol.
- C. Hidden form fields
- D. URL rewriting
- E. It cannot be done without cookie support.

21. Which of the following are valid JSP scriptlets?

Select 2 correct options.

- A. <% String uid = LoginHelper.login(request) %>
- B. <% String uid = LoginHelper.login(request); %>
- C. <%! String uid = LoginHelper.login(request) %>
- D. <%@ String uid = LoginHelper.login(request) %>
- E. <% for(int i=0; i< 10; i++) { out.println(i); } %>

22. Consider the following web.xml code snippet:

```
<servlet>
    <servlet-name>BankServlet</servlet-name>
    <servlet-class>com.abc.bankapp.BankServlet</servlet-class>
    <security-role-ref>
        <role-name>manager</role-name>
        <role-link>supervisor</role-link>
    </security-role-ref>
</servlet>
```

Which of the following statements are correct?

Select 1 correct option.

- A. The servlet code should use "manager" as a parameter in request.isUserInRole() method
- B. The servlet code can use "manager" or "supervisor" as a parameter in request.isUserInRole() method
- C. The servlet code should use "supervisor" as a parameter in request.isUserInRole() method
- D. The role of "manager" must be defined in the servlet container
- E. None of these

22. You are designing a complex webapp that uses multi tier architecture. The application must provide interfaces for HTML as well as XML and should be maintainable. Which design pattern would you use?

Select 1 correct option.

- A. Data Access Object
- B. Business Delegate
- C. MVC

D.Remote Method Invocation

E.Transfer Object

23.Which of the following directives are applicable ONLY for tag files?

Select 3 correct options.

A.attribute

B.variable

C.page

D.include

E.import

F.tag

24.Which of the following are correct about FORM based authentication mechanism?

Select 3 correct options.

A.HTML FORM is used to capture the username and password of the user

B.Password is transmitted as plain text.

C.Password is transmitted in an encrypted form

D.Password is transmitted either in encrypted text or in plain text depending on the browser

E.This mechanism can be used over HTTPS.

25.Which pattern allows you to replace the presentation logic without much impact on the data representation?

Select 1 correct option.

A.Model View Controller

B.Business Delegate

C.Transfer Object

D.Data Access Object

E.Bimodal DataAccess

26.Identify the elements that help describe the attribute characteristics of a JSP custom tag in a TLD file.

Select 3 correct options.

A.value

B.name

C.description

D.rtexprvalue

E.class

27.Select the correct return types for ServletContext.getResource() and ServletContext.getResourceAsStream() methods.

Select 1 correct option.

A.java.io.Resource and java.io.InputStream

B.java.io.Resource and java.io.BufferedInputStream

C.java.net.URL and java.io.InputStream

D.java.io.File and java.io.InputStream

E.java.net.URL and java.io.FileInputStream

28.Which of the following are valid values for the <transport-guarantee> element?

Select 3 correct options.

- A.CONFIDENTIAL
- B.INTEGRAL
- C.SECURE
- D.ENCRYPTED
- E.NONE

29.Which method of ServletResponse would you use to set its content type?

Select 1 correct option.

- A.setParameter
- B.setHeader
- C.setAttribute
- D.setContentType
- E.None of the above

30.Which of the following lines can be used to retrieve a servlet initialization parameter "dbname" from the init() method of a servlet?

```
public void init()  
{  
    String dbname = //1 : Insert line here  
}
```

Select 2 correct options.

- A.getServletConfig().getParameter("dbname");
- B.getServletConfig().getInitParameter("dbname");
- C.getServletContext().getInitParameter("dbname");
- D.getInitParameter("dbname"); E.getInitParameterValue("dbname");

SCWCD 1.4 Mock Questions - 7 (Answers)

1)D

2)A,D

3)A

4)A

5)B

6)A,C

7)D

8)B,C

9)A,B

10)E

11)C

12)E

The Function class must have a method with signature: public static String transformString(String arg);

A is incorrect. The method name must be the same as given in <function-signature> element.

B is incorrect. Method name is correct but it should also be static.

13)D

14)A,C

15)A,B,E

16)A,B,C

17)D

This is a scriptlet and they always go inside the service method. ie. they are local to the request. A is incorrect. You can put JSP code anywhere

18) D

Following is the mapping of Listeners and their corresponding events.

HttpSessionListener : HttpSessionEvent

HttpSessionAttributeListener : HttpSessionBindingEvent

HttpSessionBindingListener : HttpSessionBindingEvent

Observe that both - HttpSessionAttributeListener and HttpSessionBindingListener, use HttpSessionBindingEvent.

19) B

C is incorrect It'll give an exception saying value is not initialized!

20) B,C,D

B is correct Unlike HTTP, HTTPS uses SSL which is a stateful protocol.

C is correct Remember that this is a non-standard and obsolete way.

This can only be done in an application specific way and requires that the page has a form.

21)B,E

A is incorrect. It does not have an ending semicolon

C is incorrect. ! is used for declarations

D is incorrect. @ is used for directives

22)A

D is incorrect "supervisor" must be defined in the container. For example, in conf/tomcat-users.xml for Tomcat.

<security-role-ref> is used to map the role names hard coded in the servlet code to the actual role names defined in the servlet container.

22)C

The statement "...should provide XML and HTML interfaces..." means the same data is represented in different ways, therefore this is MVC.

23) A,B,F

B Valid only for tag files.

D is incorrect include directive is valid for regular JSP file also

E is incorrect No such directive

24)A,B,E

C is incorrect This is done in HTTP Digest authentication mechanism

25)A

A is correct. A view (ie. the presentation of data) knows how to present the data and so can be replaced with another view without any impact on the data representation.

B is incorrect. This allows plug and play between back end logic and the front end.

D is incorrect. This allows plug and play between the data container (the DB) and data requestor.

26)B,C,D

An attribute element describes the attribute for a tag. Following is its definition.

<!ELEMENT attribute (name, required?, rtxprvalue?, type?, description?) >

27)C

28)A,B,E

29)D

30)B,D

Calling getServletConfig() from the init() method returns the ServletConfig object for this servlet and calling getInitParameter(...) on the ServletConfig object returns the value of that parameter.

ePad

Sun 310-083

Sun Certified Web Component Developer for J2EE 5

276 Q&A

Version 2.73

ePad Tool

www.ExamWorx.com

Important Note, Please Read Carefully

Other ExamWorx products

A) Offline Testing engine

Use the offline Testing engine product to practice the questions in an exam environment.

Build a foundation of knowledge which will be useful also after passing the exam.

Latest Version

We are constantly reviewing our products. New material is added and old material is revised. Free updates are available for 90 days after the purchase. You should check your member zone at ExamWorx and update 3-4 days before the scheduled exam date.

Here is the procedure to get the latest version:

1.Go to www.ExamWorx.com

2.Click on **Log in**

3.The latest versions of all purchased products are downloadable from here. Just click the links.

For most updates,it is enough just to print the new questions at the end of the new version, not the whole document.

Feedback

If you spot a possible improvement then please let us know. We are always interested in improving product quality. Feedback should be sent to feedback@ExamWorx.com. You should include the following: Exam number, version, page number, question number, and your login Email.

Our experts will answer your mail promptly.

Copyright

Each iPAD file is a green exe file. If we find out that a particular iPAD Viewer file is being distributed by you, ExamWorx reserves the right to take legal action against you according to the International Copyright Laws.

Explanations

This product does not include explanations at the moment. If you are interested in providing explanations for this exam, please contact feedback@ExamWorx.com.

www.ExamWorx.com Q: 1 To take advantage of the capabilities of modern browsers that use web standards, such as XHTML and CSS, your web application is being converted from simple JSP pages to JSP Document format. However, one of your JSPs, /scripts/screenFunctions.jsp, generates a JavaScript file. This file is included in several web forms to create screen-specific validation functions and are included in these pages with the following statement:

```
10. <head>
11. <script src='/scripts/screenFunctions.jsp'
12.     language='javascript'
13.     type='application/javascript'> </script>
14. </head>
15. <!-- body of the web form -->
```

Which JSP code snippet declares that this JSP Document is a JavaScript file?

- A. <%@ page contentType='application/javascript' %>
- B. <jsp:page contentType='application/javascript' />
- C. <jsp:document contentType='application/javascript' />
- D. <jsp:directive.page contentType='application/javascript' />
- E. No declaration is needed because the web form XHTML page already declares the MIME type of the /scripts/screenFunctions.jsp file in the <script> tag.

Answer: D

www.ExamWorx.com Q: 2 Given the JSP code:

```
10. <html>
11. <body>
12. <jsp:useBean id='customer' class='com.example.Customer' />
13. Hello, ${customer.title} ${customer.lastName}, welcome
14. to Squeaky Beans, Inc.
15. </body>
16. </html>
```

Which three types of JSP code are used? (Choose three.)

- A. Java code
- B. template text
- C. scripting code
- D. standard action

E. expression language

Answer: B, D, E

www.ExamWorx.com Q: 3 You have built a collection of custom tags for your web application. The TLD file is located in the file: /WEB-INF/myTags.xml. You refer to these tags in your JSPs using the symbolic name: myTags. Which deployment descriptor element must you use to make this link between the symbolic name and the TLD file name?

A. <taglib>

```
<name>myTags</name>
<location>/WEB-INF/myTags.xml</location>
</taglib>
```

B. <tags>

```
<name>myTags</name>
<location>/WEB-INF/myTags.xml</location>
</tags>
```

C. <tags>

```
<tags-uri>myTags</taglib-uri>
<tags-location>/WEB-INF/myTags.xml</tags-location>
</tags>
```

D. <taglib>

```
<taglib-uri>myTags</taglib-uri>
<taglib-location>/WEB-INF/myTags.xml</taglib-location>
</taglib>
```

Answer: D

www.ExamWorx.com Q: 4 Which implicit object is used in a JSP page to retrieve values associated with <context-param> entries in the deployment descriptor?

A. config

B. request

C. session

D. application

Answer: D

www.ExamWorx.com Q: 5 Click the Task button.

Place the events in the order they occur.

Drag and Drop

Place the events in the order they occur.

Order of Steps	Events
1st	jsplit is called
2nd	JSP page implementation class is loaded
3rd	JSP page is compiled
4th	jspDestroy is called
5th	JSP page implementation is instantiated
6th	JSP page is translated
7th	_jspService is called

Done

Answer: Check ExamWorx eEngine, Download from Member Center

www.ExamWorx.com Q: 6 Click the Task button.

Place the code snippets in the proper order to construct the JSP code to import static content into a JSP page at translation-time.

Drag and Drop

Place the code snippets in the proper order to construct the JSP code to import static content into a JSP page at translation-time.

JSP Code:

Place here. Place here. Place here.

Code Snippets:

import='foo.jsp'	/>	file='foo.jsp'
<%@ include	<jsp:import	page='foo.jsp'
%>	<%@ import	<jsp:include

Done

The interface consists of a title bar labeled "Drag and Drop" with standard window controls. Below it is a text area with three yellow rectangular boxes, each containing the placeholder text "Place here." followed by a space. To the right of these boxes is a "Done" button. Underneath the yellow boxes is a section titled "Code Snippets" with three rows of cyan rectangular boxes. Each row contains three JSP-related tags: the first row has "import='foo.jsp'", "/>", and "file='foo.jsp'"; the second row has "<%@ include", "<jsp:import", and "page='foo.jsp'"; the third row has "%>", "<%@ import", and "<jsp:include>".

Answer: Check ExamWorx eEngine, Download from Member Center

www.ExamWorx.com Q: 7 You have created a JSP that includes instance variables and a great deal of scriptlet code. Unfortunately, after extensive load testing, you have discovered several race conditions in your JSP scriptlet code. To fix these problems would require significant recoding, but you are already behind schedule. Which JSP code snippet can you use to resolve these concurrency problems?

- A. <%@ page isThreadSafe='false' %>
- B. <%@ implements SingleThreadModel %>
- C. <%! implements SingleThreadModel %>
- D. <%@ page useSingleThreadModel='true' %>
- E. <%@ page implements='SingleThreadModel' %>

Answer: A

www.ExamWorx.com Q: 8 Click the Exhibit button.

The attribute "name" has a value of "Foo,"

What is the result if this tag handler's tag is invoked?

```
5. public class MyTagHandler extends
   TagSupport {
  6.     public int doStartTag() throws
   JspException {
  7.         try {
  8.             Writer out =
pageContext.getResponse().getWriter();
  9.             String name =
pageContext.findAttribute("name");
10.             out.print(name);
11.         } catch(Exception ex) { /* handle
exception */ }
12.         return SKIP_BODY;
13.     }
14.
15.     public int doAfterBody() throws
   JspException {
16.         try {
17.             Writer out =
pageContext.getResponse().getWriter();
18.             out.print("done");
19.         } catch(Exception ex) { /* handle
exception */ }
20.         return EVAL_PAGE;
21.     }
22. }
```

- A. Foo
- B. done
- C. Foodone
- D. An exception is thrown at runtime.
- E. No output is produced from this code.
- F. Compilation fails because of an error in this code.

Answer: A

www.ExamWorx.com Q: 9 You are building a web application that will be used throughout the European Union; therefore, it has significant internationalization requirements. You have been tasked to create a custom tag that generates a message using the `java.text.MessageFormat` class. The tag will take the `resourceKey` attribute and a variable number of argument attributes with the format, `arg<N>`. Here is an example use of this tag and its output:

```
<t:message resourceKey='diskFileMsg' arg0='MyDisk' arg1='1247' />
```

generates:

The disk "MyDisk" contains 1247 file(s).

Which Simple tag class definition accomplishes this goal of handling a variable number of tag attributes?

A. public class MessageTag extends SimpleTagSupport
implements VariableAttributes {
private Map attributes = new HashMap();
public void setVariableAttribute(String uri,
String name, Object value) {
this.attributes.put(name, value);
}
// more tag handler methods
}

B. The Simple tag model does NOT support a variable number of attributes.

C. public class MessageTag extends SimpleTagSupport
implements DynamicAttributes {
private Map attributes = new HashMap();
public void putAttribute(String name, Object value) {
this.attributes.put(name, value);
}
// more tag handler methods
}

D. public class MessageTag extends SimpleTagSupport
implements VariableAttributes {
private Map attributes = new HashMap();
public void putAttribute(String name, Object value) {
this.attributes.put(name, value);
}
// more tag handler methods
}

E. public class MessageTag extends SimpleTagSupport
implements DynamicAttributes {
private Map attributes = new HashMap();

```
public void setDynamicAttribute(String uri, String name,  
Object value) {  
this.attributes.put(name, value);  
}  
// more tag handler methods  
}
```

Answer: E

www.ExamWorx.com Q: 10 Given the JSP code:

```
<% request.setAttribute("foo", "bar"); %>
```

and the Classic tag handler code:

5. public int doStartTag() throws JspException {
6. // insert code here
7. // return int
8. }

Assume there are no other "foo" attributes in the web application.

Which invocation on the pageContext object, inserted at line 6, assigns "bar" to the variable x?

- A. String x = (String) pageContext.getAttribute("foo");
- B. String x = (String) pageContext.getRequestScope("foo");
- C. It is NOT possible to access the pageContext object from within doStartTag.
- D. String x = (String)
pageContext.getRequest().getAttribute("foo");
- E. String x = (String) pageContext.getAttribute("foo",
PageContext.ANY_SCOPE);

Answer: D

www.ExamWorx.com Q: 11 Which two statements about tag files are true? (Choose two.)

- A. Classic tag handlers and tag files CANNOT reside in the same tag library.
- B. A file named foo.tag, located in /WEB-INF/tags/bar, is recognized as a tag file by the container.
- C. A file named foo.tag, bundled in a JAR file but NOT defined in a TLD, triggers a container translation error.
- D. A file named foo.tag, located in a web application's root directory, is recognized as a tag file by the container.

- E. If files foo1.tag and foo2.tag both reside in /WEB-INF/tags/bar, the container will consider them part of the same tag library.

Answer: B, E

www.ExamWorx.com Q: 12 The sl:shoppingList and sl:item tags output a shopping list to the response and are used as follows:

11. <sl:shoppingList>
12. <sl:item name="Bread" />
13. <sl:item name="Milk" />
14. <sl:item name="Eggs" />
15. </sl:shoppingList>

The tag handler for sl:shoppingList is ShoppingListTag and the tag handler for sl:item is ItemSimpleTag.

ShoppingListTag extends BodyTagSupport and ItemSimpleTag extends SimpleTagSupport.

Which is true?

- A. ItemSimpleTag can find the enclosing instance of ShoppingListTag by calling getParent() and casting the result to ShoppingListTag.
- B. ShoppingListTag can find the child instances of ItemSimpleTag by calling super.getChildren() and casting each to an ItemSimpleTag.
- C. It is impossible for ItemSimpleTag and ShoppingListTag to find each other in a tag hierarchy because one is a Simple tag and the other is a Classic tag.
- D. ShoppingListTag can find the child instances of ItemSimpleTag by calling getChildren() on the PageContext and casting each to an ItemSimpleTag.
- E. ItemSimpleTag can find the enclosing instance of ShoppingListTag by calling findAncestorWithClass() on the PageContext and casting the result to ShoppingListTag.

Answer: A

www.ExamWorx.com Q: 13 Servlet A receives a request that it forwards to servlet B within another web application in the same web container. Servlet A needs to share data with servlet B and that data must not be visible to other servlets in A's web application. In which object can the data that A shares with B be stored?

- A. HttpSession
- B. ServletConfig
- C. ServletContext

- D. HttpServletRequest
- E. HttpServletResponse

Answer: D

www.ExamWorx.com Q: 14 Your web site has many user-customizable features, for example font and color preferences on web pages. Your IT department has already built a subsystem for user preferences using the Java SE platform's lang.util.prefs package APIs, and you have been ordered to reuse this subsystem in your web application. You need to create an event listener that constructs the preferences factory and stores it in the application scope for later use. Furthermore, this factory requires that the URL to a database must be declared in the deployment descriptor like this:

- 42. <context-param>
- 43. <param-name>prefsDbURL</param-name>
- 44. <param-value>
- 45. jdbc:pointbase:server://dbhost:4747/prefsDB
- 46. </param-value>
- 47. </context-param>

Which partial listener class will accomplish this goal?

- A. public class PrefsFactoryInitializer implements ContextListener {
public void contextInitialized(ServletContextEvent e) {
ServletContext ctx = e.getServletContext();
String prefsURL = ctx.getParameter("prefsDbURL");
PreferencesFactory myFactory = makeFactory(prefsURL);
ctx.putAttribute("myPrefsFactory", myFactory);
}
// more code here
}
B. public class PrefsFactoryInitializer implements ServletContextListener {
public void contextCreated(ServletContext ctx) {
String prefsURL = ctx.getInitParameter("prefsDbURL");
PreferencesFactory myFactory = makeFactory(prefsURL);
ctx.setAttribute("myPrefsFactory", myFactory);
}
// more code here
}
C. public class PrefsFactoryInitializer implements ServletContextListener {
public void contextInitialized(ServletContextEvent e) {
ServletContext ctx = e.getServletContext();
String prefsURL = ctx.getInitParameter("prefsDbURL");
PreferencesFactory myFactory = makeFactory(prefsURL);
ctx.setAttribute("myPrefsFactory", myFactory);

```

}
// more code here
}
D. public class PrefsFactoryInitializer implements ContextListener {
public void contextCreated(ServletContext ctx) {
String prefsURL = ctx.getParameter("prefsDbURL");
PreferencesFactory myFactory = makeFactory(prefsURL);
ctx.putAttribute("myPrefsFactory", myFactory);
}
// more code here
}

```

Answer: C

www.ExamWorx.com Q: 15 A developer wants a web application to be notified when the application is about to be shut down. Which two actions are necessary to accomplish this goal? (Choose two.)

- A. include a listener directive in a JSP page
- B. configure a listener in the TLD file using the <listener> element
- C. include a <servlet-destroy> element in the web application deployment descriptor
- D. configure a listener in the application deployment descriptor, using the <listener> element
- E. include a class implementing ServletContextListener as part of the web application deployment
- F. include a class implementing ContextDestroyedListener as part of the web application deployment
- G. include a class implementing HttpSessionAttributeListener as part of the web application deployment

Answer: D, E

www.ExamWorx.com Q: 16 You want to create a filter for your web application and your filter will implement javax.servlet.Filter.

Which two statements are true? (Choose two.)

- A. Your filter class must implement an init method and a destroy method.
- B. Your filter class must also implement javax.servlet.FilterChain.
- C. When your filter chains to the next filter, it should pass the same arguments it received in its doFilter method.
- D. The method that your filter invokes on the object it received that implements javax.servlet.FilterChain can invoke either another filter or a servlet.
- E. Your filter class must implement a doFilter method that takes, among other things, an HttpServletRequest object and an HttpServletResponse object.

Answer: A, D

www.ExamWorx.com Q: 17 Which three are true about the HttpServletRequestWrapper class? (Choose three.)

- A. The HttpServletRequestWrapper is an example of the Decorator pattern.
- B. The HttpServletRequestWrapper can be used to extend the functionality of a servlet request.
- C. A subclass of HttpServletRequestWrapper CANNOT modify the behavior of the getReader method.
- D. An HttpServletRequestWrapper may be used only by a class implementing the javax.servlet.Filter interface.
- E. An HttpServletRequestWrapper CANNOT be used on the request passed to the RequestDispatcher.include method.
- F. An HttpServletRequestWrapper may modify the header of a request within an object implementing the javax.servlet.Filter interface.

Answer: A, B, F

www.ExamWorx.com Q: 18 Click the Exhibit button.

The resource requested by the RequestDispatcher is available and implemented by the DestinationServlet.

What is the result?

```

// From file SourceServlet.java
11. public class SourceServlet extends
HttpServlet {
12.     public void service(HttpServletRequest
request,
13.                         HttpServletResponse
response)
14.         throws ServletException,
IOException {
15.     ServletContext
cxt=getServletConfig().getServletContext();
16.     RequestDispatcher rd =
17.             cxt.getRequestDispatcher("/dest
n");
18.     response.getWriter().println("hello
from source");
19.     response.flushBuffer();
20.     rd.forward(request, response);
21. }
22. }

// From file DestinationServlet.java
11. public class DestinationServlet extends
HttpServlet {
12.     public void service(HttpServletRequest
request,
13.                         HttpServletResponse
response)
14.         throws ServletException,
IOException {
15.     response.getWriter().println("hello
from dest");
17.     response.flushBuffer();
18. }
19. }

```

- A. An exception is thrown at runtime by SourceServlet.
- B. An exception is thrown at runtime by DestinationServlet.
- C. Only "hello from dest" appears in the response output stream.
- D. Both "hello from source" and "hello from dest" appear in the response output stream.

Answer: A

www.ExamWorx.com Q: 19 A developer wants to make a name attribute available to all servlets associated with a particular user, across multiple requests from that user, from the same browser instance.

Which two provide this capability from within a tag handler? (Choose two.)

- A. pageContext.setAttribute("name", theValue);
- B. pageContext.setAttribute("name", getSession());
- C. pageContext.getRequest().setAttribute("name", theValue);
- D. pageContext.getSession().setAttribute("name", theValue);
- E. pageContext.setAttribute("name", theValue,
PageContext.PAGE_SCOPE);
- F. pageContext.setAttribute("name", theValue,
PageContext.SESSION_SCOPE);

Answer: D, F

www.ExamWorx.com Q: 20 Given the definition of MyServlet:

```
11. public class MyServlet extends HttpServlet {  
12.     public void service(HttpServletRequest request,  
13.                           HttpServletResponse response)  
14.         throws ServletException, IOException {  
15.     HttpSession session = request.getSession();  
16.     session.setAttribute("myAttribute","myAttributeValue");  
17.     session.invalidate();  
18.     response.getWriter().println("value=" +  
19.                               session.getAttribute("myAttribute"));  
20. }  
21. }
```

What is the result when a request is sent to MyServlet?

- A. An IllegalStateException is thrown at runtime.
- B. An InvalidSessionException is thrown at runtime.
- C. The string "value=null" appears in the response stream.
- D. The string "value=myAttributeValue" appears in the response stream.

Answer: A

www.ExamWorx.com Q: 21 You need to store a Java long primitive attribute, called customerOID, into the session scope. Which two code snippets allow you to insert this value into the session? (Choose two.)

- A. long customerOID = 47L;
session.setAttribute("customerOID", new Long(customerOID));
- B. long customerOID = 47L;
session.setLongAttribute("customerOID", new Long(customerOID));
- C. long customerOID = 47L;
session.setAttribute("customerOID", customerOID);
- D. long customerOID = 47L;
session.setNumericAttribute("customerOID", new Long(customerOID));
- E. long customerOID = 47L;
session.setLongAttribute("customerOID", customerOID);
- F. long customerOID = 47L;
session.setNumericAttribute("customerOID", customerOID);

Answer: A, C

www.ExamWorx.com Q: 22 A developer for the company web site has been told that users may turn off cookie support in their browsers. What must the developer do to ensure that these customers can still use the web application?

- A. The developer must ensure that every URL is properly encoded using the appropriate URL rewriting APIs.
- B. The developer must provide an alternate mechanism for managing sessions and abandon the HttpSession mechanism entirely.
- C. The developer can ignore this issue. Web containers are required to support automatic URL rewriting when cookies are not supported.
- D. The developer must add the string id=<sessionid> to the end of every URL to ensure that the conversation with the browser can continue.

Answer: A

www.ExamWorx.com Q: 23 Your web application requires the adding and deleting of many session attributes during a complex use case. A bug report has come in that indicates that an important session attribute is being deleted too soon and a NullPointerException is being thrown several interactions after the fact. You have decided to create a session event listener that will log when attributes are being deleted so you can track down when the attribute is erroneously being deleted.

Which listener class will accomplish this debugging goal?

- A. Create an HttpSessionAttributeListener class and implement the attributeDeleted method and log the attribute name using the getName method on the event object.
- B. Create an HttpSessionAttributeListener class and implement the attributeRemoved method and log the attribute name using the getName method on the event object.
- C. Create an SessionAttributeListener class and implement the attributeRemoved method and log the attribute name using the getAttributeName method on the event object.
- D. Create an SessionAttributeListener class and implement the attributeDeleted method and log the attribute name using the getAttributeName method on the event object.

Answer: B

www.ExamWorx.com Q: 24 As a convenience feature, your web pages include an Ajax request every five minutes to a special servlet that monitors the age of the user's session. The client-side JavaScript that handles the Ajax callback displays a message on the screen as the session ages. The Ajax call does NOT pass any cookies, but it passes the session ID in a request parameter called sessionID. In addition, assume that your webapp keeps a hashmap of session objects by the ID. Here is a partial implementation of this servlet:

```

10. public class SessionAgeServlet extends HttpServlet {
11.   public void service(HttpServletRequest request, HttpServletResponse) throws IOException {
12.     String sessionID = request.getParameter("sessionID");
13.     HttpSession session = getSession(sessionID);
14.     long age = // your code here
15.     response.getWriter().print(age);
16.   } ... // more code here
47. }
```

Which code snippet on line 14, will determine the age of the session?

- A. session.getMaxInactiveInterval();
- B. session.getLastAccessed().getTime() - session.getCreationTime().getTime();
- C. session.getLastAccessedTime().getTime() - session.getCreationTime().getTime();
- D. session.getLastAccessed() - session.getCreationTime();
- E. session.getMaxInactiveInterval() - session.getCreationTime();
- F. session.getLastAccessedTime() - session.getCreationTime();

Answer: F

www.ExamWorx.com Q: 25 Which statement is true about web container session management?

- A. Access to session-scoped attributes is guaranteed to be thread-safe by the web container.
- B. To activate URL rewriting, the developer must use the HttpServletResponse.setURLRewriting method.

- C. If the web application uses HTTPS, then the web container may use the data on the HTTPS request stream to identify the client.
- D. The JSESSIONID cookie is stored permanently on the client so that a user may return to the web application and the web container will rejoin that session.

Answer: C

www.ExamWorx.com Q: 26 One of the use cases in your web application uses many session-scoped attributes. At the end of the use case, you want to clear out this set of attributes from the session object. Assume that this static variable holds this set of attribute names:

```
201. private static final Set<String> USE_CASE_ATTRS;  
202. static {  
203.     USE_CASE_ATTRS.add("customerOID");  
204.     USE_CASE_ATTRS.add("custMgrBean");  
205.     USE_CASE_ATTRS.add("orderOID");  
206.     USE_CASE_ATTRS.add("orderMgrBean");  
207. }
```

Which code snippet deletes these attributes from the session object?

- A. session.removeAll(USE_CASE_ATTRS);
- B. for (String attr : USE_CASE_ATTRS) {
session.removeAttribute(attr);
}
- C. for (String attr : USE_CASE_ATTRS) {
session.removeAttribute(attr);
}
- D. for (String attr : USE_CASE_ATTRS) {
session.deleteAttribute(attr);
}
- E. session.deleteAllAttributes(USE_CASE_ATTRS);

Answer: C

www.ExamWorx.com Q: 27 Assume that a news tag library contains the tags lookup and item:

**lookup Retrieves the latest news headlines and executes the tag body once for each headline.
Exposes a NESTED page-scoped attribute called headline of type com.example.Headline containing details for that headline.**

item Outputs the HTML for a single news headline. Accepts an attribute info of type com.example.Headline containing details for the headline to be rendered. Which snippet of JSP code returns the latest news headlines in an HTML table, one per row?

- A. <table>
<tr>
<td>
<news:lookup />
<news:item info="\${headline}" />
</td>
</tr>
</table>
- B. <news:lookup />
<table>
<tr>
<td><news:item info="\${headline}" /></td>
</tr>
</table>
- C. <table>
<news:lookup>
<tr>
<td><news:item info="\${headline}" /></td>
</tr>
</news:lookup>
</table>
- D. <table>
<tr>
<news:lookup>
<td><news:item info="\${headline}" /></td>
</news:lookup>
</tr>
</table>

Answer: C

www.ExamWorx.com Q: 28 Which JSTL code snippet can be used to perform URL rewriting?

- A. <a href='<c:url url="foo.jsp"/>' />
- B. <a href='<c:link url="foo.jsp"/>' />
- C. <a href='<c:url value="foo.jsp"/>' />
- D. <a href='<c:link value="foo.jsp"/>' />

Answer: C

www.ExamWorx.com Q: 29 Assume the scoped attribute priority does NOT yet exist. Which two create and set a new request-scoped attribute priority to the value "medium"? (Choose two.)

- A. \${priority = 'medium'}
- B. \${requestScope['priority'] = 'medium'}
- C. <c:set var="priority" value="medium" />
- D. <c:set var="priority" scope="request">medium</c:set>
- E. <c:set var="priority" value="medium" scope="request" />
- F. <c:set property="priority" scope="request">medium</c:set>
- G. <c:set property="priority" value="medium" scope="request" />

Answer: D, E

www.ExamWorx.com Q: 30 You are creating a JSP page to display a collection of data. This data can be displayed in several different ways so the architect on your project decided to create a generic servlet that generates a comma-delimited string so that various pages can render the data in different ways. This servlet takes on request parameter: objectID. Assume that this servlet is mapped to the URL pattern: /WEB-INF/data.

In the JSP you are creating, you need to split this string into its elements separated by commas and generate an HTML list from the data.

Which JSTL code snippet will accomplish this goal?

- A. <c:import varReader='dataString' url='/WEB-INF/data'>
<c:param name='objectID' value='\${currentOID}' />
</c:import>

<c:forTokens items='\${dataString.split(",")}' var='item'>
\${item}
</c:forTokens>

- B. <c:import varReader='dataString' url='/WEB-INF/data'>
<c:param name='objectID' value='\${currentOID}' />
</c:import>

<c:forTokens items='\${dataString}' delims=',' var='item'>
\${item}
</c:forTokens>

- C. <c:import var='dataString' url='/WEB-INF/data'>

```

<c:param name='objectID' value='${currentOID}' />
</c:import>
<ul>
<c:forTokens items='${dataString.split(",")}' var='item'>
<li>${item}</li>
</c:forTokens>
</ul>
D. <c:import var='dataString' url='/WEB-INF/data'>
<c:param name='objectID' value='${currentOID}' />
</c:import>
<ul>
<c:forTokens items='${dataString}' delims=',' var='item'>
<li>${item}</li>
</c:forTokens>
</ul>

```

Answer: D

www.ExamWorx.com Q: 31 Which three are true about TLD files? (Choose three.)

- A. The web container recognizes TLD files placed in any subdirectory of WEB-INF.
- B. When deployed inside a JAR file, TLD files must be in the META-INF directory, or a subdirectory of it.
- C. A tag handler's attribute must be included in the TLD file only if the attribute can accept request-time expressions.
- D. The web container can generate an implicit TLD file for a tag library comprised of both simple tag handlers and tag files.
- E. The web container can automatically extend the tag library map described in a web.xml file by including entries extracted from the web application's TLD files.

Answer: A, B, E

www.ExamWorx.com Q: 32 Your management has required that all JSPs be created to generate XHTML-compliant content and to facilitate that decision, you are required to create all JSPs using the JSP Document format. In the reviewOrder.jspx page, you need to use several core JSTL tags to process the collection of order items in the customer's shopping cart. Which JSP code snippets must you use in the reviewOrder.jspx page?

- A. <html xmlns:jspt="http://java.sun.com/JSP/Page"
version="2.0">
<jsp:directive.taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" />
<!-- page content -->

```

</html>
B. <html xmlns:jsp="http://java.sun.com/JSP/Page"
version="2.0"
xmlns:c="http://java.sun.com/jsp/jstl/core">
<!-- page content -->
</html>
C. <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
version="2.0">
<jsp:directive.taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" />
<!-- page content -->
</jsp:root>
D. <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
version="2.0"
xmlns:c="http://java.sun.com/jsp/jstl/core">
<!-- page content -->
</jsp:root>

```

Answer: D

www.ExamWorx.com Q: 33 Which two JSTL URL-related tags perform URL rewriting? (Choose two.)

- A. url
- B. link
- C. param
- D. import
- E. redirect

Answer: A, E

www.ExamWorx.com Q: 34 A custom JSP tag must be able to support an arbitrary number of attributes whose names are unknown when the tag class is designed. Which two are true? (Choose two.)

- A. The <body-content> element in the echo tag TLD must have the value JSP.
- B. The echo tag handler must define the setAttribute(String key, String value) method.
- C. The <dynamic-attributes>true</dynamic-attributes> element must appear in the echo tag TLD.
- D. The class implementing the echo tag handler must implement the javax.servlet.jsp.tagext.IterationTag interface.
- E. The class implementing the echo tag handler must implement the javax.servlet.jsp.tagext.DynamicAttributes interface.

Answer: C, E

www.ExamWorx.com Q: 35 A developer has used this code within a servlet:

```
62. if(request.isUserInRole("vip")) {  
63.   // VIP-related logic here  
64. }
```

What else must the developer do to ensure that the intended security goal is achieved?

- A. create a user called vip in the security realm
- B. define a group within the security realm and call it vip
- C. define a security-role named vip in the deployment descriptor
- D. declare a security-role-ref for vip in the deployment descriptor

Answer: D

www.ExamWorx.com Q: 36 Given:

```
3. class MyServlet extends HttpServlet {  
4.   public void doPut(HttpServletRequest req, HttpServletResponse resp) throws ServletException,  
IOException {  
5.     // servlet code here ...  
26.   }  
27. }
```

If the DD contains a single security constraint associated with MyServlet and its only <http-method> tags and <auth-constraint> tags are:

```
<http-method>GET</http-method>  
<http-method>PUT</http-method>  
<auth-constraint>Admin</auth-constraint>
```

Which four requests would be allowed by the container? (Choose four.)

- A. A user whose role is Admin can perform a PUT.
- B. A user whose role is Admin can perform a GET.
- C. A user whose role is Admin can perform a POST.
- D. A user whose role is Member can perform a PUT.
- E. A user whose role is Member can perform a POST.
- F. A user whose role is Member can perform a GET.

Answer: A, B, C, E

www.ExamWorx.com Q: 37 What is true about Java EE authentication mechanisms?

- A. If your deployment descriptor correctly declares an authentication type of CLIENT_CERT, your users must have a certificate from an official source before they can use your application.
- B. If your deployment descriptor correctly declares an authentication type of BASIC, the container automatically requests a user name and password whenever a user starts a new session.
- C. If you want your web application to support the widest possible array of browsers, and you want to perform authentication, the best choice of Java EE authentication mechanisms is DIGEST.
- D. To use Java EE FORM authentication, you must declare two HTML files in your deployment descriptor, and you must use a predefined action in the HTML file that handles your user's login.

Answer: D

www.ExamWorx.com Q: 38 If you want to use the Java EE platform's built-in type of authentication that uses a custom HTML page for authentication, which two statements are true? (Choose two.)

- A. Your deployment descriptor will need to contain this tag:
<auth-method>CUSTOM</auth-method>.
- B. The related custom HTML login page must be named loginPage.html.
- C. When you use this type of authentication, SSL is turned on automatically.
- D. You must have a tag in your deployment descriptor that allows you to point to both a login HTML page and an HTML page for handling any login errors.
- E. In the HTML related to authentication for this application, you must use predefined variable names for the variables that store the user and password values.

Answer: D, E

www.ExamWorx.com Q: 39 Given this fragment in a servlet:

```
23. if(req.isUserInRole("Admin")) {  
24.   // do stuff  
25. }
```

And the following fragment from the related Java EE deployment descriptor:

```
812. <security-role-ref>  
813.   <role-name>Admin</role-name>  
814.   <role-link>Administrator</role-link>  
815. </security-role-ref>  
900. <security-role>
```

901. <role-name>Admin</role-name>
902. <role-name>Administrator</role-name>
903. </security-role>

What is the result?

- A. Line 24 can never be reached.
- B. The deployment descriptor is NOT valid.
- C. If line 24 executes, the user's role will be Admin.
- D. If line 24 executes, the user's role will be Administrator.
- E. If line 24 executes the user's role will NOT be predictable.

Answer: D

www.ExamWorx.com Q: 40 Given the security constraint in a DD:

101. <security-constraint>
102. <web-resource-collection>
103. <web-resource-name>Foo</web-resource-name>
104. <url-pattern>/Bar/Baz/*</url-pattern>
105. <http-method>POST</http-method>
106. </web-resource-collection>
107. <auth-constraint>
108. <role-name>DEVELOPER</role-name>
109. </auth-constraint>
110. </security-constraint>

And given that "MANAGER" is a valid role-name, which four are true for this security constraint? (Choose four.)

- A. MANAGER can do a GET on resources in the /Bar/Baz directory.
- B. MANAGER can do a POST on any resource in the /Bar/Baz directory.
- C. MANAGER can do a TRACE on any resource in the /Bar/Baz directory.
- D. DEVELOPER can do a GET on resources in the /Bar/Baz directory.
- E. DEVELOPER can do only a POST on resources in the /Bar/Baz directory.
- F. DEVELOPER can do a TRACE on any resource in the /Bar/Baz directory.

Answer: A, C, D, F

www.ExamWorx.com Q: 41 Which three are valid URL mappings to a servlet in a web deployment descriptor? (Choose three.)

- A. /*
- B. *.do
- C. MyServlet
- D. /MyServlet
- E. /MyServlet/*
- F. MyServlet/*.jsp

Answer: B, D, E

www.ExamWorx.com Q: 42 Click the Task button.

Place the appropriate element names on the left on the web application deployment descriptor on the right so that files ending in ".mpg" are associated with the MIME type "video/mpeg."

Drag and Drop

Place the appropriate element names on the left on the web application deployment descriptor on the right so that files ending in ".mpg" are associated with the MIME type "video/mpeg."

Web Application Deployment Descriptor Snippet

Element Names	
< Place here. >	file-type
< Place here. > mpg </ Place here. >	extension
< Place here. > video/mpeg </ Place here. >	mime
</ Place here. >	content-type
	suffix
	mime-type
Done	mime-mapping

Answer: Check ExamWorx eEngine, Download from Member Center

www.ExamWorx.com Q: 43 Which three web application deployment descriptor elements allow web components to gain references to resources or EJB components? (Choose three.)

- A. ejb-ref
- B. jdbc-ref
- C. servlet-ref
- D. resource-ref
- E. javamail-ref
- F. ejb-remote-ref
- G. resource-env-ref

Answer: A, D, G

www.ExamWorx.com Q: 44 After a merger with another small business, your company has inherited a legacy WAR file but the original source files were lost. After reading the documentation of that web application, you discover that the WAR file contains a useful tag library that you want to reuse in your own webapp packaged as a WAR file.

What do you need to do to reuse this tag library?

- A. Simply rename the legacy WAR file as a JAR file and place it in your webapp's library directory.
- B. Unpack the legacy WAR file, move the TLD file to the META-INF directory, repackage the whole thing as a JAR file, and place that JAR file in your webapp's library directory.
- C. Unpack the legacy WAR file, move the TLD file to the META-INF directory, move the class files to the top-level directory, repackage the whole thing as a JAR file, and place that JAR file in your webapp's library directory.
- D. Unpack the legacy WAR file, move the TLD file to the META-INF directory, move the class files to the top-level directory, repackage the WAR, and place that WAR file in your webapp's WEB-INF directory.

Answer: C

www.ExamWorx.com Q: 45 Which two actions protect a resource file from direct HTTP access within a web application? (Choose two.)

- A. placing it in the /secure directory
- B. placing it in the /WEB-INF directory
- C. placing it in the /META-INF/secure directory
- D. creating a <web-resource> element within the deployment descriptor
- E. creating a <secure-resource> element within the deployment descriptor

Answer: B, C

www.ExamWorx.com Q: 46 Given that www.example.com/SCWCDtestApp is a validly deployed Java EE web application and that all of the JSP files specified in the requests below exist in the locations specified. Which two requests, issued from a browser, will return an HTTP 404 error? (Choose two.)

- A. http://www.example.com/SCWCDtestApp/test.jsp
- B. http://www.example.com/SCWCDtestApp/WEB-INF/test.jsp
- C. http://www.example.com/SCWCDtestApp/WEB-WAR/test.jsp
- D. http://www.example.com/SCWCDtestApp/Customer/test.jsp
- E. http://www.example.com/SCWCDtestApp/META-INF/test.jsp
- F. http://www.example.com/SCWCDtestApp/Customer/Update/test.jsp

Answer: B, E

www.ExamWorx.com Q: 47 Which two about WAR files are true? (Choose two.)

- A. WAR files must be located in the web application library directory.
- B. WAR files must contain the web application deployment descriptor.
- C. WAR files must be created by using archive tools designed specifically for that purpose.
- D. The web container must serve the content of any META-INF directory located in a WAR file.
- E. The web container must allow access to resources in JARs in the web application library directory.

Answer: B, E

www.ExamWorx.com Q: 48 Given this fragment from a Java EE deployment descriptor:

- 124. <welcome-file>beta.html</welcome-file>
- 125. <welcome-file>alpha.html</welcome-file>

And this request from a browser:

<http://www.sun.com/SCWCDtestApp/register>

Which statement is correct, when the container receives this request?

- A. This deployment descriptor is NOT valid.
- B. The container first looks in the register directory for beta.html.
- C. The container first looks in the register directory for alpha.html.
- D. The container first looks for a servlet mapping in the deployment descriptor.

Answer: D

www.ExamWorx.com Q: 49 Which EL expression evaluates to the request URI?

- A. \${requestURI}
- B. \${request.URI}
- C. \${request.getURI}
- D. \${request.requestURI}
- E. \${requestScope.requestURI}
- F. \${pageContext.request.requestURI}
- G. \${requestScope.request.requestURI}

Answer: F

www.ExamWorx.com Q: 50 Given:

1. <% int[] nums = {42,420,4200};
2. request.setAttribute("foo", nums); %>
3. \${5 + 3 lt 6}
4. \${requestScope['foo'][0] ne 10 div 0}
5. \${10 div 0}

What is the result?

- A. true true
- B. false true
- C. false true 0
- D. true true Infinity
- E. false true Infinity
- F. An exception is thrown.
- G. Compilation or translation fails.

Answer: E

www.ExamWorx.com Q: 51 You have created a web application that you license to real estate brokers. The webapp is highly customizable including the email address of the broker, which is placed on the footer of each page. This is configured as a context parameter in the deployment descriptor:

10. <context-param>
11. <param-name>footerEmail</param-name>
12. <param-value>joe@estates-r-us.biz</param-value>

13. </context-param>

Which EL code snippet will insert this context parameter into the footer?

- A. Contact me
- B. Contact me
- C. Contact me
- D. Contact me
- E. Contact me

Answer: C

www.ExamWorx.com Q: 52 Given an EL function foo, in namespace func, that requires a long as a parameter and returns a Map, which two are valid invocations of function foo? (Choose two.)

- A. \${func(1)}
- B. \${foo:func(4)}
- C. \${func:foo(2)}
- D. \${foo(5):func}
- E. \${func:foo("easy")}
- F. \${func:foo("3").name}

Answer: C, F

www.ExamWorx.com Q: 53 Click the Exhibit button.

The Appliance class is a Singleton that loads a set of properties into a Map from an external data source. Assume:

An instance of the Appliance class exists in the application-scoped attribute, appl
The appliance object includes the name property that maps to the value Cobia
The request-scoped attribute, prop, has the value name.

Which two EL code snippets will display the string Cobia? (Choose two.)

```
1. package com.example;
2. import java.util.*;
3. public class Appliance {
4.     private Map<String, String> props;
5.     public Appliance() {
6.         this.props = new
HashMap<String, String>();
7.         initialize();
8.     }
9.     public Map<String, String>
getProperties() {
10.        return this.props;
11.    }
12.    private void initialize() {
13.        // code to load appliance properties
14.    }
15. }
```

- A. \${appl.properties.name}
- B. \${appl.properties.prop}
- C. \${appl.properties[prop]}
- D. \${appl.properties[name]}
- E. \${appl.getProperties().get(prop)}
- F. \${appl.getProperties().get('name')}

Answer: A, C

www.ExamWorx.com Q: 54 Squeaky Beans Inc. hired an outside consultant to develop their web application. To finish the job quickly, the consultant created several dozen JSP pages that directly communicate with the database. The Squeaky business team has since purchased a set of business objects to model their system, and the Squeaky developer charged with maintaining the web application must now refactor all the JSPs to work with the new system. Which pattern can the developer use to solve this problem?

- A. Transfer Object
- B. Service Locator
- C. Intercepting Filter
- D. Business Delegate

Answer: D

www.ExamWorx.com Q: 55 A developer is designing a web application that must verify for each request:

- The originating request is from a trusted network.**
- The client has a valid session.**
- The client has been authenticated.**

Which design pattern provides a solution in this situation?

- A. Transfer Object
- B. Session Facade
- C. Intercepting Filter
- D. Template Method
- E. Model-View-Controller

Answer: C

www.ExamWorx.com Q: 56 The Squeaky Bean company has decided to port their web application to a new J2EE 1.4 container. While reviewing the application, a developer realizes that in multiple places within the current application, nearly duplicate code exists that finds enterprise beans. Which pattern should be used to eliminate this duplicate code?

- A. Transfer Object
- B. Front Controller
- C. Service Locator
- D. Intercepting Filter
- E. Business Delegate
- F. Model-View-Controller

Answer: C

www.ExamWorx.com Q: 57 Which two are characteristics of the Transfer Object design pattern? (Choose two.)

- A. It reduces network traffic by collapsing multiple remote requests into one.
- B. It increases the complexity of the remote interface by removing coarse-grained methods.
- C. It increases the complexity of the design due to remote synchronization and version control issues.
- D. It increases network performance introducing multiple fine-grained remote requests which return very small amounts of data.

Answer: A, C

www.ExamWorx.com Q: 58 A developer has created a special servlet that is responsible for generating XML content that is sent to a data warehousing subsystem. This subsystem uses HTTP to request these large data files, which are compressed by the servlet to save internal network bandwidth. The developer has received a request from management to create several more of these data warehousing servlets. The developer is about to copy and paste the compression code into each new servlet. Which design pattern can consolidate this compression code to be used by all of the data warehousing servlets?

- A. Facade
- B. View Helper
- C. Transfer Object
- D. Intercepting Filter
- E. Composite Facade

Answer: D

www.ExamWorx.com Q: 59 Which two are characteristics of the Service Locator pattern? (Choose two.)

- A. It encapsulates component lookup procedures.
- B. It increases source code duplication and decreases reuse.
- C. It improves client performance by caching context and factory objects.
- D. It degrades network performance due to increased access to distributed lookup services.

Answer: A, C

www.ExamWorx.com Q: 60 Click the Task button.

Given a servlet mapped to /control, place the correct URI segment returned as a String on the corresponding HttpServletRequest method call for the URI: /myapp/control/processorder.

Drag and Drop

Given a servlet mapped to /control, place the correct URI segment returned as a String on the corresponding HttpServletRequest method call for the URI: /myapp/control/processorder.

HttpServletRequest Method	URI Segment
getServletPath	/myapp
getPathInfo	/control
getContext	/processorder

Done

Answer: Check ExamWorx eEngine, Download from Member Center

www.ExamWorx.com Q: 61 You are creating a web form with this HTML:

```
11. <form action="sendOrder.jsp">
12.   <input type="text" name="creditCard">
13.   <input type="text" name="expirationDate">
14.   <input type="submit">
15. </form>
```

Which HTTP method is used when sending this request from the browser?

- A. GET
- B. PUT
- C. POST

- D. SEND
- E. FORM

Answer: A

www.ExamWorx.com Q: 62 Given an HttpSession session, a ServletRequest request, and a ServletContext context, which retrieves a URL to /WEB-INF/myconfig.xml within a web application?

- A. session.getResource("/WEB-INF/myconfig.xml")
- B. request.getResource("/WEB-INF/myconfig.xml")
- C. context.getResource("/WEB-INF/myconfig.xml")
- D. getClass().getResource("/WEB-INF/myconfig.xml")

Answer: C

www.ExamWorx.com Q: 63 Your company has a corporate policy that prohibits storing a customer's credit card number in any corporate database. However, users have complained that they do NOT want to re-enter their credit card number for each transaction. Your management has decided to use client-side cookies to record the user's credit card number for 120 days. Furthermore, they also want to protect this information during transit from the web browser to the web container; so the cookie must only be transmitted over HTTPS.

Which code snippet creates the "creditCard" cookie and adds it to the out going response to be stored on the user's web browser?

- A. 10. Cookie c = new Cookie("creditCard", usersCard);
11. c.setSecure(true);
12. c.setAge(10368000);
13. response.addCookie(c);
- B. 10. Cookie c = new Cookie("creditCard", usersCard);
11. c.setHttps(true);
12. c.setMaxAge(10368000);
13. response.setCookie(c);
- C. 10. Cookie c = new Cookie("creditCard", usersCard);
11. c.setSecure(true);
12. c.setMaxAge(10368000);
13. response.addCookie(c);
- D. 10. Cookie c = new Cookie("creditCard", usersCard);
11. c.setHttps(true);
12. c.setAge(10368000);
13. response.addCookie(c);
- E. 10. Cookie c = new Cookie("creditCard", usersCard);

```
11. c.setSecure(true);  
12. c.setAge(10368000);  
13. response.setCookie(c);
```

Answer: C

www.ExamWorx.com Q: 64 Given a header in an HTTP request:

X-Retries: 4

Which two retrieve the value of the header from a given HttpServletRequest request? (Choose two.)

- A. request.getHeader("X-Retries")
- B. request.getIntHeader("X-Retries")
- C. request.getRequestHeader("X-Retries")
- D. request.getHeaders("X-Retries").get(0)
- E. request.getRequestHeaders("X-Retries").get(0)

Answer: A, B

www.ExamWorx.com Q: 65 For a given ServletResponse response, which two retrieve an object for writing text data? (Choose two.)

- A. response.getWriter()
- B. response.getOutputStream()
- C. response.getOutputWriter()
- D. response.getWriter().getOutputStream()
- E. response.getWriter(Writer.OUTPUT_TEXT)

Answer: A, B

www.ExamWorx.com Q: 66 Which JSP standard action can be used to import content from a resource called foo.jsp?

- A. <jsp:import file='foo.jsp' />
- B. <jsp:import page='foo.jsp' />
- C. <jsp:include page='foo.jsp' />
- D. <jsp:include file='foo.jsp' />
- E. <jsp:import>foo.jsp</jsp:import>
- F. <jsp:include>foo.jsp</jsp:include>

Answer: C

www.ExamWorx.com Q: 67 Click the Task button.

A servlet context listener loads a list of com.example.Product objects from a database and stores that list into the catalog attribute of the ServletContext object.

Place code snippets to construct a jsp:useBean standard action to access this catalog.

Drag and Drop



A servlet context listener loads a list of com.example.Product objects from a database and stores that list into the catalog attribute of the ServletContext object.

Place code snippets to construct a jsp:useBean standard action to access this catalog.

The jsp:useBean Standard Action

<jsp:useBean

Place here.

Place here.

Place here.

/>

Code Snippets

id='product'

scope='application'

type='java.util.List'

id='catalog'

name='catalog'

scope='context'

scope='servletContext'

type='com.example.Product'

Done

Answer: Check ExamWorx eEngine, Download from Member Center

www.ExamWorx.com Q: 68 A session-scoped attribute is stored by a servlet, and then that servlet forwards to a JSP page. Which three `jsp:useBean` attributes must be used to access this attribute in the JSP page? (Choose three.)

- A. id
- B. name
- C. bean
- D. type
- E. scope
- F. beanName

Answer: A, D, E

www.ExamWorx.com Q: 69 You need to create a JavaBean object that is used only within the current JSP page. It must NOT be accessible to any other page including those that this page might import. Which JSP standard action can accomplish this goal?

- A. `<jsp:useBean id='pageBean' type='com.example.MyBean' />`
- B. `<jsp:useBean id='pageBean' class='com.example.MyBean' />`
- C. `<jsp:makeBean id='pageBean' type='com.example.MyBean' />`
- D. `<jsp:makeBean id='pageBean' class='com.example.MyBean' />`
- E. `<jsp:useBean name='pageBean' class='com.example.MyBean' />`
- F. `<jsp:makeBean name='pageBean' class='com.example.MyBean' />`

Answer: B

www.ExamWorx.com Q: 70 Given an `HttpServletRequest` request and `HttpServletResponse` response, which sets a cookie "username" with the value "joe" in a servlet?

- A. `request.addCookie("username", "joe")`
- B. `request.setCookie("username", "joe")`
- C. `response.addCookie("username", "joe")`
- D. `request.addHeader(new Cookie("username", "joe"))`
- E. `request.addCookie(new Cookie("username", "joe"))`
- F. `response.addCookie(new Cookie("username", "joe"))`
- G. `response.addHeader(new Cookie("username", "joe"))`

Answer: F

www.ExamWorx.com Q: 71 Your web page includes a Java SE v1.5 applet with the following declaration:

11. <object classid='clsid:CAFEEFAC-0015-0000-0000-ABCDEFFEDCBA'
12. width='200' height='200'>
13. <param name='code' value='Applet.class' />
14. </object>

Which HTTP method is used to retrieve the applet code?

- A. GET
- B. PUT
- C. POST
- D. RETRIEVE

Answer: A

www.ExamWorx.com Q: 72 You are creating a servlet that generates stock market graphs. You want to provide the web browser with precise information about the amount of data being sent in the response stream. Which two HttpServletResponse methods will you use to provide this information? (Choose two.)

- A. response.setLength(numberOfBytes);
- B. response.setContentLength(numberOfBytes);
- C. response.setHeader("Length", numberOfBytes);
- D. response.setIntHeader("Length", numberOfBytes);
- E. response.setHeader("Content-Length", numberOfBytes);
- F. response.setIntHeader("Content-Length", numberOfBytes);

Answer: B, F

www.ExamWorx.com Q: 73 You need to retrieve the username cookie from an HTTP request. If this cookie does NOT exist, then the c variable will be null. Which code snippet must be used to retrieve this cookie object?

- A. 10. Cookie c = request.getCookie("username");
- B. 10. Cookie c = null;

```

11. for ( Iterator i = request.getCookies();
12.     i.hasNext(); ) {
13.     Cookie o = (Cookie) i.next();
14.     if ( o.getName().equals("username") ) {
15.         c = o;
16.         break;
17.     }
18. }
C. 10. Cookie c = null;
11. for ( Enumeration e = request.getCookies();
12.     e.hasMoreElements(); ) {
13.     Cookie o = (Cookie) e.nextElement();
14.     if ( o.getName().equals("username") ) {
15.         c = o;
16.         break;
17.     }
18. }
D. 10. Cookie c = null;
11. Cookie[] cookies = request.getCookies();
12. for ( int i = 0; i < cookies.length; i++ ) {
13.     if ( cookies[i].getName().equals("username") ) {
14.         c = cookies[i];
15.         break;
16.     }
17. }

```

Answer: D

www.ExamWorx.com Q: 74 Given:

```

10. public void service(ServletRequest request,
11.                     ServletResponse response) {
12.     ServletInputStream sis =
13.     // insert code here
14. }

```

Which retrieves the binary input stream on line 13?

- A. request.getWriter();
- B. request.getReader();
- C. request.getInputStream();
- D. request.getResourceAsStream();
- E. request.getResourceAsStream(ServletRequest.REQUEST);

Answer: C

www.ExamWorx.com Q: 75 Click the Exhibit button.

As a maintenance feature, you have created this servlet to allow you to upload and remove files on your web server. Unfortunately, while testing this servlet, you try to upload a file using an HTTP request and on this servlet, the web container returns a 404 status.

What is wrong with this servlet?

```
1. package com.example;
2.
3. import javax.servlet.http.*;
4.
5. public class MyWebDAV extends HttpServlet
{
6.     private String resourceDirectory;
7.
8.     public MyWebDAV(String resDir) {
9.         this.resourceDirectory = resDir;
10.    }
11.    public void doPut(HttpServletRequest
req,
12.                      HttpServletResponse
resp) {
13.        // store file to resourceDirectory
14.        // (code not shown)
15.    }
16.    public void doDelete(HttpServletRequest
req,
17.                          HttpServletResponse resp) {
18.        // remove file from resourceDirectory
19.        // (code not shown)
20.    }
21.}
```

- A. HTTP does NOT support file upload operations.
- B. The servlet constructor must NOT have any parameters.
- C. The servlet needs a service method to dispatch the requests to the helper methods.

- D. The doPut and doDelete methods do NOT map to the proper HTTP methods.

Answer: B

www.ExamWorx.com Q: 76 You have built a web application with tight security. Several directories of your webapp are used for internal purposes and you have overridden the default servlet to send an HTTP 403 status code for any request that maps to one of these directories. During testing, the Quality Assurance director decided that they did NOT like seeing the bare response page generated by Firefox and Internet Explorer. The director recommended that the webapp should return a more user-friendly web page that has the same look-and-feel as the webapp plus links to the webapp's search engine. You have created this JSP page in the /WEB-INF/jsp/error403.jsp file. You do NOT want to alter the complex logic of the default servlet. How can you declare that the web container must send this JSP page whenever a 403 status is generated?

- A. <error-page>
<error-code>403</error-code>
<url>/WEB-INF/jsp/error403.jsp</url>
</error-page>
- B. <error-page>
<status-code>403</status-code>
<url>/WEB-INF/jsp/error403.jsp</url>
</error-page>
- C. <error-page>
<error-code>403</error-code>
<location>/WEB-INF/jsp/error403.jsp</location>
</error-page>
- D. <error-page>
<status-code>403</status-code>
<location>/WEB-INF/jsp/error403.jsp</location>
</error-page>

Answer: C

www.ExamWorx.com Q: 77 You want to create a valid directory structure for your Java EE web application, and your application uses tag files and a JAR file. Which three must be located directly in your WEB-INF directory (NOT in a subdirectory of WEB-INF)? (Choose three.)

- A. The JAR file
- B. A directory called lib
- C. A directory called tags
- D. A directory called TLDs
- E. A directory called classes

- F. A directory called META-INF

Answer: B, C, E

www.ExamWorx.com Q: 78 Given:

```
11. public class MyServlet extends HttpServlet {  
12.     public void service(HttpServletRequest request,  
13.                           HttpServletResponse response)  
14.         throws ServletException, IOException {  
15.     // insert code here  
16. }  
17. }
```

and this element in the web application's deployment descriptor:

```
<error-page>  
<error-code>302</error-code>  
<location>/html/error.html</location>  
</error-page>
```

Which, inserted at line 15, causes the container to redirect control to the error.html resource?

- A. response.setError(302);
- B. response.sendError(302);
- C. response.setStatus(302);
- D. response.sendRedirect(302);
- E. response.sendErrorRedirect(302);

Answer: B

www.ExamWorx.com Q: 79 Which element of the web application deployment descriptor defines the servlet class associated with a servlet instance?

- A. <class>
- B. <webapp>
- C. <servlet>
- D. <codebase>
- E. <servlet-class>
- F. <servlet-mapping>

Answer: E

www.ExamWorx.com Q: 80 Within the web application deployment descriptor, which defines a valid JNDI environment entry?

- A. <env-entry>
<env-entry-type>java.lang.Boolean</env-entry-type>
<env-entry-value>true</env-entry-value>
</env-entry>
- B. <env-entry>
<env-entry-name>param/MyExampleString</env-entry-name>
<env-entry-value>This is an Example</env-entry-value>
</env-entry>
- C. <env-entry>
<env-entry-name>param/MyExampleString</env-entry-name>
<env-entry-type>int</env-entry-type>
<env-entry-value>10</env-entry-value>
</env-entry>
- D. <env-entry>
<env-entry-name>param/MyExampleString</env-entry-name>
<env-entry-type>java.lang.String</env-entry-type>
<env-entry-value>This is an Example</env-entry-value>
</env-entry>

Answer: D

www.ExamWorx.com Q: 81 Which three are described in the standard web application deployment descriptor? (Choose three.)

- A. session configuration
- B. MIME type mappings
- C. context root for the application
- D. servlet instance pool configuration
- E. web container default port bindings
- F. ServletContext initialization parameters

Answer: A, B, F

www.ExamWorx.com Q: 82 Which two are true regarding a web application class loader? (Choose two.)

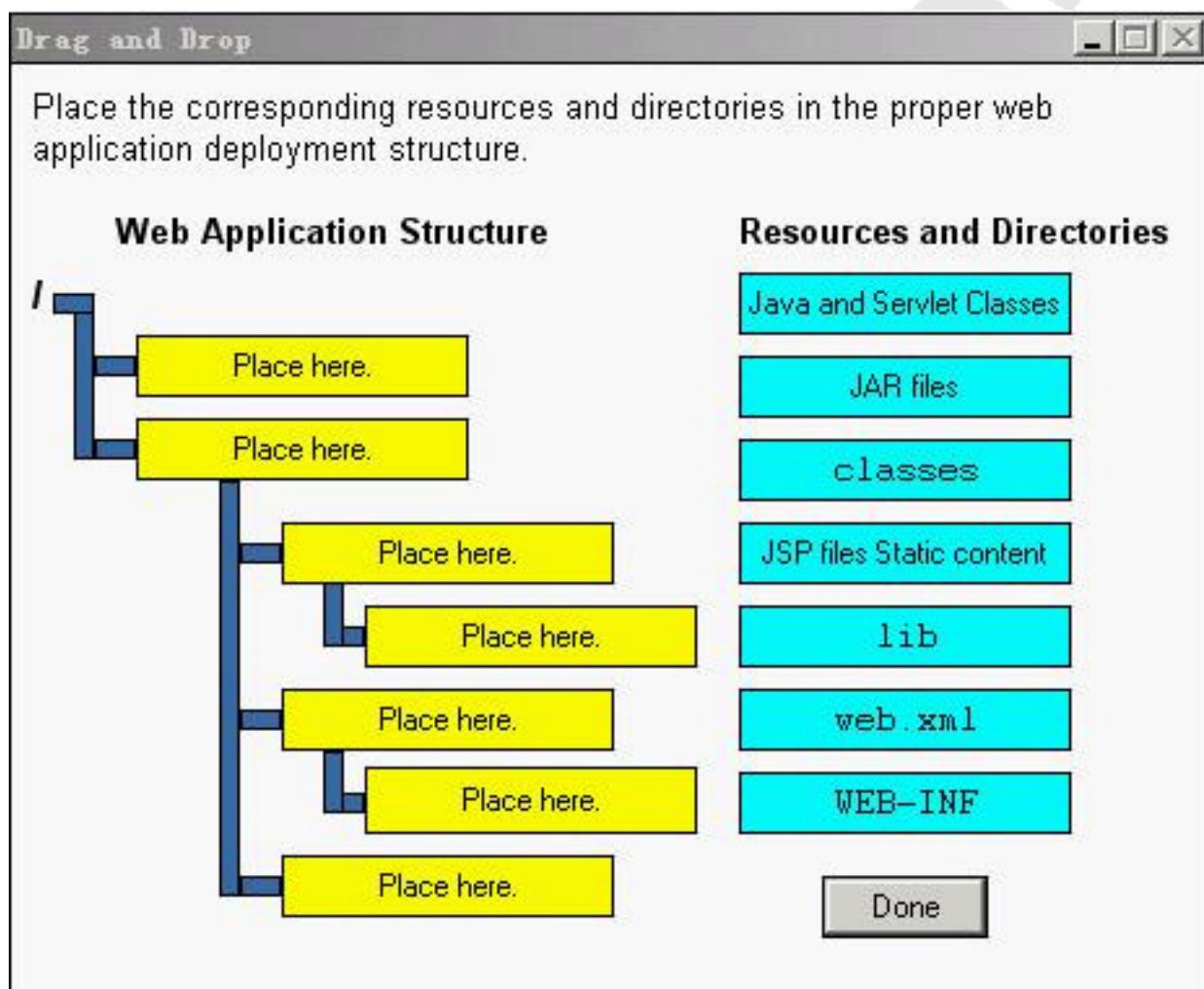
- A. A web application may override the web container's implementation classes.

- B. A web application running in a J2EE product may override classes in the javax.* namespace.
- C. A web application class loader may NOT override any classes in the java.* and javax.* namespaces.
- D. Resources in the WAR class directory or in any of the JAR files within the library directory may be accessed using the J2SE semantics of getResource.
- E. Resources in the WAR class directory or in any of the JAR files within the library directory CANNOT be accessed using the J2SE semantics of getResource.

Answer: C, D

www.ExamWorx.com Q: 83 Click the Task button.

Place the corresponding resources and directories in the proper web application deployment structure.



Answer: Check ExamWorx eEngine, Download from Member Center

www.ExamWorx.com Q: 84 You are building JSP pages that have a set of menus that are visible based on a user's security role. These menus are hand-crafted by your web design team; for example, the SalesManager role has a menu in the file /WEB-INF/html/sales-mgr-menu.html. Which JSP code snippet should be used to make this menu visible to the user?

- A. <% if (request.isUserInRole("SalesManager")) { %>
<%@ include file='/WEB-INF/html/sales-mgr-menu.html' %>
<% } %>
- B. <jsp:if test='request.isUserInRole("SalesManager")'%>
<%@ include file='/WEB-INF/html/sales-mgr-menu.html' %>
</jsp:if>
- C. <% if (request.isUserInRole("SalesManager")) { %>
<jsp:include file='/WEB-INF/html/sales-mgr-menu.html' />
<% } %>
- D. <jsp:if test='request.isUserInRole("SalesManager")'%>
<jsp:include file='/WEB-INF/html/sales-mgr-menu.html' />
</jsp:if>

Answer: A

www.ExamWorx.com Q: 85 For debugging purposes, you need to record how many times a given JSP is invoked before the user's session has been created. The JSP's destroy method stores this information to a database. Which JSP code snippet keeps track of this count for the lifetime of the JSP page?

- A. <%! int count = 0; %>
<% if (request.getSession(false) == null) count++; %>
- B. <%@ int count = 0; %>
<% if (request.getSession(false) == null) count++; %>
- C. <% int count = 0;
if (request.getSession(false) == null) count++; %>
- D. <%@ int count = 0;
if (request.getSession(false) == null) count++; %>
- E. <%! int count = 0;
if (request.getSession(false) == null) count++; %>

Answer: A

www.ExamWorx.com Q: 86 For manageability purposes, you have been told to add a "count" instance variable to a critical JSP Document so that a JMX MBean can track how frequent this JSP is being invoked. Which JSP code snippet must you use to declare this instance variable in the JSP Document?

- A. <jsp:declaration>
int count = 0;
<jsp:declaration>
- B. <%! int count = 0; %>
- C. <jsp:declaration.instance>
int count = 0;
<jsp:declaration.instance>
- D. <jsp:scriptlet.declaration>
int count = 0;
<jsp:scriptlet.declaration>

Answer: A

www.ExamWorx.com Q: 87 In a JSP-centric web application, you need to create a catalog browsing JSP page. The catalog is stored as a List object in the catalog attribute of the webapp's ServletContext object. Which scriptlet code snippet gives you access to the catalog object?

- A. <% List catalog = config.getAttribute("catalog"); %>
- B. <% List catalog = context.getAttribute("catalog"); %>
- C. <% List catalog = application.getAttribute("catalog"); %>
- D. <% List catalog = servletContext.getAttribute("catalog"); %>

Answer: C

www.ExamWorx.com Q: 88 Given the element from the web application deployment descriptor:

```
<jsp-property-group>
  <url-pattern>/main/page1.jsp</url-pattern>
  <scripting-invalid>true</scripting-invalid>
</jsp-property-group>
```

and given that /main/page1.jsp contains:

```
<% int i = 12; %>
<b><%= i %></b>
```

What is the result?

- A.
- B. 12

- C. The JSP fails to execute.
- D. <% int i = 12 %>
<%= i %>

Answer: C

www.ExamWorx.com Q: 89 You are creating a new JSP page and you need to execute some code that acts when the page is first executed, but only once. Which three are possible mechanisms for performing this initialization code? (Choose three.)

- A. In the init method.
- B. In the jspInit method.
- C. In the constructor of the JSP's Java code.
- D. In a JSP declaration, which includes an initializer block.
- E. In a JSP declaration, which includes a static initializer block.

Answer: B, D, E

www.ExamWorx.com Q: 90 You are writing a JSP that includes scriptlet code to declare a List variable and initializes that variable to an ArrayList object. Which two JSP code snippets can you use to import these list types? (Choose two.)

- A. <%! import java.util.*; %>
- B. <%! import java.util.List;
import java.util.ArrayList; %>
- C. <%@ page import='java.util.List'
import='java.util.ArrayList' %>
- D. <%@ import types='java.util.List'
types='java.util.ArrayList' %>
- E. <%@ page import='java.util.List,java.util.ArrayList' %>
- F. <%@ import types='java.util.List,java.util.ArrayList' %>

Answer: C, E

www.ExamWorx.com Q: 91 Assume the custom tag my:errorProne always throws a java.lang.RuntimeException with the message "File not found."

An error page has been configured for this JSP page.

Which option prevents the exception thrown by my:errorProne from invoking the error page mechanism, and outputs the message "File not found" in the response?

A. <c:try catch="ex">
<my:errorProne />
</c:try>
 \${ex.message}
B. <c:catch var="ex">
<my:errorProne />
</c:catch>
 \${ex.message}
C. <c:try>
<my:errorProne />
</c:try>
<c:catch var="ex" />
 \${ex.message}
D. <c:try>
<my:errorProne />
<c:catch var="ex" />
 \${ex.message}
</c:try>
E. <my:errorProne>
<c:catch var="ex">
 \${ex.message}
</c:catch>
</my:errorProne>

Answer: B

www.ExamWorx.com Q: 92 A JSP page contains a taglib directive whose uri attribute has the value dbtags. Which XML element within the web application deployment descriptor defines the associated TLD?

A. <tld>
<uri>dbtags</uri>
<location>/WEB-INF/tlds/dbtags.tld</location>
</tld>
B. <taglib>
<uri>dbtags</uri>
<location>/WEB-INF/tlds/dbtags.tld</location>
</taglib>
C. <tld>
<tld-uri>dbtags</tld-uri>
<tld-location>/WEB-INF/tlds/dbtags.tld</tld-location>
</tld>

D. <taglib>
<taglib-uri>dbtags</taglib-uri>
<taglib-location>
/WEB-INF/tlds/dbtags.tld
</taglib-location>
</taglib>

Answer: D

www.ExamWorx.com Q: 93 Assume that a news tag library contains the tags lookup and item:

lookup Retrieves the latest news headlines and executes the tag body once for each headline. Exposes a NESTED page-scoped attribute called headline of type com.example.Headline containing details for that headline.

item Outputs the HTML for a single news headline. Accepts an attribute info of type com.example.Headline containing details for the headline to be rendered. Which snippet of JSP code returns the latest news headlines in an HTML table, one per row?

A. <table>
<tr>
<td>
<news:lookup />
<news:item info="\${headline}" />
</td>
</tr>
</table>

B. <news:lookup />
<table>
<tr>
<td><news:item info="\${headline}" /></td>
</tr>
</table>

C. <table>
<news:lookup>
<tr>
<td><news:item info="\${headline}" /></td>
</tr>
</news:lookup>
</table>

D. <table>
<tr>
<news:lookup>
<td><news:item info="\${headline}" /></td>

```
</news:lookup>
</tr>
</table>
```

Answer: C

www.ExamWorx.com Q: 94 Click the Exhibit button.

Assuming the tag library in the exhibit is imported with the prefix stock, which custom tag invocation outputs the contents of the variable exposed by the quote tag?

```
1. <?xml version="1.0" encoding="UTF-8" ?>
2.
3. <taglib
4.   xmlns="http://java.sun.com/xml/ns/j2ee"
5.   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6.   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-jsptaglibrary_2_0.xsd"
7.   version="2.0">
8.   <tlib-version>1.0</tlib-version>
9.   <short-name>stock</short-name>
10.  <uri>http://example.com/tld/stock</uri>
11.  <tag>
12.    <name>quote</name>
13.    <tag-class>com.example.QuoteTag</tag-class>
14.    <body-content>empty</body-content>
15.    <variable>
16.      <name-from-attribute>var</name-from-attribute>
17.      <scope>AT_BEGIN</scope>
18.    </variable>
19.    <attribute>
20.      <name>symbol</name>
21.      <required>true</required>
22.      <rteval>true</rteval>
23.    <attribute>
24.      <name>var</name>
25.      <required>true</required>
26.      <rteval>false</rteval>
27.    </attribute>
28.  </tag>
29. </taglib>
```

- A. <stock:quote symbol="SUNW" />
 \${var}
- B. \${var}
<stock:quote symbol="SUNW" />
- C. <stock:quote symbol="SUNW">
 \${var}
</stock:quote>
- D. <stock:quote symbol="SUNW" var="quote" />

```
 ${quote}
E. <stock:quote symbol="SUNW" var="quote">
<%= quote %>
</stock:quote>
```

Answer: D

www.ExamWorx.com Q: 95 Which two are true about the JSTL core iteration custom tags? (Choose two.)

- A. It may iterate over arrays, collections, maps, and strings.
- B. The body of the tag may contain EL code, but not scripting code.
- C. When looping over collections, a loop status object may be used in the tag body.
- D. It may iterate over a map, but only the key of the mapping may be used in the tag body.
- E. When looping over integers (for example begin='1' end='10'), a loop status object may not be used in the tag body.

Answer: A, C

www.ExamWorx.com Q: 96 Assume a JavaBean com.example.GradedTestBean exists and has two attributes. The attribute name is of type java.lang.String and the attribute score is of type java.lang.Integer.

An array of com.example.GradedTestBean objects is exposed to the page in a request-scoped attribute called results. Additionally, an empty java.util.HashMap called resultMap is placed in the page scope.

A JSP page needs to add the first entry in results to resultMap, storing the name attribute of the bean as the key and the score attribute of the bean as the value.

Which code snippet of JSTL code satisfies this requirement?

- A. \${resultMap[results[0].name] = results[0].score}
- B. <c:set var="\${resultMap}" key="\${results[0].name}" value="\${results[0].score}" />
- C. <c:set var="resultMap" property="\${results[0].name}">
\${results[0].value}
</c:set>
- D. <c:set var="resultMap" property="\${results[0].name}" value="\${results[0].score}" />
- E. <c:set target="\${resultMap}" property="\${results[0].name}" value="\${results[0].score}" />

Answer: E

www.ExamWorx.com Q: 97 You are creating a JSP page to display a collection of data. This data can be displayed in several different ways so the architect on your project decided to create a generic servlet that generates a comma-delimited string so that various pages can render the data in different ways. This servlet takes on request parameter: objectID. Assume that this servlet is mapped to the URL pattern: /WEB-INF/data.

In the JSP you are creating, you need to split this string into its elements separated by commas and generate an HTML list from the data.

Which JSTL code snippet will accomplish this goal?

- A. <c:import varReader='dataString' url='/WEB-INF/data'><c:param name='objectID' value='\${currentOID}' /></c:import><c:forTokens items='\${dataString.split(",")}' var='item'>\${item}</c:forTokens>
- B. <c:import varReader='dataString' url='/WEB-INF/data'><c:param name='objectID' value='\${currentOID}' /></c:import><c:forTokens items='\${dataString}' delims=',' var='item'>\${item}</c:forTokens>
- C. <c:import var='dataString' url='/WEB-INF/data'><c:param name='objectID' value='\${currentOID}' /></c:import><c:forTokens items='\${dataString.split(",")}' var='item'>\${item}</c:forTokens>
- D. <c:import var='dataString' url='/WEB-INF/data'><c:param name='objectID' value='\${currentOID}' /></c:import><c:forTokens items='\${dataString}' delims=',' var='item'>\${item}</c:forTokens>

Answer: D

www.ExamWorx.com Q: 98 A web application contains a tag file called beta.tag in /WEB-INF/tags/alpha. A JSP page called sort.jsp exists in the web application and contains only this JSP code:

1. <%@ taglib prefix="x"
2. tagdir="/WEB-INF/tags/alpha" %>
3. <x:beta />

The sort.jsp page is requested.

Which two are true? (Choose two.)

- A. Tag files can only be accessed using a tagdir attribute.
- B. The sort.jsp page translates successfully and invokes the tag defined by beta.tag.
- C. The sort.jsp page produces a translation error because a taglib directive must always have a uri attribute.
- D. Tag files can only be placed in /WEB-INF/tags, and NOT in any subdirectories of /WEB-INF/tags.
- E. The tagdir attribute in line 2 can be replaced by a uri attribute if a TLD referring to beta.tag is created and added to the web application.
- F. The sort.jsp page produces a translation error because the tagdir attribute on lines 1-2 specifies a directory other than /WEB-INF/tags, which is illegal.

Answer: B, E

www.ExamWorx.com Q: 99 What is the purpose of session management?

- A. To manage the user's login and logout activities.
- B. To store information on the client-side between HTTP requests.
- C. To store information on the server-side between HTTP requests.
- D. To tell the web container to keep the HTTP connection alive so it can make subsequent requests without the delay of making the TCP connection.

Answer: C

www.ExamWorx.com Q: 100 The Squeaky Beans Inc. shopping application was initially developed for a non-distributed environment. The company recently purchased the Acme Application Server, which supports distributed HttpSession objects. When deploying the application to the server, the deployer marks it as distributable in the web application descriptor to take advantage of this feature.

Given this scenario, which two must be true? (Choose two.)

- A. The J2EE web container must support migration of objects that implement Serializable.
- B. The J2EE web container must use the native JVM Serialization mechanism for distributing HttpSession objects.
- C. As per the specification, the J2EE web container ensures that distributed HttpSession objects will be stored in a database.
- D. Storing references to Enterprise JavaBeans components in the HttpSession object might NOT be supported by J2EE web containers.

Answer: A, D

www.ExamWorx.com Q: 101 In your web application, you need to execute a block of code whenever the session object is first created. Which design will accomplish this goal?

- A. Create an HttpSessionListener class and implement the sessionInitialized method with that block of code.
- B. Create an HttpSessionActivationListener class and implement the sessionCreated method with that block of code.
- C. Create a Filter class, call the getSession(false) method, and if the result was null, then execute that block of code.
- D. Create an HttpSessionListener class and implement the sessionCreated method with that block of code.
- E. Create a Filter class, call the getSession(true) method, and if the result was NOT null, then execute that block of code.

Answer: D

www.ExamWorx.com Q: 102 Which interface must a class implement so that instances of the class are notified after any object is added to a session?

- A. javax.servlet.http.HttpSessionListener
- B. javax.servlet.http.HttpSessionValueListener
- C. javax.servlet.http.HttpSessionBindingListener
- D. javax.servlet.http.HttpSessionAttributeListener

Answer: D

www.ExamWorx.com Q: 103 Which method must be used to encode a URL passed as an argument to `HttpServletResponse.sendRedirect` when using URL rewriting for session tracking?

- A. `ServletResponse.encodeURL`
- B. `HttpServletResponse.encodeURL`
- C. `ServletResponse.encodeRedirectURL`
- D. `HttpServletResponse.encodeRedirectURL`

Answer: D

www.ExamWorx.com Q: 104 Users of your web application have requested that they should be able to set the duration of their sessions. So for example, one user might want a webapp to stay connected for an hour rather than the webapp's default of fifteen minutes; another user might want to stay connected for a whole day.

Furthermore, you have a special login servlet that performs user authentication and retrieves the User object from the database. You want to augment this code to set up the user's specified session duration.

Which code snippet in the login servlet will accomplish this goal?

- A. `User user = // retrieve the User object from the database
session.setDurationInterval(user.getSessionDuration());`
- B. `User user = // retrieve the User object from the database
session.setMaxDuration(user.getSessionDuration());`
- C. `User user = // retrieve the User object from the database
session.setInactiveInterval(user.getSessionDuration());`
- D. `User user = // retrieve the User object from the database
session.setDuration(user.getSessionDuration());`
- E. `User user = // retrieve the User object from the database
session.setMaxInactiveInterval(user.getSessionDuration());`
- F. `User user = // retrieve the User object from the database
session.setMaxDurationInterval(user.getSessionDuration());`

Answer: E

www.ExamWorx.com Q: 105 Which two classes or interfaces provide a `getSession` method? (Choose two.)

- A. `javax.servlet.http.HttpServletRequest`

- B. javax.servlet.http.HttpSessionContext
- C. javax.servlet.http.HttpServletResponse
- D. javax.servlet.http.HttpSessionBindingEvent
- E. javax.servlet.http.HttpSessionAttributeEvent

Answer: A, D

www.ExamWorx.com Q: 106 Given the security constraint in a DD:

- 101. <security-constraint>
- 102. <web-resource-collection>
- 103. <web-resource-name>Foo</web-resource-name>
- 104. <url-pattern>/Bar/Baz/*</url-pattern>
- 105. <http-method>POST</http-method>
- 106. </web-resource-collection>
- 107. <auth-constraint>
- 108. <role-name>DEVELOPER</role-name>
- 109. </auth-constraint>
- 110. </security-constraint>

**And given that "MANAGER" is a valid role-name, which four are true for this security constraint?
(Choose four.)**

- A. MANAGER can do a GET on resources in the /Bar/Baz directory.
- B. MANAGER can do a POST on any resource in the /Bar/Baz directory.
- C. MANAGER can do a TRACE on any resource in the /Bar/Baz directory.
- D. DEVELOPER can do a GET on resources in the /Bar/Baz directory.
- E. DEVELOPER can do only a POST on resources in the /Bar/Baz directory.
- F. DEVELOPER can do a TRACE on any resource in the /Bar/Baz directory.

Answer: A, C, D, F

www.ExamWorx.com Q: 107 Which activity supports the data integrity requirements of an application?

- A. using HTTPS as a protocol
- B. using an LDAP security realm
- C. using HTTP Basic authentication
- D. using forms-based authentication

Answer: A

www.ExamWorx.com Q: 108 Which mechanism requires the client to provide its public key certificate?

- A. HTTP Basic Authentication
- B. Form Based Authentication
- C. HTTP Digest Authentication
- D. HTTPS Client Authentication

Answer: D

www.ExamWorx.com Q: 109 Given the two security constraints in a deployment descriptor:

- 101. <security-constraint>
- 102. <!--a correct url-pattern and http-method goes here-->
- 103. <auth-constraint><role-name>SALES</role-name></auth-constraint>
- 104. <role-name>SALES</role-name>
- 105. </auth-constraint>
- 106. </security-constraint>
- 107. <security-constraint>
- 108. <!--a correct url-pattern and http-method goes here-->
- 109. <!-- Insert an auth-constraint here -->
- 110. </security-constraint>

If the two security constraints have the same url-pattern and http-method, which two, inserted independently at line 109, will allow users with role names of either SALES or MARKETING to access this resource? (Choose two.)

- A. <auth-constraint/>
- B. <auth-constraint><role-name>*</role-name></auth-constraint>
- C. <auth-constraint><role-name>ANY</role-name></auth-constraint>
- D. <auth-constraint><role-name>MARKETING</role-name></auth-constraint>

Answer: B, D

www.ExamWorx.com Q: 110 Given this fragment in a servlet:

```
23. if(req.isUserInRole("Admin")) {  
24. // do stuff  
25. }
```

And the following fragment from the related Java EE deployment descriptor:

```
812. <security-role-ref>  
813.   <role-name>Admin</role-name>  
814.   <role-link>Administrator</role-link>  
815. </security-role-ref>  
  
900. <security-role>  
901.   <role-name>Admin</role-name>  
902.   <role-name>Administrator</role-name>  
903. </security-role>
```

What is the result?

- A. Line 24 can never be reached.
- B. The deployment descriptor is NOT valid.
- C. If line 24 executes, the user's role will be Admin.
- D. If line 24 executes, the user's role will be Administrator.
- E. If line 24 executes the user's role will NOT be predictable.

Answer: D

www.ExamWorx.com Q: 111 Which two are true about authentication? (Choose two.)

- A. Form-based logins should NOT be used with HTTPS.
- B. When using Basic Authentication the target server is NOT authenticated.
- C. J2EE compliant web containers are NOT required to support the HTTPS protocol.
- D. Web containers are required to support unauthenticated access to unprotected web resources.
- E. Form-based logins should NOT be used when sessions are maintained by cookies or SSL session information.

Answer: B, D

www.ExamWorx.com Q: 112 Given:

```
11. <%  
12. request.setAttribute("vals", new String[]{"1","2","3","4"});  
13. request.setAttribute("index", "2");
```

14. %>

15. <%-- insert code here --%>

Which three EL expressions, inserted at line 15, are valid and evaluate to "3"? (Choose three.)

- A. \${vals.2}
- B. \${vals["2"]}
- C. \${vals.index}
- D. \${vals[index]}
- E. \${vals}[index]
- F. \${vals.(vals.index)}
- G. \${vals[vals[index-1]]}

Answer: B, D, G

www.ExamWorx.com Q: 113 Given:

```
11. <% java.util.Map map = new java.util.HashMap();  
12. request.setAttribute("map", map);  
13. map.put("a", "true");  
14. map.put("b", "false");  
15. map.put("c", "42"); %>
```

Which three EL expressions are valid and evaluate to true? (Choose three.)

- A. \${not map.c}
- B. \${map.d or map.a}
- C. \${map.a and map.d}
- D. \${map.false or map.true}
- E. \${map.a and map.b or map.a}
- F. \${map['true'] or map['false']}

Answer: A, B, E

www.ExamWorx.com Q: 114 Given:

<http://com.example/myServlet.jsp?num=one&num=two&num=three>

Which two produce the output "one, two and three"? (Choose two.)

- A. \${param.num[0],[1] and [2]}

- B. \${paramValues[0],[1] and [2]}
- C. \${param.num[0]}, \${param.num[1]} and \${param.num[2]}
- D. \${paramValues.num[0]}, \${paramValues.num[1]} and \${paramValues.num[2]}
- E. \${paramValues["num"][0]}, \${paramValues["num"][1]} and \${paramValues["num"][2]}
- F. \${parameterValues.num[0]}, \${parameterValues.num[1]} and \${parameterValues.num[2]}
- G. \${parameterValues["num"]["0"]}, \${parameterValues["num"]["1"]} and \${parameterValues["num"]["2"]}

Answer: D, E

www.ExamWorx.com Q: 115 Given a web application in which the cookie userName is expected to contain the name of the user. Which EL expression evaluates to that user name?

- A. \${userName}
- B. \${cookie.userName}
- C. \${cookie.user.name}
- D. \${cookies.userName[0]}
- E. \${cookies.userName}[1]
- F. \${cookies.get('userName')}

Answer: B

www.ExamWorx.com Q: 116 Given an EL function declared with:

11. <function>
12. <name>spin</name>
13. <function-class>com.example.Spinner</function-class>
14. <function-signature>
15. java.lang.String spinIt()
16. </function-signature>
17. </function>

Which two are true? (Choose two.)

- A. The function method must have the signature:
public String spin().
- B. The method must be mapped to the logical name "spin" in the web.xml file.
- C. The function method must have the signature:
public String spinIt().
- D. The function method must have the signature
public static String spin().
- E. The function method must have the signature:
public static String spinIt().

- F. The function class must be named Spinner, and must be in the package com.example.

Answer: E, F

www.ExamWorx.com Q: 117 Given a JSP page:

- 11. <n:recurse>
- 12. <n:recurse>
- 13. <n:recurse>
- 14. <n:recurse />
- 15. </n:recurse>
- 16. </n:recurse>
- 17. </n:recurse>

The tag handler for n:recurse extends SimpleTagSupport.

Assuming an n:recurse tag can either contain an empty body or another n:recurse tag, which strategy allows the tag handler for n:recurse to output the nesting depth of the deepest n:recurse tag?

- A. It is impossible to determine the deepest nesting depth because it is impossible for tag handlers that extend SimpleTagSupport to communicate with their parent and child tags.
- B. Create a private non-static attribute in the tag handler class called count of type int initialized to 0. Increment count in the doTag method. If the tag has a body, invoke the fragment for that body. Otherwise, output the value of count.
- C. Start a counter at 1. Call getChildTags(). If it returns null, output the value of the counter. Otherwise, increment counter and continue from where getChildTags() is called. Skip processing of the body.
- D. If the tag has a body, invoke the fragment for that body. Otherwise, start a counter at 1. Call getParent(). If it returns null, output the value of the counter. Otherwise, increment the counter and continue from where getParent() is called.

Answer: D

www.ExamWorx.com Q: 118 Click the Exhibit button.

The h:highlight tag renders its body, highlighting an arbitrary number of words, each of which is passed as an attribute (word1, word2, ...). For example, a JSP page can invoke the h:highlight tag as follows:

- 11. <h:highlight color="yellow" word1="high" word2="low">
- 12. high medium low
- 13. </h:highlight>

Given that **HighlightTag** extends **SimpleTagSupport**, which three steps are necessary to implement the tag handler for the highlight tag? (Choose three).

```
1. <?xml version="1.0" encoding="UTF-8" ?>
2.
3. <taglib
4.   xmlns="http://java.sun.com/xml/ns/j2ee"
5.   xmlns:xsi="http://www.w3.org/2001/XMLSchema-
a-instance"
6.   xsi:schemaLocation="http://java.sun.com/xm-
l/ns/j2ee web-jsptaglibrary_2_0.xsd"
7.   version="2.0">
8.   <tlib-version>1.0</tlib-version>
9.   <short-name>h</short-name>
10.  <uri>http://example.com/tld/highlight</uri>
11.  <tag>
12.    <name>highlight</name>
13.    <tag-class>com.example.HighlightTag</tag-
lass>
14.    <body-content>scriptless</body-content>
15.    <attribute>
16.      <name>color</name>
17.      <required>true</required>
18.    </attribute>
19.    <dynamic-attributes>true</dynamic-attributes>
20.  </taglib>
```

- A. add a doTag method
- B. add a doStartTag method
- C. add a getter and setter for the color attribute
- D. create and implement a TagExtraInfo class
- E. implement the DynamicAttributes interface
- F. add a getter and setter for the word1 and word2 attributes

Answer: A, C, E

www.ExamWorx.com Q: 119 Given:

```
5. public class MyTagHandler extends TagSupport {  
6.   public int doStartTag() throws JspException {  
7.     try {  
8.       // insert code here  
9.     } catch(Exception ex) { /* handle exception */ }  
10.    return super.doStartTag();  
11.  }  
...  
42. }
```

Which code snippet, inserted at line 8, causes the value foo to be output?

- A. JspWriter w = pageContext.getOut();
w.print("foo");
- B. JspWriter w = pageContext.getWriter();
w.print("foo");
- C. JspWriter w = new JspWriter(pageContext.getWriter());
w.print("foo");
- D. JspWriter w = new JspWriter(pageContext.getResponse());
w.print("foo");

Answer: A

www.ExamWorx.com Q: 120 Given:

```
6. <myTag:foo bar='42'>  
7. <%="processing" %>  
8. </myTag:foo>
```

and a custom tag handler for foo which extends TagSupport.

Which two are true about the tag handler referenced by foo? (Choose two.)

- A. The doStartTag method is called once.
- B. The doAfterBody method is NOT called.
- C. The EVAL_PAGE constant is a valid return value for the doEndTag method.
- D. The SKIP_PAGE constant is a valid return value for the doStartTag method.
- E. The EVAL_BODY_BUFFERED constant is a valid return value for the doStartTag method.

Answer: A, C

www.ExamWorx.com Q: 121 Which three are valid values for the body-content attribute of a tag directive in a tag file? (Choose three.)

- A. EL
- B. JSP
- C. empty
- D. dynamic
- E. scriptless
- F. tagdependent

Answer: C, E, F

www.ExamWorx.com Q: 122 The Squeaky Bean company has decided to port their web application to a new J2EE 1.4 container. While reviewing the application, a developer realizes that in multiple places within the current application, nearly duplicate code exists that finds enterprise beans. Which pattern should be used to eliminate this duplicate code?

- A. Transfer Object
- B. Front Controller
- C. Service Locator
- D. Intercepting Filter
- E. Business Delegate
- F. Model-View-Controller

Answer: C

www.ExamWorx.com Q: 123 A developer is designing a web application that makes many fine-grained remote data requests for each client request. During testing, the developer discovers that the volume of remote requests significantly degrades performance of the application. Which design pattern provides a solution for this problem?

- A. Flyweight
- B. Transfer Object
- C. Service Locator
- D. Dispatcher View
- E. Business Delegate
- F. Model-View-Controller

Answer: B

www.ExamWorx.com Q: 124 In an n-tier application, which two invocations are typically remote, not local? (Choose two.)

- A. JSP to Transfer Object
- B. Service Locator to JNDI
- C. Controller to request object
- D. Transfer Object to Entity Bean
- E. Controller to Business Delegate
- F. Business Delegate to Service Locator

Answer: B, D

www.ExamWorx.com Q: 125 A developer has created a special servlet that is responsible for generating XML content that is sent to a data warehousing subsystem. This subsystem uses HTTP to request these large data files, which are compressed by the servlet to save internal network bandwidth. The developer has received a request from management to create several more of these data warehousing servlets. The developer is about to copy and paste the compression code into each new servlet. Which design pattern can consolidate this compression code to be used by all of the data warehousing servlets?

- A. Facade
- B. View Helper
- C. Transfer Object
- D. Intercepting Filter
- E. Composite Facade

Answer: D

www.ExamWorx.com Q: 126 A developer is designing the presentation tier for a web application which requires a centralized request handling to complete common processing required by each request. Which design pattern provides a solution to this problem?

- A. Remote Proxy
- B. Front Controller
- C. Service Activator
- D. Intercepting Filter
- E. Business Delegate
- F. Data Access Object

Answer: B

www.ExamWorx.com Q: 127 You are designing an n-tier Java EE application. You have already decided that some of your JSPs will need to get data from a Customer entity bean. You are trying to decide whether to use a Customer stub object or a Transfer Object. Which two statements are true? (Choose two.)

- A. The stub will increase network traffic.
- B. The Transfer Object will decrease data staleness.
- C. The stub will increase the logic necessary in the JSPs.
- D. In both cases, the JSPs can use EL expressions to get data.
- E. Only the Transfer Object will need to use a Business Delegate.
- F. Using the stub approach allows you to design the application without using a Service Locator.

Answer: A, D

www.ExamWorx.com Q: 128 You have a simple web application that has a single Front Controller servlet that dispatches to JSPs to generate a variety of views. Several of these views require further database processing to retrieve the necessary order object using the orderID request parameter. To do this additional processing, you pass the request first to a servlet that is mapped to the URL pattern /WEB-INF/retrieveOrder.do in the deployment descriptor. This servlet takes two request parameters, the orderID and the jspURL. It handles the database calls to retrieve and build the complex order objects and then it dispatches to the jspURL.

Which code snippet in the Front Controller servlet dispatches the request to the order retrieval servlet?

- A. request.setAttribute("orderID", orderID);
request.setAttribute("jspURL", jspURL);
RequestDispatcher view
= context.getRequestDispatcher("/WEB-INF/retrieveOrder.do");
view.forward(request, response);
- B. request.setParameter("orderID", orderID);
request.setParameter("jspURL", jspURL);
Dispatcher view
= request.getDispatcher("/WEB-INF/retrieveOrder.do");
view.forwardRequest(request, response);
- C. String T="/WEB-INF/retrieveOrder.do?orderID=%d&jspURL=%s";
String url = String.format(T, orderID, jspURL);
RequestDispatcher view
= context.getRequestDispatcher(url);
view.forward(request, response);
- D. String T="/WEB-INF/retrieveOrder.do?orderID=%d&jspURL=%s";
String url = String.format(T, orderID, jspURL);
Dispatcher view

```
= context.getDispatcher(url);
view.forwardRequest(request, response);
```

Answer: C

www.ExamWorx.com Q: 129 You have built a web application that you license to small businesses. The webapp uses a context parameter, called licenseExtension, which enables certain advanced features based on your client's license package. When a client pays for a specific service, you provide them with a license extension key that they insert into the <context-param> of the deployment descriptor. Not every client will have this context parameter so you need to create a context listener to set up a default value in the licenseExtension parameter. Which code snippet will accomplish this goal?

A. You cannot do this because context parameters CANNOT be altered programmatically.

B. String ext = context.getParameter('licenseExtension');

```
if ( ext == null ) {
```

```
context.setParameter('licenseExtension', DEFAULT);
```

```
}
```

C. String ext = context.getAttribute('licenseExtension');

```
if ( ext == null ) {
```

```
context.setAttribute('licenseExtension', DEFAULT);
```

```
}
```

D. String ext = context.getInitParameter('licenseExtension');

```
if ( ext == null ) {
```

```
context.resetInitParameter("licenseExtension", DEFAULT);
```

```
}
```

E. String ext = context.getInitParameter('licenseExtension');

```
if ( ext == null ) {
```

```
context.setInitParameter('licenseExtension', DEFAULT);
```

```
}
```

Answer: A

www.ExamWorx.com Q: 130 You have a use case in your web application that adds several session-scoped attributes. At the end of the use case, one of these objects, the manager attribute, is removed and then it needs to decide which of the other session-scoped attributes to remove.

How can this goal be accomplished?

A. The object of the manager attribute should implement the HttpSessionBindingListener and it should call the removeAttribute method on the appropriate session attributes.

B. The object of the manager attribute should implement the HttpSessionListener and it should call the removeAttribute method on the appropriate session attributes.

- C. The object of the manager attribute should implement the HttpSessionBindingListener and it should call the deleteAttribute method on the appropriate session attributes.
- D. The object of the manager attribute should implement the HttpSessionListener and it should call the deleteAttribute method on the appropriate session attributes.

Answer: A

www.ExamWorx.com Q: 131 You want to create a filter for your web application and your filter will implement javax.servlet.Filter.

Which two statements are true? (Choose two.)

- A. Your filter class must implement an init method and a destroy method.
- B. Your filter class must also implement javax.servlet.FilterChain.
- C. When your filter chains to the next filter, it should pass the same arguments it received in its doFilter method.
- D. The method that your filter invokes on the object it received that implements javax.servlet.FilterChain can invoke either another filter or a servlet.
- E. Your filter class must implement a doFilter method that takes, among other things, an HttpServletRequest object and an HttpServletResponse object.

Answer: A, D

www.ExamWorx.com Q: 132 Your web site has many user-customizable features, for example font and color preferences on web pages. Your IT department has already built a subsystem for user preferences using Java SE's lang.util.prefs package APIs and you have been ordered to reuse this subsystem in your web application. You need to create an event listener that stores the user's Preference object when an HTTP session is created. Also, note that user identification information is stored in an HTTP cookie.

Which partial listener class can accomplish this goal?

- A. public class UserPrefLoader implements HttpSessionListener {
public void sessionCreated(HttpSessionEvent se) {
MyPrefsFactory myFactory = (MyPrefsFactory) se.getServletContext().getAttribute("myPrefsFactory");
User user = getUserFromCookie(se);
myFactory.setThreadLocalUser(user);
Preferences userPrefs = myFactory.userRoot();
se.getSession().setAttribute("prefs", userPrefs);
}
// more code here
}
B. public class UserPrefLoader implements SessionListener {

```

public void sessionCreated(SessionEvent se) {
    MyPrefsFactory myFactory = (MyPrefsFactory) se.getContext().getAttribute("myPrefsFactory");
    User user = getUserFromCookie(se);
    myFactory.setThreadLocalUser(user);
    Preferences userPrefs = myFactory.userRoot();
    se.getSession().addAttribute("prefs", userPrefs);
}
// more code here
}

C. public class UserPrefLoader implements HttpSessionListener {
public void sessionInitialized(HttpSessionEvent se) {
    MyPrefsFactory myFactory = (MyPrefsFactory) se.getServletContext().getAttribute("myPrefsFactory");
    User user = getUserFromCookie(se);
    myFactory.setThreadLocalUser(user);
    Preferences userPrefs = myFactory.userRoot();
    se.getHttpSession().setAttribute("prefs", userPrefs);
}
// more code here
}

D. public class UserPrefLoader implements SessionListener {
public void sessionInitialized(SessionEvent se) {
    MyPrefsFactory myFactory = (MyPrefsFactory) se.getServletContext().getAttribute("myPrefsFactory");
    User user = getUserFromCookie(se);
    myFactory.setThreadLocalUser(user);
    Preferences userPrefs = myFactory.userRoot();
    se.getSession().addAttribute("prefs", userPrefs);
}
// more code here
}

```

Answer: A

www.ExamWorx.com Q: 133 Given the web application deployment descriptor elements:

11. <filter>
12. <filter-name>ParamAdder</filter-name>
13. <filter-class>com.example.ParamAdder</filter-class>
14. </filter>
- ...
24. <filter-mapping>
25. <filter-name>ParamAdder</filter-name>
26. <servlet-name>MyServlet</servlet-name>
27. <!-- insert element here -->
28. </filter-mapping>

Which element, inserted at line 27, causes the ParamAdder filter to be applied when MyServlet is invoked by another servlet using the RequestDispatcher.include method?

- A. <include/>
- B. <dispatcher>INCLUDE</dispatcher>
- C. <dispatcher>include</dispatcher>
- D. <filter-condition>INCLUDE</filter-condition>
- E. <filter-condition>include</filter-condition>

Answer: B

www.ExamWorx.com Q: 134 Your web application uses a simple architecture in which servlets handle requests and then forward to a JSP using a request dispatcher. You need to pass information calculated by the servlet to the JSP; furthermore, that JSP uses a custom tag and must also process this information. This information must NOT be accessible to any other servlet, JSP or session in the webapp. How can you accomplish this goal?

- A. Store the data in a public instance variable in the servlet.
- B. Add an attribute to the request object before using the request dispatcher.
- C. Add an attribute to the context object before using the request dispatcher.
- D. This CANNOT be done as the tag handler has no means to extract this data.

Answer: B

www.ExamWorx.com Q: 135 A JSP page needs to set the property of a given JavaBean to a value that is calculated with the JSP page. Which three `jsp:setProperty` attributes must be used to perform this initialization? (Choose three.)

- A. id
- B. val
- C. name
- D. param
- E. value
- F. property
- G. attribute

Answer: C, E, F

www.ExamWorx.com Q: 136 Your web application views all have the same header, which includes the <title> tag in the <head> element of the rendered HTML. You have decided to remove this redundant HTML code from your JSPs and put it into a single JSP called /WEB-INF/jsp/header.jsp. However, the title of each page is unique, so you have decided to use a variable called pageTitle to parameterize this in the header JSP, like this:

10. <title>\${param.pageTitle}</title>

Which JSP code snippet should you use in your main view JSPs to insert the header and pass the pageTitle variable?

- A. <jsp:insert page='/WEB-INF/jsp/header.jsp'>
 \${pageTitle='Welcome Page'}
</jsp:insert>
- B. <jsp:include page='/WEB-INF/jsp/header.jsp'>
 \${pageTitle='Welcome Page'}
</jsp:include>
- C. <jsp:include file='/WEB-INF/jsp/header.jsp'>
 \${pageTitle='Welcome Page'}
</jsp:include>
- D. <jsp:insert page='/WEB-INF/jsp/header.jsp'>
 <jsp:param name='pageTitle' value='Welcome Page' />
</jsp:insert>
- E. <jsp:include page='/WEB-INF/jsp/header.jsp'>
 <jsp:param name='pageTitle' value='Welcome Page' />
</jsp:include>

Answer: E

www.ExamWorx.com Q: 137 A JSP page needs to instantiate a JavaBean to be used by only that page. Which two jsp:useBean attributes must be used to access this attribute in the JSP page? (Choose two.)

- A. id
- B. type
- C. name
- D. class
- E. scope
- F. create

Answer: A, D

www.ExamWorx.com Q: 138 Click the Exhibit button.

Given the HTML form:

```
1. <html>
2. <body>
3.   <form action="submit.jsp">
4.     Name: <input type="text" name="i1"><br>
5.     Price: <input type="text" name="i2"><br>
6.     <input type="submit">
7.   </form>
8. </body>
9. </html>
```

Assume the product attribute does NOT yet exist in any scope.

Which code snippet, in submit.jsp, instantiates an instance of com.example.Product that contains the results of the form submission?

```
1. package com.example;
2.
3. public class Product {
4.     private String name;
5.     private double price;
6.
7.     public Product() {
8.         this( "Default", 0.0 );
9.     }
10.
11.    public Product( String name, double
price ) {
12.        this.name = name;
13.        this.price = price;
14.    }
15.
16.    public String getName() {
17.        return name;
18.    }
19.
20.    public void setName(String name) {
21.        this.name = name;
22.    }
23.
24.    public double getPrice() {
25.        return price;
26.    }
27.
28.    public void setPrice(double price) {
29.        this.price = price;
30.    }
31. }
```

- A. <jsp:useBean id="com.example.Product" />
<jsp:setProperty name="product" property="*" />
- B. <jsp:useBean id="product" class="com.example.Product" />
 \${product.name = param.i1}
 \${product.price = param.i2}
- C. <jsp:useBean id="product" class="com.example.Product">
<jsp:setProperty name="product" property="name"
param="i1" />
<jsp:setProperty name="product" property="price"

```
param="i2" />
</jsp:useBean>
D. <jsp:useBean id="product" type="com.example.Product">
<jsp:setProperty name="product" property="name"
value="<% request.getParameter( "i1" ) %>" />
<jsp:setProperty name="product" property="price"
value="<% request.getParameter( "i2" ) %>" />
</jsp:useBean>
```

Answer: C

www.ExamWorx.com Q: 139 Click the Task button.

Place the events in the order they occur.

Drag and Drop

Place the events in the order they occur.

Order of Steps	Event
1st	web container instantiates the servlet
2nd	web container calls the servlet's destroy() method
3rd	web container loads the servlet class
4th	web container calls the servlet's init() method
5th	web container calls the servlet's service() method

Done

Answer: Check ExamWorx eEngine, Download from Member Center

www.ExamWorx.com Q: 140 For an `HttpServletResponse` response, which two create a custom header? (Choose two.)

- A. response.setHeader("X-MyHeader", "34");
- B. response.addHeader("X-MyHeader", "34");
- C. response.setHeader(new HttpHeader("X-MyHeader", "34"));
- D. response.addHeader(new HttpHeader("X-MyHeader", "34"));
- E. response.addHeader(new ServletHeader("X-MyHeader", "34"));
- F. response.setHeader(new ServletHeader("X-MyHeader", "34"));

Answer: A, B

www.ExamWorx.com Q: 141 You need to create a servlet filter that stores all request headers to a database for all requests to the web application's home page "/index.jsp". Which HttpServletRequest method allows you to retrieve all of the request headers?

- A. String[] getHeaderNames()
- B. String[] getRequestHeaders()
- C. java.util.Iterator getHeaderNames()
- D. java.util.Iterator getRequestHeaders()
- E. java.util Enumeration getHeaderNames()
- F. java.util Enumeration getRequestHeaders()

Answer: E

www.ExamWorx.com Q: 142 Given an HttpServletRequest request and HttpServletResponse response, which sets a cookie "username" with the value "joe" in a servlet?

- A. request.addCookie("username", "joe")
- B. request.setCookie("username", "joe")
- C. response.addCookie("username", "joe")
- D. request.addHeader(new Cookie("username", "joe"))
- E. request.addCookie(new Cookie("username", "joe"))
- F. response.addCookie(new Cookie("username", "joe"))
- G. response.addHeader(new Cookie("username", "joe"))

Answer: F

www.ExamWorx.com Q: 143 Click the Task button.

Given a request from mybox.example.com, with an IP address of 10.0.1.11 on port 33086, place the appropriate ServletRequest methods onto their corresponding return values.

Drag and Drop

Given a request from mybox.example.com, with an IP address of 10.0.1.11 on port 33086, place the appropriate ServletRequest methods onto their corresponding return values.

Proxy / Client Settings:

- mybox.example.com
- 10.0.1.11
- 33086

ServletRequest Methods:

- getServerPort
- getServerAddr
- getServerName
- getRemotePort
- getRemoteAddr
- getRemoteHost

Done

Answer: Check ExamWorx eEngine, Download from Member Center

www.ExamWorx.com Q: 144 Your web application requires the ability to load and remove web files dynamically to the web container's file system. Which two HTTP methods are used to perform these actions? (Choose two.)

- A. PUT
- B. POST
- C. SEND
- D. DELETE
- E. REMOVE

F. DESTROY

Answer: A, D

www.ExamWorx.com Q: 145 Every page of your web site must include a common set of navigation menus at the top of the page. This menu is static HTML and changes frequently, so you have decided to use JSP's static import mechanism. Which JSP code snippet accomplishes this goal?

- A. <%@ import file='/common/menu.html' %>
- B. <%@ page import='/common/menu.html' %>
- C. <%@ import page='/common/menu.html' %>
- D. <%@ include file='/common/menu.html' %>
- E. <%@ page include='/common/menu.html' %>
- F. <%@ include page='/common/menu.html' %>

Answer: D

www.ExamWorx.com Q: 146 For manageability purposes, you have been told to add a "count" instance variable to a critical JSP Document so that a JMX MBean can track how frequent this JSP is being invoked. Which JSP code snippet must you use to declare this instance variable in the JSP Document?

- A. <jsp:declaration>
int count = 0;
<jsp:declaration>
- B. <%! int count = 0; %>
- C. <jsp:declaration.instance>
int count = 0;
<jsp:declaration.instance>
- D. <jsp:scriptlet.declaration>
int count = 0;
<jsp:scriptlet.declaration>

Answer: A

www.ExamWorx.com Q: 147 You have a new IT manager that has mandated that all JSPs must be refactored to include no scriplet code. The IT manager has asked you to enforce this. Which deployment descriptor element will satisfy this constraint?

- A. <jsp-property-group>
<url-pattern>*.jsp</url-pattern>

```

<permit-scripting>false</permit-scripting>
</jsp-property-group>
B. <jsp-config>
<url-pattern>*.jsp</url-pattern>
<permit-scripting>false</permit-scripting>
</jsp-config>
C. <jsp-config>
<url-pattern>*.jsp</url-pattern>
<scripting-invalid>true</scripting-invalid>
</jsp-config>
D. <jsp-property-group>
<url-pattern>*.jsp</url-pattern>
<scripting-invalid>true</scripting-invalid>
</jsp-property-group>

```

Answer: D

www.ExamWorx.com Q: 148 You need to create a JSP that generates some JavaScript code to populate an array of strings used on the client-side. Which JSP code snippet will create this array?

```

A. MY_ARRAY = new Array();
<% for ( int i = 0; i < serverArray.length; i++ ) {
MY_ARRAY[<%= i %>] = '<%= serverArray[i] %>';
} %>
B. MY_ARRAY = new Array();
<% for ( int i = 0; i < serverArray.length; i++ ) {
MY_ARRAY[$i] = '${serverArray[i]}';
} %>
C. MY_ARRAY = new Array();
<% for ( int i = 0; i < serverArray.length; i++ ) { %>
MY_ARRAY[<%= i %>] = '<%= serverArray[i] %>';
<% } %>
D. MY_ARRAY = new Array();
<% for ( int i = 0; i < serverArray.length; i++ ) { %>
MY_ARRAY[$i] = '${serverArray[i]}';
<% } %>

```

Answer: C

www.ExamWorx.com Q: 149 You are building a Front Controller using a JSP page and you need to determine if the user's session has NOT been created yet and perform some special processing for this case. Which scriptlet code snippet will perform this test?

- A. <% if (request.getSession(false) == null) {
// special processing
} %>
- B. <% if (request.getHttpSession(false) == null) {
// special processing
} %>
- C. <% if (requestObject.getSession(false) == null) {
// special processing
} %>
- D. <% if (requestObject.getHttpSession(false) == null) {
// special processing
} %>

Answer: A

www.ExamWorx.com Q: 150 You are creating a new JSP page and you need to execute some code that acts when the page is first executed, but only once. Which three are possible mechanisms for performing this initialization code? (Choose three.)

- A. In the init method.
- B. In the jspInit method.
- C. In the constructor of the JSP's Java code.
- D. In a JSP declaration, which includes an initializer block.
- E. In a JSP declaration, which includes a static initializer block.

Answer: B, D, E

www.ExamWorx.com Q: 151 Click the Task button.

Place the code snippets in the proper order to construct the JSP code to include dynamic content into a JSP page at request-time.

Drag and Drop

Place the code snippets in the proper order to construct the JSP code to include dynamic content into a JSP page at request-time.

JSP Code:

Place here. Place here. Place here.

Code Snippets:

import='foo.jsp'	/>	file='foo.jsp'
<%@ include	<jsp:import	page='foo.jsp'
%>	<%@ import	<jsp:include

Done

The interface consists of a window titled "Drag and Drop". At the top, there are three yellow rectangular boxes with the placeholder text "Place here.". Below this, under the heading "Code Snippets:", is a 3x3 grid of cyan rectangular boxes containing JSP code fragments. The first row contains "import='foo.jsp'", "/>", and "file='foo.jsp'". The second row contains "<%@ include", "<jsp:import", and "page='foo.jsp'". The third row contains "%>", "<%@ import", and "<jsp:include>". At the bottom center of the window is a grey rectangular button labeled "Done".

Answer: Check ExamWorx eEngine, Download from Member Center

www.ExamWorx.com Q: 152 Given:

6. <% int[] nums = {42, 420, 4200};
7. request.setAttribute("foo", nums); %>

Which two successfully translate and result in a value of true? (Choose two.)

- A. \${true or false}
- B. \${requestScope[foo][0] > 500}
- C. \${requestScope['foo'][1] = 420}
- D. \${((requestScope['foo'][0] lt 50) && (3 gt 2)}

Answer: A, D

www.ExamWorx.com Q: 153 Click the Exhibit button.

Given:

11. <% com.example.Advisor advisor = new com.example.Advisor(); %>
12. <% request.setAttribute("foo", advisor); %>

Assuming there are no other "foo" attributes in the web application, which three are valid EL expressions for retrieving the advice property of advisor? (Choose three.)

```
1. package com.example;
2.
3. public class Advisor {
4.     private String advice="take out the
garbage";
5.     public String getAdvice() {
6.         return advice;
7.     }
8.     public void setAdvice(String advice) {
9.         this.advice = advice;
10.    }
11. }
```

- A. \${foo.advice}
- B. \${request.foo.advice}
- C. \${requestScope.foo.advice}
- D. \${requestScope[foo[advice]]}
- E. \${requestScope["foo"]["advice"]}
- F. \${requestScope["foo"]["advice"]}]

Answer: A, C, E

www.ExamWorx.com Q: 154 You are creating an error page that provides a user-friendly screen whenever a server exception occurs. You want to hide the stack trace, but you do want to provide the exception's error message to the user so the user can provide it to the customer service agent at your company. Which EL code snippet inserts this error message into the error page?

- A. Message: \${exception.message}
- B. Message: \${exception.errorMessage}

- C. Message: \${request.exception.message}
- D. Message: \${pageContext.exception.message}
- E. Message: \${request.exception.errorMessage}
- F. Message: \${pageContext.exception.errorMessage}

Answer: D

www.ExamWorx.com Q: 155 You are building a dating web site. The client's date of birth is collected along with lots of other information. You have created an EL function with the signature: calcAge(java.util.Date):int and it is assigned to the name, age, in the namespace, funct. In one of your JSPs you need to print a special message to clients who are younger than 25. Which EL code snippet will return true for this condition?

- A. \${calcAge(client.birthDate) < 25}
- B. \${calcAge[client.birthDate] < 25}
- C. \${funct:age(client.birthDate) < 25}
- D. \${funct:age[client.birthDate] < 25}
- E. \${funct:calcAge(client.birthDate) < 25}
- F. \${funct:calcAge[client.birthDate] < 25}

Answer: C

www.ExamWorx.com Q: 156 Given:

```
11. <% java.util.Map map = new java.util.HashMap();  
12.   request.setAttribute("map", map);  
13.   map.put("a", "b");  
14.   map.put("b", "c");  
15.   map.put("c", "d"); %>  
16. <%-- insert code here --%>
```

Which three EL expressions, inserted at line 16, are valid and evaluate to "d"? (Choose three.)

- A. \${map.c}
- B. \${map[c]}
- C. \${map["c"]}
- D. \${map.map.b}
- E. \${map[map.b]}
- F. \${map.(map.b)}

Answer: A, C, E

www.ExamWorx.com Q: 157 Assume the tag handler for a st:simple tag extends SimpleTagSupport. In what way can scriptlet code be used in the body of st:simple?

- A. set the body content type to JSP in the TLD
- B. Scriptlet code is NOT legal in the body of st:simple.
- C. add scripting-enabled="true" to the start tag for the st:simple element
- D. add a pass-through Classic tag with a body content type of JSP to the body of st:simple, and place the scriptlet code in the body of that tag

Answer: B

www.ExamWorx.com Q: 158 Which statement is true if the doStartTag method returns EVAL_BODY_BUFFERED ?

- A. The tag handler must implement BodyTag.
- B. The doAfterBody method is NOT called.
- C. The setBodyContent method is called once.
- D. It is never legal to return EVAL_BODY_BUFFERED from doStartTag.

Answer: C

www.ExamWorx.com Q: 159 You are creating a library of custom tags that mimic the HTML form tags. When the user submits a form that fails validation, the JSP form is forwarded back to the user. The <t:textField> tag must support the ability to re-populate the form field with the request parameters from the user's last request. For example, if the user entered "Samantha" in the text field called firstName, then the form is re-populated like this:

```
<input type='text' name='firstName' value='Samantha' />
```

Which tag handler method will accomplish this goal?

```
A. public int doStartTag() throws JspException {  
    JspContext ctx = getJspContext();  
    String value = ctx.getParameter(this.name);  
    if ( value == null ) value = "";  
    JspWriter out = pageContext.getOut();  
    try {  
        out.write(String.format(INPUT, this.name, value));  
    } (Exception e) { throw new JspException(e); }  
    return SKIP_BODY;
```

```

}

private static String INPUT
= "<input type='text' name='%s' value='%s' />";
B. public void doTag() throws JspException {
JspContext ctx = getJspContext();
String value = ctx.getParameter(this.name);
if ( value == null ) value = "";
JspWriter out = pageContext.getOut();
try {
out.write(String.format(INPUT, this.name, value));
} (Exception e) { throw new JspException(e); }
}

private static String INPUT
= "<input type='text' name='%s' value='%s' />";
C. public int doStartTag() throws JspException {
ServletRequest request = pageContext.getRequest();
String value = request.getParameter(this.name);
if ( value == null ) value = "";
JspWriter out = pageContext.getOut();
try {
out.write(String.format(INPUT, this.name, value));
} (Exception e) { throw new JspException(e); }
return SKIP_BODY;
}

private static String INPUT
= "<input type='text' name='%s' value='%s' />";
D. public void doTag() throws JspException {
ServletRequest request = pageContext.getRequest();
String value = request.getParameter(this.name);
if ( value == null ) value = "";
JspWriter out = pageContext.getOut();
try {
out.write(String.format(INPUT, this.name, value));
} (Exception e) { throw new JspException(e); }
}

private static String INPUT
= "<input type='text' name='%s' value='%s' />";

```

Answer: C

www.ExamWorx.com Q: 160 Which two directives are applicable only to tag files? (Choose two.)

- A. tag
- B. page

- C. taglib
- D. include
- E. variable

Answer: A, E

www.ExamWorx.com Q: 161 The **tl:taskList** and **tl:task** tags output a set of tasks to the response and are used as follows:

11. **<tl:taskList>**
12. **<tl:task name="Mow the lawn" />**
13. **<tl:task name="Feed the dog" />**
14. **<tl:task name="Do the laundry" />**
15. **</tl:taskList>**

The **tl:task** tag supplies information about a single task while the **tl:taskList** tag does the final output. The tag handler for **tl:taskList** is **TaskListTag**. The tag handler for **tl:task** is **TaskTag**. Both tag handlers extend **BodyTagSupport**.

Which allows the tl:taskList tag to get the task names from its nested tl:task children?

- A. It is impossible for a tag handler that extends BodyTagSupport to communicate with its parent and child tags.
- B. In the **TaskListTag.doStartTag** method, call **super.getChildTags()** and iterate through the results. Cast each result to a **TaskTag** and call **getName()**.
- C. In the **TaskListTag.doStartTag** method, call **getChildTags()** on the **PageContext** and iterate through the results. Cast each result to a **TaskTag** and call **getName()**.
- D. Create an **addTaskName** method in **TaskListTag**. Have the **TaskListTag.doStartTag** method, return **BodyTag.EVAL_BODY_BUFFERED**. In the **TaskTag.doStartTag** method, call **super.getParent()**, cast it to a **TaskListTag**, and call **addTaskName()**.
- E. Create an **addTaskName** method in **TaskListTag**. Have the **TaskListTag.doStartTag** method, return **BodyTag.EVAL_BODY_BUFFERED**. In the **TaskTag.doStartTag** method, call **findAncestorWithClass()** on the **PageContext**, passing **TaskListTag** as the class to find. Cast the result to **TaskListTag** and call **addTaskName()**.

Answer: D

www.ExamWorx.com Q: 162 Click the Exhibit button.

Given:

10. **<form action='create_product.jsp'>**
11. **Product Name: <input type='text' name='prodName' />
**
12. **Product Price: <input type='text' name='prodPrice' />
**

13. </form>

For a given product instance, which three jsp:setProperty attributes must be used to initialize its properties from the HTML form? (Choose three.)

```
1. package com.example;
2.
3. public class Product {
4.     private String name;
5.     private double price;
6.
7.     public Product() {
8.         this( "Default", 0.0 );
9.     }
10.
11.    public Product( String name, double
price ) {
12.        this.name = name;
13.        this.price = price;
14.    }
15.
16.    public String getName() {
17.        return name;
18.    }
19.
20.    public void setName(String name) {
21.        this.name = name;
22.    }
23.
24.    public double getPrice() {
25.        return price;
26.    }
27.
28.    public void setPrice(double price) {
29.        this.price = price;
30.    }
31. }
```

- A. id
- B. name
- C. type
- D. param

- E. property
- F. reqParam
- G. attribute

Answer: B, D, E

www.ExamWorx.com Q: 163 Given:

```
1. package com.example;  
2.  
3. public abstract class AbstractItem {  
4.     private String name;  
...  
13. }
```

Assume a concrete class com.example.ConcreteItem extends com.example.AbstractItem. A servlet sets a session-scoped attribute called "item" that is an instance of com.example.ConcreteItem and then forwards to a JSP page.

Which two are valid standard action invocations that expose a scripting variable to the JSP page? (Choose two.)

- A. <jsp:useBean id="com.example.ConcreteItem" scope="session" />
- B. <jsp:useBean id="item" type="com.example.ConcreteItem" scope="session" />
- C. <jsp:useBean id="item" class="com.example.ConcreteItem" scope="session" />
- D. <jsp:useBean id="item" type="com.example.ConcreteItem" class="com.example.AbstractItem" scope="session" />

Answer: B, C

www.ExamWorx.com Q: 164 Your web application views all have the same header, which includes the <title> tag in the <head> element of the rendered HTML. You have decided to remove this redundant HTML code from your JSPs and put it into a single JSP called /WEB-INF/jsp/header.jsp. However, the title of each page is unique, so you have decided to use a variable called pageTitle to parameterize this in the header JSP, like this:

10. <title>\${param.pageTitle}</title>

Which JSP code snippet should you use in your main view JSPs to insert the header and pass the pageTitle variable?

- A. <jsp:insert page='/WEB-INF/jsp/header.jsp'>
 \${pageTitle='Welcome Page'}
</jsp:insert>
- B. <jsp:include page='/WEB-INF/jsp/header.jsp'>
 \${pageTitle='Welcome Page'}
</jsp:include>
- C. <jsp:include file='/WEB-INF/jsp/header.jsp'>
 \${pageTitle='Welcome Page'}
</jsp:include>
- D. <jsp:insert page='/WEB-INF/jsp/header.jsp'>
 <jsp:param name='pageTitle' value='Welcome Page' />
</jsp:insert>
- E. <jsp:include page='/WEB-INF/jsp/header.jsp'>
 <jsp:param name='pageTitle' value='Welcome Page' />
</jsp:include>

Answer: E

www.ExamWorx.com Q: 165 Click the Exhibit button.

Given the JSP code:

1. <%
2. pageContext.setAttribute("product",
3. new com.example.Product("Pizza", 0.99));
4. %>
5. <%-- insert code here --%>

Which two, inserted at line 5, output the name of the product in the response? (Choose two.)

```
1. package com.example;
2.
3. public class Product {
4.     private String name;
5.     private double price;
6.
7.     public Product() {
8.         this( "Default", 0.0 );
9.     }
10.
11.    public Product( String name, double
price ) {
12.        this.name = name;
13.        this.price = price;
14.    }
15.
16.    public String getName() {
17.        return name;
18.    }
19.
20.    public void setName(String name) {
21.        this.name = name;
22.    }
23.
24.    public double getPrice() {
25.        return price;
26.    }
27.
28.    public void setPrice(double price) {
29.        this.price = price;
30.    }
31. }
```

- A. <%= product.getName() %>
- B. <jsp:useBean id="product" class="com.example.Product" />
<%= product.getName() %>
- C. <jsp:useBean id="com.example.Product" scope="page">
<%= product.getName() %>
</jsp:useBean>
- D. <jsp:useBean id="product" type="com.example.Product"
scope="page" />
<%= product.getName() %>
- E. <jsp:useBean id="product" type="com.example.Product">

```
<%= product.getName() %>
</jsp:useBean>
```

Answer: B, D

www.ExamWorx.com Q: 166 If you want to use the Java EE platform's built-in type of authentication that uses a custom HTML page for authentication, which two statements are true? (Choose two.)

- A. Your deployment descriptor will need to contain this tag:
<auth-method>CUSTOM</auth-method>.
- B. The related custom HTML login page must be named loginPage.html.
- C. When you use this type of authentication, SSL is turned on automatically.
- D. You must have a tag in your deployment descriptor that allows you to point to both a login HTML page and an HTML page for handling any login errors.
- E. In the HTML related to authentication for this application, you must use predefined variable names for the variables that store the user and password values.

Answer: D, E

www.ExamWorx.com Q: 167 Given the two security constraints in a deployment descriptor:

- 101. <security-constraint>
- 102. <!--a correct url-pattern and http-method goes here-->
- 103. <auth-constraint><role-name>SALES</role-name></auth-constraint>
- 104. <role-name>SALES</role-name>
- 105. </auth-constraint>
- 106. </security-constraint>
- 107. <security-constraint>
- 108. <!--a correct url-pattern and http-method goes here-->
- 109. <!-- Insert an auth-constraint here -->
- 110. </security-constraint>

If the two security constraints have the same url-pattern and http-method, which two, inserted independently at line 109, will allow users with role names of either SALES or MARKETING to access this resource? (Choose two.)

- A. <auth-constraint/>
- B. <auth-constraint><role-name>*</role-name></auth-constraint>
- C. <auth-constraint>

```
<role-name>ANY</role-name>
</auth-constraint>
D. <auth-constraint>
<role-name>MARKETING</role-name>
</auth-constraint>
```

Answer: B, D

www.ExamWorx.com Q: 168 Which two are valid values for the <transport-guarantee> element inside a <security-constraint> element of a web application deployment descriptor? (Choose two.)

- A. NULL
- B. SECURE
- C. INTEGRAL
- D. ENCRYPTED
- E. CONFIDENTIAL

Answer: C, E

www.ExamWorx.com Q: 169 Which basic authentication type is optional for a J2EE 1.4 compliant web container?

- A. HTTP Basic Authentication
- B. Form Based Authentication
- C. HTTP Digest Authentication
- D. HTTPS Client Authentication

Answer: C

www.ExamWorx.com Q: 170 Which security mechanism uses the concept of a realm?

- A. authorization
- B. data integrity
- C. confidentiality
- D. authentication

Answer: D

www.ExamWorx.com Q: 171 Which two security mechanisms can be directed through a sub-element of the <user-data-constraint> element in a web application deployment descriptor? (Choose two.)

- A. authorization
- B. data integrity
- C. confidentiality
- D. authentication

Answer: B, C

www.ExamWorx.com Q: 172 You are developing several tag libraries that will be sold for development of third-party web applications. You are about to publish the first three libraries as JAR files: container-tags.jar, advanced-html-form-tags.jar, and basic-html-form-tags.jar. Which two techniques are appropriate for packaging the TLD files for these tag libraries? (Choose two.)

- A. The TLD must be located within the WEB-INF directory of the JAR file.
- B. The TLD must be located within the META-INF directory of the JAR file.
- C. The TLD must be located within the META-INF/tld/ directory of the JAR file.
- D. The TLD must be located within a subdirectory of WEB-INF directory of the JAR file.
- E. The TLD must be located within a subdirectory of META-INF directory of the JAR file.
- F. The TLD must be located within a subdirectory of META-INF/tld/ directory of the JAR file.

Answer: B, E

www.ExamWorx.com Q: 173 A custom tag is defined to take three attributes. Which two correctly invoke the tag within a JSP page? (Choose two.)

- A. <prefix:myTag a="foo" b="bar" c="baz" />
- B. <prefix:myTag attributes={"foo","bar","baz"} />
- C. <prefix:myTag jsp:attribute a="foo" b="bar" c="baz" />
- D. <prefix:myTag>
<jsp:attribute a:foo b:bar c:baz />
</prefix:myTag>
- E. <prefix:myTag>
<jsp:attribute \${"foo", "bar", "baz"} />
</prefix:myTag>
- F. <prefix:myTag>
<jsp:attribute a="foo" b="bar" c="baz"/>
</prefix:myTag>
- G. <prefix:myTag>
<jsp:attribute name="a">foo</jsp:attribute>

```
<jsp:attribute name="b">bar</jsp:attribute>
<jsp:attribute name="c">baz</jsp:attribute>
</prefix:myTag>
```

Answer: A, G

www.ExamWorx.com Q: 174 In a JSP-centric shopping cart application, you need to move a client's home address of the Customer object into the shipping address of the Order object. The address data is stored in a value object class called Address with properties for: street address, city, province, country, and postal code. Which two JSP code snippets can be used to accomplish this goal? (Choose two.)

- A. <c:set var='order' property='shipAddress'
value='\${client.homeAddress}' />
- B. <c:set target='\${order}' property='shipAddress'
value='\${client.homeAddress}' />
- C. <jsp:setProperty name='\${order}' property='shipAddress'
value='\${client.homeAddress}' />
- D. <c:set var='order' property='shipAddress'>
<jsp:getProperty name='client' property='homeAddress' />
</c:store>
- E. <c:set target='\${order}' property='shipAddress'>
<jsp:getProperty name='client' property='homeAddress' />
</c:set>
- F. <c:setProperty name='\${order}' property='shipAddress'>
<jsp:getProperty name='client' property='homeAddress' />
</c:setProperty>

Answer: B, E

www.ExamWorx.com Q: 175 You have been contracted to create a web site for a free dating service. One feature is the ability for one client to send a message to another client, which is displayed in the latter client's private page. Your contract explicitly states that security is a high priority. Therefore, you need to prevent cross-site hacking in which one user inserts JavaScript code that is then rendered and invoked when another user views that content. Which two JSTL code snippets will prevent cross-site hacking in the scenario above? (Choose two.)

- A. <c:out>\${message}</c:out>
- B. <c:out value='\${message}' />
- C. <c:out value='\${message}' escapeXml='true' />
- D. <c:out eliminateXml='true'>\${message}</c:out>
- E. <c:out value='\${message}' eliminateXml='true' />

Answer: B, C

www.ExamWorx.com Q: 176 Click the Exhibit button.

Assuming the tag library in the exhibit is imported with the prefix forum, which custom tag invocation produces a translation error in a JSP page?

```
1. <?xml version="1.0" encoding="UTF-8" ?>
2.
3. <taglib
4.   xmlns="http://java.sun.com/xml/ns/j2ee"
5.   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6.   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd"
7.   version="2.0">
8.   <tlib-version>1.0</tlib-version>
9.   <short-name>forum</short-name>
10.  <uri>http://example.com/tld/forum</uri>
11.  <tag>
12.    <name>message</name>
13.
<tag-class>com.example.MessageTag</tag-class>
14.  <body-content>scriptless</body-content>
15.  <attribute>
16.    <name>from</name>
17.    <rteprvalue>true</rteprvalue>
18.  </attribute>
19.  <attribute>
20.    <name>subject</name>
21.    <required>false</required>
22.    <rteprvalue>true</rteprvalue>
23.  </attribute>
24. </tag>
</taglib>
```

- A. <forum:message from="My Name" subject="My Subject" />
- B. <forum:message subject="My Subject">
My message body.

```
</forum:message>
C. <forum:message from="My Name" subject="${param.subject}">
${param.body}
</forum:message>
D. <forum:message from="My Name" subject="My Subject">
<%= request.getParameter( "body" ) %>
</forum:message>
E. <forum:message from="My Name"
subject="<%= request.getParameter( "subject" ) %>">
My message body.
</forum:message>
```

Answer: D

www.ExamWorx.com Q: 177 Which JSTL code snippet can be used to import content from another web resource?

- A. <c:import url="foo.jsp"/>
- B. <c:import page="foo.jsp"/>
- C. <c:include url="foo.jsp"/>
- D. <c:include page="foo.jsp"/>
- E. Importing cannot be done in JSTL. A standard action must be used instead.

Answer: A

www.ExamWorx.com Q: 178 Click the Exhibit button.

Assume the tag library in the exhibit is placed in a web application in the path /WEB-INF/tld/example.tld.

- 1.
2. <ex:hello />

Which JSP code, inserted at line 1, completes the JSP code to invoke the hello tag?

```

1. <?xml version="1.0" encoding="UTF-8" ?>
2.
3. <taglib
4.   xmlns="http://java.sun.com/xml/ns/j2ee"
5.   xmlns:xsi="http://www.w3.org/2001/XMLSchema-
6.     xsi:schemaLocation="http://java.sun.com/xml/
7.     ns/j2ee web-jsptaglibrary_2_0.xsd"
8.   version="2.0">
9.   <tlib-version>1.0</tlib-version>
10.  <short-name>ex</short-name>
11.
<uri>http://example.com/tld/example</uri>
12.  <tag>
13.    <name>hello</name>
14.    <tag-class>com.example.HelloTag</tag-class>
15.  </taglib>
```

- A. <%@ taglib prefix="ex" uri="/WEB-INF/tld" %>
- B. <%@ taglib uri="/WEB-INF/tld/example.tld" %>
- C. <%@ taglib prefix="ex"
uri="http://localhost:8080/tld/example.tld" %>
- D. <%@ taglib prefix="ex"
uri="http://example.com/tld/example" %>

Answer: D

www.ExamWorx.com Q: 179 Which JSTL code snippet produces the output "big number" when X is greater than 42, but outputs "small number" in all other cases?

- A. <c:if test='<%= (X > 42) %>'>
<c:then>big number</c:then>
<c:else>small number</c:else>
</c:if>
- B. <c:if>
<c:then test='<%= (X > 42) %>'>big number</c:then>

```

<c:else>small number</c:else>
</c:if>
C. <c:choose test='<%= (X > 42) %>'>
<c:then>big number</c:when>
<c:else>small number</c:otherwise>
</c:choose>
D. <c:choose test='<%= (X > 42) %>'>
<c:when>big number</c:when>
<c:otherwise>small number</c:otherwise>
</c:choose>
E. <c:choose>
<c:when test='<%= (X > 42) %>'>big number</c:when>
<c:otherwise>small number</c:otherwise>
</c:choose>

```

Answer: E

www.ExamWorx.com Q: 180 A developer chooses to avoid using SingleThreadModel but wants to ensure that data is updated in a thread-safe manner. Which two can support this design goal? (Choose two.)

- A. Store the data in a local variable.
- B. Store the data in an instance variable.
- C. Store the data in the HttpSession object.
- D. Store the data in the ServletContext object.
- E. Store the data in the ServletRequest object.

Answer: A, E

www.ExamWorx.com Q: 181 Your web application uses a simple architecture in which servlets handle requests and then forward to a JSP using a request dispatcher. You need to pass information calculated in the servlet to the JSP for view generation. This information must NOT be accessible to any other servlet, JSP or session in the webapp. Which two techniques can you use to accomplish this goal? (Choose two.)

- A. Add attributes to the session object.
- B. Add attributes on the request object.
- C. Add parameters to the request object.
- D. Use the pageContext object to add request attributes.
- E. Add parameters to the JSP's URL when generating the request dispatcher.

Answer: B, E

www.ExamWorx.com Q: 182 For which three events can web application event listeners be registered? (Choose three.)

- A. when a session is created
- B. after a servlet is destroyed
- C. when a session has timed out
- D. when a cookie has been created
- E. when a servlet has forwarded a request
- F. when a session attribute value is changed

Answer: A, C, F

www.ExamWorx.com Q: 183 Given:

String value = getServletContext().getInitParameter("foo");

in an HttpServlet and a web application deployment descriptor that contains:

```
<context-param>
  <param-name>foo</param-name>
  <param-value>frodo</param-value>
</context-param>
```

Which two are true? (Choose two.)

- A. The foo initialization parameter CANNOT be set programmatically.
- B. Compilation fails because getInitParameter returns type Object.
- C. The foo initialization parameter is NOT a servlet initialization parameter.
- D. Compilation fails because ServletContext does NOT have a getInitParameter method.
- E. The foo parameter must be defined within the <servlet> element of the deployment descriptor.
- F. The foo initialization parameter can also be retrieved using getServletConfig().getInitParameter("foo").

Answer: A, C

www.ExamWorx.com Q: 184 Click the Exhibit button. Given the web application deployment descriptor elements:

11. <filter>
12. <filter-name>ParamAdder</filter-name>
13. <filter-class>com.example.ParamAdder</filter-class>

14. </filter>
...
31. <filter-mapping>
32. <filter-name>ParamAdder</filter-name>
33. <servlet-name>Destination</servlet-name>
34. </filter-mapping>
...
55. <servlet-mapping>
56. <servlet-name>Destination</servlet-name>
57. <url-pattern>/dest/Destination</url-pattern>
58. </servlet-mapping>

What is the result of a client request of the Source servlet with no query string?

```

// Source Servlet : Source.java
10. public class Source extends HttpServlet {
11.     public void service(HttpServletRequest request,
12.                          HttpServletResponse response)
13.                             throws ServletException, IOException {
14.         RequestDispatcher rd =
15.             request.getRequestDispatcher("/dest/Destination");
16.         rd.forward(request, response);
17.     }
18. }

// Filter : ParamAdder.java
12. public class ParamAdder implements Filter {
13.     // ...
14.     public void doFilter(ServletRequest request,
15.                           HttpServletResponse response,
16.                           FilterChain chain)
17.         throws ServletException, IOException {
18.     request.setAttribute("filterAdded", "addedByFilter");
19.     chain.doFilter(request, response);
20. }
21.     // ...
22. }

// Destination Servlet Destination.java
10. public class Destination extends HttpServlet {
11.     public void service(HttpServletRequest request,
12.                          HttpServletResponse response)
13.                             throws ServletException, IOException {
14.         String filterParam =
15.             (String) request.getAttribute("filterAdded");
16.         response.getWriter().println("filterAdded = "
17.                                     + filterParam);
18.     }
19. }

```

- A. The output "filterAdded = null" is written to the response stream.
- B. The output "filterAdded = addedByFilter" is written to the response stream.
- C. An exception is thrown at runtime within the service method of the Source servlet.
- D. An exception is thrown at runtime within the service method of the Destination servlet.

Answer: A

www.ExamWorx.com Q: 185 Given a Filter class definition with this method:

```
21. public void doFilter(ServletRequest request,  
22.           ServletResponse response,  
23.           FilterChain chain)  
24.      throws ServletException, IOException {  
25. // insert code here  
26. }
```

Which should you insert at line 25 to properly invoke the next filter in the chain, or the target servlet if there are no more filters?

- A. chain.forward(request, response);
- B. chain.doFilter(request, response);
- C. request.forward(request, response);
- D. request.doFilter(request, response);

Answer: B

www.ExamWorx.com Q: 186 Servlet A forwarded a request to servlet B using the forward method of RequestDispatcher. What attribute in B's request object contains the URI of the original request received by servlet A?

- A. REQUEST_URI
- B. javax.servlet.forward.request_uri
- C. javax.servlet.forward.REQUEST_URI
- D. javax.servlet.request_dispatcher.request_uri
- E. javax.servlet.request_dispatcher.REQUEST_URI

Answer: B

www.ExamWorx.com Q: 187 You want to create a valid directory structure for your Java EE web application, and you want to put your web application into a WAR file called MyApp.war. Which two are true about the WAR file? (Choose two.)

- A. At deploy time, Java EE containers add a directory called META-INF directly into the MyApp directory.
- B. At deploy time, Java EE containers add a file called MANIFEST.MF directly into the MyApp directory.
- C. It can instruct your Java EE container to verify, at deploy time, whether you have properly configured your application's classes.
- D. At deploy time, Java EE containers add a directory call META-WAR directly into the MyApp directory.

Answer: A, C

www.ExamWorx.com Q: 188 Which two from the web application deployment descriptor are valid? (Choose two.)

- A. <error-page>
<exception-type>*</exception-type>
<location>/error.html</location>
</error-page>
- B. <error-page>
<exception-type>java.lang.Error</exception-type>
<location>/error.html</location>
</error-page>
- C. <error-page>
<exception-type>java.lang.Throwable</exception-type>
<location>/error.html</location>
</error-page>
- D. <error-page>
<exception-type>java.io.IOException</exception-type>
<location>/error.html</location>
</error-page>
- E. <error-page>
<exception-type>NullPointerException</exception-type>
<location>/error.html</location>
</error-page>

Answer: C, D

www.ExamWorx.com Q: 189 After a merger with another small business, your company has inherited a legacy WAR file but the original source files were lost. After reading the documentation of that web application, you discover that the WAR file contains a useful tag library that you want to reuse in your own webapp packaged as a WAR file.

What do you need to do to reuse this tag library?

- A. Simply rename the legacy WAR file as a JAR file and place it in your webapp's library directory.
- B. Unpack the legacy WAR file, move the TLD file to the META-INF directory, repackage the whole thing as a JAR file, and place that JAR file in your webapp's library directory.
- C. Unpack the legacy WAR file, move the TLD file to the META-INF directory, move the class files to the top-level directory, repackage the whole thing as a JAR file, and place that JAR file in your webapp's library directory.

- D. Unpack the legacy WAR file, move the TLD file to the META-INF directory, move the class files to the top-level directory, repackage the WAR, and place that WAR file in your webapp's WEB-INF directory.

Answer: C

www.ExamWorx.com Q: 190 Which path is required to be present within a WAR file?

- A. /classes
- B. /index.html
- C. /MANIFEST-INF
- D. /WEB-INF/web.xml
- E. /WEB-INF/classes
- F. /WEB-INF/index.html
- G. /META-INF/index.xml

Answer: D

www.ExamWorx.com Q: 191 Given:

- 11. <servlet>
- 12. <servlet-name>catalog</servlet-name>
- 13. <jsp-file>/catalogTemplate.jsp</jsp-file>
- 14. <load-on-startup>10</load-on-startup>
- 15. </servlet>

Which two are true? (Choose two.)

- A. Line 13 is not valid for a servlet declaration.
- B. Line 14 is not valid for a servlet declaration.
- C. One instance of the servlet will be loaded at startup.
- D. Ten instances of the servlet will be loaded at startup.
- E. The servlet will be referenced by the name catalog in mappings.

Answer: C, E

www.ExamWorx.com Q: 192 You have built a web application with tight security. Several directories of your webapp are used for internal purposes and you have overridden the default servlet to send an HTTP 403 status code for any request that maps to one of these directories. During testing, the Quality Assurance director decided that they did NOT like seeing the bare response page generated by Firefox and Internet Explorer. The director recommended that the webapp should return a more user-friendly web page that has the same look-and-feel as the webapp plus links to the webapp's search engine. You have created this JSP page in the /WEB-INF/jsp/error403.jsp file. You do NOT want to alter the complex logic of the default servlet. How can you declare that the web container must send this JSP page whenever a 403 status is generated?

A. <error-page>

```
<error-code>403</error-code>
<url>/WEB-INF/jsp/error403.jsp</url>
</error-page>
```

B. <error-page>

```
<status-code>403</status-code>
<url>/WEB-INF/jsp/error403.jsp</url>
</error-page>
```

C. <error-page>

```
<error-code>403</error-code>
<location>/WEB-INF/jsp/error403.jsp</location>
</error-page>
```

D. <error-page>

```
<status-code>403</status-code>
<location>/WEB-INF/jsp/error403.jsp</location>
</error-page>
```

Answer: C

www.ExamWorx.com Q: 193 Given a portion of a valid Java EE web application's directory structure:

```
MyApp
|
|-- Directory1
|   |-- File1.html
|
|-- META-INF
|   |-- File2.html
|
|-- WEB-INF
    |-- File3.html
```

You want to know whether File1.html, File2.html, and/or File3.html is protected from direct access by your web client's browsers.

What statement is true?

- A. All three files are directly accessible.
- B. Only File1.html is directly accessible.
- C. Only File2.html is directly accessible.
- D. Only File3.html is directly accessible.
- E. Only File1.html and File2.html are directly accessible.
- F. Only File1.html and File3.html are directly accessible.
- G. Only File2.html and File3.html are directly accessible.

Answer: B

www.ExamWorx.com Q: 194 A web component accesses a local EJB session bean with a component interface of com.example.Account with a home interface of com.example.AccountHome and a JNDI reference of ejb/Account. Which makes the local EJB component accessible to the web components in the web application deployment descriptor?

- A. <env-ref>
<ejb-ref-name>ejb/Account</ejb-ref-name>
<ejb-ref-type>Session</ejb-ref-type>
<local-home>com.example.AccountHome</local-home>
<local>com.example.Account</local>
</env-ref>
- B. <resource-ref>
<ejb-ref-name>ejb/Account</ejb-ref-name>
<ejb-ref-type>Session</ejb-ref-type>
<local-home>com.example.AccountHome</local-home>
<local>com.example.Account</local>
</resource-ref>
- C. <ejb-local-ref>
<ejb-ref-name>ejb/Account</ejb-ref-name>
<ejb-ref-type>Session</ejb-ref-type>
<local-home>com.example.AccountHome</local-home>
<local>com.example.Account</local>
</ejb-local-ref>
- D. <ejb-remote-ref>
<ejb-ref-name>ejb/Account</ejb-ref-name>
<ejb-ref-type>Session</ejb-ref-type>
<local-home>com.example.AccountHome</local-home>
<local>com.example.Account</local>
</ejb-remote-ref>

Answer: C

www.ExamWorx.com Q: 195 One of the use cases in your web application uses many session-scoped attributes. At the end of the use case, you want to clear out this set of attributes from the session object. Assume that this static variable holds this set of attribute names:

```
201. private static final Set<String> USE_CASE_ATTRS;  
202. static {  
203.     USE_CASE_ATTRS.add("customerOID");  
204.     USE_CASE_ATTRS.add("custMgrBean");  
205.     USE_CASE_ATTRS.add("orderOID");  
206.     USE_CASE_ATTRS.add("orderMgrBean");  
207. }
```

Which code snippet deletes these attributes from the session object?

- A. session.removeAll(USE_CASE_ATTRS);
- B. for (String attr : USE_CASE_ATTRS) {
session.remove(attr);
}
- C. for (String attr : USE_CASE_ATTRS) {
session.removeAttribute(attr);
}
- D. for (String attr : USE_CASE_ATTRS) {
session.deleteAttribute(attr);
}
- E. session.deleteAllAttributes(USE_CASE_ATTRS);

Answer: C

www.ExamWorx.com Q: 196 Given an HttpServletRequest request:

```
22. String id = request.getParameter("jsessionid");  
23. // insert code here  
24. String name = (String) session.getAttribute("name");
```

Which three can be placed at line 23 to retrieve an existing HttpSession object? (Choose three.)

- A. HttpSession session = request.getSession();
- B. HttpSession session = request.getSession(id);
- C. HttpSession session = request.getSession(true);
- D. HttpSession session = request.getSession(false);

E. HttpSession session = request.getSession("jsessionid");

Answer: A, C, D

www.ExamWorx.com Q: 197 A developer for the company web site has been told that users may turn off cookie support in their browsers. What must the developer do to ensure that these customers can still use the web application?

- A. The developer must ensure that every URL is properly encoded using the appropriate URL rewriting APIs.
- B. The developer must provide an alternate mechanism for managing sessions and abandon the HttpSession mechanism entirely.
- C. The developer can ignore this issue. Web containers are required to support automatic URL rewriting when cookies are not supported.
- D. The developer must add the string ?id=<sessionid> to the end of every URL to ensure that the conversation with the browser can continue.

Answer: A

www.ExamWorx.com Q: 198 You need to store a floating point number, called Tsquare, in the session scope. Which two code snippets allow you to retrieve this value? (Choose two.)

- A. float Tsquare = session.getFloatAttribute("Tsquare");
- B. float Tsquare = (Float) session.getAttribute("Tsquare");
- C. float Tsquare = (float) session.getNumericAttribute("Tsquare");
- D. float Tsquare = ((Float) session.getAttribute("Tsquare")).floatValue();
- E. float Tsquare = ((Float) session.getFloatAttribute("Tsquare")).floatValue();
- F. float Tsquare = ((Float) session.getNumericAttribute("Tsquare")).floatValue();

Answer: B, D

www.ExamWorx.com Q: 199 Given the definition of MyObject and that an instance of MyObject is bound as a session attribute:

```
8. package com.example;  
9. public class MyObject implements  
10.    javax.servlet.http.HttpSessionBindingListener {  
11.    // class body code here  
12. }
```

Which is true?

- A. Only a single instance of MyObject may exist within a session.
- B. The unbound method of the MyObject instance is called when the session to which it is bound times out.
- C. The com.example.MyObject must be declared as a servlet event listener in the web application deployment descriptor.
- D. The valueUnbound method of the MyObject instance is called when the session to which it is bound times out.

Answer: D

www.ExamWorx.com Q: 200 As a convenience feature, your web pages include an Ajax request every five minutes to a special servlet that monitors the age of the user's session. The client-side JavaScript that handles the Ajax callback displays a message on the screen as the session ages. The Ajax call does NOT pass any cookies, but it passes the session ID in a request parameter called sessionID. In addition, assume that your webapp keeps a hashmap of session objects by the ID. Here is a partial implementation of this servlet:

```

10. public class SessionAgeServlet extends HttpServlet {
11.   public void service(HttpServletRequest request, HttpServletResponse) throws IOException {
12.     String sessionID = request.getParameter("sessionID");
13.     HttpSession session = getSession(sessionID);
14.     long age = // your code here
15.     response.getWriter().print(age);
16.   }
... // more code here
47. }
```

Which code snippet on line 14, will determine the age of the session?

- A. session.getMaxInactiveInterval();
- B. session.getLastAccessed().getTime() - session.getCreationTime().getTime();
- C. session.getLastAccessedTime().getTime() - session.getCreationTime().getTime();
- D. session.getLastAccessed() - session.getCreationTime();
- E. session.getMaxInactiveInterval() - session.getCreationTime();
- F. session.getLastAccessedTime() - session.getCreationTime();

Answer: F

www.ExamWorx.com Q: 201 Which statement is true about web container session management?

- A. Access to session-scoped attributes is guaranteed to be thread-safe by the web container.
- B. To activate URL rewriting, the developer must use the HttpServletResponse.setURLRewriting method.

- C. If the web application uses HTTPS, then the web container may use the data on the HTTPS request stream to identify the client.
- D. The JSESSIONID cookie is stored permanently on the client so that a user may return to the web application and the web container will rejoin that session.

Answer: C

www.ExamWorx.com Q: 202 You are designing an n-tier Java EE application. You have already decided that some of your JSPs will need to get data from a Customer entity bean. You are trying to decide whether to use a Customer stub object or a Transfer Object. Which two statements are true? (Choose two.)

- A. The stub will increase network traffic.
- B. The Transfer Object will decrease data staleness.
- C. The stub will increase the logic necessary in the JSPs.
- D. In both cases, the JSPs can use EL expressions to get data.
- E. Only the Transfer Object will need to use a Business Delegate.
- F. Using the stub approach allows you to design the application without using a Service Locator.

Answer: A, D

www.ExamWorx.com Q: 203 Which two are characteristics of the Intercepting Filter pattern? (Choose two.)

- A. It provides centralized request handling for incoming requests.
- B. It forces resource authentication to be distributed across web components.
- C. It reduces coupling between presentation-tier clients and underlying business services.
- D. It can be added and removed unobtrusively, without requiring changes to existing code.
- E. It allows preprocessing and postprocessing on the incoming requests and outgoing responses.

Answer: D, E

www.ExamWorx.com Q: 204 A developer has created a web application that includes a servlet for each use case in the application. These servlets have become rather difficult to maintain because the request processing methods have become very large. There is also common processing code in many servlets because these use cases are very similar. Which two design patterns can be used together to refactor and simplify this web application? (Choose two.)

- A. Proxy
- B. View Helper

- C. Front Controller
- D. Session Facade
- E. Business Delegate
- F. Model-View-Controller

Answer: C, F

www.ExamWorx.com Q: 205 A developer is designing a multi-tier web application and discovers a need to hide the details of establishing and maintaining remote communications from the client. In addition, the application needs to find, in a transparent manner, the heterogeneous business components used to service the client's requests. Which design patterns, working together, address these issues?

- A. Business Delegate and Transfer Object
- B. Business Delegate and Service Locator
- C. Front Controller and Business Delegate
- D. Intercepting Filter and Transfer Object
- E. Model-View-Controller and Intercepting Filter

Answer: B

www.ExamWorx.com Q: 206 A developer is designing a web application that must support multiple interfaces, including:

an XML web service for B2B
HTML for web-based clients
WML for wireless customers

Which design pattern provides a solution for this problem?

- A. Session Facade
- B. Business Delegate
- C. Data Access Object
- D. Model-View-Controller
- E. Chain of Responsibility

Answer: D

www.ExamWorx.com Q: 207 A developer is designing a web application which extensively uses EJBs and JMS. The developer finds that there is a lot of duplicated code to build the JNDI contexts to access the beans and queues. Further, because of the complexity, there are numerous errors in the code. Which J2EE design pattern provides a solution for this problem?

- A. Command
- B. Transfer Object
- C. Service Locator
- D. Session Facade
- E. Business Delegate
- F. Data Access Object

Answer: C

www.ExamWorx.com Q: 208 A developer is designing a web application that must support multiple interfaces, including:

- an XML web service for B2B
- HTML for web-based clients
- WML for wireless customers

Which design pattern provides a solution for this problem?

- A. Session Facade
- B. Business Delegate
- C. Data Access Object
- D. Model-View-Controller
- E. Chain of Responsibility

Answer: D

www.ExamWorx.com Q: 209 Which two are characteristics of the Front Controller pattern? (Choose two.)

- A. It simplifies remote interfaces to distributed objects.
- B. It promotes cleaner application partitioning and encourages reuse.
- C. It provides an initial point of contact for handling all related requests.
- D. It reduces maintainability due to the increased complexity of the design.
- E. It provides loosely coupled handlers that can be combined in various permutations.

Answer: B, C

www.ExamWorx.com Q: 210 Squeaky Beans Inc. hired an outside consultant to develop their web application. To finish the job quickly, the consultant created several dozen JSP pages that directly communicate with the database. The Squeaky business team has since purchased a set of business objects to model their system, and the Squeaky developer charged with maintaining the web application must now refactor all the JSPs to work with the new system. Which pattern can the developer use to solve this problem?

- A. Transfer Object
- B. Service Locator
- C. Intercepting Filter
- D. Business Delegate

Answer: D

www.ExamWorx.com Q: 211 A developer is designing the presentation tier for a web application that relies on a complex session bean. The session bean is still being developed and the APIs for it are NOT finalized. Any changes to the session bean API directly impacts the development of the presentation tier. Which design pattern provides a means to manage the uncertainty in the API?

- A. View Helper
- B. Front Controller
- C. Composite View
- D. Intercepting Filter
- E. Business Delegate
- F. Chain of Responsibility

Answer: E

www.ExamWorx.com Q: 212 A developer is designing a multi-tier web application and discovers a need to log each incoming client request. Which two patterns, taken independently, provide a solution for this problem? (Choose two.)

- A. Transfer Object
- B. Service Locator
- C. Front Controller
- D. Intercepting Filter
- E. Business Delegate
- F. Model-View-Controller

Answer: C, D

www.ExamWorx.com Q: 213 A developer is designing a multi-tier web application and discovers a need to hide the details of establishing and maintaining remote communications from the client. In addition, because the business and resource tiers are distributed, the application needs to minimize the inter-tier network traffic related to servicing client requests. Which design patterns, working together, address these issues?

- A. Front Controller and Transfer Object
- B. Front Controller and Service Locator
- C. Business Delegate and Transfer Object
- D. Business Delegate and Intercepting Filter
- E. Model-View-Controller and Intercepting Filter

Answer: C

www.ExamWorx.com Q: 214 Click the Task button.

Place the servlet name onto every request URL, relative to the web application context root, that will invoke that servlet. Every request URL must be filled.

Given the servlets and their path patterns:

Servlet Name	Path Pattern
ControlServlet	*.do
DataServlet	/data/*

Place the servlet name onto every request URL, relative to the web application context root, that will invoke that servlet. Every request URL must be filled.

Request URL	Servlet Name
/data/	ControlServlet
/data/index.jsp	DataServlet
/secure/command.do	
/data/command.do	
/data.do	Done

Answer: Check ExamWorx eEngine, Download from Member Center

www.ExamWorx.com Q: 215 Given a portion of a valid Java EE web application's directory structure:

```
MyApp
|
|-- File1.html
|
|-- Directory1
|   |-- File2.html |
|-- META-INF
|   |-- File3.html
```

You want to know whether File1.html, File2.html, and/or File3.html will be directly accessible by your web client's browsers.

Which statement is true?

- A. All three files are directly accessible.
- B. Only File1.html is directly accessible.
- C. Only File2.html is directly accessible.
- D. Only File3.html is directly accessible.
- E. Only File1.html and File2.html are directly accessible.
- F. Only File1.html and File3.html are directly accessible.
- G. Only File2.html and File3.html are directly accessible.

Answer: E

www.ExamWorx.com Q: 216 Which three are described in the standard web application deployment descriptor? (Choose three.)

- A. session configuration
- B. MIME type mappings
- C. context root for the application
- D. servlet instance pool configuration
- E. web container default port bindings
- F. ServletContext initialization parameters

Answer: A, B, F

www.ExamWorx.com Q: 217 You have created a servlet that generates weather maps. The data for these maps is calculated by a remote host. The IP address of this host is usually stable, but occasionally does have to change as the corporate network grows and changes. This IP address used to be hard coded, but after the fifth change to the IP address in two years, you have decided that this value should be declared in the deployment descriptor so you do NOT have the recompile the web application every time the IP address changes. Which deployment descriptor snippet accomplishes this goal?

- A. <serlvet-param>
<name>WeatherServlet.hostIP</name>
<value>127.0.4.20</value>
</serlvet-param>
- B. <init-param>
<name>WeatherServlet.hostIP</name>
<value>127.0.4.20</value>

```
</init-param>
C. <servlet>
<!-- servlet definition here -->
<param-name>WeatherServlet.hostIP</param-name>
<param-value>127.0.4.20</param-value>
</servlet>
D. <init-param>
<param-name>WeatherServlet.hostIP</param-name>
<param-value>127.0.4.20</param-value>
</init-param>
E. <serlvet-param>
<param-name>WeatherServlet.hostIP</param-name>
<param-value>127.0.4.20</param-value>
</servlet-param>
```

Answer: D

www.ExamWorx.com Q: 218 In which two locations can library dependencies be defined for a web application? (Choose two.)

- A. the web application deployment descriptor
- B. the /META-INF/dependencies.xml file
- C. the /META-INF/MANIFEST.MF manifest file
- D. the /META-INF/MANIFEST.MF manifest of a JAR in the web application classpath

Answer: C, D

www.ExamWorx.com Q: 219 Which two about WAR files are true? (Choose two.)

- A. WAR files must be located in the web application library directory.
- B. WAR files must contain the web application deployment descriptor.
- C. WAR files must be created by using archive tools designed specifically for that purpose.
- D. The web container must serve the content of any META-INF directory located in a WAR file.
- E. The web container must allow access to resources in JARs in the web application library directory.

Answer: B, E

www.ExamWorx.com Q: 220 Given this fragment from a Java EE deployment descriptor:

341. <error-page>
342. <exception-type>java.lang.Throwable</exception-type>

343. <location>/mainError.jsp</location>
344. </error-page>
345. <error-page>
346. <exception-type>java.lang.ClassCastException</exception-type>
347. <location>/castError.jsp</location>
348. </error-page>

If the web application associated with the fragment above throws a ClassCastException.

Which statement is true?

- A. The deployment descriptor is invalid.
- B. The container invokes mainError.jsp.
- C. The container invokes castError.jsp.
- D. Neither mainError.jsp nor castError.jsp is invoked.

Answer: C

www.ExamWorx.com Q: 221 Which defines the welcome files in a web application deployment descriptor?

- A. <welcome>
<welcome-file>/welcome.jsp</welcome-file>
</welcome>
<welcome>
<welcome-file>/index.html</welcome-file>
</welcome>
- B. <welcome-file-list>
<welcome-file>welcome.jsp</welcome-file>
<welcome-file>index.html</welcome-file>
</welcome-file-list>
- C. <welcome>
<welcome-file>welcome.jsp</welcome-file>
</welcome>
<welcome>
<welcome-file>/index.html</welcome-file>
</welcome>
- D. <welcome-file-list>
<welcome-file>/welcome.jsp</welcome-file>
<welcome-file>/index.html</welcome-file>
</welcome-file-list>
- E. <welcome>
<welcome-file>

```
<welcome-name>Welcome</welcome-name>
<location>welcome.jsp</location>
</welcome-file>
<welcome-file>
<welcome-name>Index</welcome-name>
<location>index.html</location>
</welcome-file>
</welcome>
```

Answer: B

www.ExamWorx.com Q: 222 Click the Exhibit button.

Assume the product attribute does NOT yet exist in any scope.

Which two create an instance of com.example.Product and initialize the name and price properties to the name and price request parameters? (Choose two.)

```
1. package com.example;
2.
3. public class Product {
4.     private String name;
5.     private double price;
6.
7.     public Product() {
8.         this( "Default", 0.0 );
9.     }
10.
11.    public Product( String name, double
price ) {
12.        this.name = name;
13.        this.price = price;
14.    }
15.
16.    public String getName() {
17.        return name;
18.    }
19.
20.    public void setName(String name) {
21.        this.name = name;
22.    }
23.
24.    public double getPrice() {
25.        return price;
26.    }
27.
28.    public void setPrice(double price) {
29.        this.price = price;
30.    }
31. }
```

- A. <jsp:useBean id="product" class="com.example.Product" />
<jsp:setProperty name="product" property="*" />
- B. <jsp:useBean id="product" class="com.example.Product" />
<% product.setName(request.getParameter("name")); %>
<% product.setPrice(request.getParameter("price")); %>
- C. <jsp:useBean id="product" class="com.example.Product" />
<jsp:setProperty name="product" property="name"
value="\${param.name}" />
<jsp:setProperty name="product" property="price"

value="\${param.price}" />
D. <jsp:useBean id="product" class="com.example.Product">
<jsp:setProperty name="product" property="name"
value="\${name}" />
<jsp:setProperty name="product" property="price"
value="\${price}" />
</jsp:useBean>

Answer: A, C

www.ExamWorx.com Q: 223 Click the Exhibit button.

A session-scoped attribute, product, is stored by a servlet. That servlet then forwards to a JSP page. This attribute holds an instance of the com.example.Product class with a name property of "The Matrix" and price property of 39.95.

Given the JSP page code snippet:

1. <jsp:useBean id='product' class='com.example.Product'>
2. <jsp:setProperty name='product' property='price' value='49.95' />
3. </jsp:useBean>
4. <%= product.getName() %> costs <%= product.getPrice() %>

What is the response output of this JSP page code snippet?

```
1. package com.example;
2.
3. public class Product {
4.     private String name;
5.     private double price;
6.
7.     public Product() {
8.         this( "Default", 0.0 );
9.     }
10.
11.    public Product( String name, double
price ) {
12.        this.name = name;
13.        this.price = price;
14.    }
15.
16.    public String getName() {
17.        return name;
18.    }
19.
20.    public void setName(String name) {
21.        this.name = name;
22.    }
23.
24.    public double getPrice() {
25.        return price;
26.    }
27.
28.    public void setPrice(double price) {
29.        this.price = price;
30.    }
31. }
```

- A. Default costs 0.0
- B. Default costs 49.95
- C. Default costs 39.95
- D. The Matrix costs 0.0
- E. The Matrix costs 49.95
- F. The Matrix costs 39.95

Answer: B

www.ExamWorx.com Q: 224 Click the Exhibit button.

A servlet sets a session-scoped attribute product with an instance of com.example.Product and forwards to a JSP.

Which two output the name of the product in the response? (Choose two.)

```
1. package com.example;
2.
3. public class Product {
4.     private String name;
5.     private double price;
6.
7.     public Product() {
8.         this( "Default", 0.0 );
9.     }
10.
11.    public Product( String name, double
price ) {
12.        this.name = name;
13.        this.price = price;
14.    }
15.
16.    public String getName() {
17.        return name;
18.    }
19.
20.    public void setName(String name) {
21.        this.name = name;
22.    }
23.
24.    public double getPrice() {
25.        return price;
26.    }
27.
28.    public void setPrice(double price) {
29.        this.price = price;
30.    }
31. }
```

-
- A. \${product.name}

- B. <jsp:getProperty name="product" property="name" />
- C. <jsp:useBean id="com.example.Product" />
<%= product.getName() %>
- D. <jsp:getProperty name="product" class="com.example.Product"
property="name" />
- E. <jsp:useBean id="product" type="com.example.Product">
<%= product.getName() %>
</jsp:useBean>

Answer: A, B

www.ExamWorx.com Q: 225 You have built your own light-weight templating mechanism. Your servlets, which handle each request, dispatch the request to one of a small set of template JSP pages. Each template JSP controls the layout of the view by inserting the header, body, and footer elements into specific locations within the template page. The URLs for these three elements are stored in request-scoped variables called, headerURL, bodyURL, and footerURL, respectively. These attribute names are never used for other purposes. Which JSP code snippet should be used in the template JSP to insert the JSP content for the body of the page?

- A. <jsp:insert page='\${bodyURL}' />
- B. <jsp:insert file='\${bodyURL}' />
- C. <jsp:include page='\${bodyURL}' />
- D. <jsp:include file='\${bodyURL}' />
- E. <jsp:insert page='<%= bodyURL %>' />
- F. <jsp:include page='<%= bodyURL %>' />

Answer: C

www.ExamWorx.com Q: 226 Given:

```
3. public class MyTagHandler extends TagSupport {  
4.   public int doStartTag() {  
5.     // insert code here  
6.     // return an int  
7.   }  
8.   // more code here  
...  
18. }
```

There is a single attribute foo in the session scope.

Which three code fragments, inserted independently at line 5, return the value of the attribute? (Choose three.)

- A. Object o = pageContext.getAttribute("foo");
- B. Object o = pageContext.findAttribute("foo");
- C. Object o = pageContext.getAttribute("foo",
PageContext.SESSION_SCOPE);
- D. HttpSession s = pageContext.getSession();
Object o = s.getAttribute("foo");
- E. HttpServletRequest r = pageContext.getRequest();
Object o = r.getAttribute("foo");

Answer: B, C, D

www.ExamWorx.com Q: 227 You are creating a content management system (CMS) with a web application front-end. The JSP that displays a given document in the CMS has the following general structure:

1. <%-- tag declaration --%>
2. <t:document>
- ...
11. <t:paragraph>... <t:citation docID='xyz' /> ...</t:paragraph>
- ...
99. </t:document>

The citation tag must store information in the document tag for the document tag to generate a reference section at the end of the generated web page.

The document tag handler follows the Classic tag model and the citation tag handler follows the Simple tag model. Furthermore, the citation tag could also be embedded in other custom tags that could have either the Classic or Simple tag handler model.

Which tag handler method allows the citation tag to access the document tag?

- A. public void doTag() {
JspTag docTag = findAncestorWithClass(this, DocumentTag.class);
((DocumentTag)docTag).addCitation(this.docID);
}
- B. public void doStartTag() {
JspTag docTag = findAncestorWithClass(this, DocumentTag.class);
((DocumentTag)docTag).addCitation(this.docID);
}
- C. public void doTag() {
Tag docTag = findAncestor(this, DocumentTag.class);
((DocumentTag)docTag).addCitation(this.docID);

```
}

D. public void doStartTag() {
    Tag docTag = findAncestor(this, DocumentTag.class);
    ((DocumentTag)docTag).addCitation(this.docID);
}
```

Answer: A

www.ExamWorx.com Q: 228 Given:

6. <myTag:foo bar='42'>
7. <%="processing" %>
8. </myTag:foo>

and a custom tag handler for foo which extends TagSupport.

Which two are true about the tag handler referenced by foo? (Choose two.)

- A. The doStartTag method is called once.
- B. The doAfterBody method is NOT called.
- C. The EVAL_PAGE constant is a valid return value for the doEndTag method.
- D. The SKIP_PAGE constant is a valid return value for the doStartTag method.
- E. The EVAL_BODY_BUFFERED constant is a valid return value for the doStartTag method.

Answer: A, C

www.ExamWorx.com Q: 229 Which two are true concerning the objects available to developers creating tag files? (Choose two.)

- A. The session object must be declared explicitly.
- B. The request and response objects are available implicitly.
- C. The output stream is available through the implicit outStream object.
- D. The servlet context is available through the implicit servletContext object.
- E. The JspContext for the tag file is available through the implicit jspContext object.

Answer: B, E

www.ExamWorx.com Q: 230 Your web application uses a lot of Java enumerated types in the domain model of the application. Built into each enum type is a method, `getDisplay()`, which returns a localized, user-oriented string. There are many uses for presenting enums within the web application, so your manager has asked you to create a custom tag that iterates over the set of enum values and processes the body of the tag once for each value; setting the value into a page-scoped attribute called, `enumValue`. Here is an example of how this tag is used:

```
10. <select name='season'>
11. <t:everyEnum type='com.example.Season'>
12.   <option value='${enumValue}'>${enumValue.display}</option>
13. </t:everyEnum>
14. </select>
```

You have decided to use the Simple tag model to create this tag handler.

Which tag handler method will accomplish this goal?

- A.

```
public void doTag() throw JspException {
try {
for ( Enum value : getEnumValues() ) {
pageContext.setAttribute("enumValue", value);
getJspBody().invoke(getOut());
}
} (Exception e) { throw new JspException(e); }
}
```
- B.

```
public void doTag() throw JspException {
try {
for ( Enum value : getEnumValues() ) {
getJspContext().setAttribute("enumValue", value);
getJspBody().invoke(null);
}
} (Exception e) { throw new JspException(e); }
}
```
- C.

```
public void doTag() throw JspException {
try {
for ( Enum value : getEnumValues() ) {
getJspContext().setAttribute("enumValue", value);
getJspBody().invoke(getJspContext().getWriter());
}
} (Exception e) { throw new JspException(e); }
}
```
- D.

```
public void doTag() throw JspException {
try {
for ( Enum value : getEnumValues() ) {
pageContext.setAttribute("enumValue", value);

```

```
getJspBody().invoke(getJspContext().getWriter());
}
} (Exception e) { throw new JspException(e); }
}
```

Answer: B

www.ExamWorx.com Q: 231 Which two statements are true about the security-related tags in a valid Java EE deployment descriptor? (Choose two.)

- A. Every <security-constraint> tag must have at least one <http-method> tag.
- B. A <security-constraint> tag can have many <web-resource-collection> tags.
- C. A given <auth-constraint> tag can apply to only one <web-resource-collection> tag.
- D. A given <web-resource-collection> tag can contain from zero to many <url-pattern> tags.
- E. It is possible to construct a valid <security-constraint> tag such that, for a given resource, no user roles can access that resource.

Answer: B, E

www.ExamWorx.com Q: 232 Which element of a web application deployment descriptor <security-constraint> element is required?

- A. <realm-name>
- B. <auth-method>
- C. <security-role>
- D. <transport-guarantee>
- E. <web-resource-collection>

Answer: E

www.ExamWorx.com Q: 233 Which two are required elements for the <web-resource-collection> element of a web application deployment descriptor? (Choose two.)

- A. <realm-name>
- B. <url-pattern>
- C. <description>
- D. <web-resource-name>
- E. <transport-guarantee>

Answer: B, D

www.ExamWorx.com Q: 234 Given:

```
3. class MyServlet extends HttpServlet {  
4.     public void doPut(HttpServletRequest req,  
                         HttpServletResponse resp)  
             throws ServletException, IOException {  
5.         // servlet code here  
...  
26.    }  
27. }
```

If the DD contains a single security constraint associated with MyServlet and its only <http-method> tags and <auth-constraint> tags are:

```
<http-method>GET</http-method>  
<http-method>PUT</http-method>  
<auth-constraint>Admin</auth-constraint>
```

Which four requests would be allowed by the container? (Choose four.)

- A. A user whose role is Admin can perform a PUT.
- B. A user whose role is Admin can perform a GET.
- C. A user whose role is Admin can perform a POST.
- D. A user whose role is Member can perform a PUT.
- E. A user whose role is Member can perform a POST.
- F. A user whose role is Member can perform a GET.

Answer: A, B, C, E

www.ExamWorx.com Q: 235 What is true about Java EE authentication mechanisms?

- A. If your deployment descriptor correctly declares an authentication type of CLIENT_CERT, your users must have a certificate from an official source before they can use your application.
- B. If your deployment descriptor correctly declares an authentication type of BASIC, the container automatically requests a user name and password whenever a user starts a new session.
- C. If you want your web application to support the widest possible array of browsers, and you want to perform authentication, the best choice of Java EE authentication mechanisms is DIGEST.
- D. To use Java EE FORM authentication, you must declare two HTML files in your deployment descriptor, and you must use a predefined action in the HTML file that handles your user's login.

Answer: D

www.ExamWorx.com Q: 236 Which two statements are true about using the isUserInRole method to implement security in a Java EE application? (Choose two.)

- A. It can be invoked only from the doGet or doPost methods.
- B. It can be used independently of the getRemoteUser method.
- C. Can return "true" even when its argument is NOT defined as a valid role name in the deployment descriptor.
- D. Using the isUserInRole method overrides any declarative authentication related to the method in which it is invoked.
- E. Using the isUserInRole method overrides any declarative authorization related to the method in which it is invoked.

Answer: B, C

www.ExamWorx.com Q: 237 Given an HttpServletRequest request and an HttpServletResponse response:

```
41. HttpSession session = null;  
42. // insert code here  
43. if(session == null) {  
44.   // do something if session does not exist  
45. } else {  
46.   // do something if session exists  
47. }
```

To implement the design intent, which statement must be inserted at line 42?

- A. session = response.getSession();
- B. session = request.getSession();
- C. session = request.getSession(true);
- D. session = request.getSession(false);
- E. session = request.getSession("jsessionid");

Answer: D

www.ExamWorx.com Q: 238 You need to store a floating point number, called Tsquare, in the session scope. Which two code snippets allow you to retrieve this value? (Choose two.)

- A. float Tsquare = session.getFloatAttribute("Tsquare");
- B. float Tsquare = (Float) session.getAttribute("Tsquare");

- C. float Tsquare = (float) session.getAttribute("Tsquare");
- D. float Tsquare = ((Float) session.getAttribute("Tsquare")).floatValue();
- E. float Tsquare = ((Float) session.getFloatAttribute("Tsquare")).floatValue();
- F. float Tsquare = ((Float) session.getAttribute("Tsquare")).floatValue();

Answer: B, D

www.ExamWorx.com Q: 239 A web application uses the HttpSession mechanism to determine if a user is "logged in." When a user supplies a valid user name and password, an HttpSession is created for that user.

The user has access to the application for only 15 minutes after logging in. The code must determine how long the user has been logged in, and if this time is greater than 15 minutes, must destroy the HttpSession.

Which method in HttpSession is used to accomplish this?

- A. getCreationTime
- B. invalidateAfter
- C. getLastAccessedTime
- D. getMaxInactiveInterval

Answer: A

www.ExamWorx.com Q: 240 Which method must be used to encode a URL passed as an argument to HttpServletResponse.sendRedirect when using URL rewriting for session tracking?

- A. ServletResponse.encodeURL
- B. HttpServletResponse.encodeURL
- C. ServletResponse.encodeRedirectURL
- D. HttpServletResponse.encodeRedirectURL

Answer: D

www.ExamWorx.com Q: 241 Which interface must a session attribute implement if it needs to be notified when a web container persists a session?

- A. javax.servlet.http.HttpSessionListener
- B. javax.servlet.http.HttpSessionBindingListener
- C. javax.servlet.http.HttpSessionAttributeListener

D. javax.servlet.http.HttpSessionActivationListener

Answer: D

www.ExamWorx.com Q: 242 What is the purpose of session management?

- A. To manage the user's login and logout activities.
- B. To store information on the client-side between HTTP requests.
- C. To store information on the server-side between HTTP requests.
- D. To tell the web container to keep the HTTP connection alive so it can make subsequent requests without the delay of making the TCP connection.

Answer: C

www.ExamWorx.com Q: 243 You need to store a Java long primitive attribute, called customerOID, into the session scope. Which two code snippets allow you to insert this value into the session? (Choose two.)

- A. long customerOID = 47L;
session.setAttribute("customerOID", new Long(customerOID));
- B. long customerOID = 47L;
session.setLongAttribute("customerOID", new Long(customerOID));
- C. long customerOID = 47L;
session.setAttribute("customerOID", customerOID);
- D. long customerOID = 47L;
session.setNumericAttribute("customerOID", new Long(customerOID));
- E. long customerOID = 47L;
session.setLongAttribute("customerOID", customerOID);
- F. long customerOID = 47L;
session.setNumericAttribute("customerOID", customerOID);

Answer: A, C

www.ExamWorx.com Q: 244 Click the Exhibit button.

Assuming the tag library in the exhibit is imported with the prefix forum, which custom tag invocation produces a translation error in a JSP page?

```

1. <?xml version="1.0" encoding="UTF-8" ?>
2.
3. <taglib
4.   xmlns="http://java.sun.com/xml/ns/j2ee"
5.   xmlns:xsi="http://www.w3.org/2001/XMLSchema-
a-instance"
6.   xsi:schemaLocation="http://java.sun.com/xm-
l/ns/j2ee web-jsptaglibrary_2_0.xsd"
7.   version="2.0">
8.   <tlib-version>1.0</tlib-version>
9.   <short-name>forum</short-name>
10.  <uri>http://example.com/tld/forum</uri>
11.  <tag>
12.    <name>message</name>
13.    <tag-class>com.example.MessageTag</tag-class>
14.    <attribute>
15.      <name>from</name>
16.      <rteprvalue>true</rteprvalue>
17.    </attribute>
18.    <attribute>
19.      <name>subject</name>
20.      <required>false</required>
21.      <rteprvalue>true</rteprvalue>
22.    </attribute>
23.  </tag>
24. </taglib>

```

- A. <forum:message from="My Name" subject="My Subject" />
- B. <forum:message subject="My Subject">
My message body.
</forum:message>
- C. <forum:message from="My Name" subject="\${param.subject}">
\${param.body}
</forum:message>
- D. <forum:message from="My Name" subject="My Subject">
<%= request.getParameter("body") %>
</forum:message>
- E. <forum:message from="My Name"
subject=<%= request.getParameter("subject") %>">

My message body.
</forum:message>

Answer: D

www.ExamWorx.com Q: 245 Given in a single JSP page:

```
<%@ taglib prefix='java' uri='myTags' %>
<%@ taglib prefix='JAVA' uri='moreTags' %>
```

Which two are true? (Choose two.)

- A. The prefix 'java' is reserved.
- B. The URI 'myTags' must be properly mapped to a TLD file by the web container.
- C. A translation error occurs because the prefix is considered identical by the web container.
- D. For the tag usage <java:tag1/>, the tag1 must be unique in the union of tag names in 'myTags' and 'moreTags'.

Answer: A, B

www.ExamWorx.com Q: 246 In a JSP-centric shopping cart application, you need to move a client's home address of the Customer object into the shipping address of the Order object. The address data is stored in a value object class called Address with properties for: street address, city, province, country, and postal code. Which two JSP code snippets can be used to accomplish this goal? (Choose two.)

- A. <c:set var='order' property='shipAddress'
value='\${client.homeAddress}' />
- B. <c:set target='\${order}' property='shipAddress'
value='\${client.homeAddress}' />
- C. <jsp:setProperty name='\${order}' property='shipAddress'
value='\${client.homeAddress}' />
- D. <c:set var='order' property='shipAddress'>
<jsp:getProperty name='client' property='homeAddress' />
</c:store>
- E. <c:set target='\${order}' property='shipAddress'>
<jsp:getProperty name='client' property='homeAddress' />
</c:set>
- F. <c:setProperty name='\${order}' property='shipAddress'>
<jsp:getProperty name='client' property='homeAddress' />
</c:setProperty>

Answer: B, E

www.ExamWorx.com Q: 247 Given that a scoped attribute cart exists only in a user's session, which two, taken independently, ensure the scoped attribute cart no longer exists? (Choose two.)

- A. \${cart = null}
- B. <c:remove var="cart" />
- C. <c:remove var="\${cart}" />
- D. <c:remove var="cart" scope="session" />
- E. <c:remove scope="session">cart</c:remove>
- F. <c:remove var="\${cart}" scope="session" />
- G. <c:remove scope="session">\${cart}</c:remove>

Answer: B, D

www.ExamWorx.com Q: 248 In which three directories, relative to a web application's root, may a tag library descriptor file reside when deployed directly into a web application? (Choose three.)

- A. /WEB-INF
- B. /META-INF
- C. /WEB-INF/tlds
- D. /META-INF/tlds
- E. /WEB-INF/resources
- F. /META-INF/resources

Answer: A, C, E

www.ExamWorx.com Q: 249 You have been contracted to create a web site for a free dating service. One feature is the ability for one client to send a message to another client, which is displayed in the latter client's private page. Your contract explicitly states that security is a high priority. Therefore, you need to prevent cross-site hacking in which one user inserts JavaScript code that is then rendered and invoked when another user views that content. Which two JSTL code snippets will prevent cross-site hacking in the scenario above? (Choose two.)

- A. <c:out>\${message}</c:out>
- B. <c:out value='\${message}' />
- C. <c:out value='\${message}' escapeXml='true' />
- D. <c:out eliminateXml='true'>\${message}</c:out>
- E. <c:out value='\${message}' eliminateXml='true' />

Answer: B, C

www.ExamWorx.com Q: 250 A custom tag is defined to take three attributes. Which two correctly invoke the tag within a JSP page? (Choose two.)

- A. <prefix:myTag a="foo" b="bar" c="baz" />
- B. <prefix:myTag attributes={"foo","bar","baz"} />
- C. <prefix:myTag jsp:attribute a="foo" b="bar" c="baz" />
- D. <prefix:myTag>
<jsp:attribute a:foo b:bar c:baz />
</prefix:myTag>
- E. <prefix:myTag>
<jsp:attribute \${"foo", "bar", "baz"} />
</prefix:myTag>
- F. <prefix:myTag>
<jsp:attribute a="foo" b="bar" c="baz"/>
</prefix:myTag>
- G. <prefix:myTag>
<jsp:attribute name="a">foo</jsp:attribute>
<jsp:attribute name="b">bar</jsp:attribute>
<jsp:attribute name="c">baz</jsp:attribute>
</prefix:myTag>

Answer: A, G

www.ExamWorx.com Q: 251 Which two are true about the JSTL core iteration custom tags? (Choose two.)

- A. It may iterate over arrays, collections, maps, and strings.
- B. The body of the tag may contain EL code, but not scripting code.
- C. When looping over collections, a loop status object may be used in the tag body.
- D. It may iterate over a map, but only the key of the mapping may be used in the tag body.
- E. When looping over integers (for example begin='1' end='10'), a loop status object may not be used in the tag body.

Answer: A, C

www.ExamWorx.com Q: 252 Which two are valid and equivalent? (Choose two.)

- A. <%! int i; %>
- B. <%= int i; %>

- C. <jsp:expr>int i;</jsp:expr>
- D. <jsp:scriptlet>int i;</jsp:scriptlet>
- E. <jsp:declaration>int i;</jsp:declaration>

Answer: A, E

www.ExamWorx.com Q: 253 The JSP developer wants a comment to be visible in the final output to the browser. Which comment style needs to be used in a JSP page?

- A. <!-- this is a comment -->
- B. <% // this is a comment %>
- C. <%-- this is a comment --%>
- D. <% /* this is a comment */ %>

Answer: A

www.ExamWorx.com Q: 254 Which is a benefit of precompiling a JSP page?

- A. It avoids initialization on the first request.
- B. It provides the ability to debug runtime errors in the application.
- C. It provides better performance on the first request for the JSP page.
- D. It avoids execution of the _jspService method on the first request.

Answer: C

www.ExamWorx.com Q: 255 Given tutorial.jsp:

2. <h1>EL Tutorial</h1>
3. <h2>Example 1</h2>
4. <p>
5. Dear \${my:nickname(user)}
6. </p>

Which, when added to the web application deployment descriptor, ensures that line 5 is included verbatim in the JSP output?

- A. <jsp-config>
<url-pattern>*.jsp</url-pattern>
<el-ignored>true</el-ignored>
</jsp-config>

B. <jsp-config>
<url-pattern>*.jsp</url-pattern>
<isELIgnored>true</isELIgnored>
</jsp-config>
C. <jsp-config>
<jsp-property-group>
<el-ignored>*.jsp</el-ignored>
</jsp-property-group>
</jsp-config>
D. <jsp-config>
<jsp-property-group>
<url-pattern>*.jsp</url-pattern>
<el-ignored>true</el-ignored>
</jsp-property-group>
</jsp-config>
E. <jsp-config>
<jsp-property-group>
<url-pattern>*.jsp</url-pattern>
<isELIgnored>true</isELIgnored>
</jsp-property-group>
</jsp-config>

Answer: D

www.ExamWorx.com Q: 256 In a JSP-centric web application, you need to create a catalog browsing JSP page. The catalog is stored as a List object in the catalog attribute of the webapp's ServletContext object. Which scriptlet code snippet gives you access to the catalog object?

- A. <% List catalog = config.getAttribute("catalog"); %>
- B. <% List catalog = context.getAttribute("catalog"); %>
- C. <% List catalog = application.getAttribute("catalog"); %>
- D. <% List catalog = servletContext.getAttribute("catalog"); %>

Answer: C

www.ExamWorx.com Q: 257 You are building your own layout mechanism by including dynamic content for the page's header and footer sections. The footer is always static, but the header generates the <title> tag that requires the page name to be specified dynamically when the header is imported. Which JSP code snippet performs the import of the header content?

- A. <jsp:include page='/WEB-INF/jsp/header.jsp'>
<jsp:param name='pageName' value='Welcome Page' />

```
</jsp:include>
B. <jsp:import page='/WEB-INF/jsp/header.jsp'>
<jsp:param name='pageName' value='Welcome Page' />
</jsp:import>
C. <jsp:include page='/WEB-INF/jsp/header.jsp'>
<jsp:attribute name='pageName' value='Welcome Page' />
</jsp:include>
D. <jsp:import page='/WEB-INF/jsp/header.jsp'>
<jsp:attribute name='pageName' value='Welcome Page' />
</jsp:import>
```

Answer: A

www.ExamWorx.com Q: 258 Which ensures that a JSP response is of type "text/plain"?

- A. <%@ page mimeType="text/plain" %>
- B. <%@ page contentType="text/plain" %>
- C. <%@ page pageEncoding="text/plain" %>
- D. <%@ page contentEncoding="text/plain" %>
- E. <% response.setEncoding("text/plain"); %>
- F. <% response.setContentType("text/plain"); %>

Answer: B

www.ExamWorx.com Q: 259 Your web application uses a simple architecture in which servlets handle requests and then forward to a JSP using a request dispatcher. You need to pass information calculated in the servlet to the JSP for view generation. This information must NOT be accessible to any other servlet, JSP or session in the webapp. Which two techniques can you use to accomplish this goal? (Choose two.)

- A. Add attributes to the session object.
- B. Add attributes on the request object.
- C. Add parameters to the request object.
- D. Use the pageContext object to add request attributes.
- E. Add parameters to the JSP's URL when generating the request dispatcher.

Answer: B, E

www.ExamWorx.com Q: 260 All of your JSPs need to have a link that permits users to email the web master. This web application is licensed to many small businesses, each of which have a different email address for the web master. You have decided to use a context parameter that you specify in the deployment descriptor, like this:

42. <context-param>
43. <param-name>webmasterEmail</param-name>
44. <param-value>master@example.com</param-value>
45. </context-param>

Which JSP code snippet creates this email link?

- A. contact us
- B. contact us
- C. contact us
- D. contact us

Answer: D

www.ExamWorx.com Q: 261 Which three are true about servlet filters? (Choose three.)

- A. A filter must implement the destroy method.
- B. A filter must implement the doFilter method.
- C. A servlet may have multiple filters associated with it.
- D. A servlet that is to have a filter applied to it must implement the javax.servlet.FilterChain interface.
- E. A filter that is part of a filter chain passes control to the next filter in the chain by invoking the FilterChain.forward method.
- F. For each <filter> element in the web application deployment descriptor, multiple instances of a filter may be created by the web container.

Answer: A, B, C

www.ExamWorx.com Q: 262 Click the Task button.

Place the XML elements in the web application deployment descriptor solution to configure a servlet context event listener named com.example.MyListener.

Place the XML elements in the web application deployment descriptor solution to configure a servlet context event listener named com.example.MyListener.

Web Application Deployment Descriptor Solution

< Place here. >

< Place here. > com.example.MyListener </ Place here. >

</ Place here. >

XML Elements

class	listener-resource
listener	servlet-listener
context-listener	listener-class
class-name	resource-class

Done

```
<!-- Place here. -->
<!-- Place here. > com.example.MyListener </!-- Place here. -->
</!-- Place here. >

XML Elements

|                  |                   |
|------------------|-------------------|
| class            | listener-resource |
| listener         | servlet-listener  |
| context-listener | listener-class    |
| class-name       | resource-class    |



Done


```

Answer: Check ExamWorx eEngine, Download from Member Center

www.ExamWorx.com Q: 263 Which is true about the web container request processing model?

- A. The init method on a filter is called the first time a servlet mapped to that filter is invoked.
- B. A filter defined for a servlet must always forward control to the next resource in the filter chain.
- C. Filters associated with a named servlet are applied in the order they appear in the web application deployment descriptor file.
- D. If the init method on a filter throws an UnavailableException, then the container will make no further attempt to execute it.

Answer: C

www.ExamWorx.com Q: 264 Your IT department is building a lightweight Front Controller servlet that invokes an application logic object with the interface:

```
public interface ApplicationController {  
    public String invoke(HttpServletRequest request)  
}
```

The return value of this method indicates a symbolic name of the next view. From this name, the Front Controller servlet looks up the JSP URL in a configuration table. This URL might be an absolute path or a path relative to the current request. Next, the Front Controller servlet must send the request to this JSP to generate the view. Assume that the servlet variable `request` is assigned the current `HttpServletRequest` object and the variable `context` is assigned the webapp's `ServletContext`.

Which code snippet of the Front Controller servlet accomplishes this goal?

A. Dispatcher view

```
= context.getDispatcher(viewURL);  
view.forwardRequest(request, response);
```

B. Dispatcher view

```
= request.getDispatcher(viewURL);  
view.forwardRequest(request, response);
```

C. RequestDispatcher view

```
= context.getRequestDispatcher(viewURL);  
view.forward(request, response);
```

D. RequestDispatcher view

```
= request.getRequestDispatcher(viewURL);  
view.forward(request, response);
```

Answer: D

www.ExamWorx.com Q: 265 Given that a web application consists of two `HttpServlet` classes, `ServletA` and `ServletB`, and the `ServletA.service` method:

```
20. String key = "com.example.data";  
21. session.setAttribute(key, "Hello");  
22. Object value = session.getAttribute(key);  
23.
```

Assume `session` is an `HttpSession`, and is not referenced anywhere else in `ServletA`.

Which two changes, taken together, ensure that value is equal to "Hello" on line 23? (Choose two.)

A. ensure that the `ServletB.service` method is synchronized

- B. ensure that the ServletA.service method is synchronized
- C. ensure that ServletB synchronizes on the session object when setting session attributes
- D. enclose lines 21-22 in a synchronized block:

```
synchronized(this) {  
    session.setAttribute(key, "Hello");  
    value = session.getAttribute(key);  
}
```

- E. enclose lines 21-22 in a synchronized block:

```
synchronized(session) {  
    session.setAttribute(key, "Hello");  
    value = session.getAttribute(key);  
}
```

Answer: C, E

www.ExamWorx.com Q: 266 Which retrieves all cookies sent in a given HttpServletRequest request?

- A. request.getCookies()
- B. request.getAttributes()
- C. request.getSession().getCookies()
- D. request.getSession().getAttributes()

Answer: A

www.ExamWorx.com Q: 267 Your company has a corporate policy that prohibits storing a customer's credit card number in any corporate database. However, users have complained that they do NOT want to re-enter their credit card number for each transaction. Your management has decided to use client-side cookies to record the user's credit card number for 120 days. Furthermore, they also want to protect this information during transit from the web browser to the web container; so the cookie must only be transmitted over HTTPS. Which code snippet creates the "creditCard" cookie and adds it to the outgoing response to be stored on the user's web browser?

- A. 10. Cookie c = new Cookie("creditCard", usersCard);
11. c.setSecure(true);
12. c.setAge(10368000);
13. response.addCookie(c);
- B. 10. Cookie c = new Cookie("creditCard", usersCard);
11. c.setHttps(true);
12. c.setMaxAge(10368000);
13. response.setCookie(c);
- C. 10. Cookie c = new Cookie("creditCard", usersCard);
11. c.setSecure(true);

```
12. c.setMaxAge(10368000);
13. response.addCookie(c);
D. 10. Cookie c = new Cookie("creditCard", usersCard);
11. c.setHttps(true);
12. c.setAge(10368000);
13. response.addCookie(c);
E. 10. Cookie c = new Cookie("creditCard", usersCard);
11. c.setSecure(true);
12. c.setAge(10368000);
13. response.setCookie(c);
```

Answer: C

www.ExamWorx.com Q: 268 Click the Task button.

Given a servlet mapped to /control, place the correct URI segment returned as a String on the corresponding HttpServletRequest method call for the URI: /myapp/control/processorder.

Drag and Drop

Given a servlet mapped to /control, place the correct URI segment returned as a String on the corresponding HttpServletRequest method call for the URI: /myapp/control/processorder.

HttpServletRequest Method	URI Segment
getServletPath	/myapp
getPathInfo	/control
getContext	/processorder

Done

Answer: Check ExamWorx eEngine, Download from Member Center

www.ExamWorx.com Q: 269 A web browser need NOT always perform a complete request for a particular page that it suspects might NOT have changed. The HTTP specification provides a mechanism for the browser to retrieve only a partial response from the web server; this response includes information, such as the Last-Modified date but NOT the body of the page. Which HTTP method will the browser use to retrieve such a partial response?

- A. GET
- B. ASK
- C. SEND
- D. HEAD
- E. TRACE
- F. OPTIONS

Answer: D

www.ExamWorx.com Q: 270 You are creating a servlet that generates stock market graphs. You want to provide the web browser with precise information about the amount of data being sent in the response stream. Which two `HttpServletResponse` methods will you use to provide this information? (Choose two.)

- A. `response.setLength(numberOfBytes);`
- B. `response.setContentLength(numberOfBytes);`
- C. `response.setHeader("Length", numberOfBytes);`
- D. `response.setIntHeader("Length", numberOfBytes);`
- E. `response.setHeader("Content-Length", numberOfBytes);`
- F. `response.setIntHeader("Content-Length", numberOfBytes);`

Answer: B, F

www.ExamWorx.com Q: 271 Which two prevent a servlet from handling requests? (Choose two.)

- A. The servlet's init method returns a non-zero status.
- B. The servlet's init method throws a `ServletException`.
- C. The servlet's init method sets the `ServletResponse`'s content length to 0.
- D. The servlet's init method sets the `ServletResponse`'s content type to null.
- E. The servlet's init method does NOT return within a time period defined by the servlet container.

Answer: B, E

www.ExamWorx.com Q: 272 A web application allows the HTML title banner to be set using a servlet context initialization parameter called `titleStr`. Which two properly set the title in this scenario? (Choose two.)

- A. `<title>${titleStr}</title>`
- B. `<title>${initParam.titleStr}</title>`
- C. `<title>${params[0].titleStr}</title>`
- D. `<title>${paramValues.titleStr}</title>`
- E. `<title>${initParam['titleStr']}</title>`
- F. `<title>${servletParams.titleStr}</title>`
- G. `<title>${request.get("titleStr")}</title>`

Answer: B, E

www.ExamWorx.com Q: 273 You are building a dating service web site. Part of the form to submit a client's profile is a group of radio buttons for the person's hobbies:

20. <input type='radio' name='hobbyEnum' value='HIKING'>Hiking

21. <input type='radio' name='hobbyEnum' value='SKIING'>Skiing

22. <input type='radio' name='hobbyEnum' value='SCUBA'>SCUBA Diving
23. <!-- and more options -->

After the user submits this form, a confirmation screen is displayed with these hobbies listed. Assume that an application-scoped variable, hobbies, holds a map between the Hobby enumerated type and the display name.

Which EL code snippet will display Nth element of the user's selected hobbies?

- A. \${hobbies[hobbyEnum[N]]}
- B. \${hobbies[paramValues.hobbyEnum[N]]}
- C. \${hobbies[paramValues@'hobbyEnum'@N]}
- D. \${hobbies.get(paramValues.hobbyEnum[N])}
- E. \${hobbies[paramValues.hobbyEnum.get(N)]}

Answer: B

www.ExamWorx.com Q: 274 Given a web application in which the request parameter productID contains a product identifier. Which two EL expressions evaluate the value of the productID? (Choose two.)

- A. \${productID}
- B. \${param.productID}
- C. \${params.productID}
- D. \${params.productID[1]}
- E. \${paramValues.productID}
- F. \${paramValues.productID[0]}
- G. \${pageContext.request.productID}

Answer: B, F

www.ExamWorx.com Q: 275 You are building a web application with a scheduling component. On the JSP, you need to show the current date, the date of the previous week, and the date of the next week. To help you present this information, you have created the following EL functions in the 'd' namespace:

name: curDate; signature: java.util.Date currentDate()

name: addWeek; signature: java.util.Date addWeek(java.util.Date, int)
name: dateString; signature: java.util.String getDateString(java.util.Date)

Which EL code snippet will generate the string for the previous week?

- A. \${d:dateString(addWeek(curDate(), -1))}
- B. \${d:dateString[addWeek[curDate[], -1]]}
- C. \${d:dateString[d:addWeek[d:curDate[], -1]]}
- D. \${d:dateString(d:addWeek(d:curDate(), -1))}

Answer: D

www.ExamWorx.com Q: 276 You are building a dating web site. The client's date of birth is collected along with lots of other information. The Person class has a derived method, getAge():int, which returns the person's age calculated from the date of birth and today's date. In one of your JSPs you need to print a special message to clients within the age group of 25 through 35. Which two EL code snippets will return true for this condition? (Choose two.)

- A. \${client.age in [25,35]}
- B. \${client.age between [25,35]}
- C. \${client.age between 25 and 35}
- D. \${client.age <= 35 && client.age >= 25}
- E. \${client.age le 35 and client.age ge 25}
- F. \${not client.age > 35 && client.age < 25}

Answer: D, E

SCJP 1.6 (Sun Certified Java Programmer (SCJP) Mock Questions)

Final Test Questions

Questions: 71

www.techfaq360.com

[Buy 800 Questions with details explanations](#)

Question - 1

What is the output for the below code ?

```
1. public class A {  
2.     int add(int i, int j){  
3.         return i+j;  
4.     }  
5. }  
6. public class B extends A{  
7.     public static void main(String argv[]){  
8.         short s = 9;  
9.         System.out.println(add(s,6));  
10.    }  
11. }
```

Options are

- A.Compile fail due to error on line no 2
- B.Compile fail due to error on line no 9
- C.Compile fail due to error on line no 8
- D.15

Answer :

B is the correct answer.

Cannot make a static reference to the non-static method add(int, int) from the type A. The short s is autoboxed correctly, but the add() method cannot be invoked from a static method because add() method is not static.

Question - 2

What is the output for the below code ?

```
public class A {  
    int k;  
    boolean istrue;  
    static int p;  
    public void printValue() {  
        System.out.print(k);  
        System.out.print(istrue);  
        System.out.print(p);  
  
    }  
  
}  
public class Test{  
  
    public static void main(String argv[]){  
  
        A a = new A();  
  
        a.printValue();  
    }  
}
```

Options are

- A.0 false 0
- B.0 true 0
- C.0 0 0
- D.Compile error - static variable must be initialized before use.

Answer :

A is the correct answer.

Global and static variable need not be initialized before use. Default value of global and static int variable is zero. Default value of boolean variable is false. Remember local variable must be initialized before use.

Question - 3

What is the output for the below code ?

```
public class Test{
    int $;
    int $7;
    int do;
    public static void main(String argv[]){
        Test test = new Test();
        test.$7=7;
        test.do=9;

        System.out.println(test.$7);
        System.out.println(test.do);
        System.out.println(test._$);

    }
}
```

Options are

- A.7 9 0
- B.7 0 0
- C.Compile error - \$7 is not valid identifier.
- D.Compile error - do is not valid identifier.

Answer :

D is the correct answer.

\$7 is valid identifier. Identifiers must start with a letter, a currency character (\$), or underscore (_). Identifiers cannot start with a number. You can't use a Java keyword as an identifier. do is a Java keyword.

Question - 4

What is the output for the below code ?

```
package com;
class Animal {

    public void printName(){
        System.out.println("Animal");
    }

}
```

```

package exam;
import com.Animal;
public class Cat extends Animal {

    public void printName(){
        System.out.println("Cat");
    }

}

package exam;
import com.Animal;

public class Test {

    public static void main(String[] args){
        Animal a = new Cat();
        a.printName();

    }

}

```

Options are

- A.Animal
- B.Cat
- C.Animal Cat
- D.Compile Error

Answer :

D is the correct answer.

Cat class won't compile because its superclass, Animal, has default access and is in a different package. Only public superclass can be accessible for different package.

Question - 5

What is the output for the below code ?

```

public class A {
    int i = 10;
    public void printValue() {

```

```

        System.out.println("Value-A");
    } ;

}

public class B extends A{
    int i = 12;
    public void printValue() {
        System.out.print("Value-B");
    }
}

public class Test{
    public static void main(String argv[]){
        A a = new B();
        a.printValue();
        System.out.println(a.i);
    }
}

```

Options are

- A.Value-B 11
- B.Value-B 10
- C.Value-A 10
- D.Value-A 11

Answer :

B is the correct answer.

If you create object of subclass with reference of super class like (A a = new B();) then subclass method and super class variable will be executed.

Question - 6

What is the output for the below code ?

```

public enum Test {
    BREAKFAST(7, 30), LUNCH(12, 15), DINNER(19, 45);

    private int hh;

    private int mm;
}

```

```

    Test(int hh, int mm) {
        assert (hh >= 0 && hh <= 23) : "Illegal hour.";
        assert (mm >= 0 && mm <= 59) : "Illegal mins.";
        this.hh = hh;
        this.mm = mm;
    }

    public int getHour() {
        return hh;
    }

    public int getMins() {
        return mm;
    }

    public static void main(String args[]){
        Test t = new BREAKFAST;
        System.out.println(t.getHour() + ":" +t.getMins());
    }
}

```

Options are

- A.7:30
- B.Compile Error - an enum cannot be instantiated using the new operator.
- C.12:30
- D.19:45

Answer :

B is the correct answer.

As an enum cannot be instantiated using the new operator, the constructors cannot be called explicitly. You have to do like Test t = BREAKFAST;

Question - 7

What is the output for the below code ?

```

public class A {
    static{System.out.println("static");}
    { System.out.println("block");}
    public A(){
        System.out.println("A");
    }

    public static void main(String[] args){
        A a = new A();
    }
}

```

}

Options are

- A.A block static
- B.static block A
- C.static A
- D.A

Answer :

B is the correct answer.

First execute static block, then statement block then constructor.

Question - 8

What is the output for the below code ?

```
1. public class Test {  
2.     public static void main(String[] args) {  
3.         int i = 010;  
4.         int j = 07;  
5.         System.out.println(i);  
6.         System.out.println(j);  
7.     }  
8. }
```

Options are

- A.8 7
- B.10 7
- C.Compilation fails with an error at line 3
- D.Compilation fails with an error at line 5

Answer :

A is the correct answer.

By placing a zero in front of the number is an integer in octal form. 010 is in octal form so its value is 8.

Question - 9

What is the output for the below code ?

```
1. public class Test {  
2.     public static void main(String[] args){  
3.         byte b = 6;  
4.         b+=8;  
5.         System.out.println(b);  
6.         b = b+7;  
7.         System.out.println(b);  
8.     }  
9. }
```

Options are

- A.14 21
- B.14 13
- C.Compilation fails with an error at line 6
- D.Compilation fails with an error at line 4

Answer :

C is the correct answer.

int or smaller expressions always resulting in an int. So compiler complain about Type mismatch: cannot convert from int to byte for b = b+7; But b += 7; // No problem because +=, -=, *=, and /= will all put in an implicit cast. b += 7 is same as b = (byte)b+7 so compiler not complain.

Question - 10

What is the output for the below code ?

```
public class Test {  
    public static void main(String[] args){  
        String value = "abc";  
        changeValue(value);  
        System.out.println(value);  
    }  
  
    public static void changeValue(String a){  
        a = "xyz";  
    }  
}
```

Options are

- A.abc
- B.xyz
- C.Compilation fails
- D.Compilation clean but no output

Answer :

A is the correct answer.

Java pass reference as value. passing the object reference, and not the actual object itself. Simply reassigning to the parameter used to pass the value into the method will do nothing, because the parameter is essentially a local variable.

Question - 11

What is the output for the below code ?

```
public class Test {  
  
    public static void printValue(int i, int j, int k){  
        System.out.println("int");  
    }  
    public static void printValue(byte...b){  
        System.out.println("long");  
    }  
  
    public static void main(String... args) {  
        byte b = 9;  
        printValue(b,b,b);  
    }  
}
```

Options are

- A.long
- B.int
- C.Compilation fails
- D.Compilation clean but throws RuntimeException

Answer :

B is the correct answer.

Primitive widening uses the smallest method argument possible. (For Example if you pass short value to a method but method with short argument is not available then compiler choose method with int argument). But in this case compiler will prefer the older style before it chooses the newer style, to keep existing code more robust. var-args method is looser than widen.

Question - 12

Fill in the gap:

```
public class Test {  
  
    public static void main(String[] args) {  
        String[] words = new String[] {"aaa", "bbb", "ccc", "aaa"};  
        Map<String, Integer> m = new TreeMap<String, Integer>();  
        for (String word : words) {  
            freq = m.get(word);  
            m.put(word, freq == null ? 1 : freq + 1);  
        }  
        System.out.println(m);  
  
    }  
}
```

Use the following fragments zero or many times

String

Integer

Boolean

Float

Question - 13

You have a java file name Test.java inside src folder of javaproject directory.

You have also classes folder inside javaproject directory.

you have issued below command from command prompt.

cd javaproject

Which of the below command puts Test.class file inside classes folder ?

Options are

- A.javac -d classes src/Test.java
- B.javac Test.java
- C.javac src/Test.java
- D.javac classes src/Test.java

Answer :

A is the correct answer.

The -d option lets you tell the compiler in which directory to put the .class file it generates (d for destination)

Question - 14

You have two class files name Test.class and Test1.class inside javaproject directory.

Test.java source code is :

```
public class Test{  
    public static void main (String[] args){  
        System.out.println("Hello Test");  
    }  
}
```

Test1.java source code is :

```
public class Test1{  
    public static void main (String[] args){  
        System.out.println("Hello Test1");  
    }  
}
```

you have issued below commands from command prompt.

```
cd javaproject  
java Test Test1
```

What is the output ?

Options are

- A.Hello Test
- B.Hello Test Hello Test1
- C.Hello Test1
- D.Run fails - class not found

Answer :

A is the correct answer.

You must specify exactly one class file to execute. If more than one then first one will be executed.

Question - 15

You have a java file name Test.java .

Test.java needs access to a class contained in app.jar in "exam" directory.

Which of the following command set classpath to compile clean?

Options are

- A.javac -classpath exam/app.jar Test.java
- B.javac -classpath app.jar Test.java
- C.javac -classpath exam Test.java
- D.None of the above

Answer :

A is the correct answer.

javac -classpath exam/app.jar Test.java is the correct command to set exam/app.jar in classpath.

Question - 16

What will be the result of compiling the following code:

```
public class SuperClass {  
    public int doIt(String str, Integer... data) throws Exception{  
        String signature = "(String, Integer[])";  
        System.out.println(str + " " + signature);  
        return 1;  
    }  
  
    public class SubClass extends SuperClass{  
        public int doIt(String str, Integer... data)  
        {  
            String signature = "(String, Integer[])";  
        }  
    }  
}
```

```

        System.out.println("Overridden: " + str + " " +
signature);
        return 0;
    }

    public static void main(String... args)
{
    SuperClass sb = new SubClass();
    sb.doIt("hello", 3);
}
}

```

Options are

- A.Overridden: hello (String, Integer[])
- B.hello (String, Integer[])
- C.Complilation fails
- D.None of the above

Answer :

C is the correct answer.

Unhandled exception type Exception.

Question - 17

**What happens when the following code is compiled and run.
Select the one correct answer.**

```

for(int i = 2; i < 4; i++)
    for(int j = 2; j < 4; j++)
        if(i < j)
            assert i!=j : i;

```

Options are

- A.The class compiles and runs, but does not print anything.
- B.The number 2 gets printed with AssertionError
- C.compile error
- D.The number 3 gets printed with AssertionError

Answer :

A is the correct answer.

When if condition returns true, the assert statement also returns true. Hence Assertion Error does not get generated. .

Question - 18

What happens when the following code is compiled and run.
Select the one correct answer.

```
for(int i = 2; i < 4; i++)
    for(int j = 2; j < 4; j++)
        assert i!=j : i;
```

Options are

- A.The class compiles and runs, but does not print anything.
- B.The number 2 gets printed with Assertion Error
- C.compile error
- D.The number 3 gets printed with Assertion Error

Answer :

B is the correct answer.

When i and j are both 2, assert condition is false, and Assertion Error gets generated. .

Question - 19

```
try{
    File f = new File("a.txt");
} catch(Exception e){
    } catch(IOException io){
}
Is this code create new file name a.txt ?
```

Options are

- A.True
- B.False
- C.Compilation Error
- D.None

Answer :

C is the correct answer.

IOException is unreachable to compiler because all exception is going to catch by Exception block.

Question - 20

```
class A {  
    A(String s) {  
    }  
    A() {  
    }  
}  
  
1. class B extends A {  
2.     B() { }  
3.     B(String s) {  
4.         super(s);  
5.     }  
6.     void test() {  
7.         // insert code here  
8.     }  
9. }
```

Which of the below code can be insert at line 7 to make clean compilation ?

Options are

- A.A a = new B();
- B.A a = new B(5);
- C.A a = new A(String s);
- D.All of the above

Answer :

A is the correct answer.

A a = new B(); is correct because anonymous inner classes are no different from any other class when it comes to polymorphism.

Question - 21

What is the output for the below code ?

```
interface A {  
    public void printValue();
```

```

}

1. public class Test{
2.     public static void main (String[] args){
3.         A a1 = new A() {
4.             public void printValue(){
5.                 System.out.println("A");
6.             }
7.         };
8.         a1.printValue();
9.     }
10. }

```

Options are

- A.Compilation fails due to an error on line 3
- B.A
- C.Compilation fails due to an error on line 8
- D.null

Answer :

B is the correct answer.

The A a1 reference variable refers not to an instance of interface A, but to an instance of an anonymous (unnamed) class. So no compilation error.

Question - 22

```

class A {
    class A1 {
        void printValue(){
            System.out.println("A.A1");
        }
    }
}

1. public class Test{
2.     public static void main (String[] args){
3.         A a = new A();
4.         // INSERT CODE
5.         a1.printValue();
6.     }
7. }

```

Which of the below code inserted at line 4, compile and produce the output "A.A1"?

Options are

- A.A.A1 a1 = new A.A1();
- B.A.A1 a1 = a.new A1();
- C.A a1 = new A.A1();
- D.All of the above

Answer :

B is the correct answer.

correct inner class instantiation syntax is A a = new A(); A.A1 a1 = a.new A1();

Question - 23

What is the output for the below code ?

```
public class A {  
    public void printValue(){  
        System.out.println("Value-A");  
    }  
}  
  
public class B extends A{  
    public void printNameB(){  
        System.out.println("Name-B");  
    }  
}  
  
public class C extends A{  
    public void printNameC(){  
        System.out.println("Name-C");  
    }  
}  
  
1. public class Test{  
2.     public static void main (String[] args) {  
3.         B b = new B();  
4.         C c = new C();  
5.         newPrint(b);  
6.         newPrint(c);  
7.     }  
8.     public static void newPrint(A a){  
9.         a.printValue();  
10.    }  
}
```

11. }

Options are

- A.Value-A Name-B
- B.Value-A Value-A
- C.Value-A Name-C
- D.Name-B Name-C

Answer :

B is the correct answer.

Class B extended Class A therefore all methods of Class A will be available to class B except private methods. Class C extended Class A therefore all methods of Class A will be available to class C except private methods.

Question - 24

What is the output for the below code ?

```
public class A {  
    public void printName(){  
        System.out.println("Value-A");  
    }  
  
public class B extends A{  
    public void printName(){  
        System.out.println("Name-B");  
    }  
  
public class C extends A{  
    public void printName(){  
        System.out.println("Name-C");  
    }  
  
1. public class Test{  
2.     public static void main (String[] args) {  
3.         B b = new B();  
4.         C c = new C();  
5.         b = c;  
6.         newPrint(b);  
7.     }  
}
```

```
8.     public static void newPrint(A a){  
9.             a.printName();  
10.        }  
11.    }
```

Options are

- A.Name-B
- B.Name-C
- C.Compilation fails due to an error on lines 5
- D.Compilation fails due to an error on lines 9

Answer :

C is the correct answer.

Reference variable can refer to any object of the same type as the declared reference OR can refer to any subtype of the declared type. Reference variable "b" is type of class B and reference variable "c" is a type of class C. So Compilation fails.

Question - 25

What is the output for the below code ?

```
public class C {  
}  
  
public class D extends C{  
}  
  
public class A {  
  
    public C getOBJ(){  
        System.out.println("class A - return C");  
        return new C();  
    }  
}  
  
public class B extends A{  
  
    public D getOBJ(){  
        System.out.println("class B - return D");  
        return new D();  
    }  
}
```

```

}

public class Test {

    public static void main(String... args) {
        A a = new B();
        a.getOBJ();

    }
}

```

Options are

- A.class A - return C
- B.class B - return D
- C.Compilation fails
- D.Compilation succeed but no output

Answer :

B is the correct answer.

From J2SE 5.0 onwards. return type in the overriding method can be same or subtype of the declared return type of the overridden (superclass) method.

Question - 26

What is the output for the below code ?

```

public class A {
    private void printName(){
        System.out.println("Value-A");
    }
}

public class B extends A{
    public void printName(){
        System.out.println("Name-B");
    }
}

public class Test{
    public static void main (String[] args) {
        B b = new B();
        b.printName();
    }
}

```

}

Options are

- A.Value-A
- B.Name-B
- C.Value-A Name-B
- D.Compilation fails - private methods can't be override

Answer :

B is the correct answer.

You can not override private method , private method is not availabe in subclass . In this case printName() method a class A is not overriding by printName() method of class B. printName() method of class B different method. So you can call printName() method of class B.

Question - 27

What is the output for the below code ?

```
import java.io.FileNotFoundException;

public class A {
    public void printName() throws FileNotFoundException {
        System.out.println("Value-A");
    }
}

public class B extends A{
    public void printName() throws NullPointerException{
        System.out.println("Name-B");
    }
}

public class Test{
    public static void main (String[] args) throws Exception{
        A a = new B();
        a.printName();
    }
}
```

Options are

- A.Value-A
- B.Compilation fails-Exception NullPointerException is not compatible with throws clause in A.printName()
- C.Name-B
- D.Compilation succeed but no output

Answer :

C is the correct answer.

The overriding method can throw any unchecked (runtime) exception, regardless of exception thrown by overridden method. NullPointerException is RuntimeException so compiler not complain.

Question - 28

What is the output for the below code ?

```
public class A {  
    public A(){  
        System.out.println("A");  
    }  
    public A(int i){  
        this();  
        System.out.println(i);  
    }  
}  
  
public class B extends A{  
    public B (){  
        System.out.println("B");  
    }  
    public B (int i){  
        this();  
        System.out.println(i+3);  
    }  
}  
  
public class Test{  
    public static void main (String[] args){  
        new B(5);  
    }  
}
```

Options are

- A.A B 8
- B.A 5 B 8
- C.A B 5
- D.B 8 A 5

Answer :

A is the correct answer.

Constructor of class B call their superclass constructor of class A (public A()) , which execute first, and that constructors can be overloaded. Then come to constructor of class B (public B (int i)).

Question - 29

What is the output for the below code ?

```
1. public interface InfA {  
2.         protected String getName();  
3.     }  
  
public class Test implements InfA{  
    public String getName(){  
        return "test-name";  
    }  
  
    public static void main (String[] args){  
        Test t = new Test();  
        System.out.println(t.getName());  
  
    }  
}
```

Options are

- A.test-name
- B.Compilation fails due to an error on lines 2
- C.Compilation fails due to an error on lines 1
- D.Compilation succeed but Runtime Exception

Answer :

B is the correct answer.

Illegal modifier for the interface method InfA.getName(); only public and abstract are permitted

Question - 30

What is the output for the below code ?

```
public class D {  
    int i;  
    int j;  
    public D(int i,int j){  
        this.i=i;  
        this.j=j;  
    }  
  
    public void printName() {  
        System.out.println("Name-D");  
    }  
}  
  
1. public class Test{  
2.     public static void main (String[] args){  
3.         D d = new D();  
4.         d.printName();  
5.     }  
6. }  
7. }
```

Options are

- A.Name-D
- B.Compilation fails due to an error on lines 3
- C.Compilation fails due to an error on lines 4
- D.Compilation succeed but no output

Answer :

B is the correct answer.

Since there is already a constructor in this class (public D(int i,int j)), the compiler won't supply a default constructor. If you want a no-argument constructor to overload the with-arguments version you already have, you have to define it by yourself. The constructor D() is undefined in class D. If you define explicit constructor then default constructor will

not be available. You have to define explicitly like public D(){ } then the above code will work. If no constructor into your class , a default constructor will be automatically generated by the compiler.

Question - 31

```
public class A {
    public void test1(){
        System.out.println("test1");
    }
}

public class B extends A{
    public void test2(){
        System.out.println("test2");
    }
}

1. public class Test{
2.     public static void main (String[] args){
3.         A a = new A();
4.         A b = new B();
5.         B b1 = new B();
6.         // insert code here
7.     }
8. }
```

Which of the following , inserted at line 6, will compile and print test2?

Options are

- A.((B)b).test2();
- B.(B)b.test2();
- C.b.test2();
- D.a.test2();

Answer :

A is the correct answer.

((B)b).test2(); is proper cast. test2() method is in class B so need to cast b then only test2() is accessible. (B)b.test2(); is not proper cast without the second set of parentheses, the compiler thinks it is an incomplete statement.

Question - 32

What is the output for the below code ?

```
1. public class Test {  
2.     public static void main(String... args) {  
3.         int x =5;  
4.         x *= 3 + 7;  
5.         System.out.println(x);  
6.     }  
7. }
```

Options are

- A.22
- B.50
- C.10
- D.Compilation fails with an error at line 4

Answer :

B is the correct answer.

$x *= 3 + 7;$ is same as $x = x * (3 + 7) = 5 * (10) = 50$ because expression on the right is always placed inside parentheses.

Question - 33

What is the output for the below code ?

```
1. public class Test {  
2.     enum Month { JAN, FEB, MAR };  
3.     public static void main(String... args) {  
4.         Month m1 = Month.JAN;  
5.         Month m2 = Month.JAN;  
6.         Month m3 = Month.FEB;  
7.         System.out.println(m1 == m2);  
8.         System.out.println(m1.equals(m2));  
9.         System.out.println(m1 == m3);  
10.        System.out.println(m1.equals(m3));  
11.    }  
12. }
```

Options are

- A.true true true false
- B.true true false false
- C.false false true true
- D.Compilation fails with an error at line 10

Answer :

B is the correct answer.

m1 and m2 refer to the same enum constant So m1 == m2 returns true BUT m1 and m3 refer to different enum constant So m1 == m3 returns false. m1.equals(m2) returns true because enum constant value is same (JAN and JAN). m1.equals(m3) return false because enum constants values are different (JAN and FEB).

Question - 34

What is the output for the below code ?

```
1. public class Test {  
2.     public static void main(String... args) {  
3.         int [] index = new int[5];  
4.         System.out.println(index instanceof Object);  
5.     }  
6. }
```

Options are

- A.true
- B.false
- C.Compilation fails with an error at line 3
- D.Compilation fails with an error at line 4

Answer :

A is the correct answer.

An array is always an instance of Object

Question - 35

What is the output for the below code ?

```
public class Test {
```

```

public static void main(String... args) {
    int a =5 , b=6, c =7;
    System.out.println("Value is "+ b +c);
    System.out.println(a + b +c);
    System.out.println("String "+(b+c));
}

```

Options are

- A.Value is 67 18 String 13
- B.Value is 13 18 String 13
- C.Value is 13 18 String
- D.Compilation fails

Answer :

A is the correct answer.

If the left hand operand is not a String then + operator treat as plus BUT if left hand operand is a String then + perform String concatenation.

Question - 36

What is the output for the below code?

```

public class A {
    public A() {
        System.out.println("A");
    }
}

public class B extends A implements Serializable {
    public B() {
        System.out.println("B");
    }
}

public class Test {

    public static void main(String... args) throws Exception {
        B b = new B();

        ObjectOutputStream save = new ObjectOutputStream(new
FileOutputStream("datafile"));
        save.writeObject(b);
        save.flush();
    }
}

```

```

        ObjectInputStream restore = new ObjectInputStream(new
FileInputStream("datafile"));
        B z = (B) restore.readObject();

    }

}

```

Options are

- A.A B A
- B.A B A B
- C.B B
- D.B

Answer :

A is the correct answer.

On the time of deserialization , the Serializable object not create new object. So constructor of class B does not called. A is not Serializable object so constructor is called.

Question - 37

What is the output for the below code?

```

public class A {
    public A() {
        System.out.println("A");
    }
}

public class Test {

    public static void main(String... args) throws Exception {
        A a = new A();

        ObjectOutputStream save = new ObjectOutputStream(new
FileOutputStream("datafile")));
        save.writeObject(a);
        save.flush();

        ObjectInputStream restore = new ObjectInputStream(new
FileInputStream("datafile"));
        A z = (A) restore.readObject();
    }
}

```

```
    }  
}
```

Options are

- A.A A
- B.A
- C.java.io.NotSerializableException
- D.None of the above

Answer :

C is the correct answer.

Class A does not implements Serializable interface. So throws NotSerializableException on trying to Serialize a non Serializable object.

Question - 38

What will be the result of compiling and run the following code:

```
public class Test {  
  
    public static void main(String... args) throws Exception {  
        Integer i = 34;  
        int l = 34;  
        if(i.equals(l)){  
            System.out.println(true);  
        }else{  
            System.out.println(false);  
        }  
    }  
}
```

Options are

- A.true
- B.false
- C.Compile error
- D.None of the above

Answer :

A is the correct answer.

equals() method for the integer wrappers will only return true if the two primitive types and the two values are equal.

Question - 39

What will be the result of compiling and run the following code:

```
public class Test {  
  
    public static void main(String... args) throws Exception {  
        File file = new File("test.txt");  
        System.out.println(file.exists());  
        file.createNewFile();  
        System.out.println(file.exists());  
    }  
  
}
```

Options are

- A.true true
- B.false true
- C.false true
- D.None of the above

Answer :

B is the correct answer.

creating a new instance of the class File, you're not yet making an actual file, you're just creating a filename. So file.exists() return false. createNewFile() method created an actual file.so file.exists() return true.

Question - 40

What is the output for the below code ?

```
public class A {}  
  
public class B implements Serializable {  
    private static A a = new A();  
    public static void main(String... args){  
        B b = new B();  
        try{  
            FileOutputStream fs = new  
FileOutputStream("b.ser");  
            ObjectOutputStream os = new  
ObjectOutputStream(fs);
```

```

        os.writeObject(b);
        os.close();

    }catch(Exception e){
        e.printStackTrace();
    }

}

```

Options are

- A.Compilation Fail
- B.java.io.NotSerializableException: Because class A is not Serializable.
- C.No Exception at Runtime
- D.None of the above

Answer :

C is the correct answer.

No java.io.NotSerializableException, Because class A variable is static. static variables are not Serializable.

Question - 41

What will happen when you attempt to compile and run the following code ?

```

1. public class Test extends Thread{
2.     public static void main(String argv[]){
3.         Test t = new Test();
4.         t.run();
5.         t.start();
6.     }
7.     public void run(){
8.         System.out.println("run-test");
9.     }
10. }

```

Options are

- A.run-test run-test
- B.run-test
- C.Compilation fails due to an error on line 4

D.Compilation fails due to an error on line 7

Answer :

A is the correct answer.

t.run() Legal, but does not start a new thread , it is like a method call of class Test BUT t.start() create a thread and call run() method.

Question - 42

What is the output for the below code ?

```
class A implements Runnable{
    public void run(){
        System.out.println("run-a");
    }
}

1. public class Test {
2.     public static void main(String... args) {
3.         A a = new A();
4.         Thread t = new Thread(a);
5.         t.start();
6.         t.start();
7.     }
8. }
```

Options are

- A.run-a
- B.run-a run-a
- C.Compilation fails with an error at line 6
- D.Compilation succeed but Runtime Exception

Answer :

D is the correct answer.

Once a thread has been started, it can never be started again. 2nd time t.start() throws java.lang.IllegalThreadStateException.

Question - 43

What is the output for the below code ?

```
class A implements Runnable{
    public void run(){
        try{
            for(int i=0;i<4;i++){
                Thread.sleep(100);

                System.out.println(Thread.currentThread().getName());
            }
        } catch(InterruptedException e){
        }
    }

public class Test {
    public static void main(String argv[]) throws Exception{
        A a = new A();
        Thread t = new Thread(a,"A");
        Thread t1 = new Thread(a,"B");
        t.start();
        t.join();
        t1.start();
    }
}
```

Options are

- A.A A A A B B B B
- B.A B A B A B A B
- C.Output order is not guaranteed
- D.Compilation succeed but Runtime Exception

Answer :

A is the correct answer.

t.join(); means Threat t must finish before Thread t1 start.

Question - 44

What is the output for the below code ?

```
public class B {
    public synchronized void printName() {
        try{
```

```

        System.out.println("printName");
        Thread.sleep(5*1000);

    }catch(InterruptedException e){

    }

}

public synchronized void printValue(){
    System.out.println("printValue");

}
}

public class Test extends Thread{
    B b = new B();
    public static void main(String argv[]) throws Exception{
Test t = new Test();
Thread t1 = new Thread(t,"t1");
Thread t2 = new Thread(t,"t2");
t1.start();
t2.start();
}

public void run(){
    if(Thread.currentThread().getName().equals("t1")){
        b.printName();
    }else{
        b.printValue();
    }
}
}

```

Options are

- A.print : printName , then wait for 5 seconds then print : printValue
- B.print : printName then print : printValue
- C.print : printName then wait for 5 minutes then print : printValue
- D.Compilation succeed but Runtime Exception

Answer :

A is the correct answer.

There is only one lock per object, if one thread has picked up the lock, no other thread can pick up the lock until the first thread releases the lock. printName() method acquire the lock for 5 seconds, So other threads can not access the object. If one synchronized method of an instance is executing then other synchronized method of the same instance should wait.

Question - 45

What is the output for the below code ?

```
public class B {
    public static synchronized void printName(){
        try{
            System.out.println("printName");
            Thread.sleep(5*1000);

        }catch(InterruptedException e){
            }

    }

    public synchronized void printValue(){
        System.out.println("printValue");
    }
}

public class Test extends Thread{
    B b = new B();
    public static void main(String argv[]) throws Exception{
        Test t = new Test();
        Thread t1 = new Thread(t,"t1");
        Thread t2 = new Thread(t,"t2");
        t1.start();
        t2.start();
    }

    public void run(){
        if(Thread.currentThread().getName().equals("t1")){
            b.printName();
        }else{
            b.printValue();
        }
    }
}
```

Options are

- A.print : printName , then wait for 5 seconds then print : printValue
- B.print : printName then print : printValue
- C.print : printName then wait for 5 minutes then print : printValue
- D.Compilation succeed but Runtime Exception

Answer :

B is the correct answer.

There is only one lock per object, if one thread has picked up the lock, no other thread can pick up the lock until the first thread releases the lock. In this case printName() is static , So lock is in class B not instance b, both method (one static and other no-static) can run simultaneously. A static synchronized method and a non static synchronized method will not block each other.

Question - 46

What is the output for the below code ?

```
class A extends Thread{
    int count = 0;
    public void run(){
        System.out.println("run");
        synchronized (this) {
            for(int i =0; i < 50 ; i++){
                count = count + i;
            }
            notify();
        }
    }
}

public class Test{
    public static void main(String argv[]) {
        A a = new A();
        a.start();
        synchronized (a) {
            System.out.println("waiting");
            try{
                a.wait();
            }catch(InterruptedException e){
                }
            System.out.println(a.count);
        }
    }
}
```

Options are

- A.waiting run 1225
- B.waiting run 0
- C.waiting run and count can be anything
- D.Compilation fails

Answer :

A is the correct answer.

a.wait(); put thread on wait until not get notified. A thread gets on this waiting list by executing the wait() method of the target object. It doesn't execute any further instructions until the notify() method of the target object is called. A thread to call wait() or notify(), the thread has to be the owner of the lock for that object.

Question - 47

Which of the following statements about this code are true?

```
class A extends Thread{
    public void run(){
        for(int i =0; i < 2; i++){
            System.out.println(i);
        }
    }
}

public class Test{
    public static void main(String argv[]){
        Test t = new Test();
        t.check(new A() {});
    }
    public void check(A a){
        a.start();
    }
}
```

Options are

- A.0 0
- B.Compilation error, class A has no start method
- C.0 1
- D.Compilation succeed but runtime exception

Answer :

C is the correct answer.

class A extends Thread means the anonymous instance that is passed to check() method has a start method which then calls the run method.

Question - 48

HashMap can be synchronized by _____ ?

Options are

- A.Map m = Collections.synchronizedMap(hashMap);
- B.Map m = hashMap.synchronizedMap();
- C.Map m = Collection.synchronizedMap(hashMap);
- D.None of the above

Answer :

A is the correct answer.

HashMap can be synchronized by Map m = Collections.synchronizedMap(hashMap);

Question - 49

What is the output for the below code?

```
import java.util.LinkedList;
import java.util.Queue;

public class Test {
    public static void main(String... args) {

        Queue q = new LinkedList();
        q.add("newyork");
        q.add("ca");
        q.add("texas");
        show(q);
    }

    public static void show(Queue q) {
        q.add(new Integer(11));
        while (!q.isEmpty())
            System.out.print(q.poll() + " ");
    }
}
```

```
    }  
  
}
```

Options are

- A.Compile error : Integer can't add
- B.newyork ca texas 11
- C.newyork ca texas
- D.None of the above

Answer :

B is the correct answer.

" q was originally declared as Queue<String>, But in show() method it is passed as an untyped Queue. nothing in the compiler or JVM prevents us from adding an Integer after that.

If the show method signature is public static void show(Queue<String> q) than you can't add Integer, Only String allowed. But public static void show(Queue q) is untyped Queue so you can add Integer.

Y poll() Retrieves and removes the head of this queue, or returns null if this queue is empty.

Question - 51

What is the output for the bellow code?

```
import java.util.Iterator;  
import java.util.Set;  
import java.util.TreeSet;  
  
public class Test {  
    public static void main(String... args) {  
  
        Set s = new TreeSet();  
        s.add("7");  
        s.add(9);  
        Iterator itr = s.iterator();  
        while (itr.hasNext())  
            System.out.print(itr.next() + " ");  
  
    }  
}
```

Options are

- A.Compile error
- B.Runtime Exception
- C.7 9
- D.None of the above

Answer :

B is the correct answer.

Without generics, the compiler does not know what type is appropriate for this TreeSet, so it allows everything to compile. But at runtime the TreeSet will try to sort the elements as they are added, and when it tries to compare an Integer with a String it will throw a ClassCastException.
Exception in thread "main" java.lang.ClassCastException: java.lang.String cannot be cast to java.lang.Integer.

Question - 52

What is the output for the below code?

```
import java.util.Iterator;
import java.util.TreeSet;

public class Test {
    public static void main(String... args) {

        TreeSet s1 = new TreeSet();
        s1.add("one");
        s1.add("two");
        s1.add("three");
        s1.add("one");

        Iterator it = s1.iterator();
        while (it.hasNext() ) {
            System.out.print( it.next() + " " );
        }

    }
}
```

Options are

- A.one three two
- B.Runtime Exception
- C.one three two one
- D.one two three

Answer :

A is the correct answer.

h TreeSet assures no duplicate entries.it will return elements in natural order, which, for Strings means alphabetical.

Question - 53

If we do

```
ArrayList lst = new ArrayList();
```

What is the initial capacity of the ArrayList lst ?

Options are

- A.10
- B.8
- C.15
- D.12

Answer :

A is the correct answer.

```
/** * Constructs an empty list with an initial capacity of ten. */ public ArrayList()  
{ this(10); }
```

Question - 54

What is the output for the below code ?

```
package bean;  
  
public class Abc {  
    public static int index_val = 10;  
}  
  
package com;  
import static bean.Bean.index_val;  
  
public class Test1 {  
  
    public static void main(String... args) {  
        System.out.println(index_val);  
    }  
}
```

Options are

A.10

- B.compile error, index_val not defined
- C.Compile error at import static bean.Abc.index_val;
- D.None of the above

Answer :

A is the correct answer.

The static import construct allows unqualified access to static members without inheriting from the type containing the static members. J2SE 5.0 onwards it allows static import like import static bean.Abc.index_val; and can be use directly System.out.println(index_val);

Question - 55

Which of the following statement is true about jar command?

Options are

- A.The jar command creates the META-INF directory implicitly.
- B.The jar command creates the MANIFEST.MF file implicitly.
- C.The jar command would not place any of your files in META-INF directory.
- D.All of the above are true

Answer :

A is the correct answer.

All statements are true.

Question - 56

You have a class file name Test.class inside javaproject directory.

Test.java source code is :

```
import java.util.Properties;
class Test {
    public static void main (String[] args){
        Properties p = System.getProperties();
        System.out.println(p.getProperty("key1"));
    }
}
```

you have issued below commands from command prompt.

```
cd javaproject  
java -D key1=value1 Test
```

What is the output ?

Options are

- A.value1
- B.null
- C.Run successfully but no output
- D.Run fails - java.lang.NoClassDefFoundError: key1=value1

Answer :

D is the correct answer.

-D option , name=value pair must follow immediately, no spaces allowed. In this case there is space between -D and key1=value1 So java.lang.NoClassDefFoundError: key1=value1.

Question - 57

What is the output for the below code ?

```
public class A {  
    public void printValue(){  
        System.out.println("A");  
    }  
}  
  
public class B extends A {  
    public void printValue(){  
        System.out.println("B");  
    }  
}  
  
1. public class Test {  
2.     public static void main(String... args) {  
3.         A b = new B();  
4.         newValue(b);  
5.     }  
6.     public static void newValue(A a){  
7.         if(a instanceof B){  
8.             ((B)a).printValue();  
9.         }  
10.    }  
}
```

11. }

Options are

- A.A
- B.B
- C.Compilation fails with an error at line 4
- D.Compilation fails with an error at line 8

Answer :

B is the correct answer.

instanceof operator is used for object reference variables to check whether an object is of a particular type. In newValue(b); b is instance of B So works properly.

Question - 58

What is the output for the below code ?

```
1. public class Test {  
2.     static int i =5;  
3.     public static void main(String... args) {  
4.         System.out.println(i++);  
5.         System.out.println(i);  
6.         System.out.println(++i);  
7.         System.out.println(++i+i++);  
8.  
9.     }  
10. }
```

Options are

- A.5 6 7 16
- B.6 6 6 16
- C.6 6 7 16
- D.5 6 6 16

Answer :

A is the correct answer.

i++ : print value then increment (postfix - increment happens after the value of the variable is used) ++i : increment the print (prefix - increment happens before the value of the variable is used)

Question - 59

What is the output for the below code ?

```
1. public class Test {  
2.     public static void main(String... args) {  
3.         Integer i = 34;  
4.         String str = (i<21)? "jan": (i<56)? "feb": "march";  
5.         System.out.println(str);  
6.     }  
7. }
```

Options are

- A.feb
- B.jan
- C.march
- D.Compilation fails with an error at line 4

Answer :

A is the correct answer.

This is nested conditional with unbox. (*i*<21) is false goto (*i*<56), (*i*<56) is true so result is "feb".

Question - 60

What is the output ?

```
public class Test {  
  
    public static void main(String... args) {  
  
        Pattern p = Pattern.compile("a+b?c*");  
        Matcher m = p.matcher("ab");  
        boolean b = m.matches();  
        System.out.println(b);  
  
    }  
}
```

Options are

- A.true

- B.false
- C.Compile error
- D.None of the above

Answer :

A is the correct answer.

X? X, once or not at all X* X, zero or more times X+ X, one or more times

Question - 61

What is the output for the below code ?

```
1. public class Test {  
2.     public static void main(String[] args){  
3.         byte i = 128;  
4.         System.out.println(i);  
5.     }  
6. }
```

Options are

- A.128
- B.0
- C.Compilation fails with an error at line 3
- D.Compilation fails with an error at line 4

Answer :

C is the correct answer.

byte can only hold up to 127. So compiler complain about possible loss of precision.

Question - 62

What is the output for the below code ?

```
1. public class Test {  
2.     int i=8;  
3.     int j=9;  
4.     public static void main(String[] args){  
5.         add();  
6.     }  
7.     public static void add(){  
8.         int k = i+j;
```

```
9.         System.out.println(k);
10.    }
11. }
```

Options are

- A.17
- B.0
- C.Compilation fails with an error at line 5
- D.Compilation fails with an error at line 8

Answer :

D is the correct answer.

i and j are instance variable and attempting to access an instance variable from a static method. So Compilation fails .

Question - 63

Which collection class grows or shrinks its size and provides indexed access to its elements, but methods are not synchronized?

Options are

- A.java.util.ArrayList
- B.java.util.List
- C.java.util.HashSet
- D.java.util.Vector

Answer :

A is the correct answer.

ArrayList provides an index to its elements and methods are not synchronized.

Question - 64

What is the output of bellow code ?

```
public class Bean{
    private String str;

    Bean(String str ){
        this.str = str;
    }

    public String getStr() {
        return str;
    }
}
```

```

    }

    public boolean equals(Object o){
        if (!(o instanceof Bean)) {
            return false;
        }

        return ((Bean) o).getStr().equals(str);
    }

    public int hashCode() {
        return 12345;
    }

    public String toString() {
        return str;
    }
}

import java.util.HashSet;
public class Test {
    public static void main(String ... sss) {
        HashSet myMap = new HashSet();
        String s1 = new String("das");
        String s2 = new String("das");
        Bean s3 = new Bean("abcdef");
        Bean s4 = new Bean("abcdef");

        myMap.add(s1);
        myMap.add(s2);
        myMap.add(s3);
        myMap.add(s4);

        System.out.println(myMap);
    }
}

```

Options are

- A.das abcdef
- B.das abcdef das abcdef
- C.das das abcdef abcdef
- D.das

Answer :

A is the correct answer.

implemented 'equals' and 'hashCode' methods to get unique result in Set.

Question - 65

What will happen when you attempt to compile and run the following code ?

```
class A implements Runnable{
    public void run(){
        System.out.println("run-A");
    }
}

1. public class Test {
2.     public static void main(String argv[]){
3.         A a = new A();
4.         Thread t = new Thread(a);
5.         System.out.println(t.isAlive());
6.         t.start();
7.         System.out.println(t.isAlive());
8.     }
9. }
```

Options are

- A.false run-A true
- B.false run-A false
- C.true run-A true
- D.Compilation fails due to an error on line 7

Answer :

A is the correct answer.

Once the start() method is called, the thread is considered to be alive.

Question - 66

What will happen when you attempt to compile and run the following code ?

```
1. public class Test extends Thread{
2.     public static void main(String argv[]){
3.         Test t = new Test();
4.         t.run();
5.         t.start();
6.     }
7.     public void run(){

}
```

```
8.     System.out.println("run-test");
9. }
10. }
```

Options are

- A.run-test run-test
- B.run-test
- C.Compilation fails due to an error on line 4
- D.Compilation fails due to an error on line 7

Answer :

A is the correct answer.

t.run() Legal, but does not start a new thread , it is like a method call of class Test BUT t.start() create a thread and call run() method.

Question - 67

Which of the following are methods of the Thread class?

- 1) yield()
- 2) sleep(long msec)
- 3) go()
- 4) stop()

Options are

- A.1 , 2 and 4
- B.1 and 3
- C.3 only
- D.None of the above

Answer :

A is the correct answer.

Check out the Java2 Docs for an explanation

Question - 68

What notifyAll() method do?

Options are

- A.Wakes up all threads that are waiting on this object's monitor
- B.Wakes up one threads that are waiting on this object's monitor
- C.Wakes up all threads that are not waiting on this object's monitor
- D.None of the above

Answer :

A is the correct answer.

notifyAll() : Wakes up all threads that are waiting on this object's monitor.A thread waits on an object's monitor by calling one of the wait methods.

Question - 69

What is the output for the below code?

```
import java.util.NavigableMap;
import java.util.concurrent.ConcurrentSkipListMap;

public class Test {
    public static void main(String... args) {

        NavigableMap navMap = new
ConcurrentSkipListMap();

        navMap.put(4, "April");
        navMap.put(5, "May");
        navMap.put(6, "June");
        navMap.put(1, "January");
        navMap.put(2, "February");
        navMap.put(3, "March");

        navMap.pollFirstEntry();
        navMap.pollLastEntry();
        navMap.pollFirstEntry();
        System.out.println(navMap.size());

    }
}
```

Options are

- A.Compile error : No method name like pollFirstEntry() or pollLastEntry()
- B.3
- C.6
- D.None of the above

Answer :

B is the correct answer.

Y pollFirstEntry() Removes and returns a key-value mapping associated with the least key in this map, or null if the map is empty.

Y pollLastEntry() Removes and returns a key-value mapping associated with the greatest key in this map, or null if the map is empty.

Question - 70

What is the output for the below code?

```
import java.io.Console;

public class Test {
    public static void main(String... args) {

        Console con = System.console();
        boolean auth = false;

        if (con != null)
        {
            int count = 0;
            do
            {
                String uname = con.readLine(null);
                char[] pwd = con.readPassword("Enter %s's
password: ", uname);

                con.writer().write("\n\n");
            } while (!auth && ++count < 3);
        }

    }
}
```

Options are

- A.NullPointerException
- B.It works properly
- C.Compile Error : No readPassword() method in Console class.
- D.None of the above

Answer :

A is the correct answer.

\$ passing a null argument to any method in Console class will cause a NullPointerException to be thrown.

Question - 71

What is the output for the below code?

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.NavigableSet;
import java.util.TreeSet;

public class Test {
    public static void main(String... args) {

        List lst = new ArrayList ();
        lst.add(34);
        lst.add(6);
        lst.add(2);
        lst.add(8);
        lst.add(7);
        lst.add(10);

        NavigableSet nvset = new TreeSet(lst);
        System.out.println(nvset.headSet(10,true));
    }
}
```

Options are

- A.Compile error : No method name like headSet()
- B.2, 6, 7, 8, 10
- C.2, 6, 7, 8
- D.None of the above

Answer :

B is the correct answer.

headSet(10) Returns the elements elements are strictly less than 10.

headSet(10,false) Returns the elements elements are strictly less than 10.

- headSet(10,true) Returns the elements elements are strictly less than or equal to 10.

Question - 72

What is the output?

```
import java.util.ArrayList;
import java.util.List;
import java.util.NavigableSet;
```

```
import java.util.TreeSet;

public class Test {
    public static void main(String... args) {

        List lst = new ArrayList();
        lst.add(34);
        lst.add(6);
        lst.add(2);
        lst.add(8);
        lst.add(7);
        lst.add(10);

        NavigableSet nvset = new TreeSet(lst);
        System.out.println(nvset.lower(6)+" "+nvset.higher(6)+" "
"+ nvset.lower(2));

    }
}
```

Options are

- A.1 2 7 10 34 null
- B.2 7 null
- C.2 7 34
- D.1 2 7 10 34

Answer :

B is the correct answer.

lower() Returns the greatest element in this set strictly less than the given element, or

null if there is no such element.

higher() Returns the least element in this set strictly greater than the given element, or

null if there is no such element.

Question 1)

Which of the following lines will compile without warning or error.

- 1) float f=1.3;
- 2) char c="a";
- 3) byte b=257;
- 4) boolean b=null;
- 5) int i=10;

1

Question 2)

What will happen if you try to compile and run the following code

```
public class MyClass {  
    public static void main(String arguments[]) {  
        amethod(arguments);  
    }  
    public void amethod(String[] arguments) {  
        System.out.println(arguments);  
        System.out.println(arguments[1]);  
    }  
}
```

- 1) error Can't make static reference to void amethod.
- 2) error method main not correct
- 3) error array must include parameter
- 4) amethod must be declared with String

2

Question 3)

Which of the following will compile without error

1)

```
import java.awt.*;  
package Mypackage;  
class Myclass {}
```

2)

```
package MyPackage;  
import java.awt.*;  
class MyClass {}
```

3)

```
/*This is a comment */
```

```
package MyPackage;  
import java.awt.*;  
class MyClass {}
```

3

Question 4)

A byte can be of what size

- 1) -128 to 127
- 2) (-2 power 8)-1 to 2 power 8
- 3) -255 to 256
- 4) depends on the particular implementation of the Java Virtual machine

4

Question 5)

What will be printed out if this code is run with the following command line?

java myprog good morning

```
public class myprog{  
    public static void main(String argv[]){  
        System.out.println(argv[2]);  
    }  
}
```

- 1) myprog
- 2) good
- 3) morning
- 4) Exception raised: "java.lang.ArrayIndexOutOfBoundsException: 2"

5

Question 6)

Which of the following are keywords or reserved words in Java?

- 1) if
- 2) then
- 3) goto
- 4) while
- 5) case

6

Question 7)

Which of the following are legal identifiers

- 1) 2variable
- 2) variable2
- 3) _whatavariable
- 4) _3_
- 5) \$anothervar
- 6) #myvar

7

Question 8)

What will happen when you compile and run the following code?

```
public class MyClass{
    static int i;
    public static void main(String argv[]){
        System.out.println(i);
    }
}
```

1) Error Variable i may not have been initialized
2) null
3) 1
4) 0
8

Question 9)

What will happen if you try to compile and run the following code?

```
public class Q {
    public static void main(String argv[]){
        int anar[] = new int[]{1,2,3};
        System.out.println(anar[1]);
    }
}
```

- 1) 1
2) Error anar is referenced before it is initialized
3) 2
4) Error: size of array must be defined

9

Question 10)

What will happen if you try to compile and run the following code?

```
public class Q {
    public static void main(String argv[]){
        int anar[] = new int[5];
        System.out.println(anar[0]);
    }
}
```

- 1) Error: anar is referenced before it is initialized
2) null
3) 0
4) 5
10
-

Question 11)

What will be the result of attempting to compile and run the following code?

```
abstract class MineBase {
```

```
abstract void amethod();
static int i;
}
public class Mine extends MineBase {
public static void main(String argv[]){
int[] ar=new int[5];
for(i=0;i < ar.length;i++)
System.out.println(ar[i]);
}
}
```

- 1) a sequence of 5 0's will be printed
- 2) Error: ar is used before it is initialized
- 3) Error Mine must be declared abstract
- 4) IndexOutOfBoundsException

11

Question 12)

What will be printed out if you attempt to compile and run the following code ?

```
int i=1;
switch (i) {
case 0:
System.out.println("zero");
break;
case 1:
System.out.println("one");
case 2:
System.out.println("two");
default:
System.out.println("default");
}
```

- 1) one
- 2) one, default
- 3) one, two, default
- 4) default

12

Question 13)

What will be printed out if you attempt to compile and run the following code?

```
int i=9;
switch (i) {
default:
System.out.println("default");
case 0:
System.out.println("zero");
break;
case 1:
```

```
System.out.println("one");
case 2:
    System.out.println("two");
}
1) default
2) default, zero
3) error default clause not defined
4) no output displayed
13
```

Question 14)

Which of the following lines of code will compile without error

- 1)
int i=0;
if(i) {
 System.out.println("Hello");
}
- 2)
boolean b=true;
boolean b2=true;
if(b==b2) {
 System.out.println("So true");
}
- 3)
int i=1;
int j=2;
if(i==1|| j==2)
 System.out.println("OK");
- 4)
int i=1;
int j=2;
if(i==1 &| j==2)

 System.out.println("OK");

Question 15)

What will be output if you try to compile and run the following code, but there is no file called Hello.txt in the current directory?.

```
import java.io.*;
public class Mine {
    public static void main(String argv[]){
        Mine m=new Mine();
        System.out.println(m.amethod());
    }
    public int amethod() {
```

```

try {
    FileInputStream dis=new FileInputStream("Hello.txt");
}catch (FileNotFoundException fne) {
    System.out.println("No such file found");
    return -1;
}catch(IOException ioe) {
} finally{
    System.out.println("Doing finally");
}

return 0;
}

}

```

1) No such file found
 2 No such file found ,-1
 3) No such file found, Doing finally, -1
 4) 0
 15

Question 16)

Which of the following statements are true?

- 1) Methods cannot be overridden to be more private
 - 2) static methods cannot be overloaded
 - 3) private methods cannot be overloaded
 - 4) An overloaded method cannot throw exceptions not checked in the base class
- 16
-

Question 17)

What will happen if you attempt to compile and run the following code?

```

class Base {}
class Sub extends Base {}
class Sub2 extends Base {}
public class CEx{
    public static void main(String argv[]){
        Base b=new Base();
        Sub s=(Sub) b;
    }
}

```

- 1) Compile and run without error
- 2) Compile time Exception
- 3) Runtime Exception

17

Question 18)

Which of the following statements are true?

- 1) System.out.println(-1 >>> 2); will output a result larger than 10
- 2) System.out.println(-1 >>> 2); will output a positive number
- 3) System.out.println(2 >> 1); will output the number 1
- 4) System.out.println(1 <<< 2); will output the number 4

18

Question 19)

What will happen when you attempt to compile and run the following code?

```
public class Tux extends Thread{
```

```
    static String sName = "vandaleur";
    public static void main(String argv[]){
        Tux t = new Tux();
        t.piggy(sName);
        System.out.println(sName);

    }
    public void piggy(String sName){
        sName = sName + " wiggy";
        start();
    }
    public void run(){

        for(int i=0;i < 4; i++){
            sName = sName + " " + i;

        }
    }
}
```

- 1) Compile time error
- 2) Compilation and output of "vandaleur wiggy"
- 3) Compilation and output of "vandaleur wiggy 0 1 2 3"
- 4) Compilation and output of either "vandaleur", "vandaleur 0", "vandaleur 0 1" "vandaleur 0 1 2" or "vandaleur 0 1 2 3"

19

Question 20)

What will be displayed when you attempt to compile and run the following code

```
//Code start
import java.awt.*;
public class Butt extends Frame{
    public static void main(String argv[]){
```

```

        Butt MyBut=new Butt();
    }
    Butt(){
        Button HelloBut=new Button("Hello");
        Button ByeBut=new Button("Bye");
        add(HelloBut);
        add(HelloBut);
        setSize(300,300);
        setVisible(true);
    }
}
//Code end

```

- 1) Two buttons side by side occupying all of the frame, Hello on the left and Bye on the right
- 2) One button occupying the entire frame saying Hello
- 3) One button occupying the entire frame saying Bye
- 4) Two buttons at the top of the frame one saying Hello the other saying Bye

20

Question 21)

What will be output by the following code?

```

public class MyFor{
    public static void main(String argv[]){
        int i;
        int j;
        outer:
        for (i=1;i <3;i++)
            inner:
            for(j=1; j<3; j++) {
                if (j==2)
                    continue outer;
                System.out.println("Value for i=" + i + " Value for j=" +j);
            }
    }
}

```

- 1) Value for i=1 Value for j=1
- 2) Value for i=2 Value for j=1
- 3) Value for i=2 Value for j=2
- 4) Value for i=3 Value for j=1

21

Question 22)

Which statement is true of the following code?

```

public class Agg{

```

```

public static void main(String argv[]){
    Agg a = new Agg();
    a.go();
}
public void go(){
    DSRoss ds1 = new DSRoss("one");
    ds1.start();
}
}

class DSRoss extends Thread{
private String sTname="";
DSRoss(String s){
    sTname = s;
}
public void run(){
    notwait();
    System.out.println("finished");
}
public void notwait(){
    while(true){
        try{
            System.out.println("waiting");
            wait();
        }catch(InterruptedException ie){ }
        System.out.println(sTname);
        notifyAll();
    }
}
}

1) It will cause a compile time error
2) Compilation and output of "waiting"
3) Compilation and output of "waiting" followed by "finished"
4) Runtime error, an exception will be thrown

```

22

Question 23)

Which of the following methods can be legally inserted in place of the comment
//Method Here ?

```

class Base{
    public void amethod(int i) { }
}

```

```
public class Scope extends Base{  
    public static void main(String argv[]){  
    }  
    //Method Here  
    }  
1) void amethod(int i) throws Exception {}  
2) void amethod(long i) throws Exception {}  
3) void amethod(long i){}  
4) public void amethod(int i) throws Exception {}  
23
```

Question 24)

Which of the following will output -4.0

- 1) System.out.println(Math.floor(-4.7));
- 2) System.out.println(Math.round(-4.7));
- 3) System.out.println(Math.ceil(-4.7));
- 4) System.out.println(Math.min(-4.7));

24

Question 25)

What will happen if you attempt to compile and run the following code?

```
Integer ten=new Integer(10);  
Long nine=new Long (9);  
System.out.println(ten + nine);  
int i=1;  
System.out.println(i + ten);  
1) 19 followed by 20  
2) 19 followed by 11  
3) Compile time error  
4) 10 followed by 1  
25
```

Question 26)

If you run the code below, what gets printed out?

```
String s=new String("Bicycle");  
int iBegin=1;  
char iEnd=3;  
System.out.println(s.substring(iBegin,iEnd));  
1) Bic  
2) ic  
3) icy  
4) error: no method matching substring(int,char)  
26
```

Question 27)

If you wanted to find out where the position of the letter v (ie return 2) in the string s containing "Java", which of the following could you use?

- 1) mid(2,s);
- 2) charAt(2);
- 3) s.indexOf('v');
- 4) indexOf(s,'v');

Answer to Question 27

Question 28)

Given the following declarations

```
String s1=new String("Hello")
String s2=new String("there");
String s3=new String();
```

Which of the following are legal operations?

- 1) s3=s1 + s2;
- 2) s3=s1-s2;
- 3) s3=s1 & s2;
- 4) s3=s1 && s2

28

Question 29)

What is the result of the following operation?

```
System.out.println(4 | 3);
```

- 1) 6
- 2) 0
- 3) 1
- 4) 7

29

Question 30)

```
public class MyClass1 {
    public static void main(String argv[]){ }
    /*Modifier at XX */ class MyInner { }
}
```

What modifiers would be legal at XX in the above code?

- 1) public
- 2) private
- 3) static
- 4) friend

30

Question 31)

What will happen when you attempt to compile and run the following code?

```

public class Holt extends Thread{
    private String sThreadName;
    public static void main(String argv[]){
        Holt h = new Holt();
        h.go();
    }
    Holt(){}
}

Holt(String s){
    sThreadName = s;
}
public String getThreadName(){
    return sThreadName;
}

public void go(){
    Holt first = new Holt("first");
    first.start();
    Holt second = new Holt("second");
    second.start();
}

public void start(){
    for(int i = 0; i < 2; i ++){
        System.out.println(getThreadName() +i);
        try{
            Thread.sleep(100);
        } catch(InterruptedException e){System.out.println(e.getMessage());}
    }
}
}

```

- 1) Compile time error
- 2) Output of first0, second0, first0, second1
- 3) Output of first0, first1, second0, second1
- 4) Runtime error

31

Question 32)

An Applet has its Layout Manager set to the default of FlowLayout. What code would be correct to change to another Layout Manager.

- 1) setLayoutManager(new GridLayout());
- 2) setLayout(new GridLayout(2,2));
- 3) setGridLayout(2,2);
- 4) setBorderLayout();

32

Question 33)

What will happen when you attempt to compile and run the following code?.

```
class Background implements Runnable{  
    int i=0;  
    public int run(){  
        while(true){  
            i++;  
            System.out.println("i="+i);  
        } //End while  
        return 1;  
    }//End run
```

}//End class

- 1) It will compile and the run method will print out the increasing value of i.
- 2) It will compile and calling start will print out the increasing value of i.
- 3) The code will cause an error at compile time.
- 4) Compilation will cause an error because while cannot take a parameter of true.

33

Question 34)

Which of the following statements about this code are true?

```
public class Morecombe{  
    public static void main(String argv[]){  
        Morecombe m = new Morecombe();  
        m.go(new Turing());  
    }  
    public void go(Turing t){  
        t.start();  
    }  
}  
class Turing extends Thread{  
    public void run(){  
        for(int i=0; i<2; i++){  
            System.out.println(i);  
        }  
    }  
}
```

- 1) Compilation error due to malformed parameter to go method
- 2) Compilation error, class Turing has no start method
- 3) Compilation and output of 0 followed by 1

4) Compilation but runtime error

34

Question 35)

What will be the result when you attempt to compile and run the following code?.

```
public class Conv{  
    public static void main(String argv[]){  
        Conv c=new Conv();  
        String s=new String("ello");  
        c.amethod(s);  
    }  
  
    public void amethod(String s){  
        char c='H';  
        c+=s;  
        System.out.println(c);  
    }  
}
```

1) Compilation and output the string "Hello"
2) Compilation and output the string "ello"
3) Compilation and output the string elloH
4) Compile time error

35

Question 36)

Given the following code, what test would you need to put in place of the comment line?

//place test here

to result in an output of the string

Equal

```
public class EqTest{  
    public static void main(String argv[]){  
        EqTest e=new EqTest();  
    }  
  
    EqTest(){  
        String s="Java";  
        String s2="java";  
        //place test here {  
            System.out.println("Equal");  
        }else  
        {  
            System.out.println("Not equal");  
        }  
    }  
}
```

- 1) if(s==s2)
 - 2) if(s.equals(s2))
 - 3) if(s.equalsIgnoreCase(s2))
 - 4) if(s.noCaseMatch(s2))
- 36
-

Question 37)

Given the following code

```
import java.awt.*;  
public class SetF extends Frame{  
    public static void main(String argv[]){  
        SetF s=new SetF();  
        s.setSize(300,200);  
        s.setVisible(true);  
    }  
}
```

How could you set the frame surface color to pink

- 1)s.setBackground(Color.pink);
- 2)s.setColor(PINK);
- 3)s.Background(pink);
- 4)s.color=Color.pink

Question 38)

How can you change the current working directory using an instance of the File class called FileName?

- 1) FileName.chdir("DirName")
 - 2) FileName.cd("DirName")
 - 3) FileName.cwd("DirName")
 - 4) The File class does not support directly changing the current directory.
- 38
-

Question 39)

If you create a TextField with a constructor to set it to occupy 5 columns, what difference will it make if you use it with a proportional font (ie Times Roman) or a fixed pitch typewriter style font (Courier).

- 1)With a fixed font you will see 5 characters, with a proportional it will depend on the width of the characters
- 2)With a fixed font you will see 5 characters, with a proportional it will cause the field to expand to fit the text
- 3)The columns setting does not affect the number of characters displayed
- 4)Both will show exactly 5 characters

Question 40)

Given the following code how could you invoke the Base constructor that will print out the string "base constructor";

```

class Base{
    Base(int i){
        System.out.println("base constructor");
    }
    Base(){
    }
}

public class Sup extends Base{
    public static void main(String argv[]){
        Sup s= new Sup();
        //One
    }
    Sup()
    {
        //Two
    }

    public void derived()
    {
        //Three
    }
}

```

1) On the line After //One put Base(10);
 2) On the line After //One put super(10);
 3) On the line After //Two put super(10);
 4) On the line After //Three put super(10);

Question 41)

Given the following code what will be output?

```

public class Pass{
    static int j=20;
    public static void main(String argv[]){
        int i=10;
        Pass p = new Pass();
        p.amethod(i);
        System.out.println(i);
        System.out.println(j);
    }

    public void amethod(int x){
        x=x*2;
        j=j*2;
    }
}

```

```
    }
}
1) Error: amethod parameter does not match variable
2) 20 and 40
3) 10 and 40
4) 10, and 20
```

Question 42)

What code placed after the comment //For loop would result in the population of every element of the array ia[] with a value from variable i.?

```
public class Lin{
    public static void main(String argv[]){
        Lin l = new Lin();
        l.amethod();
    }
    public void amethod(){
        int ia[] = new int[4];
        //Start For loop
        {
            ia[i]=i;
            System.out.println(ia[i]);
        }
    }
}
1) for(int i=0; i < ia.length() -1; i++)
2) for (int i=0; i< ia.length(); i++)
3) for(int i=1; i < 4; i++)
4) for(int i=0; i< ia.length;i++)
```

Question 43)

What will be the result when you try to compile and run the following code?

```
private class Base{
    Base(){
        int i = 100;
        System.out.println(i);
    }
}

public class Pri extends Base{
    static int i = 200;
    public static void main(String argv[]){
        Pri p = new Pri();
        System.out.println(i);
    }
}
```

```
}
```

1) Error at compile time
2) 200
3) 100 followed by 200
4) 100
43

Question 44)

What will the following code print out?

```
public class Oct{  
    public static void main(String argv[]){  
        Oct o = new Oct();  
        o.amethod();  
    }  
    public void amethod(){  
        int oi= 012;  
        System.out.println(oi);  
    }  
}  
1)12  
2)012  
3)10  
4)10.0
```

Question 45

What will happen when you try compiling and running this code?

```
public class Ref{  
    public static void main(String argv[]){  
        Ref r = new Ref();  
        r.amethod(r);  
    }  
    public void amethod(Ref r){  
        int i=99;  
        multi(r);  
        System.out.println(i);  
    }  
    public void multi(Ref r){  
        r.i = r.i*2;  
    }  
}  
1) Error at compile time  
2) An output of 99  
3) An output of 198  
4) An error at runtime  
45
```

Question 46)

You need to create a class that will store unique object elements. You do not need to sort these elements but they must be unique.

What interface might be most suitable to meet this need?

- 1) Set
- 2) List
- 3) Map
- 4) Vector

46

Question 47)

Which of the following will successfully create an instance of the Vector class and add an element?

1) Vector v=new Vector(99);
v[1]=99;

2) Vector v=new Vector();
v.addElement(99);

3) Vector v=new Vector();
v.add(99);

4 Vector v=new Vector(100);
v.addElement("99");

Question 48)

You have created a simple Frame and overridden the paint method as follows

```
public void paint(Graphics g){  
    g.drawString("Dolly",50,10);  
}
```

What will be the result when you attempt to compile and run the program?

- 1) The string "Dolly" will be displayed at the centre of the frame
- 2) An error at compilation complaining at the signature of the paint method
- 3) The lower part of the word Dolly will be seen at the top of the frame, with the top hidden.
- 4) The string "Dolly" will be shown at the bottom of the frame.

48

Question 49)

What will be the result when you attempt to compile this program?

```
public class Rand{  
    public static void main(String argv[]){
```

```

        int iRand;
        iRand = Math.random();
        System.out.println(iRand);
    }
}

```

1) Compile time error referring to a cast problem
 2) A random number between 1 and 10
 3) A random number between 0 and 1
 4) A compile time error about random being an unrecognised method

Question 50)

Given the following code

```

import java.io.*;
public class Th{
    public static void main(String argv[]){
        Th t = new Th();
        t.amethod();
    }
    public void amethod(){
        try{
            ioCall();
        }catch( IOException ioe){ }
    }
}

```

What code would be most likely for the body of the ioCall method

- 1) public void ioCall ()throws IOException{
 DataInputStream din = new DataInputStream(System.in);
 din.readChar();
 }
 - 2) public void ioCall ()throw IOException{
 DataInputStream din = new DataInputStream(System.in);
 din.readChar();
 }
 - 3) public void ioCall (){
 DataInputStream din = new DataInputStream(System.in);
 din.readChar();
 }
 - 4) public void ioCall throws IOException(){
 DataInputStream din = new DataInputStream(System.in);
 din.readChar();
 }
- }
-

Question 51)

What will happen when you compile and run the following code?

```
public class Scope{
```

```
private int i;
public static void main(String argv[]){
    Scope s = new Scope();
    s.amethod();
}//End of main
public static void amethod(){
    System.out.println(i);
}//end of amethod
}//End of class
```

- 1) A value of 0 will be printed out
- 2) Nothing will be printed out
- 3) A compile time error
- 4) A compile time error complaining of the scope of the variable i

Question 52)

You want to lay out a set of buttons horizontally but with more space between the first button and the rest. You are going to use the GridBagLayout manager to control the way the buttons are set out. How will you modify the way the GridBagLayout acts in order to change the spacing around the first button?

- 1) Create an instance of the GridBagConstraints class, call the weightx() method and then pass the GridBagConstraints instance with the component to the setConstraints method of the GridBagLayout class.
- 2) Create an instance of the GridBagConstraints class, set the weightx field and then pass the GridBagConstraints instance with the component to the setConstraints method of the GridBagLayout class.
- 3) Create an instance of the GridBagLayout class, set the weightx field and then call the setConstraints method of the GridBagLayoutClass with the component as a parameter.
- 4) Create an instance of the GridBagLayout class, call the setWeightx() method and then pass the GridBagConstraints instance with the component to the setConstraints method of the GridBagLayout class.

Question 53)

Which of the following can you perform using the File class?

- 1) Change the current directory
- 2) Return the name of the parent directory
- 3) Delete a file
- 4) Find if a file contains text or binary information

53

Question 54)

Which statement is true of the following code?

```
public class Rpcraven{
```

```

public static void main(String argv[]){
    Pmcraven pm1 = new Pmcraven("One");
    pm1.run();
    Pmcraven pm2 = new Pmcraven("Two");
    pm2.run();

}

class Pmcraven extends Thread{
private String sTname="";
Pmcraven(String s){
    sTname = s;

}
public void run(){
    for(int i =0; i < 2 ; i++){
        try{
            sleep(1000);
        }catch(InterruptedException e){ }

        yield();
        System.out.println(sTname);
    }
}

1) Compile time error, class Rocraven does not import java.lang.Thread
2) Output of One One Two Two
3) Output of One Two One Two
4) Compilation but no output at runtime
54

```

Question 55)

You are concerned that your program may attempt to use more memory than is available. To avoid this situation you want to ensure that the Java Virtual Machine will run its garbage collection just before you start a complex routine. What can you do to be certain that garbage collection will run when you want .

- 1) You cannot be certain when garbage collection will run
- 2) Use the Runtime.gc() method to force garbage collection
- 3) Ensure that all the variables you require to be garbage collected are set to null
- 4) Use the System.gc() method to force garbage collection

55

Question 56)

You are using the GridBagLayout manager to place a series of buttons on a Frame. You want to make the size of one of the buttons bigger than the text it contains. Which of the following will allow you to do that?

- 1) The GridBagLayout manager does not allow you to do this
- 2) The setFill method of the GridBagLayout class
- 3) The setFill method of the GridBagConstraints class
- 4) The fill field of the GridBagConstraints class

56

Question 57)

Which of the following most closely describes a bitset collection?

- 1) A class that contains groups of unique sequences of bits
- 2) A method for flipping individual bits in instance of a primitive type
- 3) An array of boolean primitives that indicate zeros or ones
- 4) A collection for storing bits as on-off information, like a vector of bits

57

Question 58)

You have these files in the same directory. What will happen when you attempt to compile and run Class1.java if you have not already compiled Base.java

```
//Base.java
package Base;
class Base{
    protected void amethod(){
        System.out.println("amethod");
    }//End of amethod
}//End of class base
package Class1;
//Class1.java
public class Class1 extends Base{
    public static void main(String argv[]){
        Base b = new Base();
        b.amethod();
    }//End of main
}//End of Class1
```

- 1) Compile Error: Methods in Base not found
- 2) Compile Error: Unable to access protected method in base class
- 3) Compilation followed by the output "amethod"
- 4) Compile error: Superclass Class1.Base of class Class1.Class1 not found

58

Question 59)

What will happen when you attempt to compile and run the following code

```
class Base{  
    private void amethod(int iBase){  
        System.out.println("Base.amethod");  
    }  
}
```

```
class Over extends Base{  
    public static void main(String argv[]){  
        Over o = new Over();  
        int iBase=0;  
        o.amethod(iBase);  
    }  
  
    public void amethod(int iOver){  
        System.out.println("Over.amethod");  
    }  
}
```

- 1) Compile time error complaining that Base.amethod is private
- 2) Runtime error complaining that Base.amethod is private
- 3) Output of "Base.amethod"
- 4) Output of "Over.amethod"

59

Question 60)

You are creating an applet with a Frame that contains buttons. You are using the GridBagLayout manager and you have added Four buttons. At the moment the buttons appear in the centre of the frame from left to right. You want them to appear one on top of the other going down the screen. What is the most appropriate way to do this.

- 1) Set the gridy value of the GridBagConstraints class to a value increasing from 1 to 4
- 2) set the fill value of the GridBagConstraints class to VERTICAL
- 3) Set the ipady value of the GridBagConstraints class to a value increasing from 0 to 4
- 4) Set the fill value of the GridBagLayouts class to GridBag.VERTICAL

60

If you have a copy of the Roberts and Heller Java2 Guide that says the exam does not cover the GridBagLayout, this is an error. You can confirm this by looking at the online errata at

Answers

Answer 1)

Objective 4.5)

5) int i=10;

explanation:

1) float f=1.3;

Will not compile because the default type of a number with a floating point component is a double. This would compile with a cast as in

float f=(float) 1.3

2) char c="a";

Will not compile because a char (16 bit unsigned integer) must be defined with single quotes. This would compile if it were in the form

char c='a';

3) byte b=257;

Will not compile because a byte is eight bits. Take of one bit for the sign component you can define numbers between

-128 to +127

4) a boolean value can either be true or false, null is not allowed

.

Answer 2)

Objective 4.1

1) Can't make static reference to void amethod.

Because main is defined as static you need to create an instance of the class in order to call any non-static methods. Thus a typical way to do this would be.

```
MyClass m=new MyClass();
m.amethod();
```

Answer 2 is an attempt to confuse because the convention is for a main method to be in the form

String argv[]

That argv is just a convention and any acceptable identifier for a string array can be used. Answers 3 and 4 are just nonsense.

Answer 3)

Objective 4.1)

2 and 3 will compile without error.

1 will not compile because any package declaration must come before any other code.

Comments may appear anywhere

Answer 4)

4)

Objective 4.5)

1) A byte is a signed 8 bit integer.

Answer 5)

5)

Objective 4.2)

4) Exception raised: "java.lang.ArrayIndexOutOfBoundsException: 2"

Unlike C/C++ java does not start the parameter count with the program name. It does however start from zero. So in this case zero starts with good, morning would be 1 and there is no parameter 2 so an exception is raised.

Answer 6)

6)

Objective 4.3)

1) if

3) goto

4) while

5) case

then is not a Java keyword, though if you are from a VB background you might think it was. Goto is a reserved word in Java.

Answer 7)

7)

Objective 4.1)

2) variable2

3) _whatavariable

4) _3_

5) \$anothervar

An identifier can begin with a letter (most common) or a dollar sign(\$) or an underscore(_). An identifier cannot start with anything else such as a number, a hash, # or a dash -. An identifier cannot have a dash in its body, but it may have an underscore _. Choice 4) _3_ looks strange but it is an acceptable, if unwise form for an identifier.

Answer 8)

8)

Objective 4.4)

4) 0

Class level variables are always initialised to default values. In the case of an int this will be 0. Method level variables are not given default values and if you attempt to use one before it has been initialised it will cause the Error Variable i may not have been initialized type of error.

Answer 9)

9)

Objective 4.4)

3) 2

No error will be triggered.

Like in C/C++, arrays are always referenced from 0. Java allows an array to be populated at creation time. The size of array is taken from the number of initializers. If you put a size within any of the square brackets you will get an error.

Answer 10)

10)

Objective 4.4)

3) 0

Arrays are always initialised when they are created. As this is an array of ints it will be initialised with zeros.

Answer 11)

11)

Objective 1.2

3) Error Mine must be declared abstract

A class that contains an abstract method must itself be declared as abstract. It may however contain non abstract methods. Any class derived from an abstract class must either define all of the abstract methods or be declared abstract itself.

Answer 12)

12)

Objective 2.1)

3) one, two, default

Code will continue to fall through a case statement until it encounters a break.

Answer 13)

13)

Objective 4.1)

2) default, zero

Although it is normally placed last the default statement does not have to be the last item as you fall through the case block. Because there is no case label found matching the expression the default label is executed and the code continues to fall through until it encounters a break.

Answer 14)

14)

Objective 5.1

2,3

Example 1 will not compile because if must always test a boolean. This can catch out C/C++ programmers who expect the test to be for either 0 or not 0.

Answer 15)

15)

Objective 11.5)

3) No such file found, doing finally, -1

The no such file found message is to be expected, however you can get caught out if you are not aware that the finally clause is almost always executed, even if there is a return statement.

Answer 16)

16)

Objective 6.2)

1) Methods cannot be overridden to be more private

Static methods cannot be overridden but they can be overloaded. If you have doubts about that statement, please follow and read carefully the link given to the Sun tutorial below. There is no logic or reason why private methods should not be overloaded. Option 4 is a jumbled up version of the limitations of exceptions for overridden methods

Answer 17)

17)

Objective 6.2)

3) Runtime Exception

Without the cast to sub you would get a compile time error. The cast tells the compiler that you really mean to do this and the actual type of b does not get resolved until runtime. Casting down the object hierarchy is a problem, as the compiler cannot be sure what has been implemented in descendent classes. Casting up is not a problem because sub classes will have the features of the base classes. This can feel counter intuitive if you are aware that with primitives casting is allowed for widening operations (ie byte to int).

Answer 18)

18)

Objective 5.1)

1) System.out.println(-1 >>> 2); will output a result larger than 10

2) System.out.println(-1 >>> 2); will output a positive number

3) System.out.println(2 >> 1); will output the number 1

You can test this with the following class

```
public class shift{  
    static int i=2;  
    public static void main(String argv[]){  
        System.out.println( -1 >>> 2);  
        System.out.println( -1 >>> 2);  
        System.out.println( 2 >> 1);  
    }  
}
```

Java does not have a <<< operator. The operation 1 << 2 would output 4
Because of the way twos complement number representation works the unsigned right shift operation means a small shift in a negative number can return a very large value so the output of option 1 will be much larger than 10.
The unsigned right shift places no significance on the leading bit that indicates the sign. For this shift the value 1 of the bit sign is replaced with a zero turning the result into a positive number for option 2.

Answer 19)

19)

Objective 7.1)

4) Compilation and output of either "vandaleur", "vandaleur 0", "vandaleur 0 1"
"vandaleur 0 1 2" or "vandaleur 0 1 2 3"

If that seems a vague answer it is because you cannot be certain of the system that the underlying OS uses for allocating cycles for a Thread. The chances are that once the thread has been spun off in the call to start in the method piggy the main method will run to completion and the value of sName will still be vandeluer before the Thread modifies it. You cannot be certain of this though.

Answer 20)

20)

Objective 8.1)

3) One button occupying the entire frame saying Bye

The default layout manager for a Frame is a border layout. If directions are not given (ie North, South, East or West), any button will simply go in the centre and occupy all the space. An additional button will simply be placed over the previous button. What you would probably want in a real example is to set up a flow layout as in
setLayout(new FlowLayout());

Which would allow the buttons to both appear side by side, given the appropriate font and size.

Applets and panels have a default FlowLayout manager

Answer 21)

21)

Objective 2.2)

1,2

Value for i=1 Value for j=1
Value for i=2 Value for j=1

The statement continue outer causes the code to jump to the label outer and the for loop increments to the next number.

Answer 22)

22)

Objective 7.3)

4) Runtime error, an exception will be thrown

A call to wait/notify must be within synchronized code. With JDK1.2 this code throws the error message

java.lang.IllegalMonitorStateException: current thread not owner

at java.lang.Object.wait(Native Method)

at java.lang.Object.wait(Object.java:424)

at DSRoss.notwait(Compiled Code)

at DSRoss.run(Agg.java:21)

Answer 23)

23)

Objective 2.3)

2,3

Options 1, & 4 will not compile as they attempt to throw Exceptions not declared in the base class. Because options 2 and 3 take a parameter of type long they represent overloading not overriding and there is no such limitations on overloaded methods.

Answer 24)

24)

Objective 9.1)

3) System.out.println(Math.ceil(-4.7));

Options 1 and 2 will produce -5 and option 4 will not compile because the min method requires 2 parameters.

Answer 25)

25

Objective 4.5)

3) Compile time error

The wrapper classes cannot be used like primitives.

Depending on your compiler you will get an error that says something like "Error: Can't convert java.lang.Integer". Wrapper classes have similar names to primitives but all start with upper case letters.

Thus in this case we have int as a primitive and Integer as a wrapper. The objectives do not specifically mention the wrapper classes but don't be surprised if they come up.

Answer 26)

26)

Objective 4.5)

2) ic

This is a bit of a catch question. Anyone with a C/C++ background would figure out that addressing in strings starts with 0 so that 1 corresponds to i in the string Bicycle. The catch is that the second parameter returns the endcharacter minus 1. In this case it means instead of the "icy" being returned as intuition would expect it is only "ic".

Answer 27)

27)

Objective 9.2)

3) s.indexOf('v');

charAt returns the letter at the position rather than searching for a letter and returning the position, MID is just to confuse the Basic Programmers, indexOf(s,'v'); is how some future VB/J++ nightmare hybrid, might perform such a calculation.

Answer 28)

Objective 5.1)

28

1) s3=s1 + s2;

Java does not allow operator overloading as in C++, but for the sake of convenience the + operator is overridden for strings.

Answer 29)

29)

Objective 5.3)

4) 7

The | is known as the Or operator, you could think of it as the either/or operator. Turning the numbers into binary gives

4=100

3=011

For each position, if either number contains a 1 the result will contain a result in that position. As every position contains a 1 the result will be

111

Which is decimal 7.

Answer 30)

30

Objective 4.1)

1,2,3

public, private, static are all legal access modifiers for this inner class.

Answer 31)

31

Objective 7.1

3) Output of first0, first1, second0, second1

Note that this code overrides and calls the start method. If you wished to get the output mixed you would need to override the run method but call the start method.

Answer 32)

Objective 8.1)

2) setLayout(new GridLayout(2,2));

Changing the layout manager is the same for an Applet or an application. Answer 1 is wrong though it might have been a reasonable name for the designers to choose. Answers 3 and 4 are incorrect because changing the layout manager always requires an instance of one of the Layout Managers and these are bogus methods.

Instead of creating the anonymous instance of the Layout manager as in option 2 you can also create a named instance and pass that as a parameter. This is often what automatic code generators such as Borland/Inprise JBuilder do.

08_01Tut.htm

Answer 33)

33)

Objective 7.1)

3) The code will cause an error at compile time

The error is caused because run should have a void not an int return type.

Any class that implements an interface must create a method to match all of the methods in the interface. The Runnable interface has one method called run that has a void return type. The sun compiler gives the error

Method redefined with different return type: int run() was defined as void run();

Answer 34)

34)

Objective 7.1)

3) Compilation and output of 0 followed by 1

The creation of an anonymous class as a parameter to go is fairly strange as you would expect it to override a method in its parent class (Turing). You don't have to though. The fact that class Turing extends Thread means the anonymous instance that is passed to go has a start method which then calls the run method.

07_01Tut.htm

Answer 35)

35

Objective 5.1)

4) Compile time error

The only operator overloading offered by java is the + sign for the String class. A char is a 16 bit integer and cannot be concatenated to a string with the + operator.

Answer 36)

36

Objective 5.2)

3) if(s.equalsIgnoreCase(s2))

String comparison is case sensitive so using the equals string method will not return a match. Using the==operator just compares where memory address of the references and noCaseMatch was just something I made up to give me a fourth slightly plausible option.

Answer 37)

37

Objective 8.1)

1) s.setBackground(Color.pink);

For speakers of the more British spelt English note that there is no letter u in Color. Also the constants for colors are in lower case.

08_01Tut.htm

Answer 38)

38)

Objective 11.1)

4) The File class does not support directly changing the current directory.

This seems rather surprising to me, as changing the current directory is a very common requirement. You may be able to get around this limitation by creating a new instance of the File class passing the new directory to the constructor as the path name.

011_01Tut.htm

Answer 39)

39)

Objective 8.1)

1)With a fixed font you will see 5 characters, with a proportional it will depend on the width of the characters

With a proportional font the letter w will occupy more space than the letter i. So if you have all wide characters you may have to scroll to the right to see the entire text of a TextField.

08_01Tut.htm

Answer 40)

40)

Objective 6.2

3) On the line After //Two put super(10);

Constructors can only be invoked from within constructors.

Answer 41)

41)

Objective 5.4)

3) 10 and 40

when a parameter is passed to a method the method receives a copy of the value. The method can modify its value without affecting the original copy. Thus in this example when the value is printed out the method has not changed the value.

05_04Tut.htm

Answer 42)

42

Objective 1.1

4) for(int i=0; i< ia.length;i++)

Although you could control the looping with a literal number as with the number 4 used in option 3, it is better practice to use the length property of an array. This provides against bugs that might result if the size of the array changes. This question also checks that you know that arrays starts from zero and not One as option 3 starts from one.

Remember that array length is a field and not a function like the String length() method.

01_01Tut.htm

Answer 43)

43)

Objective 1.2

1) Error at compile time

This is a slightly sneaky one as it looks like a question about constructors, but it is attempting to test knowledge of the use of the private modifier. A top level class cannot be defined as private. If you didn't notice the modifier private, remember in the exam to be real careful to read every part of the question.

01_02Tut.htm

Answer 44)

44

Objective 4.5)

3)10

The name of the class might give you a clue with this question, Oct for Octal. Prefixing a number with a zero indicates that it is in Octal format. Thus when printed out it gets converted to base ten. 012 in octal means the first column from the right has a value of 2 and the next along has a value of one times eight. In decimal that adds up to 10.

Answer 45)

45

Objective 1.2)

1) Error at compile time

The variable i is created at the level of amethod and will not be available inside the method multi.

[01_02Tut.htm](#)

Answer 46)

46

Objective 10.1)

1) Set

The Set interface ensures that its elements are unique, but does not order the elements. In reality you probably wouldn't create your own class using the Set interface. You would be more likely to use one of the JDK classes that use the Set interface such as HashSet or TreeSet.

[10_01Tut.htm](#)

Answer 47)

47

Objective 10.1)

4) Vector v=new Vector(100);

v.addElement("99")

A vector can only store objects not primitives. The parameter "99" for the addElement method passes a string object to the Vector. Option 1) creates a vector OK but then uses array syntax to attempt to assign a primitive. Option 2 also creates a vector then uses correct Vector syntax but falls over when the parameter is a primitive instead of an object.

[10_01Tut.htm](#)

Answer 48)

Objective 8.1)

48

3) The lower part of the word Dolly will be seen at the top of the form

The Second parameter to the drawString method indicates where the baseline of the string will be placed. Thus the 3rd parameter of 10 indicates the Y coordinate to be 10 pixels from the top of the Frame. This will result in just the bottom of the string Dolly showing up or possibly only the descending part of the letter y.

[08_01Tut.htm](#)

Answer 49)

49)

Objective 9.1)

1) Compile time error referring to a cast problem

This is a bit of a sneaky one as the Math.random method returns a pseudo random number between 0 and 1, and thus option 3 is a plausible Answer. However the number returned is a double and so the compiler will complain that a cast is needed to convert a double to an int.

[09_01Tut.htm](#)

Answer 50)

Objective 2.3)

50

```
1) public void ioCall ()throws IOException{  
    DataInputStream din = new DataInputStream(System.in);  
    din.readChar();  
}
```

If a method might throw an exception it must either be caught within the method with a try/catch block, or the method must indicate the exception to any calling method by use of the throws statement in its declaration. Without this, an error will occur at compile time.

02_03Tut.htm

Answer 51)

Objective 1.2)

51)

3) A compile time error

Because only one instance of a static method exists not matter how many instance of the class exists it cannot access any non static variables. The JVM cannot know which instance of the variable to access. Thus you will get an error saying something like

Can't make a static reference to a non static variable

01_02Tut.htm

Answer 52)

Objective 8.1)

52)

2) Create an instance of the GridBagConstraints class, set the weightx field and then pass the GridBagConstraints instance with the component to the setConstraints method of the GridBagLayout class.

The Key to using the GridBagLayout manager is the GridBagConstraints class. This class is not consistent with the general naming conventions in the java API as you would expect that weightx would be set with a method, whereas it is a simple field (variable). If you have a copy of the Roberts and Heller Java2 Guide that says the exam does not cover the GridBagLayout, this is an error which is corrected in later versions of the book
08_01Tut.htm

Answer 53)

Objective 11.1)

53)

2) Return the name of the parent directory

3) Delete a file

It is surprising that you can't change the current directory. It is not so surprising that you can't tell if a file contains text or binary information.

11_01Tut.htm

Answer 54)

54)

Objective 7.1

2) Output of One One Two Two

Answer 3 would be true if the code called the start method instead of the run method (well it is on my Windows machine anyway, I'm not sure it would be for every implementation of Java Threads). If you call the run method directly it just acts as any other method and does not return to the calling code until it has finished executing.

07_01Tut.htm

Answer 55)

Objective 3.1)

55)

1) You cannot be certain when garbage collection will run

Although there is a Runtime.gc(), this only suggests that the Java Virtual Machine does its garbage collection. You can never be certain when the garbage collector will run.

Roberts and Heller is more specific about this than Boone. This uncertainty can cause consternation for C++ programmers who wish to run finalize methods with the same intent as they use destructor methods.

03_01Tut.htm

Answer 56)

Objective 8.1)

56)

4) The fill field of the GridBagConstraints class

Unlike the GridLayout manager you can set the individual size of a control such as a button using the GridBagLayout manager. A little background knowledge would indicate that it should be controlled by a setSomethingOrOther method, but it isn't.

If you have a copy of the Roberts and Heller Java2 Guide that says the exam does not cover the GridBagLayout, this is an error. You can confirm this by looking at the online errata at

.

Answer 57)

Objective 10.1)

57)

4) A collection for storing bits as on-off information, like a vector of bits

This is the description given to a bitset in Bruce Eckels "Thinking in Java" book. The reference to unique sequence of bits was an attempt to mislead because of the use of the word Set in the name bitset. Normally something called a set implies uniqueness of the members, but not in this context.

10_01Tut.htm

Answer 58)

58)

Objective 4.1)

4)Compile error: Superclass Class1.Base of class Class1.Class1 not found

Using the package statement has an effect similar to placing a source file into a different directory. Because the files are in different packages they cannot see each other. The stuff about File1 not having been compiled was just to mislead, java has the equivalent of an "automake", whereby if it was not for the package statements the other file would have been automatically compiled.

Answer 59)

59)

Objective 6.2)

4) Output of Over.amethod()

The names of parameters to an overridden method is not important, but as the version of amethod in class Base is set to be private it is not visible within Over (despite Over extending Base) and thus does not take part in overriding.

Answer 60)

Objective 8.1)

60)

1) Set the gridy value of the GridBagConstraints class to a value increasing from 1 to 4

Answer 4 is fairly obviously bogus as it is the GridBagConstraints class that does most of the magic in laying out components under the GridBagLayout manager. The fill value of the GridBagConstraints class controls the behavior inside its virtual cell and the ipady field controls the internal padding around a component.

If you have a copy of the Roberts and Heller Java2 Guide that says the exam does not cover the GridBagLayout, this is an error. You can confirm this by looking at the online errata at

Question 1)

What will happen when you attempt to compile and run this code?

```
abstract class Base{
    abstract public void myfunc();
    public void another(){
        System.out.println("Another method");
    }
}

public class Abs extends Base{
    public static void main(String argv[]){
        Abs a = new Abs();
        a.amethod();
    }
    public void myfunc(){
        System.out.println("My Func");
    }
    public void amethod(){
        myfunc();
    }
}
```

- 1) The code will compile and run, printing out the words "My Func"
 - 2) The compiler will complain that the Base class has non abstract methods
 - 3) The code will compile but complain at run time that the Base class has non abstract methods
 - 4) The compiler will complain that the method myfunc in the base class has no body, nobody at all to loooove it
-

Question 2)

What will happen when you attempt to compile and run this code?

```
public class MyMain{
public static void main(String argv){
    System.out.println("Hello cruel world");
}
}
```

- 1) The compiler will complain that main is a reserved word and cannot be used for a class
 - 2) The code will compile and when run will print out "Hello cruel world"
 - 3) The code will compile but will complain at run time that no constructor is defined
 - 4) The code will compile but will complain at run time that main is not correctly defined
-

Question 3)

Which of the following are Java modifiers?

- 1) public
 - 2) private
 - 3) friendly
 - 4) transient
 - 5) vagrant
-

Question 4)

What will happen when you attempt to compile and run this code?

```
class Base{  
    abstract public void myfunc();  
    public void another(){  
        System.out.println("Another method");  
    }  
}  
  
public class Abs extends Base{  
    public static void main(String argv[]){  
        Abs a = new Abs();  
        a.amethod();  
    }  
  
    public void myfunc(){  
        System.out.println("My func");  
    }  
  
    public void amethod(){  
        myfunc();  
    }  
}
```

- 1) The code will compile and run, printing out the words "My Func"
 - 2) The compiler will complain that the Base class is not declared as abstract.
 - 3) The code will compile but complain at run time that the Base class has non abstract methods
 - 4) The compiler will complain that the method myfunc in the base class has no body, nobody at all to looove it
-

Question 5)

Why might you define a method as native?

-
- 1) To get to access hardware that Java does not know about
 - 2) To define a new data type such as an unsigned integer
 - 3) To write optimised code for performance in a language such as C/C++
 - 4) To overcome the limitation of the private scope of a method
-

Question 6)

What will happen when you attempt to compile and run this code?

```
class Base{  
    public final void amethod(){  
        System.out.println("amethod");  
    }  
}  
  
public class Fin extends Base{  
    public static void main(String argv[]){  
        Base b = new Base();  
        b.amethod();  
    }  
}
```

- 1) Compile time error indicating that a class with any final methods must be declared final itself
 - 2) Compile time error indicating that you cannot inherit from a class with final methods
 - 3) Run time error indicating that Base is not defined as final
 - 4) Success in compilation and output of "amethod" at run time.
-

Question 7)

What will happen when you attempt to compile and run this code?

```
public class Mod{  
    public static void main(String argv[]){  
        }  
        public static native void amethod();  
    }
```

- 1) Error at compilation: native method cannot be static
- 2) Error at compilation native method must return value
- 3) Compilation but error at run time unless you have made code containing native amethod available
- 4) Compilation and execution without error

Question 8)

What will happen when you attempt to compile and run this code?

```
private class Base{}  
public class Vis{  
    transient int ival;  
    public static void main(String elephant[]){  
    }  
}
```

- 1)Compile time error: Base cannot be private
 - 2)Compile time error indicating that an integer cannot be transient
 - 3)Compile time error transient not a data type
 - 4)Compile time error malformed main method
-

Question 9)

What happens when you attempt to compile and run these two files in the same directory?

```
//File P1.java  
package MyPackage;  
class P1{  
void afancymethod(){  
    System.out.println("What a fancy method");  
}  
}  
//File P2.java  
public class P2 extends P1{  
    public static void main(String argv[]){  
        P2 p2 = new P2();  
        p2.afancymethod();  
    }  
}
```

- 1) Both compile and P2 outputs "What a fancy method" when run
 - 2) Neither will compile
 - 3) Both compile but P2 has an error at run time
 - 4) P1 compiles cleanly but P2 has an error at compile time
-

Question 10)

You want to find out the value of the last element of an array. You write the following code. What will happen when you compile and run it.?

```
public class MyAr{  
    public static void main(String argv[]){  
        int[] i = new int[5];  
        System.out.println(i[5]);  
    }  
}
```

- 1) An error at compile time
 - 2) An error at run time
 - 3) The value 0 will be output
 - 4) The string "null" will be output
-

Question 11)

You want to loop through an array and stop when you come to the last element. Being a good java programmer and forgetting everything you ever knew about C/C++ you know that arrays contain information about their size. Which of the following can you use?

- 1)myarray.length();
 - 2)myarray.length;
 - 3)myarray.size
 - 4)myarray.size();
-

Question 12)

What best describes the appearance of an application with the following code?

```
import java.awt.*;  
public class FlowAp extends Frame{  
    public static void main(String argv[]){  
        FlowAp fa=new FlowAp();  
        fa.setSize(400,300);  
        fa.setVisible(true);  
  
    }  
  
    FlowAp(){  
        add(new Button( "One" ));  
        add(new Button( "Two" ));  
        add(new Button( "Three" ));  
        add(new Button( "Four" ));  
    }  
}
```

```
    } //End of constructor  
}  
} //End of Application
```

- 1) A Frame with buttons marked One to Four placed on each edge.
 - 2) A Frame with buutons marked One to four running from the top to bottom
 - 3) A Frame with one large button marked Four in the Centre
 - 4) An Error at run time indicating you have not set a LayoutManager
-

Question 13)

How do you indicate where a component will be positioned using Flowlayout?

- 1) North, South,East,West
 - 2) Assign a row/column grid reference
 - 3) Pass a X/Y percentage parameter to the add method
 - 4) Do nothing, the FlowLayout will position the component
-

Question 14)

How do you change the current layout manager for a container

- 1) Use the setLayout method
 - 2) Once created you cannot change the current layout manager of a component
 - 3) Use the setLayoutManager method
 - 4) Use the updateLayout method
-

Question 15)

Which of the following are fields of the GridBagConstraints class?

- 1) ipadx
 - 2) fill
 - 3) insets
 - 4) width
-

Question 16)

What most closely matches the appearance when this code runs?

```
import java.awt.*;
public class CompLay extends Frame{
    public static void main(String argv[]){
        CompLay cl = new CompLay();
    }

    CompLay(){
        Panel p = new Panel();
        p.setBackground(Color.pink);
        p.add(new Button("One"));
        p.add(new Button("Two"));
        p.add(new Button("Three"));
        add("South",p);
        setLayout(new FlowLayout());
        setSize(300,300);
        setVisible(true);
    }
}
```

- 1) The buttons will run from left to right along the bottom of the Frame
 - 2) The buttons will run from left to right along the top of the frame
 - 3) The buttons will not be displayed
 - 4) Only button three will show occupying all of the frame
-

Question 17)

Which statements are correct about the *anchor* field?

- 1) It is a field of the GridBagLayout manager for controlling component placement
 - 2) It is a field of the GridBagConstraints class for controlling component placement
 - 3) A valid setting for the anchor field is GridBagConstraints.NORTH
 - 4) The anchor field controls the height of components added to a container
-

Question 18)

What will happen when you attempt to compile and run the following code?

```
public class Bground extends Thread{
    public static void main(String argv[]){
        Bground b = new Bground();
        b.run();
    }
    public void start(){
        for (int i = 0; i < 10; i++){
            System.out.println("Value of i = " + i);
        }
    }
}
```

```
        }  
    }  
}
```

- 1) A compile time error indicating that no run method is defined for the Thread class
 - 2) A run time error indicating that no run method is defined for the Thread class
 - 3) Clean compile and at run time the values 0 to 9 are printed out
 - 4) Clean compile but no output at runtime
-

Question 19)

Is the following statement true or false?

When using the GridBagLayout manager, each new component requires a new instance of the GridBagConstraints class.

- 1) true
 - 2) false
-

Question 20)

Which most closely matches a description of a Java Map?

- 1) A vector of arrays for a 2D geographic representation
 - 2) A class for containing unique array elements
 - 3) A class for containing unique vector elements
 - 4) An interface that ensures that implementing classes cannot contain duplicate keys
-

Question 21)

How does the set collection deal with duplicate elements?

- 1) An exception is thrown if you attempt to add an element with a duplicate value
 - 2) The *add* method returns false if you attempt to add an element with a duplicate value
 - 3) A set may contain elements that return duplicate values from a call to the equals method
 - 4) Duplicate values will cause an error at compile time
-

Question 22)

What can cause a thread to stop executing?

- 1) The program exits via a call to System.exit(0);
 - 2) Another thread is given a higher priority
 - 3) A call to the thread's stop method.
 - 4) A call to the halt method of the Thread class?
-

Question 23)

For a class defined inside a method, what rule governs access to the variables of the enclosing method?

- 1) The class can access any variable
 - 2) The class can only access static variables
 - 3) The class can only access transient variables
 - 4) The class can only access final variables
-

Question 24)

Under what circumstances might you use the yield method of the Thread class

- 1) To call from the currently running thread to allow another thread of the same or higher priority to run
 - 2) To call on a waiting thread to allow it to run
 - 3) To allow a thread of higher priority to run
 - 4) To call from the currently running thread with a parameter designating which thread should be allowed to run
-

Question 25)

What will happen when you attempt to compile and run the following code

```
public class Hope{  
    public static void main(String argv[]){  
        Hope h = new Hope();  
    }  
  
    protected Hope(){
```

```

        for(int i =0; i <10; i ++){
            System.out.println(i);
        }
    }
}

```

- 1) Compilation error: Constructors cannot be declared protected
 - 2) Run time error: Constructors cannot be declared protected
 - 3) Compilation and running with output 0 to 10
 - 4) Compilation and running with output 0 to 9
-

Question 26)

What will happen when you attempt to compile and run the following code

```

public class MySwitch{
public static void main(String argv[ ]){
    MySwitch ms= new MySwitch();
    ms.amethod();
}

public void amethod(){
    int k=10;
    switch(k){
        default: //Put the default at the bottom, not here
            System.out.println("This is the default output");
            break;
        case 10:
            System.out.println("ten");
        case 20:
            System.out.println("twenty");
            break;
    }
}
}

```

- 1) None of these options
 - 2) Compile time error target of switch must be an integral type
 - 3) Compile and run with output "This is the default output"
 - 4) Compile and run with output of the single line "ten"
-

Question 27)

Which of the following is the correct syntax for suggesting that the JVM performs garbage collection

-
- 1) System.free();
 - 2) System.setGarbageCollection();
 - 3) System.out.gc();
 - 4) System.gc();
-

Question 28)

What will happen when you attempt to compile and run the following code

```
public class As{  
    int i = 10;  
    int j;  
    char z= 1;  
    boolean b;  
    public static void main(String argv[]){  
        As a = new As();  
        a.amethod();  
    }  
    public void amethod(){  
        System.out.println(j);  
        System.out.println(b);  
    }  
}
```

- 1) Compilation succeeds and at run time an output of 0 and false
 - 2) Compilation succeeds and at run time an output of 0 and true
 - 3) Compile time error b is not initialised
 - 4) Compile time error z must be assigned a char value
-

Question 29)

What will happen when you attempt to compile and run the following code with the command line "hello there"

```
public class Arg{  
String[] MyArg;  
    public static void main(String argv[]){  
        MyArg=argv;  
    }  
  
    public void amethod(){  
        System.out.println(argv[1]);  
    }  
}
```

-
- 1) Compile time error
 - 2) Compilation and output of "hello"
 - 3) Compilation and output of "there"
 - 4) None of the above
-

Question 30)

What will happen when you attempt to compile and run the following code

```
public class StrEq{  
    public static void main(String argv[ ]){  
        StrEq s = new StrEq();  
    }  
    private StrEq(){  
        String s = "Marcus";  
        String s2 = new String("Marcus");  
        if(s == s2){  
            System.out.println("we have a match");  
        }else{  
            System.out.println("Not equal");  
        }  
    }  
}
```

- 1) Compile time error caused by private constructor
- 2) Output of "we have a match"
- 3) Output of "Not equal"
- 4) Compile time error by attempting to compare strings using ==

Question 31)

What will happen when you attempt to compile and run the following code

```
import java.io.*;
class Base{
    public void amethod() throws FileNotFoundException{}
}

public class ExcepDemo extends Base{
    public static void main(String argv[]){
        ExcepDemo e = new ExcepDemo();
    }

    public void amethod(){}
    protected ExcepDemo(){
        try{
            DataInputStream din = new DataInputStream(System.in);
            System.out.println("Pausing");
            din.readByte();
            System.out.println("Continuing");
            this.amethod();
        }catch(IOException ioe) {}
    }
}
```

-
- 1) Compile time error caused by protected constructor
 - 2) Compile time error caused by *amethod* not declaring Exception
 - 3) Runtime error caused by amethod not declaring Exception
 - 4) Compile and run with output of "Pausing" and "Continuing" after a key is hit

Question 32)

What will happen when you attempt to compile and run this program

```
public class Outer{
public String name = "Outer";
public static void main(String argv[]){
    Inner i = new Inner();
    i.showName();
} //End of main
private class Inner{
    String name =new String("Inner");
    void showName(){
        System.out.println(name);
    }
} //End of Inner class
}
```

- 1) Compile and run with output of "Outer"
 - 2) Compile and run with output of "Inner"
 - 3) Compile time error because Inner is declared as private
 - 4) Compile time error because of the line creating the instance of Inner
-

Question 33)

What will happen when you attempt to compile and run this code

```
//Demonstration of event handling
import java.awt.*;
import java.awt.event.*;
public class MyWc extends Frame implements WindowListener{
public static void main(String argv[]){
    MyWc mwc = new MyWc();
}
public void windowClosing(WindowEvent we){
    System.exit(0);
} //End of windowClosing
public void MyWc(){
    setSize(300,300);
```

```
        setVisible(true);
    }
}//End of class
```

- 1) Error at compile time
 - 2) Visible Frame created that can be closed
 - 3) Compilation but no output at run time
 - 4) Error at compile time because of comment before *import* statements
-

Question 34)

Which option most fully describes what will happen when you attempt to compile and run the following code

```
public class MyAr{
    public static void main(String argv[]){  
        MyAr m = new MyAr();  
        m.amethod();  
    }  
    public void amethod(){  
        static int i;  
        System.out.println(i);  
    }  
}
```

- 1) Compilation and output of the value 0
 - 2) Compile time error because i has not been initialized
 - 3) Compilation and output of null
 - 4) Compile time error
-

Question 35)

Which of the following will compile correctly

- 1) short myshort = 99S;
 - 2) String name = 'Excellent tutorial Mr Green';
 - 3) char c = 17c;
 - 4) int z = 015;
-

Question 36)

Which of the following are Java key words

- 1) double
- 2) Switch

-
- 3)then
 - 4)instanceof
-

Question 37)

What will be output by the following line?

```
System.out.println(Math.floor(-2.1));
```

- 1) -2
 - 2) 2.0
 - 3) -3
 - 4) -3.0
-

Question 38)

Given the following main method in a class called Cycle and a command line of

```
java Cycle one two
```

what will be output?

```
public static void main(String bicycle[]){
    System.out.println(bicycle[0]);
}
```

- 1) None of these options
 - 2) cycle
 - 3) one
 - 4) two
-

Question 39)

Which of the following statements are true?

- 1) At the root of the collection hierarchy is a class called Collection
- 2) The collection interface contains a method called enumerator
- 3) The iterator method returns an instance of the Vector class
- 4) The Set interface is designed for unique elements

Question 40)

Which of the following statements are correct?

- 1) If multiple listeners are added to a component only events for the last listener added will be processed
 - 2) If multiple listeners are added to a component the events will be processed for all but with no guarantee in the order
 - 3) Adding multiple listeners to a component will cause a compile time error
 - 4) You may remove as well add listeners to a component.
-

Question 41)

Given the following code

```
class Base{}  
public class MyCast extends Base{  
    static boolean b1=false;  
    static int i = -1;  
    static double d = 10.1;  
    public static void main(String argv[]){  
        MyCast m = new MyCast();  
        Base b = new Base();  
        //Here  
    }  
}
```

Which of the following, if inserted at the comment //Here will allow the code to compile and run without error

- 1) b=m;
 - 2) m=b;
 - 3) d =i;
 - 4) b1 =i;
-

Question 42)

Which of the following statements about threading are true

- 1) You can only obtain a mutually exclusive lock on methods in a class that extends Thread or implements runnable

-
- 2) You can obtain a mutually exclusive lock on any object
 - 3) A thread can obtain a mutually exclusive lock on an object by calling a synchronized method on that object.
 - 4) Thread scheduling algorithms are platform dependent
-

Question 43)

Your chief Software designer has shown you a sketch of the new Computer parts system she is about to create. At the top of the hierarchy is a Class called Computer and under this are two child classes. One is called LinuxPC and one is called WindowsPC.

The main difference between the two is that one runs the Linux operating System and the other runs the Windows System (of course another difference is that one needs constant re-booting and the other runs reliably). Under the WindowsPC are two Sub classes one called Server and one Called Workstation. How might you appraise your designers work?

- 1) Give the goahead for further design using the current scheme
 - 2) Ask for a re-design of the hierarchy with changing the Operating System to a field rather than Class type
 - 3) Ask for the option of WindowsPC to be removed as it will soon be obsolete
 - 4) Change the hierarchy to remove the need for the superfluous Computer Class.
-

Question 44)

Which of the following statements are true

- 1) An inner class may be defined as static
 - 2) There are NO circumstances where an inner class may be defined as private
 - 3) A programmer may only provide one constructor for an anonymous class
 - 4) An inner class may extend another class
-

Question 45)

What will happen when you attempt to compile and run the following code

```
int Output=10;
boolean b1 = false;
if((b1==true) && ((Output+=10)==20)){
    System.out.println("We are equal "+Output);
} else {
    System.out.println("Not equal! "+Output);
}
```

- 1) Compile error, attempting to perform binary comparison on logical data type
 - 2) Compilation and output of "We are equal 10"
 - 3) Compilation and output of "Not equal! 20"
 - 4) Compilation and output of "Not equal! 10"
-

Question 46)

Given the following variables which of the following lines will compile without error?

```
String s = "Hello";
long l = 99;
double d = 1.11;
int i = 1;
int j = 0;
1) j= i <<s;
2) j= i<<j;
3) j=i<<d;
4) j=i<<l;
```

Question 47)

What will be output by the following line of code?

```
System.out.println(010|4);
```

- 1) 14
 - 2) 0
 - 3) 6
 - 4) 12
-

Question 48)

Given the following variables

```
char c = 'c';
int i = 10;
double d = 10;
long l = 1;
String s = "Hello";
```

Which of the following will compile without error?

- 1)c=c+i;
 - 2)s+=i;
 - 3)i+=s;
 - 4)c+=s;
-

Question 49)

Which of the following will compile without error?

- 1) File f = new File("/", "autoexec.bat");
 - 2) DataInputStream d = new DataInputStream(System.in);
 - 3) OutputStreamWriter o = new OutputStreamWriter(System.out);
 - 4) RandomAccessFile r = new RandomAccessFile("OutFile");
-

Question 50)

Given the following classes which of the following will compile without error?

```
interface IFace{}
class CFace implements IFace{}
class Base{}
public class ObRef extends Base{
    public static void main(String argv[]){
        ObRef ob = new ObRef();
        Base b = new Base();
        Object o1 = new Object();
        IFace o2 = new CFace();
    }
}
1)o1=o2;
2)b=ob;
3)ob=b;
4)o1=b;
```

Question 51)

Given the following code what will be the output?

```
class ValHold{
    public int i = 10;
}

public class ObParm{
public static void main(String argv[]){
    ObParm o = new ObParm();
    o.amethod();
}
public void amethod(){
    int i = 99;
    ValHold v = new ValHold();
    v.i=30;
    another(v,i);
    System.out.print( v.i );
} //End of amethod

public void another(ValHold v, int i){
    i=0;
    v.i = 20;
    ValHold vh = new ValHold();
    v = vh;
    System.out.print(v.i);
System.out.print(i);
} //End of another
}
```

- 1) 10030
 - 2) 20030
 - 3) 209930
 - 4) 10020
-

Question 52)

Given the following class definition, which of the following methods could be legally placed after the comment

```
//Here
public class Rid{
    public void amethod(int i, String s){}
    //Here
}
```

- 1) public void amethod(String s, int i){}
- 2) public int amethod(int i, String s){}

-
- 3)public void amethod(int i, String mystring){ }
4) public void Amethod(int i, String s) {}
-

Question 53)

Given the following class definition which of the following can be legally placed after the comment line

//Here ?

```
class Base{  
    public Base(int i){}  
}  
  
public class MyOver extends Base{  
    public static void main(String arg[]){  
        MyOver m = new MyOver(10);  
    }  
    MyOver(int i){  
        super(i);  
    }  
  
    MyOver(String s, int i){  
        this(i);  
        //Here  
    }  
}
```

- 1)MyOver m = new MyOver();
2)super();
3)this("Hello",10);
4)Base b = new Base(10);
-

Question 54)

Given the following class definition, which of the following statements would be legal after the comment //Here

```
class InOut{  
  
    String s= new String("Between");  
    public void amethod(final int iArgs){  
        int iam;  
        class Bicycle{  
            public void sayHello(){  
                //Here
```

```
        }
    } //End of bicycle class
} //End of amethod
public void another(){
int iOther;
}
}
```

- 1) System.out.println(s);
 - 2) System.out.println(iOther);
 - 3) System.out.println(iam);
 - 4) System.out.println(iArgs);
-

Question 55)

Which of the following are methods of the Thread class?

- 1) yield()
 - 2) sleep(long msec)
 - 3) go()
 - 4) stop()
-

Question 56)

Which of the following methods are members of the Vector class and allow you to input a new element

- 1) addElement
 - 2) insert
 - 3) append
 - 4) addItem
-

Question 57)

Which of the following statements are true?

-
- 1) Adding more classes via import statements will cause a performance overhead, only import classes you actually use.
 - 2) Under no circumstances can a class be defined with the *private* modifier
 - 3) A inner class may under some circumstances be defined with the *protected* modifier
 - 4) An interface cannot be instantiated
-

Question 58)

Which of the following are correct event handling methods

- 1) mousePressed(MouseEvent e){}
 - 2) MousePressed(MouseClick e){}
 - 3) functionKey(KeyPress k){}
 - 4) componentAdded(ContainerEvent e){}
-

Question 59)

Which of the following are methods of the Collection interface?

- 1) iterator
 - 2) isEmpty
 - 3) toArray
 - 4) setText
-

Question 60)

Which of the following best describes the use of the synchronized keyword?

- 1) Allows two process to run in parallel but to communicate with each other
- 2) Ensures only one thread at a time may access a method or object
- 3) Ensures that two or more processes will start and end at the same time
- 4) Ensures that two or more Threads will start and end at the same time

Answers

Answer 1)

Objective 1.2)

- 1) The code will compile and run, printing out the words "My Func"

A class that contains an abstract method must be declared abstract itself, but may contain non abstract methods.

http://www.jchq.net/tutorial/01_02Tut.htm

Answer 2)

Objective 4.1)

- 4) The code will compile but will complain at run time that main is not correctly defined

In this example the parameter is a string not a string array as needed for the correct main method

http://www.jchq.net/tutorial/04_01Tut.htm

Answer 3)

Objective 4.3)

- 1) public
- 2) private
- 4) transient

The keyword transient is easy to forget as is not frequently used. Although a method may be considered to be friendly like in C++ it is not a Java keyword.

http://www.jchq.net/tutorial/04_03Tut.htm

Answer 4)

Objective 1.2)

- 2) The compiler will complain that the Base class is not declared as abstract.

If a class contains abstract methods it must itself be declared as abstract

http://www.jchq.net/tutorial/01_02Tut.htm

Answer 5)

Objective 1.2)

- 1) To get to access hardware that Java does not know about
- 3) To write optimised code for performance in a language such as C/C++

http://www.jchq.net/tutorial/01_02Tut.htm

Answer 6)

Objective 1.2)

- 4) Success in compilation and output of "amethod" at run time.

A final method cannot be overridden in a sub class, but apart from that it does not cause any other restrictions.

http://www.jchq.net/tutorial/01_02Tut.htm

Answer 7)

Objective 1.2)

- 4) Compilation and execution without error

It would cause a run time error if you had a call to amethod though.

http://www.jchq.net/tutorial/01_02Tut.htm

Answer 8)

Objective 1.2)

- 1)Compile time error: Base cannot be private

A top level (non nested) class cannot be private.

Answer 9)

Objective 1.2)

4) P1 compiles cleanly but P2 has an error at compile time

The package statement in P1.java is the equivalent of placing the file in a different directory to the file P2.java and thus when the compiler tries to compile P2 an error occurs indicating that superclass P1 cannot be found.

http://www.jchq.net/tutorial/01_02Tut.htm

Answer 10)

Objective 4.2)

2) An error at run time

This code will compile, but at run-time you will get an `ArrayIndexOutOfBoundsException` exception. This because counting in Java starts from 0 and so the 5th element of this array would be `i[4]`.

Remember that arrays will always be initialized to default values wherever they are created.

http://www.jchq.net/tutorial/04_02Tut.htm

Answer 11)

Objective 1.1)

2)`myarray.length;`

The `String` class has a `length()` method to return the number of characters. I have sometimes become confused between the two.

http://www.jchq.net/tutorial/01_01Tut.htm

Answer 12)

Objective 8.2)

3) A Frame with one large button marked Four in the Centre

The default layout manager for a Frame is the BorderLayout manager. This Layout manager defaults to placing components in the centre if no constraint is passed with the call to the add method.

http://www.jchq.net/tutorial/08_02Tut.htm

Answer 13)

Objective 8.2)

4) Do nothing, the FlowLayout will position the component

Answer 14)

Objective 8.2)

1) Use the setLayout method

Answer 15)

Objective 8.2)

- 1) ipadx
- 2) fill
- 3) insets

http://www.jchq.net/tutorial/08_02Tut.htm

Answer 16)

Objective 8.2)

2) The buttons will run from left to right along the top of the frame

The call to the setLayout(new FlowLayout()) resets the Layout manager for the entire frame and so the buttons end up at the top rather than the bottom.

http://www.jchq.net/tutorial/08_02Tut.htm

Answer 17)

Objective 8.2)

- 2) It is a field of the GridBagConstraints class for controlling component placement
- 3) A valid setting for the anchor field is GridBagConstraints.NORTH

http://www.jchq.net/tutorial/08_02Tut.htm

Answer 18)

Objective 7.1)

- 4) Clean compile but no output at runtime

This is a bit of a sneaky one as I have swapped around the names of the methods you need to define and call when running a thread. If the for loop were defined in a method called public void run() and the call in the main method had been to b.start() The list of values from 0 to 9 would have been output.

http://www.jchq.net/tutorial/07_01Tut.htm

Answer 19)

Objective 8.2)

- 2) false

You can re-use the same instance of the GridBagConstraints when added successive components.

http://www.jchq.net/tutorial/08_02Tut.htm

Answer 20)

Objective 10.1)

- 4) An interface that ensures that implementing classes cannot contain duplicates

http://www.jchq.net/tutorial/10_01Tut.htm

Answer 21)

Objective 10.1)

- 2) The *add* method returns false if you attempt to add an element with a duplicate value

I find it a surprise that you do not get an exception.

http://www.jchq.net/tutorial/10_01Tut.htm

Answer 22)

Objective 7.1)

- 1) The program exits via a call to exit(0);
- 2) The priority of another thread is increased
- 3) A call to the stop method of the Thread class

Note that this question asks what can cause a thread to stop executing, not what will cause a thread to stop executing. Java threads are somewhat platform dependent and you should be carefull when making assumptions about Thread priorities. On some platforms you may find that a Thread with higher priorities gets to "hog" the processor. You can read up on this in more detail at

<http://java.sun.com/docs/books/tutorial/essential/threads/priority.html>

http://www.jchq.net/tutorial/07_01Tut.htm

Answer 23)

Objective 4.1)

- 4) The class can only access final variables

http://www.jchq.net/tutorial/04_01Tut.htm

Answer 24)

Objective 7.1)

- 1) To call from the currently running thread to allow another thread of the same or higher priority to run

Option 3 looks plausible but there is no guarantee that the thread that grabs the cpu time will be of a higher priority. It will depend on the threading algorithm of the Java Virtual Machine and the underlying operating system

http://www.jchq.net/tutorial/07_01Tut.htm

Answer 25)

Objective 6.2)

4) Compilation and running with output 0 to 9

http://www.jchq.net/tutorial/06_02Tut.htm

Answer 26)

Objective 2.1)

1) None of these options

Because of the lack of a break statement after the break 10; statement the actual output will be

"ten" followed by "twenty"

http://www.jchq.net/tutorial/02_01Tut.htm

Answer 27)

Objective 3.1)

4) System.gc();

http://www.jchq.net/tutorial/03_01Tut.htm

Answer 28)

Objective 4.4)

1) Compilation succeeds and at run time an output of 0 and false

The default value for a boolean declared at class level is false, and integer is 0;

http://www.jchq.net/tutorial/04_04Tut.htm

Answer 29)

Objective 1.2)

1) Compile time error

You will get an error saying something like "Cant make a static reference to a non static variable". Note that the main method is static. Even if main was not static the array argv is local to the main method and would thus not be visible within amethod.

http://www.jchq.net/tutorial/01_02Tut.htm

Answer 30)

Objective 5.2)

3) Output of "Not equal"

Despite the actual character strings matching, using the == operator will simply compare memory location. Because the one string was created with the new operator it will be in a different location in memory to the other string.

http://www.jchq.net/tutorial/05_02Tut.htm

Answer 31)

Objective 2.3)

4) Compile and run with output of "Pausing" and "Continuing" after a key is hit

An overriden method in a sub class must not throw Exceptions not thrown in the base class. In the case of the method amethod it throws no exceptions and will thus compile without complain. There is no reason that a constructor cannot be protected.

http://www.jchq.net/tutorial/02_03Tut.htm

Answer 32)

Objective 6.3)

4) Compile time error because of the line creating the instance of Inner

This looks like a question about inner classes but it is also a reference to the fact that the main method is static and thus you cannot directly access a non static method. The line causing the error could be fixed by changing it to say

```
Inner i = new Outer().new Inner();
```

Then the code would compile and run producing the output "Inner"

http://www.jchq.net/tutorial/06_03Tut.htm

Answer 33)

Objective 4.6)

1) Error at compile time

If you implement an interface you must create bodies for all methods in that interface. This code will produce an error saying that MyWc must be declared abstract because it does not define all of the methods in WindowListener. Option 4 is nonsense as comments can appear anywhere. Option 3 suggesting that it might compile but not produce output is meant to mislead on the basis that what looks like a constructor is actually an ordinary method as it has a return type.

http://www.jchq.net/tutorial/04_06Tut.htm

Answer 34)

Objective 1.2)

4) Compile time error

An error will be caused by attempting to define an integer as static within a method. The lifetime of a field within a method is the duration of the running of the method. A static field exists once only for the class. An approach like this does work with Visual Basic.

http://www.jchq.net/tutorial/01_02Tut.htm

Answer 35)

Objective 4.5)

4) int z = 015;

The letters c and s do not exist as literal indicators and a String must be enclosed with double quotes, not single as in this case.

http://www.jchq.net/tutorial/04_05Tut.htm

Answer 36)

Objective 4.3)

- 1)double
- 4)instanceof

Note the upper case S on switch means it is not a keyword and the word then is part of Visual Basic but not Java. Also, *instanceof* looks like a method but is actually a keyword,

http://www.jchq.net/tutorial/04_03Tut.htm

Answer 37)

Objective 9.1)

- 4) -3.0

http://www.jchq.net/tutorial/09_02Tut.htm

Answer 38)

Objective 4.2)

- 3) one

Command line parameters start from 0 and from the first parameter after the name of the compile (normally Java)

http://www.jchq.net/tutorial/04_02Tut.htm

Answer 39)

Objective 10.1)

- 4) The Set is designed for unique elements.

Collection is an interface, not a class. The Collection interface includes a method called iterator. This returns an instance of the Iterator class which has some similarities with Enumerators.

The name set should give away the purpose of the Set interface as it is analogous to the Set concept in relational databases which implies uniqueness.

http://www.jchq.net/tutorial/10_01Tut.htm

Answer 40)

Objective 8.1)

- 2) If multiple listeners are added to a component the events will be processed for all but with no guarantee in the order
- 4) You may remove as well add listeners to a component.

It ought to be fairly intuitive that a component ought to be able to have multiple listeners. After all, a text field might want to respond to both the mouse and keyboard

http://www.jchq.net/tutorial/08_01Tut.htm

Answer 41)

Objective 5.1)

- 1) b=m;
- 3) d =i;

You can assign up the inheritance tree from a child to a parent but not the other way without an explicit casting. A boolean can only ever be assigned a boolean value.

http://www.jchq.net/tutorial/08_01Tut.htm

Answer 42)

Objective 7.3)

- 2) You can obtain a mutually exclusive lock on any object
- 3) A thread can obtain a mutually exclusive lock on an object by calling a synchronized method on that object.
- 4) Thread scheduling algorithms are platform dependent

Yes that says dependent and not independent.

http://www.jchq.net/tutorial/07_03Tut.htm

Answer 43)

Objective 6.1)

2) Ask for a re-design of the hierarchy with changing the Operating System to a field rather than Class type

This question is about the requirement to understand the difference between the "is-a" and the "has-a" relationship. Where a class is inherited you have to ask if it represents the "is-a" relationship. As the difference between the root and the two children are the operating system you need to ask are Linux and Windows types of computers. The answer is no, they are both types of Operating Systems. So option two represents the best of the options. You might consider having operating system as an interface instead but that is another story.

Of course there are as many ways to design an object hierarchy as ways to pronounce Bjarne Strousjou, but this is the sort of answer that Sun will probably be looking for in the exam. Questions have been asked in discussion forums if this type of question really comes up in the exam. I think this is because some other mock exams do not contain any questions like this. I assure you that this type of question can come up in the exam. These types of question are testing your understanding of the difference between the is-a and has-a relationship in class design.

http://www.jchq.net/tutorial/06_01Tut.htm

Answer 44)

Objective 4.1)

- 1) An inner class may be defined as static
- 4) An inner class may extend another class

A static inner class is also sometimes known as a top level nested class. There is some debate if such a class should be called an inner class. I tend to think it should be on the basis that it is created inside the opening braces of another class. How could a programmer provide a constructor for an anonymous class?. Remember a constructor is a method with no return type and the same name as the class. Inner classes may be defined as private

http://www.jchq.net/tutorial/04_01Tut.htm

Answer 45)

Objective 5.3)

4) Compilation and output of "Not equal! 10"

The output will be "Not equal 10". This illustrates that the Output $+=10$ calculation was never performed because processing stopped after the first operand was evaluated to be false. If you change the value of b1 to true processing occurs as you would expect and the output is "We are equal 20";.

http://www.jchq.net/tutorial/05_03Tut.htm

Answer 46)

Objective 5.1)

2) $j = i \ll j;$

4) $j = i \ll 1;$

http://www.jchq.net/tutorial/05_01Tut.htm

Answer 47)

Objective 5.3)

4) 12

As well as the binary OR objective this questions requires you to understand the octal notation which means that the leading letter zero (not the letter O)) means that the first 1 indicates the number contains one eight and nothing else. Thus this calculation in decimal mean

$8 | 4$

To convert this to binary means

1000

0100

1100

Which is 12 in decimal

The | bitwise operator means that for each position where there is a 1, results in a 1 in the same position in the answer.

http://www.jchq.net/tutorial/05_03Tut.htm

Answer 48)

Objective 5.1)

2)s+=i;

Only a String acts as if the + operator were overloaded

http://www.jchq.net/tutorial/05_01Tut.htm

Answer 49)

Objective 10.1)

Although the objectives do not specifically mention the need to understand the I/O Classes, feedback from people who have taken the exam indicate that you will get questions on this topic. As you will probably need to know this in the real world of Java programming, get familiar with the basics. I have assigned these questions to Objective 10.1 as that is a fairly vague objective.

- 1) File f = new File("/", "autoexec.bat");
- 2) DataInputStream d = new DataInputStream(System.in);
- 3) OutputStreamWriter o = new OutputStreamWriter(System.out);

Option 4, with the RandomAccess file will not compile because the constructor must also be passed a mode parameter which must be either "r" or "rw"

http://www.jchq.net/tutorial/10_01Tut.htm

Answer 50)

Objective 5.1)

- 1) o1=o2;
- 2) b=ob;

4) o1=b;
http://www.jchq.net/tutorial/05_01Tut.htm

Answer 51)

Objective 5.4)

4) 10020

In the call

another(v,i);

A reference to v is passed and thus any changes will be intact after this call.

http://www.jchq.net/tutorial/05_01Tut.htm

Answer 52)

Objective 6.2)

- 1) public void amethod(String s, int i){}
- 4) public void Amethod(int i, String s) {}

Overloaded methods are differentiated only on the number, type and order of parameters, not on the return type of the method or the names of the parameters.

http://www.jchq.net/tutorial/06_02Tut.htm

Answer 53)

Objective 6.2)

4) Base b = new Base(10);

Any call to this or super must be the first line in a constructor. As the method already has a call to this, no more can be inserted.

http://www.jchq.net/tutorial/06_02Tut.htm

Answer 54)

Objective 4.1)

- 1) System.out.println(s);
- 4) System.out.println(iArgs);

A class within a method can only see final variables of the enclosing method. However it the normal visibility rules apply for variables outside the enclosing method.

http://www.jchq.net/tutorial/04_01Tut.htm

Answer 55)

Objective 7.2)

- 1) yield()
- 2) sleep
- 4) stop()

Note, the methods stop and suspend have been deprecated with the Java2 release, and you may get questions on the exam that expect you to know this. Check out the Java2 Docs for an explanation

http://www.jchq.net/tutorial/07_02Tut.htm

Answer 56)

Objective 10.1)

- 1) addElement

http://www.jchq.net/tutorial/10_01Tut.htm

Answer 57)

Objective 4.1)

The import statement allows you to use a class directly instead of fully qualifying it with the full package name, adding more classes with the import statement does not cause a runtime performance overhead. I assure you this is true. An inner class can be defined with the protected modifier, though I am not certain why you would want to do it. An inner

class can be defined with the private modifier, try compiling some sample code before emailing me to ask about this.

- 3) A inner class may under some circumstances be defined with the protected modifier
- 4) An interface cannot be instantiated

http://www.jchq.net/tutorial/04_01Tut.htm

Answer 58)

Objective 4.6)

- 1) mousePressed(MouseEvent e){ }
- 4) componentAdded(ContainerEvent e){ }

http://www.jchq.net/tutorial/04_06Tut.htm

Answer 59)

Objective 10.1)

- 1) iterator
- 2) isEmpty
- 3) toArray

http://www.jchq.net/tutorial/10_01Tut.htm

Answer 60)

Objective 7.3)

- 2) Ensures only one thread at a time may access a method or object

http://www.jchq.net/tutorial/07_03Tut.htm

Question 1)

Which of the following are legal statements?

- 1) float f=1/3;
- 2) int i=1/3;
- 3) float f=1.01;
- 4) double d=999d;

Answer to Question 1)

Question 2)

Which of the following are Java keywords?

- 1) NULL
- 2) new
- 3) instanceof
- 4) wend

Answer to Question 2)

Question 3)

Which of the following are valid statements?

- 1) System.out.println(1+1);
- 2) int i=2+'2';
- 3) String s="on"+'one';
- 4) byte b=255;

Answer to Question 3)

Question 4)

Which of the following statements are true?

- 1) The garbage collection algorithm in Java is vendor implemented
- 2) The size of primitives is platform dependent
- 3) The default type for a numerical literal with decimal component is a float.
- 4) You can modify the value in an Instance of the Integer class with the setValue method

Answer to Question 4)

Question 5)

Which of the following are true statements?

- 1) I/O in Java can only be performed using the Listener classes

- 2) The RandomAccessFile class allows you to move directly to any point a file.
- 3) The creation of a named instance of the File class creates a matching file in the underlying operating system only when the close method is called.
- 4) The characteristics of an instance of the File class such as the directory separator, depend on the current underlying operating system

Answer to Question 5)

Question 6).

Which of the following statements are true?

- 1) The instanceof operator can be used to determine if a reference is an instance of a class, but not an interface.
- 2) The instanceof operator can be used to determine if a reference is an instance of a particular primitive wrapper class
- 3) The instanceof operator will only determine if a reference is an instance of a class immediately above in the hierarchy but no further up the inheritance chain
- 4) The instanceof operator can be used to determine if one reference is of the same class as another reference thus

Answer to Question 6)

Question 7)

Which of the following statements are true?

- 1) An interface can only contain method and not variables
- 2) Interfaces cannot have constructors
- 3) A class may extend only one other class and implement only one interface
- 4) Interfaces are the Java approach to addressing its lack of multiple inheritance, but require implementing classes to create the functionality of the Interfaces.

Answer to Question 7)

Question 8)

Which of the following are valid statements

- 1) public class MyCalc extends Math
- 2) Math.max(s);
- 3) Math.round(9.99,1);
- 4) Math.mod(4,10);

Answer to Question 8)

Question 9)

Which of the following are methods of the Runnable interface

- 1) run
- 2) start
- 3) yield
- 4) stop

Answer to Question 9)

Question 10)

Which of the following statements are true?

- 1) A byte can represent between -128 to 127
- 2) A byte can represent between -127 to 128
- 3) A byte can represent between -256 to 256
- 4) A char can represent between $-2 \times 2^{16} \leq \text{char} \leq 2^{16} - 1$

Answer to Question 10)

Question 11)

What will happen when you attempt to compile and run the following code

```
class Base{  
    public void Base(){  
        System.out.println("Base");  
    }  
}  
public class In extends Base{  
    public static void main(String argv[]){  
        In i=new In();  
    }  
}
```

- 1) Compile time error Base is a keyword
- 2) Compilation and no output at runtime
- 3) Output of Base
- 4) Runtime error Base has no valid constructor

Answer to Question 11)

Question 12)

You have a public class called myclass with the main method defined as follows

```
public static void main(String parm[]){  
    System.out.println(parm[0]);  
}
```

If you attempt to compile the class and run the program as follows

`java myclass hello`

What will happen?

- 1) Compile time error, main is not correctly defined
- 2) Run time error, main is not correctly defined
- 3) Compilation and output of java
- 4) Compilation and output of hello

Answer to Question 12)

Question 13)

Which of the following statements are true?

- 1) If a class has any abstract methods it must be declared abstract itself.
- 2) All methods in an abstract class must be declared as abstract
- 3) When applied to a class, the final modifier means it cannot be sub-classed
- 4) transient and volatile are Java modifiers

Answer to Question 13)

Question 14)

Which of the following are valid methods?

- 1) public static native void amethod(){ }
- 2) public static void amethod(){ }
- 3) private protected void amethod(){ }
- 4) static native void amethod();

Answer to Question 14)

Question 15)

Which of the following statements are true?

- 1) Constructors cannot have a visibility modifier
- 2) Constructors can be marked public and protected, but not private
- 3) Constructors can only have a primitive return type
- 4) Constructors are not inherited

Answer to Question 15)

Question 16)

What will happen when you attempt to compile and run the following class?

```
class Base{  
Base(int i){  
    System.out.println("Base");  
}  
}  
  
class Severn extends Base{  
public static void main(String argv[]){  
    Severn s = new Severn();  
}  
void Severn(){  
    System.out.println("Severn");  
}  
}
```

- 1) Compilation and output of the string "Severn" at runtime
- 2) Compile time error
- 3) Compilation and no output at runtime
- 4) Compilation and output of the string "Base"

Answer to Question 16)

Question 17)

Which of the following statements are true?

- 1) static methods do not have access to the implicit variable called this
- 2) A static method may be called without creating an instance of its class
- 3) A static method may not be overridden to be non-static
- 4) A static method may not be overloaded

Answer to question 17)

Question 18)

Which of the following will compile without error?

1)

```
char c='1';  
System.out.println(c>>1);
```

2)

```
Integer i=Integer("1");
System.out.println(i>>1);
```

3)

```
int i=1;
System.out.println(i<<<1);
```

4)

```
int i=1;
System.out.println(i<<1);
```

Answer to Question 18)

Question 19)

Which of the following are true?

- 1) A component may have only one event listener attached at a time
- 2) An event listener may be removed from a component
- 3) The ActionListener interface has no corresponding Adapter class
- 4) The processing of an event listener requires a try/catch block

Answer to Question 19)

Question 20)

Which of the following are Java keywords?

- 1) sizeof
- 2) main
- 3) transient
- 4) volatile

Answer to Question 20)

Question 21)

Which of the following statements are true?

- 1) The default constructor has a return type of void
- 2) The default constructor takes a parameter of void
- 3) The default constructor takes no parameters
- 4) The default constructor is not created if the class has any constructors of its own.

Answer to Question 21)

Question 22)

Which of the following statements are true?

- 1) All of the variables in an interface are implicitly static
- 2) All of the variables in an interface are implicitly final
- 3) All of the methods in an interface are implicitly abstract
- 4) A method in an interface can access class level variables

Answer to Question 22)

Question 23)

Which of the following statements are true?

- 1) The String class is implemented as a char array, elements are addressed using the stringname[] convention
- 2) The + operator is overloaded for concatenation for the String class
- 3) Strings are a primitive type in Java and the StringBuffer is used as the matching wrapper type
- 4) The size of a string can be retrieved using the length property

Answer to Question 23)

Question 24)

Which of the following statements are true?

- 1) A method in an interface must not have a body
- 2) A class may extend one other class plus at most one interface
- 3) A class may extends at most one other class plus implement many interfaces
- 4) An class accesses an interface via the keyword uses

Answer to Question 24)

Question 25)

Which of the following statements are true?

- 1) The following statement will produce a result of 1. System.out.println(-1 >>>2);
- 2) Performing an unsigned left shift (<<<) on a negative number will always produce a negative number result
- 3) The following statement will produce a result of zero, System.out.println(1 >>1);
- 4) All the Java integral types are signed numbers

Answer to Question 25)

Question 26)

Which of the following statements are true?

- 1) The elements in a Java array can only be of primitive types, not objects
- 2) Arrays elements are initialized to default values wherever they are created using the keyword new
- 3) An array may be dynamically resized using the setSize method
- 4) You can find out the size of an array using the size method

Answer to Question 26)

Question 27)

Given the following class

```
public class Ombersley{  
    public static void main(String argv[]){  
        boolean b1 = true;  
        if((b1 ==true) || place(true)){  
            System.out.println("Hello Crowle");  
        }  
    }  
  
    public static boolean place(boolean location){  
        if(location==true){  
            System.out.println("Borcetshire");  
        }  
        System.out.println("Powick");  
        return true;  
    }  
}
```

What will happen when you attempt to compile and run it?

- 1) Compile time error
- 2) Output of "Hello Crowle"
- 3) Output of Borcetshire and Powick followed by "Hello Crowle"
- 4) No output

Answer to Question 27)

Question 28)

You are given a class hierarchy with an instance of the class Dog. The class Dog is a child of mammal and the class Mammal is a child of the class Vertebrate. The class

Vertebrate has a method called move which prints out the string "move". The class mammal overrides this method and prints out the string "walks". The class Dog overrides this method and prints out the string "walks on paws". Given an instance of the class Dog,, how can you access the ancestor method move in Vertebrate so it prints out the string "move";

- 1) d.super().super().move();
- 2) d.parent().parent().move();
- 3) d.move();
- 4) none of the above;

Answer to Question 28)
Question 29)

Which of the following most closely describes the process of overriding?

- 1) A class with the same name replaces the functionality of a class defined earlier in the hierarchy
- 2) A method with the same name completely replaces the functionality of a method earlier in the hierarchy
- 3) A method with the same name but different parameters gives multiple uses for the same method name
- 4) A class is prevented from accessing methods in its immediate ancestor

Answer to Question 29)
Question 30)

Which of the following statements are true?

- 1) The % is used to calculate a percentage thus: 10 % 20=50
- 2) The / operator is used to divide one value by another
- 3) The # symbol may not be used as the first character of a variable
- 4) The \$ symbol may not be used as the first character of a variable

Answer to Question 30)
Question 31)

Which of the following statements are true?

- 1) The default layout manager for an Applet is FlowLayout
- 2) The default layout manager for a Frame is FlowLayout
- 3) A layout manager must be assigned to an Applet before the setSize method is called
- 4) The FlowLayout manager attempts to honor the preferred size of any components

Answer to Question 31)
Question 32)

Which of the following statements are true about a variable created with the static modifier?

- 1) Once assigned the value of a static variable may not be altered
- 2) A static variable created in a method will keep the same value between calls
- 3) Only one instance of a static variable will exist for any amount of class instances
- 4) The static modifier can only be applied to a primitive value

Answer to Question 32)

Question 33)

Which of the following statements are true?

- 1) Java uses a system called UTF for I/O to support international character sets
- 2) The RandomAccessFile is the most suitable class for supporting international character sets
- 3) An instance of FileInputStream may not be chained to an instance of FileOutputStream
- 4) File I/O activities requires use of Exception handling

Answer to Question 33)

Question 34)

What will happen when you attempt to compile and run the following code?

```
import java.io.*;
class ExBase{
    abstract public void martley(){
    }

}

public class MyEx extends ExBase{
    public static void main(String argv[]){
        DataInputStream fi = new DataInputStream(System.in);
        try{
            fi.readChar();
        }catch(IOException e){
            System.exit(0);
        }
        finally {System.out.println("Doing finally");}
    }
}
```

- 1) Compile time error
- 2) It will run, wait for a key press and then exit

- 3) It will run, wait for a keypress, print "Doing finally" then exit
- 4) At run and immediately exit

Answer to Question 34)

Question 35)

What will happen when you attempt to compile and run the following code

```
public class Borley extends Thread{  
    public static void main(String argv[]){  
        Borley b = new Borley();  
        b.start();  
    }  
    public void run(){  
        System.out.println("Running");  
    }  
}
```

- 1) Compilation and run but no output
- 2) Compilation and run with the output "Running"
- 3) Compile time error with complaint of no Thread target
- 4) Compile time error with complaint of no access to Thread package

Answer to Question 35)

Question 36)

Assuming any exception handling has been set up, which of the following will create an instance of the RandomAccessFile class

- 1) RandomAccessFile raf=new RandomAccessFile("myfile.txt","rw");
- 2) RandomAccessFile raf=new RandomAccessFile(new DataInputStream());
- 3) RandomAccessFile raf=new RandomAccessFile("myfile.txt");
- 4) RandomAccessFile raf=new RandomAccessFile(new File("myfile.txt"));

Answer to Question 36)

Question 37)

Given the following class definition

```
public class Upton{  
    public static void main(String argv[]){  
    }  
    public void amethod(int i){ }  
    //Here  
}
```

Which of the following would be legal to place after the comment //Here ?

- 1) public int amethod(int z){ }
- 2) public int amethod(int i,int j){return 99;}
- 3) protected void amethod(long l){ }
- 4) private void anothermethod(){ }

Answer to Question 37)

Question 38)

Which of the following statements are true?

- 1) Code must be written if the programmer wants a frame to close on selecting the system close menu
- 2) The default layout for a Frame is the BorderLayout Manager
- 3) The layout manager for a Frame cannot be changed once it has been assigned
- 4) The GridBagLayout manager makes extensive use of the the GridBagConstraints class.

Answer to Question 38)

Question 39)

Given the following class definition

```
public class Droitwich{
    class one{
        private class two{
            public void main(){
                System.out.println("two");
            }
        }
    }
}
```

Which of the following statements are true

- 1) The code will not compile because the classes are nested to more than one level
- 2) The code will not compile because class two is marked as private
- 3) The code will compile and output the string two at runtime
- 4) The code will compile without error

Answer to Question 39)

Question 40)

Given the following code

```
class Base{
```

```

static int oak=99;
}

public class Doverdale extends Base{
    public static void main(String argv[]){
        Doverdale d = new Doverdale();
        d.amethod();
    }
    public void amethod(){
        //Here
    }
}

```

Which of the following if placed after the comment //Here, will compile and modify the value of the variable oak?

- 1) super.oak=1;
- 2) oak=33;
- 3) Base.oak=22;
- 4) oak=50.1;

Answer to Question 40)

Question 41)

You are creating an application that has a form with a text entry field used to enter a persons age. Which of the following is appropriate for capturing this information.

- 1) Use the Text field of a TextField and parse the result using Integer
- 2) Use the getInteger method of the TextField
- 3) Use the getText method of a TextBox and parse the result using the getInt method of Integer class
- 4) Use the getText method of a TextField and use the parseInt method of the Integer class

Answer to Question 41)

Question 42)

Given the following declaration

```
Integer i=new Integer(99);
```

How can you now set the value of i to 10?

- 1) i=10;
- 2) i.setValue(10);
- 3) i.parseInt(10);
- 4) none of the above

Answer to Question 42)

Question 43)

Which of the following statements are true

- 1) constructors cannot be overloaded
- 2) constructors cannot be overridden
- 3) a constructor can return a primitive or an object reference
- 4) constructor code executes from the current class up the hierarchy to the ancestor class

Answer to Question 43)

Question 44)

Given a reference called

t

to a class which extends Thread, which of the following will cause it to give up cycles to allow another thread to execute.

- 1) t.yield();
- 2) Thread.yield();
- 3) yield(100); //Or some other suitable amount in milliseconds
- 4) yield(t);

Answer to Question 44)

Question 45)

What will happen when you attempt to compile and run the following code?

```
public class Sandys{  
    private int court;  
    public static void main(String argv[]){  
        Sandys s = new Sandys(99);  
        System.out.println(s.court);  
    }  
    Sandys(int ballcount){
```

```
    court=ballcount;  
}  
}
```

- 1) Compile time error, the variable court is defined as private
- 2) Compile time error, s is not initialized when the System.out method is called
- 3) Compilation and execution with no output
- 4) Compilation and run with an output of 99

Answer to Question 45)

Question 46)

Which of the following statements are true?

- 1) A method cannot be overloaded to be less public in a child class
- 2) To be overridden a method only needs the same name and parameter types
- 3) To be overridden a method must have the same name, parameter and return types
- 4) An overridden method must have the same name, parameter names and parameter types

Answer to Question 46)

Question 47)

What will happen when you attempt to compile and run the following code?

```
class Base{  
Base(){  
    System.out.println("Base");  
}  
}  
  
public class Checket extends Base{  
public static void main(String argv[]){  
    Checket c = new Checket();  
    super();  
}  
  
Checket(){  
    System.out.println("Checket");  
}  
}
```

- 1) Compile time error
- 2) Checket followed by Base
- 3) Base followed by Checket
- 4) runtime error

Answer to Question 47)

Question 48)

Which of the following statements are true?

- 1) Static methods cannot be overriden to be non static
- 2) Static methods cannot be declared as private
- 3) Private methods cannot be overloaded
- 4) An overloaded method cannot throw exceptions not checked in the base class

Answer to Question 48)

Question 49)

Which of the following statements are true?

- 1) The automatic garbage collection of the JVM prevents programs from ever running out of memory
- 2) A program can suggest that garbage collection be performed but not force it
- 3) Garbage collection is platform independent
- 4) An object becomes eligible for garbage collection when all references denoting it are set to null.

Answer to Question 49)

Question 50)

Given the following code

```
public class Sutch{  
int x=2000;  
public static void main(String argv[]){  
    System.out.println("Ms "+argv[1]+"Please pay $" +x);  
}  
}
```

What will happen if you attempt to compile and run this code with the command line

java Sutch Jones Diggle

- 1) Compilation and output of Ms Diggle Please pay \$2000
- 2) Compile time error

- 3) Compilation and output of Ms Jones Please pay \$2000
- 4) Compilation but runtime error

Answer to Question 50)

Question 51)

What will happen when you attempt to compile and run the following code

```
class Base{  
protected int i = 99;  
}  
public class Ab{  
private int i=1;  
public static void main(String argv[]){  
Ab a = new Ab();  
a.hallow();  
}  
  
abstract void hallow(){  
System.out.println("Claines "+i);  
}  
}
```

- 1) Compile time error
- 2) Compilation and output of Claines 99
- 3) Compilation and output of Claines 1
- 4) Compilation and not output at runtime

Answer to Question 51)

Question 52)

You have been asked to create a scheduling system for a hotel and catering organisation.

You have been given the following information and asked to create a set of classes to represent it.

On the catering side of the organisation they have

Head Chefs
Chefs
Apprentice Chefs

The system needs to store an employeeid, salary and the holiday entitlement

How would you best represent this information in Java given the following information and asked to create a set of classes to represent it.

How would you best represent this information in Java

- 1) Create classes for Head Chef, Chef, Apprentice Chef and store the other values in fields
- 2) Create an employee class and derive sub classes for Head Chef, Chef, Apprentice Chef and store the other values in fields.
- 3) Create an employee class with fields for Job title and fields for the other values.
- 4) Create classes for all of the items mentioned and create a container class to represent employees

Answer to Question 52)

Question 53)

You need to read in the lines of a large text file containing tens of megabytes of data.
Which of the following would be most suitable for reading in such a file

- 1) new FileInputStream("file.name")
- 2) new InputStreamReader(new FileInputStream("file.name"))
- 3) new BufferedReader(new InputStreamReader(new FileInputStream("file.name")));
- 4) new RandomAccessFile(raf=new RandomAccessFile("myfile.txt","+rw");

Answer to Question 53)

Question 54)

What will happen when you attempt to compile and run the following code?

```
public class Inc{  
    public static void main(String argv[]){  
        Inc inc = new Inc();  
        int i =0;  
        inc.fermin(i);  
        i = i++;  
        System.out.println(i);  
    }  
    void fermin(int i){  
        i++;  
    }  
}
```

- 1) Compile time error
- 2) Output of 2
- 3) Output of 1
- 4) Output of 0

Answer to Question 54)
Question 55)

What will happen when you attempt to compile and run the following code?

```
public class Agg{  
    static public long i=10;  
    public static void main(String argv[]){  
        switch(i){  
            default:  
                System.out.println("no value given");  
            case 1:  
                System.out.println("one");  
            case 10:  
                System.out.println("ten");  
            case 5:  
                System.out.println("five");  
        }  
    }  
}
```

- 1) Compile time error
- 2) Output of "ten" followed by "five"
- 3) Output of "ten"
- 4) Compilation and run time error because of location of default

Answer to Question 55)
Question 56)

Given the following class

```
public class ZeroPrint{  
    public static void main(String argv[]){  
        int i =0;  
        //Here  
    }  
}
```

Which of the following lines if placed after the comment //Here will print out 0.

- 1) System.out.println(i++);
- 2) System.out.println(i+'0');
- 3) System.out.println(i);
- 4) System.out.println(i--);

Answer to Question 56)
Question 57)

Given the following code

```
class Base { }

class Agg extends Base{
    public String getFields(){
        String name = "Agg";
        return name;
    }
}

public class Avf{
    public static void main(String argv[]){
        Base a = new Agg();
        //Here
    }
}
```

What code placed after the comment //Here will result in calling the getFields method resulting in the output of the string "Agg"?

- 1) System.out.println(a.getFields());
- 2) System.out.println(a.name);
- 3) System.out.println((Base) a.getFields());
- 4) System.out.println(((Agg) a).getFields());

Answer to Question 57)
Question 58)

What will happen when you attempt to compile and run the following code.

```
public class Pvf{

    static boolean Paddy;
    public static void main(String argv[]){
        System.out.println(Paddy);
    }
}
```

- 1) Compile time error
- 2) compilation and output of false

- 3) compilation and output of true
- 4) compilation and output of null

Answer to Question 58)

Question 59)

Which of the following statements are true?

- 1) The x,y coordinates of an instance of MouseEvent can be obtained using the getX() and getY() methods
- 2) The x,y coordinates of an instance of MouseEvent can be obtained using the X and Y integer fields
- 3) The time of a MouseEvent can be extracted using the getTime() method
- 4) The time of a MouseEvent can be extracted using the getWhen method

Answer to Question 59)

Question 60)

Given the following code

```
import java.io.*;  
  
public class Ppvg{  
    public static void main(String argv[]){  
        Ppvg p = new Ppvg();  
        p.fliton();  
    }  
    public int fliton(){  
        try{  
            FileInputStream din = new FileInputStream("Ppvg.java");  
            din.read();  
        }catch( IOException ioe){  
            System.out.println("flytwick");  
            return 99;  
        }finally{  
            System.out.println("fliton");  
        }  
        return -1;  
    }  
}
```

Assuming the file Ppvg.java is available to be read which of the following statements are true if you try to compile and run the program?

- 1) The program will run and output only "flytwick"

- 2) The program will run and output only "fliton"
- 3) The program will run and output both "fliton" and "flytwick"
- 4) An error will occur at compile time because the method fliton attempts to return two values

Answer to Question 60)

Answers

Answer to Question 1)

Objective 4.5)

- 1) float f=1/3;
- 2) int i=1/3;
- 4) double d=999d;

The fact that option 3 does not compile may be a surprise. The problem is because the default type for a number with a decimal component is a double and not a float. The additional trailing d in the option with 999 doesn't help, but it doesn't harm.

http://www.jchq.net/tutorial/04_05Tut.htm

Answer to Question 2)

Objective 4.3)

- 2) new

The option NULL (note the upper case letter) is definitely not a keyword. There is some discussion as to i. There is some discussion as to if null is a keyword but for the purpose of the exam you should probably assume it is a keyword.

The option instanceof is a bit of a misleading option that would probably not occur on the exam. The real keyword is instanceof (note that the of has no capital letter O). I had the incorrect version in an earlier version of this tutorial as it looks more likely to my eyes. The instanceof keyword looks like a method, but it is actually an operator.

The option wend is probably valid in some other language to indicate the end of a while loop, but Java has no such keyword.

http://www.jchq.net/tutorial/04_03Tut.htm

Answer to Question 3)

Objective 4.5)

- 1) System.out.println(1+1);
- 2) int i=2+'2';

Option 3 is not valid because single quotes are used to indicate a character constant and not a string. Several people have emailed me to say that option 3 will compile. When they eventually compiled the exact code they have agreed, it will not compile. Let me re-state that

String s="on"+'one';

Will NOT compile.

Option 4 will not compile because 255 is out of the range of a byte

http://www.jchq.net/tutorial/04_05Tut.htm

Answer to Question 4)

Objective 7.1)

- 1) The garbage collection algorithm in Java is vendor implemented

Threading and garbage collection are two of the few areas that are platform dependent. This is one of the reasons why Java is not suitable for realtime programming. It is not a good idea use it to control your plane or nuclear power station. Once an instance of the Integer class has a value it cannot be changed.

http://www.jchq.net/tutorial/07_01Tut.htm

Answer to Question 5)

Objective 10.1)

(Not on the official sub objectives but this topic does come up on the exam)

- 2) The RandomAccessFile class allows you to move directly to any point a file.
- 4) The characteristics of an instance of the File class such as the directory separator, depend on the current underlying operating system

The File class can be considered to represent information about a file rather than a real file object. You can create a file in the underlying operating system by passing an instance of a file to a stream such as FileOutputStream. The file will be created when you call the close method of the stream.

http://www.jchq.net/tutorial/10_01Tut.htm

Answer to Question 6)

Objective 5.1)

2) The instanceof operator can be used to determine if a reference is an instance of a particular primitive wrapper class

The instanceof operator can only be used to make a static comparison with a class type. Java 1.1 added the `isInstance` method to the class `Class` to allow you to dynamically determine a class type. The exam does not test you on `isInstance`.

http://www.jchq.net/tutorial/05_01Tut.htm

Answer to Question 7)

Objective 4.1)

2) Interfaces cannot have constructors

If you try to create a constructor for an Interface the compiler will give you an error message something like

"interface can't have constructors".

4) Interfaces are the Java approach to addressing the single inheritance model, but require implementing classes to create the functionality of the Interfaces.

An interface may contain variables as well as methods. However any variables are final by default and must be assigned values on creation. A class can only extend one other class (single inheritance) but may implement as many interfaces as you like (or is sensible).

http://www.jchq.net/tutorial/04_01Tut.htm

Answer to Question 8)

Objective 9.1)

None of these are valid statements. The `Math` class is final and cannot be extended. The `max` method takes two parameters, `round` only takes one parameter and there is no `mod` parameter. You may get questions in the exam that have no apparently correct answer. If you are absolutely sure this is the case, do not check any of the options.

http://www.jchq.net/tutorial/09_01Tut.htm

Answer to Question 9)

Objective 7.1)

1) The Runnable interface has only one method run that needs to be created in any class that implements it. The start method is used to actually call and start the run method executing.

http://www.jchq.net/tutorial/07_01Tut.htm

Answer to Question 10)

Objective 4.5)

1) A byte can represent between -128 to 127

The char type is the only unsigned type in Java and thus cannot represent a negative number.

For more information on this topic go to

http://www.jchq.net/tutorial/04_05Tut.htm

Answer to Question 11)

Objective 1.2)

2) Compilation and no output at runtime

Because the method in Base called Base has a return type it is not a constructor and therefore does not get called on creation of an instance of its child class In

For more information on this topic go to

http://www.jchq.net/tutorial/01_02Tut.htm

Answer to Question 12)

Objective 4.2)

4) Compilation and output of hello

This type of question is particularly calculated to catch out C/C++ programmers who might expect parameter zero to be the name of the compiler.

For more information on this topic go to

http://www.jchq.net/tutorial/04_02Tut.htm

Answer to Question 13)

Objective 1.2)

- 1) If a class has any abstract methods it must be declared abstract itself.
- 3) The final modifier means that a class cannot be sub-classed
- 4) transient and volatile are Java modifiers

An abstract class may have non abstract methods. Any class that descends from an abstract class must implement the abstract methods of the base class or declare them as abstract itself.

For more information on this topic go to

http://www.jchq.net/tutorial/01_02Tut.htm

Answer to Question 14)

Objective 1.2)

- 2) public static void amethod(){ }
- 4) static native void amethod();

Option 1 is not valid because it has braces and the native modifier means that the method can have no body. This is because the body must be implemented in some other language (often C/C++). Option 3 is not valid because private and protected contradict themselves.

For more information on this topic go to

http://www.jchq.net/tutorial/01_02Tut.htm

Answer to Question 15)

Objective 6.2)

- 4) Constructors are not inherited

Constructors can be marked public, private or protected. Constructors do not have a return type.

For more information on this topic go to

http://www.jchq.net/tutorial/06_02Tut.htm

Answer to Question 16)

Objective 1.3)

2) Compile time error

An error occurs when the class `Severn` attempts to call the zero parameter constructor in the class `Base`. Because the `Base` class has an integer constructor Java does not provide the "behind the scenes" zero parameter constructor.

For more information on this topic go to

http://www.jchq.net/tutorial/01_03Tut.htm

Answer to Question 17)

Objective 1.2)

- 1) static methods do not have access to the implicit variable called this
- 2) A static method may be called without creating an instance of its class
- 3) a static may not be overriden to be non-static

The implicit variable `this` refers to the current instance of a class and thus by its nature a static method cannot have access to it.

For more information on this topic go to

http://www.jchq.net/tutorial/01_02Tut.htm

Answer to Question 18)

Objective 5.1)

1)

```
char c='1';
System.out.println(c>>1);
```

4)

```
int i=1;
System.out.println(i<<1);
```

Be aware that Integer (not the upper case I) is a wrapper class and thus cannot be treated like a primitive. The fact that option 1 will compile may be a surprise, but although the char type is normally used to store character types, it is actually an unsigned integer type. The reason option 3 does not compile is that Java has a >>> operator but not a <<< operator. ;>> operator but not a <<< operator.

For more information on this topic go to

http://www.jchq.net/tutorial/05_01Tut.htm

Answer to Question 19)

Objective 4.6)

- 2) An event listener may be removed from a component
- 3) The ActionListener interface has no corresponding Adapter class

A component may have multiple event listeners attached. Thus a field may need to respond to both the mouse and the keyboard, requiring multiple event handlers. The ActionListener has not matching Adapter class because it has only one method, the idea of the Adapter classes is to eliminate the need to create blank methods.

For more information on this topic go to

http://www.jchq.net/tutorial/04_06Tut.htm

Answer to Question 20)

Objective 4.3)

- 3) transient
- 4) volatile

Option 1, sizeof is designed to catch out the C/C++ programmers. Java does not have a sizeof keyword as the size of primitives should be consistent on all Java implementations. Although a program needs a main method with the standard signature to start up it is not a keyword. The real keywords are less commonly used and therefore might not be so familiar to you.

For more information on this topic go to

http://www.jchq.net/tutorial/04_03Tut.htm

Answer to Question 21)

Objective 1.3)

- 3) The default constructor takes no parameters
- 4) The default constructor is not created if the class has any constructors of its own.

Option 1 is fairly obviously wrong as constructors never have a return type. Option 2 is very dubious as well as Java does not offer void as a type for a method or constructor.

For more information on this topic go to

http://www.jchq.net/tutorial/05_01Tut.htm

Answer to Question 22)

Objective 4.1)

- 1) All of the variables in an interface are implicitly static
- 2) All of the variables in an interface are implicitly final
- 3) All of the methods in an interface are implicitly abstract

All the variables in an interface are implicitly static and final. Any methods in an interface have no body, so may not access any type of variable

http://www.jchq.net/tutorial/04_01Tut.htm

Answer to Question 23)

Objective 4.5)

- 2) The + operator is overloaded for concatenation for the String class

In Java Strings are implemented as a class within the Java.lang package with the special distinction that the + operator is overloaded. If you thought that the String class is implemented as a char array, you may have a head full of C++ that needs emptying. There is not "wrapper class" for String as wrappers are only for primitive types.

If you are surprised that option 4 is not a correct answer it is because length is a method for the String class, but a property for an array and it is easy to get the two confused.

http://www.jchq.net/tutorial/05_01Tut.htm

Answer to Question 24)

Objective 6.1)

- 1) A method in an interface must not have a body
- 3) A class may extends one other class plus many interfaces

A class accesses an interface using the implements keyword (not uses)

http://www.jchq.net/tutorial/06_01Tut.htm

Answer to Question 25)

Objective 5.1)

- 3) The following statement will produce a result of zero, System.out.println(1 >>1);

Although you might not know the exact result of the operation -1 >>> 2 a knowledge of the way the bits will be shifted will tell you that the result is not plus 1. (The result is more like 1073741823) There is no such Java operator as the unsigned left shift.

Although it is normally used for storing characters rather than numbers the char Java primitive is actually an unsigned integer type.

http://www.jchq.net/tutorial/05_01Tut.htm

And for information on the size of primitives see

http://www.jchq.net/tutorial/04_05Tut.htm

Answer to Question 26)

Objective 4.4)

- 2) Arrays elements are initialized to default values wherever they are created using the keyword new.

You can find the size of an array using the length field. The method length is used to return the number of characters in a String. An array can contain elements of any type but they must all be of the same type. The size of an array is fixed at creation. If you want to change its size you can of course create a new array and assign the old one to it. A more flexible approach can be to use a collection class such as Vector.

http://www.jchq.net/tutorial/04_04Tut.htm

Answer to Question 27)

Objective 5.3)

- 2) Output of "Hello Crowle"

This code is an example of a short circuited operator. Because the first operand of the || (or) operator returns true Java sees no reason to evaluate the second. Whatever the value of the second the overall result will always be true. Thus the method called place is never called.

http://www.jchq.net/tutorial/05_03Tut.htm

Answer to Question 28)

Objective 6.2)

- 4) none of the above;

You may access methods of a direct parent class through the use of super but classes further up the hierarchy are not visible.

http://www.jchq.net/tutorial/06_02Tut.htm

Answer to Question 29)

Objective 6.1)

- 2) A method with the same name completely replaces the functionality of a method earlier in the hierarchy

Option 3 is more like a description of overloading. I like to remind myself of the difference between overloading and overriding in that an overridden method is like something overridden in the road, it is squashed, flat no longer used and replaced by something else. An overloaded method has been given extra work to do (it is loaded up with work), but it is still being used in its original format. This is just my little mind trick and doesn't match to anything that Java is doing.

http://www.jchq.net/tutorial/06_01Tut.htm

Answer to Question 30)

Objective 1.2)

- 2) The / operator is used to divide one value by another
- 3) The # symbol may not be used as the first character of a variable

The % is the modulo operator and returns the remainder after a division. Thus 10 % 3=1
The \$ symbol may be used as the first character of a variable, but I would suggest that it is generally not a good idea. The # symbol cannot be used anywhere in the name of a variable. Knowing if a variable can start with the # or \$ characters may seem like arbitrary and non essential knowlege but questions like this do come up on the exam.

http://www.jchq.net/tutorial/01_02Tut.htm

Answer to Question 31)

Objective 8.1)

- 1) The default layout manager for an Applet is FlowLayout
- 4) The FlowLayout manager attempts to honor the preferred size of any components

The default layout manager for an Application is BorderLayout. An applet will use the default of FlowLayout if one is not specifically applied.

http://www.jchq.net/tutorial/08_01Tut.htm

Answer to Question 32)

Objective 1.2)

- 3) Only one instance of a static variable will exist for any amount of class instances

Option 1) is more a description of a final variable. Option 2 is designed to fool Visual Basic programmers like me as this is how you can use the keyword static in VB. The modifier static can be applied to a class (only an inner class), method or variable.

http://www.jchq.net/tutorial/01_02Tut.htm

Answer to Question 33)

Objective 11.1)

- 1) Java uses a system called UTF for I/O to support international character sets
- 3) An instance of FileInputStream may not be chained to an instance of FileOutputStream
- 4) File I/O activities requires use of Exception handling

Internally Java uses Unicode which are 16 bit characters. For I/O Java uses UTF which may be more than 16 bits per character than 16 bits per character.

Generally InputStreams can only be chained to other InputStreams and OutputStreams can only be chained to other OutputStreams. The piped streams are an exception to this.

http://www.jchq.net/tutorial/11_01Tut.htm

Answer to Question 34)

Objective 1.2)

1) Compile time error

It will produce an error like "Abstract and native method can't have a body. This is typical of the more misleading question where you might think it is asking you about the circumstances under which the finally clause runs, but actually it is about something else.

http://www.jchq.net/tutorial/07_02Tut.htm

Answer to Question 35)

Objective 7.1)

2) Compilation and run with the output "Running"

This is perfectly legitimate if useless sample of creating an instance of a Thread and causing its run method to execute via a call to the start method. The Thread class is part of the core java.lang package and does not need any explicit import statement. The reference to a Thread target is an attempt to mislead with a reference to the method of using the Runnable interface instead of simply inheriting from the Thread super class.

http://www.jchq.net/tutorial/07_01Tut.htm

Answer to Question 36)

Objective 11.1)

1) RandomAccessFile raf=new RandomAccessFile("myfile.txt","rw");

The RandomAccessFile is an anomaly in the Java I/O architecture. It descends directly from Object and is not part of the Streams architecture.

http://www.jchq.net/tutorial/11_01Tut.htm

Answer to Question 37)

Objective 6.2)

- 2) public int amethod(int i, int j) {return 99;}
- 3) protected void amethod (long l){}
- 4) private void anothermethod(){}

Option 1 will not compile on two counts. One is the obvious one that it claims to return an integer. The other is that it is effectively an attempt to redefine a method within the same class. The change of name of the parameter from i to z has no effect and a method cannot be overridden within the same class.

http://www.jchq.net/tutorial/06_02Tut.htm

Answer to Question 38)

Objective 8.1)

- 1) Code must be written to cause a frame to close on selecting the system close menu
- 2) The default layout for a Frame is the BorderLayout Manager
- 3) The GridBagLayout manager makes extensive use of the the GridBagConstraints class.

You can change the layout manager for a Frame or any other container whenever you like.

http://www.jchq.net/tutorial/08_01Tut.htm

Answer to Question 39)

Objective 1.2)

- 4) The code will compile without error

There are no restrictions on the level of nesting for inner/nested classes. Inner classes may be marked private. The main method is not declared as public static void main, and assuming that the commandline was java Droitwich it would not be invoked anyway.

http://www.jchq.net/tutorial/01_02Tut.htm

Answer to Question 40)

Objective 1.2)

- 1) super.oak=1;
- 2) oak=33;
- 3) Base.oak=22;

Because the variable oak is declared as static only one copy of it will exist. Thus it can be changed either through the name of its class or through the name of any instance of that class. Because it is created as an integer it cannot be assigned a fractional component without a cast.

http://www.jchq.net/tutorial/01_02Tut.htm

Answer to Question 41)

Obj Question 41)

Objective 4.6)

4) Use the getText method of a Textfield and use the parseInt method of the Integer class

Here is an example of how you might do this

```
Integer.parseInt(txtInputValue.getText());
```

I'm not sure that a question on this actually will come up in the exam but it is a very useful thing to know in the real world.

http://www.jchq.net/tutorial/04_06Tut.htm

Answer to Question 42)

Objective 4.6)

4) none of the above

The wrapper classes are immutable. Once the value has been set it cannot be changed. A common use of the wrapper classes is to take advantage of their static methods such as Integer.parseInt(String s) that will returns an integer if the the value has been set it cannot be changed. A common use of the wrapper classes is to take advantage of their static methods such as Integer.parseInt(String s) that will returns an integer if the String contains one.

http://www.jchq.net/tutorial/04_06Tut.htm

Answer to Question 43)

Objective 6.2)

2) constructors cannot be overriden

Overloading constructors is a key technique to allow multiple ways of initialising classes. By definition, constructors have no return values so option 3 makes no sense. Option 4 is the inverse of what happens as constructor code will execute starting from the oldest ancestor class downwards. You can test this by writing a class that inherits from a base class and getting the constructor to print out a message. When you create the child class you will see the order of constructor calling.

http://www.jchq.net/tutorial/06_02Tut.htm

Answer to Question 44)

Objective 7.1)

yield is a static method and causes whatever thread is currently executing to yield its cycles.

- 1) t.yield();
- 2) Thread.yield()

(Thanks Roseanne)

http://www.jchq.net/tutorial/07_01Tut.htm

JavaDoc for the Thread class

<http://java.sun.com/products/jdk/1.2/docs/api/java/lang/Thread.html>

Answer to Question 45)

Objective 6.2)

- 4) Compilation and run with an output of 99

The fact that the variable court is declared as private does not stop the constructor from being able to initialise it.

http://www.jchq.net/tutorial/06_02Tut.htm

Answer to Question 46)

Objective 6.2)

- 3) To be overriden a method must have the same name, parameter and return types

Option 1 is a sneaky one in that it should read overriden not overloaded. An overriden method must also have the same return type. Parameter names are purely a programmer convenience and are not a factor in either overloading and overriding. Parameter order is a factor however.

http://www.jchq.net/tutorial/06_02Tut.htm

Answer to Question 47)

Objective 6.2)

- 1) Compile time error

With the sun JDK it will produce the following error

"Only constructors can invoke constructors".

If you took out the call to super that causes this error the program would compile and at runtime it would output Base and then Checket as constructors are called from the oldest ancestor class downwards.

http://www.jchq.net/tutorial/06_02Tut.htm

Answer to Question 48)

Objective 1.2)

1) Static methods cannot be overridden to be non static

The JDK1.1 compiler will issue an error message "static methods cannot be overridden" if you attempt to do this; if you attempt to do this. There is no logic or attempt to do this. There is no logic or reason why private methods should not be overloaded or that static methods should not be declared private. Option 4 is a jumbled up version of the limitations of exceptions for overridden methods

http://www.jchq.net/tutorial/01_02Tut.htm

Answer to Question 49)

Objective 3.1)

2) A program can suggest that garbage collection be performed but not force it

4) A reference becomes eligible for garbage collection when it is assigned to null

If a program keeps creating new references without any being discarded it may run out of memory. Unlike most aspects of Java garbage collection is platform dependent.

http://www.jchq.net/tutorial/03_01Tut.htm

Answer to Question 50)

Objective 1.2)

2) Compile time error

The main method is static and cannot access the non static variable x

http://www.jchq.net/tutorial/01_02Tut.htm

Answer to Question 51)

Objective 1.2)

1) Compile time error

When compiled with JDK 1.1 the following error is produced.

Abstract and native methods can't have a body: void hallow() abstract void hallow()

http://www.jchq.net/tutorial/01_02Tut.htm

Answer to Question 52)

Objective 6.1)

3) Create an employee class with fields for Job title and fields for the other values.

These questions can appear tricky as the whole business of designing class structures is more art than science. It is asking you to decide if an item of data is best represented by the "Is a" or "Has a" relationship. Thus in this case any of the job titles mentioned will always refer to something that "Is a" employee. However the employee "has a" job title that might change.

One of the important points is to ask yourself when creating a class "Could this change into another class at some point in the future". Thus in this example an apprentice chef would hope one day to turn into a chef and if she is very good will one day be head chef. Few other mock exams seem to have this type of questions but they do come up in the real exam.

http://www.jchq.net/tutorial/06_01Tut.htm

Answer to Question 53)

Objective 11.1)

3) new BufferedReader(new InputStreamReader(new FileInputStream("file.name")));

The key to this question is that it asks about tens of megabytes of data, implying that performance is an issue. A BufferedReader will optimise the performance of accessing a file. Although the objectives do not specifically mention it questions on I/O do come up on the exam.

http://www.jchq.net/tutorial/11_01Tut.htm

Answer to Question 54)

Objective 5.4)

4) Output of 0

The method `fermin` only receives a copy of the variable `i` and any modifications to it are not reflected in the version in the calling method. The post increment operator `++` effectively modifies the value of `i` after the initial value has been assigned to the left hand side of the equals operator. This can be a very tricky concept to understand.

http://www.jchq.net/tutorial/05_04Tut.htm

Answer to Question 55)

Objective 2.1)

1) Compile time error

This might be considered a "gocha" or deliberate attempt to mislead you because `i` has been given the data type of long and the parameter must be of long and the parameter must be either a byte, char, short or int. If you attempt to compile this code with JDK 1.2 you will get an error that says something like "Incompatible type for switch, Explicit cast needed to convert long to int". Answering with option 2 would have been reasonable because if the parameter had been an integer type the lack of break statements would have caused this output. If you gave either of the answers you should probably revise the subject.

http://www.jchq.net/tutorial/02_02Tut.htm

Answer to Question 56)

Objective 5.1)

- 1) `System.out.println(i++);`
- 3) `System.out.println(i);`
- 4) `System.out.println(i--);`

The options for this question might look suspiciously easy if you are not aware of the effects of the post-increment operators. The `++` and `--` operations for examples 1 and 4 only come into effect after the output operations, ie after whatever else is done to them on that line of code. Option 2 should be fairly obvious as you should know that the single quote characters indicate a char value, ie storing the character rather than the numerical value for 0.

http://www.jchq.net/tutorial/05_01Tut.htm

Answer to Question 57)

Objective 6.2)

4) System.out.println(((Agg) a).getFields());

The Base type reference to the instance of the class Agg needs to be cast from Base to Agg to get access to its methods. The method invoked depends on the object itself, not on the declared type. So, a.getField() tries to invoke a getField method in Base which does not exist. But the call to ((Agg)a).getField() will invoke the getField() in the Agg class. You will be unlucky to get a question as complex as this on the exam. If you think option 1 is valid, have a go at compiling the code.

http://www.jchq.net/tutorial/06_02Tut.htm

Answer to Question 58)

Objective 4.4)

2) compilation and output of false

A variable defined at class level will always be given a default value and the default value for the primitive type boolean is false

http://www.jchq.net/tutorial/04_04Tut.htm

Answer to Question 59)

Objective 4.6)

1) The x,y coordinates of an instance of MouseEvent can be obtained using the getX() and getY() methods

4) The time of a MouseEvent can be extracted using the getWhen method

If you chose option 4, referring to the mythical getTime method you have made a reasonable guess based on the normal conventions of Java. However the conventions do not always hold true. If you chose option 3 perhaps you are not as aware of the conventions as you should be.

http://www.jchq.net/tutorial/04_06Tut.htm

Answer to Question 60)

Objective 2.3

2) The program will run and output only "fliton"

This question tests your knowledge of the principle that the finally clause will almost always run.

http://www.jchq.net/tutorial/02_03Tut.htm

operators

Question 1

```
class EBH019 {
    public static void main (String args[]) {
        int i1 = 0xffffffff, i2 = i1 << 1;
        int i3 = i1 >> 1, i4 = i1 >>> 1;
        System.out.print(Integer.toHexString(i2) + ", ");
        System.out.print(Integer.toHexString(i3) + ", ");
        System.out.print(Integer.toHexString(i4));
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: ffffffff,fffffff,fffffff
- b. Prints: ffffffff,fffffff,7fffffff
- c. Prints: ffffffff,7fffffff,fffffff
- d. Prints: ffffffff,7fffffe,7fffffe
- e. Prints: ffffffe,fffffff,fffffff
- f. ANS Prints: ffffffe,fffffff,7fffffe
- g. Prints: ffffffe,7fffffff,fffffff
- h. Prints: ffffffe,7fffffff,7fffffff
- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 2

```
class EBH201 {
    public static void main (String[] args) {
        int a = 1 || 2 ^ 3 && 5;
        int b = ((1 || 2) ^ 3) && 5;
        int c = 1 || (2 ^ (3 && 5));
        System.out.print(a + "," + b + "," + c);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0
- b. Prints: 0,0,3
- c. Prints: 0,3,0
- d. Prints: 0,3,3
- e. Prints: 3,0,0
- f. Prints: 3,0,3
- g. Prints: 3,3,0
- h. Prints: 3,3,3
- i. Run-time error
- j. ANS Compile-time error
- k. None of the above

Question 3

```
cl
ass EBH202 {
    static boolean a, b, c;
    public static void main (String[] args) {
        boolean x = (a = true) || (b = true) && (c = true);
        System.out.print(a + "," + b + "," + c);
```

}}

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e.ANS Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 4

```
class EBH203 {  
    static boolean a, b, c;  
    public static void main (String[] args) {  
        boolean x = a || (b = true) && (c = true);  
        System.out.print(a + "," + b + "," + c);  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d.ANS Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 5

```
class EBH011 {  
    public static void main (String[] args) {  
        float a = Float.POSITIVE_INFINITY;  
        double b = Double.POSITIVE_INFINITY;  
        double c = Double.NaN;  
        System.out.print((a == b)+", "+(c == c)+", "+(c != c));  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f.ANS Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true

- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 6

```
class EBH012 {  
    public static void main (String[] args) {  
        byte x = 3, y = 5;  
        System.out.print((-x == ~x + 1)+", "+(-y == ~y + 1));  
    }}  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. ANS Prints: true,true
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 7

```
class EBH013 {  
    public static void main (String[] args) {  
        byte x = 3, y = 5;  
        System.out.print((~x == -x - 1)+", "+(~y == -y - 1));  
    }}  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. ANS Prints: true,true
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 8

```
class EBH014 {  
    public static void main (String[] args) {  
        byte x = 3, y = 5;  
        System.out.print((y % x) + ",");  
        System.out.print(y == ((y/x)*x + (y%x)));  
    }}  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,true
- b. ANS Prints: 2,true
- c. Prints: 1,false
- d. Prints: 2,false
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 9

```

class Color {}

class Red extends Color {}
class Blue extends Color {}
class A {
    public static void main (String[] args) {
        Color color1 = new Red(); Red color2 = new Red();
        boolean b1 = color1 instanceof Color;
        boolean b2 = color1 instanceof Blue;
        boolean b3 = color2 instanceof Blue;
        System.out.print(b1+", "+b2+", "+b3);
    }
}

```

What is the result of attempting to compile and run the program?

- a. false,false,false
- b. false,false,true
- c. false,true,false
- d. false,true,true
- e. true,false,false
- f. true,false,true
- g. true,true,false
- h. true,true,true
- i. Run-time error
- j.ANS Compile-time error
- k. None of the above

Question 10

```

class EBH020 {
    public static void main (String[] args) {
        int a = 1 | 2 ^ 3 & 5;
        int b = ((1 | 2) ^ 3) & 5;
        int c = 1 | (2 ^ (3 & 5));
        System.out.print(a + "," + b + "," + c);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0
- b. Prints: 0,0,3
- c. Prints: 0,3,0
- d. Prints: 0,3,3
- e. Prints: 3,0,0
- f.ANS Prints: 3,0,3
- g. Prints: 3,3,0
- h. Prints: 3,3,3
- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 11

```

class EBH025 {
    public static void main (String args[]) {
        int i1 = 0xffffffff, i2 = i1 << 33;
        int i3 = i1 << (33 & 0x1f);
        System.out.print(Integer.toHexString(i2) + ", ");
    }
}

```

```
        System.out.print(Integer.toHexString(i3));
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0
- b. Prints: 0,ffffffe
- c. Prints: 0,fffffff
- d. Prints: ffffffff,fffffff
- e. Prints: ffffffff,ffffffe
- f. Prints: ffffffe,fffffff
- g.ANS Prints: ffffffe,ffffffe
- h. Run-time error
- i. Compile-time error
- j. None of the above

Question 12

```
class EBH001 {
    static int m(int i) {System.out.print(i + " , "); return i;}
    public static void main(String s[]) {
        m(m(1) - m(2) + m(3) * m(4));
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1, 2, 3, 4, 8,
- b.ANS Prints: 1, 2, 3, 4, 11,
- c. Prints: 3, 4, 1, 2, 11,
- d. Run-time error
- e. Compile-time error
- f. None of the above

Question 13

```
class EBH002 {
    static int m(int i) {System.out.print(i + " , "); return i;}
    public static void main(String s[]) {
        m(m(1) + m(2) % m(3) * m(4));
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1, 2, 3, 4, 0,
- b. Prints: 1, 2, 3, 4, 3,
- c.ANS Prints: 1, 2, 3, 4, 9,
- d. Prints: 1, 2, 3, 4, 12,
- e. Prints: 2, 3, 4, 1, 9,
- f. Prints: 2, 3, 4, 1, 3,
- g. Run-time error
- h. Compile-time error
- i. None of the above

Question 14

```
class EBH005 {
    public static void main (String[] s) {
        byte b = 127; b <= 2;System.out.println(b);
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: -4
- b. Prints: -3
- c. Prints: -2
- d. Prints: 0
- e. Prints: 1
- f. Prints: 127
- g. Prints: 508
- h. Run-time error
- i. Compile-time error
- j. None of the above

Question 15

```
class EBH007{  
    public static void main (String[] s) {  
        byte b = 5; System.out.println(b<<33);  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: -1
- b. Prints: 0
- c. Prints: 1
- d. Prints: 5
- e. Prints: 10
- f. Run-time error
- g. Compile-time error
- h. None of the above

Question 16

```
class EBH015 {  
    public static void main (String[] args) {  
        System.out.print((new Object() instanceof Object)+",");  
        System.out.print((new Object() instanceof String)+",");  
        System.out.print((new String() instanceof Object));  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 17

```
class EBH101 {  
    static int m(int i) {System.out.print(i + " , "); return i;}  
    public static void main(String s[]) {  
        int i = 1; m(m(++i) + m(i++) + m(-i) + m(i++));  
    } }
```

}}

What is the result of attempting to compile and run the above program?

- a. Prints: 1, 2, 3, 4, 10,
- b. Prints: 1, 2, -3, 4, 4,
- c. Prints: 2, 2, -3, -3, -2,
- d. Prints: 2, 2, -3, 3, 4,
- e. Prints: 2, 3, -3, -2, 0,
- f. Prints: 2, 3, -3, 4, 6,
- g. Prints: 2, 3, 4, 5, 14,
- h. Run-time error
- i. Compile-time error
- j. None of the above

Question 18

```
class EBH102 {  
    static int m(int i) {System.out.print(i + ","); return i;}  
    public static void main(String s[]) {  
        int i = 1, j = m(i++) + m(i++) * m(i++) + m(i++);  
        System.out.print(j % 5);  
    } }
```

What is the result of attempting to compile and run the above program?

- a. Prints: 1,2,3,4,0
- b. Prints: 1,2,3,4,1
- c. Prints: 1,2,3,4,2
- d. Prints: 1,2,3,4,3
- e. Prints: 1,2,3,4,4
- f. Prints: 1,2,3,4,5
- g. Run-time error
- h. Compile-time error
- i. None of the above

Question 19

```
class EBH103 {  
    public static void main (String[] args) {  
        byte a = 1, b = 2, c = (byte)a++, d = (byte)++b, e = (byte)a + b;  
        System.out.print(c + d + e);  
    } }
```

What is the result of attempting to compile and run the above program?

- a. Prints: 1 2 3
- b. Prints: 6
- c. Prints: 2 3 5
- d. Prints: 10
- e. Prints: 1 3 4
- f. Prints: 8
- g. Prints: 1 3 5
- h. Prints: 9
- i. Run-time error.
- j. Compile-time error.
- k. None of the above

Question 20

```

class EBH204 {
    static boolean m1(String s, boolean b) {
        System.out.print(s + (b ? "T" : "F"));
        return b;
    }
    public static void main(String[] args) {
        m1("A",m1("B",false) || m1("C",true) || m1("D",false));
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: ATBFCT
- b. Prints: ATBFCTDF
- c. Prints: BFCTAT
- d. Prints: BTCTDFAT
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 21

```

class EBH104 {
    static int m(int i) {System.out.print(i + ", "); return i;}
    public static void main(String s[]) {
        int i = 1; m(m(++i) - m(i++) + m(-i) * m(~i));
    }
}

```

What is the result of attempting to compile and run the above program?

- a. Prints: 2, 2, -3, -4, 8,
- b. Prints: 2, 2, -3, -4, 12,
- c. Prints: 2, 3, -3, -4, 7,
- d. Prints: 1, 1, 1, 1, 0,
- e. Prints: 2, 2, -2, -2, 4,
- f. Prints: 2, 3, -2, -2, 3,
- g. Prints: -1, -2, 2, 2, 0,
- h. Run-time error
- i. Compile-time error
- j. None of the above

Question 22

```

class EBH105 {
    static int m(int i) {System.out.print(i + ","); return 0;}
    public static void main (String[] args) {
        int i = 0; i = i++ + m(i); System.out.print(i);
    }
}

```

What is the result of attempting to compile and run the above program?

- a. Prints: 0,0
- b. Prints: 1,0
- c. Prints: 0,1
- d. Prints: 1,1
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 23

```
class EBH106 {
    public static void main(String args[ ]) {
        int a = 1; a += ++a + a++; System.out.print(a);
    }
}
```

What is the result of attempting to compile and run the above program?

- a. Prints: 3
- b. Prints: 4
- c. Prints: 5
- d. Prints: 6
- e. Prints: 7
- f. Run-time error
- g. Compile-time error
- h. None of the above

Question 24

```
class EBH107 {
    static int m(int i) {System.out.print(i + ","); return i;}
    public static void main(String s[]) {
        int i=0, j = m(++i) + m(++i) * m(++i) % m(++i) + m(++i);
        System.out.print(j%5);
    }
}
```

What is the result of attempting to compile and run the above program?

- a. Prints: 1,2,3,4,5,0
- b. Prints: 1,2,3,4,5,1
- c. Prints: 1,2,3,4,5,2
- d. Prints: 1,2,3,4,5,3
- e. Prints: 1,2,3,4,5,4
- f. Prints: 1,2,3,4,5,5
- g. Run-time error
- h. Compile-time error
- i. None of the above

Question 25

```
class EBH108 {
    public static void main(String s[]) {
        int i=0, j = ++i + ((++i * ++i) % ++i) + ++i;
        System.out.print(j%5);
    }
}
```

What is the result of attempting to compile and run the above program?

- a. Prints: 1
- b. Prints: 2
- c. Prints: 3
- d. Prints: 4
- e. Prints: 5
- f. Run-time error
- g. Compile-time error
- h. None of the above

Question 26

```
class EBH023 {
    static String m1(boolean b){return b?"T":"F";}
```

```

public static void main(String [] args) {
    boolean b1 = false?false:true?false:true?false:true;
    boolean b2 = false?false:(true?false:(true?false:true));
    boolean b3 = ((false?false:true)?false:true)?false:true;
    System.out.println(m1(b1) + m1(b2) + m1(b3));
}

```

What is the result of attempting to compile and run the program?

- a. Prints: FFF
- b. Prints: FFT
- c. Prints: FTF
- d. Prints: FTT
- e. Prints: TFF
- f. Prints: TFT
- g. Prints: TTF
- h. Prints: TTT
- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 27

```

class EBH024 {
    public static void main(String[] args) {
        int i1 = 15;
        String b1 = (i1>30)?"Red":(i1>20)?"Green":(i1>10)?"Blue":"Violet";
        String b2 =
(i1>30)?"Red":((i1>20)?"Green":((i1>10)?"Blue":"Violet"));
        System.out.println(b1 + "," + b2);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: Red,Red
- b. Prints: Green,Green
- c. Prints: Blue,Blue
- d. Prints: Violet,Violet
- e. Prints: Blue,Violet
- f. Prints: Violet,Blue
- g. Prints: Blue,Green
- h. Prints: Green,Blue
- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 28

```

class EBH003 {
    static int m(int i) {System.out.print(i + " , "); return i;}
    public static void main(String s[]) {
        m(m(~1) + m(1|2) + m(1&2) + m(1^3) + m(1<<1));
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: -2, 3, 0, 3, 0, 6,
- b. Prints: -2, 3, 0, 2, 1, 4,

- c. Prints: -2, 3, 0, 2, 2, 5,
- d. Prints: -2, 3, 0, 3, 2, 6,
- e. Prints: -1, 3, 0, 3, 2, 7,
- f. Prints: -2, 0, 3, 3, 0, 6,
- g. Prints: -1, 0, 3, 2, 1, 4,
- h. Prints: -2, 0, 3, 2, 2, 5,
- i. Prints: -2, 0, 3, 3, 2, 6,
- j. Prints: -1, 0, 3, 3, 2, 7,
- k. Run-time error
- l. Compile-time error
- m. None of the above

Question 29

```
class EBH021 {
    public static void main(String[] args) {
        System.out.print((-1 & 0x1f) + "," + (8 << -1));
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0
- b. Prints: -1,4
- c. Prints: 0x1f,8
- d. Prints: 31,16
- e. Run-time error
- f. Compile-time error
- g. None of the above

Answers: Answer Remark

1 f Prints: fffffffe, fffffff, 7fffffff If the left-hand operand of a shift operator, `<<`, `>>` and `>>>`, is of type int, then the shift distance is always within the range of 0 to 31, inclusive; and is specified by the least significant 5 bits of the right hand operand. Similarly, if the left-hand operand of a shift operator is of type long, then the shift distance is always within the range of 0 to 63, inclusive; and is specified by the least significant 6 bits of the right hand operand. The left shift operator, `<<`, shifts each bit of the left operand to the left a distance specified by the shift distance. A number of bits that is equal to the shift distance are shifted out of the left-most bit position and are lost. A number of bits that is equal to the shift distance are shifted in at the right. The signed right shift operator, `>>`, shifts each bit of the left operand to the right a distance specified by the shift distance. A number of bits that is equal to the shift distance are shifted out of the right-most bit position and are lost. A number of bits that is equal to the shift distance are shifted in at the left. The value of each bit that is shifted in at the left is equal to the value of the sign bit. The signed right shift operator maintains the sign of the left operand. The unsigned right shift operator, `>>>`, is similar to the signed right shift operator except for the fact that each bit shifted in at the left is zero.

2 j Compile-time error Both operands of the conditional and operator and the conditional or operator must be of type boolean.

3 e Prints: true,false,false The conditional and expression is not evaluated, because the left hand operand of the conditional or expression is true. The original expression is as follows: $x = (a = \text{true}) \mid\mid (b = \text{true}) \&\& (c = \text{true})$. The left hand operand of the conditional or expression is the result of the assignment expression, $(a = \text{true})$. Since the left hand operand of the conditional or expression is true, the right hand operand will not be evaluated. In this case, the right hand operand is the conditional and expression. Consequently, neither operand of the conditional and expression is evaluated and the variables, b and c, maintain their default values of false.

4 d Prints: false,true,true The right hand operand of the conditional or operator is evaluated only if the left hand operand is false. The right hand operand of the conditional and operator is only evaluated if the left hand operand is true. In this case, the left hand operand of the conditional or operator is false, so the right hand operand must also be evaluated. The left hand operand of the conditional and operator is the result of the conditional or expression, true, so the right hand operand is evaluated.

5 f Prints: true,false,true The positive infinity of type float is promoted to the positive infinity of type double. NaN is not equal to anything including itself.

6 d Prints: true,true The sign of an integral numeric type is changed by inverting all of the bits and by adding one.

7 d Prints: true,true The bitwise complement operator produces the same result as changing the sign and subtracting one. Please note that the operand must be an integral type. The bitwise complement operator can not be applied to a floating-point value.

8 b Prints: 2,true Suppose the left operand were divided by the right operand. The remainder operator returns the remainder of the division operation. For integral types, the identity, $(y == ((y/x)*x+(y%x)))$, is always true.

9 j Compile-time error The type of the reference color2 is Red. Since Red is not a subclass or a superclass of Blue, the expression color2 instanceof Blue is rejected at compile-time. Please note: The expression, x instanceof T, produces a compile-time error whenever the cast expression (T)x produces a compile-time error. If the program had been able to compile and run, the expression color1 instanceof Color would evaluate to true at run-time. The reference color1 refers to an instance of type Red. Since Red is a subclass of Color, the expression color1 instanceof Color would evaluate to true at run-time. The expression, color1 instanceof Blue would evaluate to false at run-time. The reference, color1, is of type Color. Since Color is a superclass of Blue, the expression, color1 instanceof Blue, is accepted at compile-time. The type of the object instance referenced by color1 is Red. Since Red is not Blue or a subclass of Blue, the expression, color1 instanceof Blue, would be false at run-time.

10 f Prints: 3,0,3 Java evaluates operands from left to right while respecting operator precedence. The order of operator precedence starting with the lowest is as follows: |, ^, &. Although complete memorization of the operator precedence chart is not necessary, it is a good idea to memorize the three levels that appear in this question.

11 g Prints: ffffffe,ffffffe For each of the three shift operators, <<, >> and >>>, the shift distance is specified by the right hand operand. If the left operand is of type int, then the shift distance

is always within the range 0 to 31, inclusive; and the following expression is always true: `(int1 << shift) == (int1 << (shift & 0x1f))`. The hexadecimal representation of decimal 31 is 0x1f and the binary representation is 0001 1111. The hexadecimal representation of decimal 33 is 0x21 and the binary representation is 0010 0001. The expression `i1 << (33 & 0x1f)` is equivalent to `(0xffffffff << (0x21 & 0x1f))`. Evaluation of the right hand operand of the shift operator produces `(0xffffffff << 1)`. The final result is 0xffffffe. Similarly, if the left operand is of type long, then the shift distance is always within the range 0 to 63, inclusive; and the following expression is always true: `(long1 << shift) == (long1 << (shift & 0x3f))`.

12 b Prints: 1, 2, 3, 4, 11, The expression can be simplified as follows: $j = 1 - 2 + 3 * 4 = 11$. The original expression is as follows: `m(m(1) - m(2) + m(3) * m(4))`. Simplification step one. Evaluate each operand from left to right: `m(1 - 2 + 3 * 4)`. Step two. Add parentheses to indicate operator precedence: `m(1 - 2 + (3 * 4))`. Step three. Evaluate the inner-most parentheses: `m(1 - 2 + 12)`. Step four: Work through the expression from left to right. $j = 11$.

13 c Prints: 1, 2, 3, 4, 9, The expression can be simplified as follows: $1 + 2 \% 3 * 4 = 9$. The original expression is as follows: `m(m(1) + m(2) \% m(3) * m(4))`. Simplification step one. Evaluate each operand from left to right: `m(1 + 2 \% 3 * 4)`. Step two. Add parentheses to indicate operator precedence and associativity: `m(1 + ((2 \% 3) * 4))`. Step three. Evaluate the inner-most parentheses: `m(1 + (2 * 4))`. Step four. Evaluate the inner-most parentheses: `m(1 + 8)`. The result is 9.

14 a Prints: -4 If the left-hand operand of the shift operator is of type byte, short, or char then the left operand is promoted to a 32 bit int and all four bytes are shifted. When a variable of type int with a value of 127 is shifted two bits to the left, the result is 508. The compound assignment operator includes an implicit cast to the type of the left-hand operand. The expression, `E1 op= E2`, is equivalent to `E1=(T)((E1) op (E2))`, where T is the type of the left hand operand. Therefore, when 508 is cast to an eight bit byte, the three most significant bytes (24 bits) are discarded leaving only the least significant byte (8 bits). The result is the binary value, 11111100, which is the two's complement representation of negative four.

15 e Prints: 10 If the left-hand operand of the shift operator is of type byte, short, or char then the left operand is promoted to a 32 bit int and all four bytes are shifted. If the promoted type of the left-hand operand is of type int, then the shift distance is always within the range of 0 to 31, inclusive; and is specified by the least significant 5 bits of the right-hand operand. In this case, the shift distance is 33, and the five least significant bits are 00001; so the shift distance is one bit. Note: If the type of the left hand operand is long, then the least significant six bits of the right hand operand are used.

16 f Prints: true,false,true The left operand of the instanceof operator must be null or a reference to an instance of an Object or a subclass of Object. The right operand of the instanceof operator must be a class type, interface type or array type. If the left operand is a reference to an instance of the type specified by the right operand or if the left operand is a reference to an instance of a subclass of the type specified by the right operand, then instanceof returns true.

17 d Prints: 2, 2, -3, 3, 4, The expression can be simplified as follows: $j = 2 + 2 + -3 + 3 = 4$. The original expression is as follows: $j = ++i + i++ + -i + i++$. Simplification step one. Evaluate the unary expressions from left to right: $j = 2 + 2 + -3 + 3$. Step two.

Complete the evaluation of the simplified expression: $j = 4$.

18 b Prints: 1,2,3,4,1 The expression can be simplified as follows: $j = 1 + (2 * 3) + 4 = 11$. The original expression is as follows: $j = m(i++) + m(i++) * m(i++) + m(i++)$. The method, m, prints and then returns the value of the parameter, so the original expression is equivalent to the following: $j = i++ + i++ * i++ + i++$. Step one. Work through the expression from left to right to evaluate the unary expressions: $j = 1 + 2 * 3 + 4$. Step two. Add parentheses to indicate operator precedence: $j = 1 + (2 * 3) + 4$. Step three. Work through the simplified expression: $j = 1 + 6 + 4 = 11$. Step four. Evaluate the expression that is the argument of the print method: $j \% 5 = 11 \% 5 = 1$.

19 j Compile-time error. The precedence of the cast operator is higher than the precedence of the addition operator, so the cast applies only to variable a and not to the result of the addition. Binary numeric promotion causes the byte variables a and b to be promoted to type int before the addition operation, and the result of the addition is also of type int. The attempt to assign the int result to the byte variable e generates a possible loss of precision error.

20 c Prints: BFCTAT The right operand of the conditional or operator is evaluated only if the left hand operand is false. In this case, the left operand of the first conditional or operator is false so the right hand operand is evaluated. No further evaluation of the expression is necessary so the right hand operand of the second conditional or operator is not evaluated.

21 b Prints: 2, 2, -3, -4, 12, The original expression is as follows: $m(m(++i) - m(i++) + m(-i) * m(~i))$. The method, m, prints and then returns the value of the parameter, so the original expression is equivalent to the following: $++i - i++ + -i * ~i$. We can use a simplification process that evaluates the expression as follows. Step one. Work through the expression from left to right to evaluate the unary expressions: $2 - 2 + -3 * -4$. Step two. Add the parentheses to indicate operator precedence: $2 - 2 + (-3 * -4)$. Step three. Evaluate the innermost parentheses: $2 - 2 + 12$. Step four. Evaluate the simplified expression to produce the result, 12. The bitwise complement operator, \sim , inverts each bit of the operand. To avoid working in binary the same result can be obtained by changing the sign of the operand and then subtracting 1. The following identity is always true $\sim x == -x - 1$.

22 b Prints: 1,0 The expression, $i = i++ + m(i)$, can be reduced to, $i = 0 + 0$. The left operand of the addition expression is found to be zero, and the value of variable i is then incremented to 1. The right hand operand of the addition operation is evaluated next. The value, 1, is passed to method m. After printing the argument value, method m returns the value zero. The two operands of the addition operation are zero as is the result of the addition. The zero value serves as the right hand operand of the assignment statement, so the value, zero, is assigned to variable i.

23 c Prints: 5 The two statements, $int a=1$ followed by $a += ++a + a++$, can be rewritten as the single statement, $a=(int)((1)+(++a + a++))$. Further evaluation produces $a=(int)((1)+(2 + 2))$. Generally speaking, a

compound assignment expression of the form E1 op= E2 can be rewritten as E1=(T)((E1)op(E2)) where T is the type of E1.

24 d Prints: 1,2,3,4,5,3 The expression can be simplified as follows: $j = 1 + ((2 * 3) \% 4) + 5 = 8$. The original expression is as follows: $j = ++i + ++i * ++i \% ++i + ++i$. Step one. Evaluate the unary expressions from left to right: $j = 1 + 2 * 3 \% 4 + 5$. Step two. Add parentheses to indicate operator precedence: $j = 1 + ((2 * 3) \% 4) + 5$. Step three. Evaluate the inner most parentheses: $j = 1 + (6 \% 4) + 5$. Repeat step three: $j = 1 + 2 + 5$. Repeat step three: $j = 8$. The argument of the print expression is: $j \% 5$. The result is: $8 \% 5 = 3$.

25 c Prints: 3 The expression can be simplified as follows: $j = 1 + ((2 * 3) \% 4) + 5 = 8$. The original expression is as follows: $j = ++i + ((++i * ++i) \% ++i) + ++i$. Step one. Evaluate the unary expressions from left to right: $j = 1 + ((2 * 3) \% 4) + 5$. Step two. Evaluate the inner-most parentheses: $j = 1 + (6 \% 4) + 5$. Step three: Evaluate the inner-most parentheses. $j = 1 + 2 + 5$. Step four: Work through the expression from left to right. $j = 8$. The argument of the print expression is: $j \% 5$. The result is: $8 \% 5 = 3$.

26 b Prints: FFT The expression used to assign variable b1 is equivalent to the expression used to assign variable b2. The results demonstrate that the conditional operator (?:) groups from right-to-left.

27 c Prints: Blue,Blue The expression used to assign variable b1 is equivalent to the expression used to assign variable b2. The results demonstrate that the conditional operator (?:) groups from right-to-left.

28 c Prints: -2, 3, 0, 2, 2, 5, The expression can be simplified as follows: $-2+3+0+2+2=5$. The original expression is as follows: $m(m(\sim 1) + m(1|2) + m(1&2) + m(1^3) + m(1<<1))$. Expr 1: $\sim 1 = -1 - 1 = -2$. Expr 2: $1|2 = 0001 | 0010 = 0011 = 3$. Expr 3: $1&2 = 0001 \& 0010 = 0000$. Expr 4: $1^3 = 0001 ^ 0011 = 0010 = 2$. Expr 5: $1<<1 = 0001 << 1 = 0010 = 2$. Note: The bitwise expressions were demonstrated using only four bits of the 32 bit int type values.

29 g None of the above Prints 31,0. The expression $(-1 \& 0x1f)$ is equal to $(0xffffffff \& 0x1f)$, and both are equal to the hex value 0x1f or decimal 31. The expression $(8 << -1)$ is equivalent to $(8 << 0xffffffff)$. If the left hand operand of a shift expression is of type int, then the right hand operand is implicitly masked with the value 0x1f. In other words, the expression $(8 << -1)$ is equivalent to $(8 << (-1 \& 0x1f))$. By replacing -1 with the hexadecimal representation we have $(8 << (0xffffffff \& 0x1f))$. By evaluating the right hand operand we have $(8 << 31)$. When 8 is shifted 31 bits to the left, the result is zero since the only non-zero bit is lost as it is shifted beyond the most significant bit of the int data type.

Arguments

Question 1

```
class GFC401 {
    static int m1(int x) {return ++x;}
    public static void main (String[] args) {
        int x = 1;
        int y = m1(x);
```

```
        System.out.println(x + "," + y);
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1
- b. Prints: 1,2
- c. Prints: 2,1
- d. Prints: 2,2
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 2

```
class GRC10 {
    public static void main (String[] s) {
        System.out.print(s[1] + s[2] + s[3]);
    }
}
java GRC10 A B C D E F
```

What is the result of attempting to compile and run the program using the specified command line?

- a. Prints: ABC
- b. Prints: BCD
- c. Prints: CDE
- d. Prints: A B C
- e. Prints: B C D
- f. Prints: C D E
- g. Compile-time error
- h. Run-time error
- i. None of the above

Question 3

```
class GFC402 {
    static int x=1;
    void m1(int i) {x++; i++;}
    public static void main (String[] args) {
        int y=3; m1(y);
        System.out.println(x + "," + y);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,3
- b. Prints: 2,3
- c. Prints: 1,4
- d. Prints: 2,4
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 4

```
class GFC403 {
    private static int x=1;
    static void m1(int i) {x++; i++;}
    public static void main (String[] args) {
```

```
    int y=3; m1(y);
    System.out.println(x + "," + y);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,3
- b. Prints: 2,3
- c. Prints: 1,4
- d. Prints: 2,4
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 5

```
class GFC404 {
    private static int x=1;
    static void m1(int x, int y) {x++; y++;}
    public static void main (String[] args) {
        int y=3; m1(x, y);
        System.out.println(x + "," + y);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,3
- b. Prints: 2,3
- c. Prints: 1,4
- d. Prints: 2,4
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 6

```
class GFC301 {
    private String name;
    public GFC301(String name) {this.name = name;}
    public void setName(String name) {this.name = name;}
    public String getName() {return name;}
    public static void m1(GFC301 r1, GFC301 r2) {
        r1.setName("Bird");
        r2 = r1;
    }
    public static void main (String[] args) {
        GFC301 pet1 = new GFC301("Dog");
        GFC301 pet2 = new GFC301("Cat");
        m1(pet1,pet2);
        System.out.println(pet1.getName() + "," + pet2.getName());
    }
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: Dog,Cat
- b. Prints: Dog,Bird
- c. Prints: Bird,Cat
- d. Prints: Bird,Bird
- e. Run-time error

- f. Compile-time error
 - g. None of the above
- Question 7

```
class GFC303 {  
    private String name;  
    public GFC303(String name) {this.name = name;}  
    public void setName(String name) {this.name = name;}  
    public String getName() {return name;}  
    public static void m1(GFC303 pet1, GFC303 pet2) {  
        pet1 = new GFC303("Fish");  
        pet2 = null;  
    }  
    public static void main (String[] args) {  
        GFC303 pet1 = new GFC303("Dog");  
        GFC303 pet2 = new GFC303("Cat");  
        m1(pet1,pet2);  
        System.out.println(pet1.getName() + "," + pet2.getName());  
    }  
}
```

- What is the result of attempting to compile and run the program?
- a. Prints: Dog,Cat
 - b. Prints: Dog,Fish
 - c. Prints: Fish,Cat
 - d. Prints: Fish,Fish
 - e. Compile-time error
 - f. Run-time error
 - g. None of the above
- Question 8

```
class GFC304 {  
    static void m1(int[] i1, int[] i2) {  
        int[] i3 = i1; i1 = i2; i2 = i3;  
    }  
    public static void main (String[] args) {  
        int[] i1 = {1}, i2 = {3}; m1(i1, i2);  
        System.out.print(i1[0] + "," + i2[0]);  
    }  
}
```

- What is the result of attempting to compile and run the program?
- a. Prints: 1,1
 - b. Prints: 1,3
 - c. Prints: 3,1
 - d. Prints: 3,3
 - e. Run-time error
 - f. Compile-time error
 - g. None of the above
- Question 9

```
class GFC305 {  
    static void m1(int[] i1, int[] i2) {  
        int i = i1[0]; i1[0] = i2[0]; i2[0] = i;  
    }  
    public static void main (String[] args) {
```

```
    int[] i1 = {1}, i2 = {3}; m1(i1, i2);
    System.out.print(i1[0] + "," + i2[0]);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1
- b. Prints: 1,3
- c. Prints: 3,1
- d. Prints: 3,3
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 10

```
class GFC306 {
    static int[] i1 = {1}, i2 = {3};
    static void m1(int[] i1) {
        int[] i3 = i1; i1 = i2; i2 = i3;
    }
    public static void main (String[] args) {
        m1(i1);
        System.out.print(i1[0] + "," + i2[0]);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1
- b. Prints: 1,3
- c. Prints: 3,1
- d. Prints: 3,3
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 11

```
class GFC307 {
    static void m1(int[] i1, int[] i2) {
        i1 = i2 = null;
    }
    public static void main (String[] args) {
        int[] i1 = {1}, i2 = {3}; m1(i1, i2);
        System.out.print(i1[0] + "," + i2[0]);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0
- b. Prints: 1,1
- c. Prints: 1,3
- d. Prints: 3,1
- e. Prints: null,null
- f. Run-time error
- g. Compile-time error
- h. None of the above

Question 12

```

class GFC308 {
    int[] i1 = {1}, i2 = {3};
    void m1() {
        m2(i1, i2);
        System.out.print(i1[0] + "," + i2[0]);
    }
    void m2(int[] i1, int[] i2) {
        int[] i3 = i1;
        this.i1 = i2;
        this.i2 = i3;
    }
    public static void main (String[] args) {
        new GFC308().m1();
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0
- b. Prints: 1,1
- c. Prints: 1,3
- d. Prints: 3,1
- e. Prints: null,null
- f. Run-time error
- g. Compile-time error
- h. None of the above

No. Answer Remark

1 b Prints: 1,2 Primitive arguments are passed by value. The method m1 increments the parameter x, and the result is returned. The local variable x of main remains unchanged.

2 b Prints: BCD The index for the first element of an array is zero so the first argument printed by this program is the second argument on the command line following the name of the class.

3 f Compile-time error Method m1 is an instance method, and must be invoked with reference to an instance of type GFC402. Method m1 can not be invoked from a static context.

4 b Prints: 2,3 Variables of primitive type are passed to methods by value: Only a copy of the value of the variable is passed to the method. While the method works with a local copy of the variable, the original variable remains unchanged by any actions performed on the method parameter. For that reason, method m1 does not change the value of the variable y in the main method. However, method m1 does have direct access to the class variable x and the content of the class variable is modified by method m1.

5 a Prints: 1,3 Variables of primitive type are passed to methods by value: Only a copy of the value of the variable is passed to the method. While the method works with a local copy of the variable, the original variable remains unchanged by any actions performed on the method parameter. For that reason, method m1 does not change the contents of the variable y in the main method or the class variable x.

6 c Prints: Bird,Cat The method m1 is invoked by the method invocation expression m1(pet1,pet2). The value of the reference variable denoted by the argument pet1 is used to initialize the method parameter r1. Inside of method m1, the method invocation expression r1.setName("Bird") uses the copy of the value of the argument pet1 to

assign a new name to the instance of GFC301 that is referenced by the local variable pet1 in the main method. Generally speaking, a reference parameter can be used to invoke methods on the referenced object and change the state of the object to the extent provided by the object's methods. The method invocation expression m1(pet1,pet2) has a second argument pet2, and the value of pet2 is used to initialize the method parameter r2. Inside of method m1, the assignment expression r2 = r1 changes the value of the method parameter r2; but the local variable of the main method denoted by the argument pet2 appearing in the method invocation expression m1(pet1,pet2) remains unchanged.

7 a Prints: Dog,Cat The method m1 is invoked by the method invocation expression m1(pet1,pet2). A copy of the reference argument pet1 is assigned to the method parameter pet1. Inside the body of method m1, the assignment expression pet1 = new GFC303("Fish") assigns a reference to a new instance of GFC303 to the method parameter pet1; but the argument pet1 that appears in the method invocation expression m1(pet1,pet2) and the local variable pet1 that is declared in the main method remain unchanged. The method invocation expression m1(pet1,pet2) has a second argument pet2, and a copy of pet2 is assigned to the method parameter pet2. Inside of method m1, the assignment expression pet2 = null changes the value of the method parameter pet2; but the argument pet2 appearing in the method invocation expression remains unchanged in the main method.

8 b Prints: 1,3 The method m1 is invoked by the method invocation expression m1(i1, i2). The argument i1 denotes a local variable of type int[] that is declared in the main method. The value of the argument is a reference to the array, and the argument value is used to initialize the method parameter i1 of method m1. Inside the body of m1, the expression i1 = i2 sets the value of parameter i1 to the value of parameter i2, but the change in the value of the parameter i1 does not change the original argument value or the local variable i1 of the main method that the argument denotes. Similarly, the assignment expression i2 = i3 in method m1 does not change the value of the local variable i1 declared in the main method.

9 c Prints: 3,1 Method m1 is not able to change the value of the local variables that are declared in the main method and serve as the arguments in the method invocation expression. However, method m1 is able to modify the contents of the arrays that are referenced by the method parameters.

10 a Prints: 1,1 The method m1 is invoked by the method invocation expression m1(i1). The argument i1 denotes the static member variable i1. Inside the declaration of method m1, the method parameter i1 shadows the static member variable i1. The assignment expression i1 = i2 assigns the value of the member variable i2 to the method parameter i1, but the member variable i1 remains unchanged. Inside of method m1, the member variable i2 is not shadowed; so the assignment expression i2 = i3 assigns the reference value contained by the method local variable i3 to the member variable i2. This question demonstrates that argument values are passed to method parameters by value, and the method parameter is only a copy of the argument value. A change made to the method parameter does not change the value of any variable that is shadowed by the parameter and does not change the value of the argument appearing in the method invocation expression.

11 c Prints: 1,3 Although the reference parameters i1 and i2 are reassigned inside of m1, the change has no impact outside of m1. Array references are passed by value: the invoked method gets a copy of the array reference.

12 d Prints: 3,1 Inside of method m2, the local variables i1 and i2 remain unchanged while the shadowed instance variables are changed.

Exception Handling

Question 1

```
class A {  
    public static void main (String[] args) {  
        Error error = new Error();  
        Exception exception = new Exception();  
        System.out.print((exception instanceof Throwable) + "," );  
        System.out.print(error instanceof Throwable);  
    } } 
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 2

```
class A {A() throws Exception {}} // 1  
class B extends A {B() throws Exception {}} // 2  
class C extends A {C() {}} // 3 
```

Which of the following statements are true?

- a. class A extends Object.
- b. Compile-time error at 1.
- c. Compile-time error at 2.
- d. Compile-time error at 3.

Question 3

```
class A {  
    public static void main (String[] args) {  
        Object error = new Error();  
        Object runtimeException = new RuntimeException();  
        System.out.print((error instanceof Exception) + "," );  
        System.out.print(runtimeException instanceof Exception);  
    } } 
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compile-time error

- f. Run-time error
 - g. None of the above
- Question 4

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 1;
        try {
            try {
                switch (x) {
                    case 1: throw new Level1Exception();
                    case 2: throw new Level2Exception();
                    case 3: throw new Level3Exception();
                } a++;
            }
            catch (Level2Exception e) {b++;}
            finally {c++;}
        }
        catch (Level1Exception e) { d++;}
        catch (Exception e) {f++;}
        finally {g++;}
        System.out.print(a+" "+b+" "+c+" "+d+" "+f+" "+g);
    }
}
```

- What is the result of attempting to compile and run the program?
- a. Prints: 0,0,0,1,0,0
 - b. Prints: 0,0,1,1,0,1
 - c. Prints: 0,1,1,1,0,1
 - d. Prints: 1,0,1,1,0,1
 - e. Prints: 1,1,1,1,0,1
 - f. Compile-time error
 - g. Run-time error
 - h. None of the above

Question 5

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 2;
        try {
            try {
                switch (x) {
                    case 1: throw new Level1Exception();
                    case 2: throw new Level2Exception();
                    case 3: throw new Level3Exception();
                } a++;
            }
        }
```

```

        catch (Level2Exception e) {b++;}
        finally {c++;}
    }
    catch (Level1Exception e) { d++;}
    catch (Exception e) {f++;}
    finally {g++;}
    System.out.print(a+" , "+b+" , "+c+" , "+d+" , "+f+" , "+g);
}
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,1,0,0,1
- b. Prints: 0,1,0,0,0,0
- c. Prints: 0,1,1,0,0,1
- d. Prints: 0,1,0,0,0,1
- e. Prints: 1,1,1,0,0,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 6

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 3;
        try {
            try {
                switch (x) {
                    case 1: throw new Level1Exception();
                    case 2: throw new Level2Exception();
                    case 3: throw new Level3Exception();
                } a++; }
            catch (Level2Exception e) {b++;}
            finally {c++;}
        }
        catch (Level1Exception e) { d++;}
        catch (Exception e) {f++;}
        finally {g++;}
        System.out.print(a+" , "+b+" , "+c+" , "+d+" , "+f+" , "+g);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1,1,0,0,1
- b. Prints: 0,1,1,0,0,1
- c. Prints: 0,1,0,0,0,0
- d. Prints: 0,1,0,0,0,1
- e. Prints: 0,0,1,0,0,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 7

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 4;
        try {
            try {
                switch (x) {
                    case 1: throw new Level1Exception();
                    case 2: throw new Level2Exception();
                    case 3: throw new Level3Exception();
                    case 4: throw new Exception();
                } a++;
            }
            catch (Level2Exception e) {b++;}
            finally{c++;}
        }
        catch (Level1Exception e) { d++;}
        catch (Exception e) {f++;}
        finally {g++;}
        System.out.print(a+" , "+b+" , "+c+" , "+d+" , "+f+" , "+g);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,0,0,1
- b. Prints: 0,0,0,0,1,0
- c. Prints: 0,0,1,0,0,1
- d. Prints: 0,0,1,0,1,1
- e. Prints: 0,1,1,1,1,1
- f. Prints: 1,1,1,1,1,1
- g. Compile-time error
- h. Run-time error
- i. None of the above

Question 8

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
    public static void main(String args[]) {
        int a,b,c,d,f,g,x;
        a = b = c = d = f = g = 0;
        x = 5;
        try {
            try {
                switch (x) {
                    case 1: throw new Level1Exception();
                    case 2: throw new Level2Exception();
                    case 3: throw new Level3Exception();
                    case 4: throw new Exception();
                } a++;
            }

```

```

        catch (Level2Exception e) {b++;}
        finally {c++;}
    }
    catch (Level1Exception e) { d++;}
    catch (Exception e) {f++;}
    finally {g++;}
    System.out.print(a+", "+b+", "+c+", "+d+", "+f+", "+g);
}}

```

What is the result of attempting to compile and run the program?

- a. Prints: 1,0,0,0,0,0
- b. Prints: 1,0,1,0,0,1
- c. Prints: 0,0,1,0,0,1
- d. Prints: 1,1,1,1,1,1
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 9

```

class ColorException extends Exception {}
class WhiteException extends ColorException {}
class White {
    void m1() throws ColorException {throw new WhiteException();}
    void m2() throws WhiteException {}
    public static void main (String[] args) {
        White white = new White();
        int a,b,d,f; a = b = d = f = 0;
        try {white.m1(); a++;} catch (ColorException e) {b++;}
        try {white.m2(); d++;} catch (WhiteException e) {f++;}
        System.out.print(a+", "+b+", "+d+", "+f+");
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,0
- c. Prints: 0,1,1,0
- d. Prints: 1,1,1,0
- e. Prints: 1,1,1,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 10

```

class ColorException extends Exception {}
class WhiteException extends ColorException {}
class White {
    void m1() throws ColorException {throw new ColorException();}
    void m2() throws WhiteException {throw new ColorException();}
    public static void main (String[] args) {
        White white = new White();
        int a,b,d,f; a = b = d = f = 0;
        try {white.m1(); a++;} catch (ColorException e) {b++;}
        try {white.m2(); d++;} catch (WhiteException e) {f++;}
        System.out.print(a+", "+b+", "+d+", "+f+");
    }
}

```

}}

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,1
- c. Prints: 0,1,0,1
- d. Prints: 0,1,1,1
- e. Prints: 1,1,1,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 11

```
class ColorException extends Exception {}  
class WhiteException extends ColorException {}  
class White {  
    void m1() throws ColorException {throw new ColorException();}  
    void m2() throws WhiteException {throw new WhiteException();}  
    public static void main (String[] args) {  
        White white = new White();  
        int a,b,d,f; a = b = d = f = 0;  
        try {white.m1(); a++;} catch (WhiteException e) {b++;}  
        try {white.m2(); d++;} catch (WhiteException e) {f++;}  
        System.out.print(a+","+b+","+d+","+f);  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,1
- c. Prints: 0,1,0,1
- d. Prints: 0,1,1,1
- e. Prints: 1,1,1,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 12

```
class Level1Exception extends Exception {}  
class Level2Exception extends Level1Exception {}  
class Level3Exception extends Level2Exception {}  
class Brown {  
    public static void main(String args[]) {  
        int a, b, c, d, f; a = b = c = d = f = 0;  
        int x = 1;  
        try {  
            switch (x) {  
                case 1: throw new Level1Exception();  
                case 2: throw new Level2Exception();  
                case 3: throw new Level3Exception();  
            } a++; }  
        catch (Level3Exception e) {b++;}  
        catch (Level2Exception e) {c++;}  
        catch (Level1Exception e) {d++;}  
        finally {f++;}
```

```
        System.out.print(a+" , "+b+" , "+c+" , "+d+" , "+f);
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,1
- b. Prints: 0,0,1,1,1
- c. Prints: 0,1,1,1,1
- d. Prints: 1,1,1,1,1
- e. Prints: 0,0,1,0,1
- f. Prints: 0,1,0,0,1
- g. Prints: 1,0,0,0,1
- h. Compile-time error
- i. Run-time error
- j. None of the above

Question 13

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Brown {
    public static void main(String args[]) {
        int a, b, c, d, f; a = b = c = d = f = 0;
        int x = 2;
        try {
            switch (x) {
                case 1: throw new Level1Exception();
                case 2: throw new Level2Exception();
                case 3: throw new Level3Exception();
            } a++;
        } catch (Level3Exception e) {b++;}
        catch (Level2Exception e) {c++;}
        catch (Level1Exception e) {d++;}
        finally {f++;}
        System.out.print(a+" , "+b+" , "+c+" , "+d+" , "+f);
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,1
- b. Prints: 0,0,1,1,1
- c. Prints: 0,1,1,1,1
- d. Prints: 1,1,1,1,1
- e. Prints: 0,0,1,0,1
- f. Prints: 0,1,0,0,1
- g. Prints: 1,0,0,0,1
- h. Compile-time error
- i. Run-time error
- j. None of the above

Question 14

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Brown {
    public static void main(String args[]) {
```

```

int a, b, c, d, f; a = b = c = d = f = 0;
int x = 4;
try {
    switch (x) {
        case 1: throw new Level1Exception();
        case 2: throw new Level2Exception();
        case 3: throw new Level3Exception();
    } a++;
}
catch (Level3Exception e) {b++;}
catch (Level2Exception e) {c++;}
catch (Level1Exception e) {d++;}
finally {f++;}
System.out.print(a+, "+b+, "+c+, "+d+, "+f);
}
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,1
- b. Prints: 0,0,1,1,1
- c. Prints: 0,1,1,1,1
- d. Prints: 1,1,1,1,1
- e. Prints: 0,0,1,0,1
- f. Prints: 0,1,0,0,1
- g. Prints: 1,0,0,0,1
- h. Compile-time error
- i. Run-time error
- j. None of the above

Question 15

```

class ColorException extends Exception {}
class WhiteException extends ColorException {}
abstract class Color {
    abstract void m1() throws ColorException;
}
class White extends Color {
    void m1() throws WhiteException {throw new WhiteException();}
    public static void main (String[] args) {
        White white = new White();
        int a,b,c; a = b = c = 0;
        try {white.m1(); a++;}
            catch (WhiteException e) {b++;}
            finally {c++;}
        System.out.print(a+, "+b+, "+c);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0
- b. Prints: 0,0,1
- c. Prints: 0,1,0
- d. Prints: 0,1,1
- e. Prints: 1,0,0
- f. Prints: 1,0,1
- g. Prints: 1,1,0
- h. Prints: 1,1,1
- i. Compile-time error

- j. Run-time error
 - k. None of the above
- Question 16

```
class RedException extends Exception {}  
class BlueException extends Exception {}  
class White {  
    void m1() throws RedException {throw new RedException();}  
    public static void main (String[] args) {  
        White white = new White();  
        int a,b,c,d; a = b = c = d = 0;  
        try {white.m1(); a++;}  
            catch (RedException e) {b++;}  
            catch (BlueException e) {c++;}  
            finally {d++;}  
        System.out.print(a+"," +b+"," +c+"," +d);  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,1
- c. Prints: 0,1,0,1
- d. Prints: 0,1,1,1
- e. Prints: 1,1,1,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 17

```
class Level1Exception extends Exception {}  
class Level2Exception extends Level1Exception {}  
class Level3Exception extends Level2Exception {}  
class Purple {  
    public static void main(String args[]) {  
        int a,b,c,d,f,g,x;  
        a = b = c = d = f = g = 0;  
        x = 1;  
        try {  
            throw new Level1Exception();  
            try {  
                switch (x) {  
                    case 1: throw new Level1Exception();  
                    case 2: throw new Level2Exception();  
                    case 3: throw new Level3Exception();  
                } a++; }  
                catch (Level2Exception e) {b++;}  
                finally {c++;}  
            }  
            catch (Level1Exception e) { d++;}  
            catch (Exception e) {f++;}  
            finally {g++;}  
        System.out.print(a+"," +b+"," +c+"," +d+"," +f+"," +g);  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1,1,0,0,1
- b. Prints: 0,1,1,0,0,1
- c. Prints: 0,1,0,0,0,0
- d. Prints: 0,1,0,0,0,1
- e. Prints: 0,0,1,0,0,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

No. Answer Remark

1 d Prints: true,true Both Error and Exception are subclasses of Throwable.

2 a d class A extends Object. Compile-time error at 3. The constructors for class B and class C both invoke the constructor for A. The constructor for class A declares Exception in the throws clause. Since the constructors for B and C invoke the constructor for A, it is necessary to declare Exception in the throws clauses of B and C. A compile-time error is generated at marker 3, because the constructor does not declare Exception in the throws clause.

3 b Prints: false,true Error is a direct subclass of Throwable. RuntimeException is a direct subclass of Exception.

4 b Prints: 0,0,1,1,0,1 The nested catch clause is able to catch a Level2Exception or any subclass of it. The switch statement throws a Level1Exception that can not be caught by the nested catch clause; so the nested finally block is executed as control passes to the first of the two outer catch clauses. The outer finally block is executed as control passes out of the try statement.

5 c Prints: 0,1,1,0,0,1 The nested catch block is able to catch a Level2Exception or any subclass of it causing b to be incremented. Both of the finally blocks are then executed.

6 b Prints: 0,1,1,0,0,1 The nested catch block is able to catch a Level2Exception or any subclass of it causing b to be incremented. Both of the finally blocks are then executed.

7 d Prints: 0,0,1,0,1,1 The nested catch clause is able to catch a Level2Exception or any subclass of it. The switch statement throws an Exception that can not be caught by the nested catch clause; so the nested finally block is executed as control passes to the second of the two outer catch clauses. The outer finally block is executed as control passes out of the try statement.

8 b Prints: 1,0,1,0,0,1 The switch statement does not throw an exception; so the switch completes normally. The subsequent statement increments the variable, a; and the try block completes normally. Both of the finally blocks are then executed.

9 c Prints: 0,1,1,0 The first try block contains two statements. The first invokes method m1, and the subsequent statement contains a post increment expression with the variable, a, as the operand. Method m1 throws a WhiteException exception, so variable a is not incremented as control passes to the catch block where b is incremented. The throws clause of m1 declares a ColorException, so the body may throw a ColorException or any subclass of ColorException. The second try block also contains two statements. The first invokes method m2, and the subsequent statement contains a post increment expression with the

variable, d, as the operand. Method m2 does not throw an exception, so d is incremented, and the try block completes normally. Although the throws clause of m2 declares a WhiteException, there is no requirement to throw any exception.

10 f Compile-time error The throws clause of White.m2 declares a WhiteException, so the body of m2 may throw a WhiteException or any subclass of WhiteException. Instead, the body of m2 throws a superclass of WhiteException. The result is a compile-time error.

11 f Compile-time error The throws clause of White.m1 declares a ColorException, but the catch clause in the main method catches only a subclass of ColorException. The result is a compile-time error.

12 a Prints: 0,0,0,1,1 The first catch clause has a parameter e of type Level3Exception, so the first catch clause is able to catch any exception type that is assignable to type Level3Exception. Since Level2Exception is the superclass of Level3Exception, an instance of Level2Exception is not assignable to a catch clause parameter of type Level3Exception. Similarly, Level1Exception is also a superclass of Level3Exception, so an instance of Level1Exception is not assignable to a catch clause parameter of type Level3Exception. The only exception type that can be caught by the first catch clause is a Level3Exception. The second catch clause has a parameter e of type Level2Exception, so the second catch clause is able to catch a Level2Exception. The Level1Exception is the superclass of Level2Exception. An instance of Level1Exception is not assignable to a catch clause parameter of type Level2Exception, so the second catch clause can not catch a Level1Exception. Since a Level3Exception is a subclass of Level2Exception an exception of type Level3Exception is assignable to a catch clause parameter type Level2Exception. All exceptions of type Level3Exception will be caught by the first catch clause, so the second catch clause in this program will not have an opportunity to catch a Level3Exception. The third catch clause has a parameter e of type Level1Exception, so the third catch clause is able to catch a Level1Exception. The exceptions of type Level2Exception and Level3Exception are assignable to the catch clause parameter of the third catch clause, but the exceptions of those subclass types will be caught by the first two catch clauses. The switch statement throws a Level1Exception. The try block completes abruptly as control passes to the third catch block where d is incremented. The finally block is also executed, so f is incremented.

13 e Prints: 0,0,1,0,1 The first catch block is able to catch a Level3Exception or any subclass of Level3Exception. The second catch block is able to catch a Level2Exception or any subclass of Level2Exception. The switch statement throws a Level2Exception. The try block completes abruptly as control passes to the second catch block where c is incremented. The finally block is also executed, so f is incremented.

14 g Prints: 1,0,0,0,1 The switch statement does not throw an exception; so the switch completes normally. The subsequent statement increments the variable, a; and the try block completes normally. The finally block is also executed, so f is incremented.

15 d Prints: 0,1,1 The try block contains two statements. The first invokes method m1, and the subsequent statement contains a post increment expression with the variable, a, as the operand. Method m1 throws a WhiteException exception, so variable a is not incremented as control passes to the catch block where b is incremented. Although

Color.m1 declares a ColorException in the throws clause, a subclass of Color is free to declare only a subclass of ColorException in the throws clause of the overriding method.

16 f Compile-time error A compile-time error is generated, because the second catch clause attempts to catch an exception that is never thrown in the try block.

17 f Compile-time error A throw statement is the first statement in the outer try block. A throw statement appearing in a try block causes control to pass out of the block. Consequently, statements can not be reached if they appear in the block after the throw statement. The switch statement that appears after the throw statement is unreachable and results in a compile-time error.

Interfaces

Question 1

```
interface A {  
    void m1();           // 1  
    public void m2();    // 2  
    protected void m3(); // 3  
    private void m4();   // 4  
}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4

Question 2

Which of the following statements is not true?

- a. An interface that is declared within the body of a class or interface is known as a nested interface.
- b. A constant can be a member of an interface.
- c. A class declaration can be a member of an interface.
- d. If an interface is named in the implements clause of a class, then the class must implement all of the methods declared within the interface.
- e. None of the above.

Question 3

Which of the following statements are true?

- a. An interface declaration can be a member of an interface.
- b. A method declared within an interface must have a body represented by empty curly braces.
- c. An interface can implement another interface.
- d. An abstract class that implements an interface must implement all abstract methods declared within the interface.
- e. An abstract method declaration can be a member of an interface.

Question 4

Which of the following are modifiers that can be applied to an interface that is a member of a directly enclosing interface?

- a. abstract
- b. implements
- c. final
- d. private
- e. protected
- f. public

Question 5

Which of the following are modifiers that can be applied to an interface that is a member of a directly enclosing class?

- a. abstract
- b. extends
- c. final
- d. private
- e. protected
- f. public

Question 6

Which of the following is a modifier that can be applied to an interface that is a member of a directly enclosing class or interface?

- a. static
- b. synchronized
- c. transient
- d. volatile
- e. implements
- f. None of the above.

Question 7

Suppose that an interface, I1, is not a member of an enclosing class or interface. Which of the following modifiers can be applied to interface I1?

- a. abstract
- b. final
- c. private
- d. protected
- e. public

Question 8

Suppose that an interface, I1, is not a member of an enclosing class or interface. Which of the following modifiers can be applied to interface I1?

- a. abstract
- b. public
- c. static
- d. synchronized
- e. transient
- f. volatile

Question 9

Which of the following are modifiers that can be applied to a field declaration within an interface?

- a. abstract
- b. const
- c. final

- d. private
 - e. protected
 - f. public
- Question 10

Which of the following is a modifier that can be applied to a field declaration within an interface?

- a. static
- b. synchronized
- c. transient
- d. volatile
- e. None of the above.

Question 11

Which of the following modifiers can be applied to a class that is declared within an enclosing interface?

- a. public
- b. protected
- c. private
- d. abstract
- e. static
- f. final

Question 12

```
interface A {  
    int a = 1;                      // 1  
    public int b = 2;                 // 2  
    public static int c = 3;          // 3  
    public static final int d = 4;     // 4  
}
```

Which field declaration results in a compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

Question 13

```
interface A {  
    protected int e = 5;              // 1  
    private int f = 6;                // 2  
    volatile int g = 7;              // 3  
    transient int h = 8;             // 4  
    public static final int d = 9;     // 5  
}
```

Which of the field declarations does not result in a compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5
- f. None of the above

Question 14

Which of the following are modifiers that can be applied to a method declaration within an interface?

- a. abstract
- b. final
- c. private
- d. protected
- e. public

Question 15

Which of the following is a modifier that can be applied to a method declaration within an interface?

- a. static
- b. synchronized
- c. transient
- d. volatile
- e. native
- f. None of the above

Question 16

```
interface A {void m1();} // 1
class B implements A {public void m1() {} } // 2
class C implements A {protected void m1() {} } // 3
class D implements A {private void m1() {} } // 4
class E implements A {void m1() {} } // 5
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5

Question 17

```
interface A {
    void m1(); // 1
    public void m2(); // 2
    protected void m3(); // 3
    private void m4(); // 4
    abstract void m5(); // 5
}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5

Question 18

```
interface A {
    final void m1(); // 1
    synchronized void m2(); // 2
```

```
    native void m3();           // 3
    abstract void m4();         // 4
    public void m5();           // 5
}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5

Question 19

```
interface A {void main(String[] args);}           // 1
interface B {public void main(String[] args);}      // 2
interface C {public static void main(String[] args);} // 3
interface D {protected void main(String[] args);}     // 4
interface E {private void main(String[] args);}       // 5
```

Which interface declarations generate a Compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5

Question 20

```
interface F {abstract void main(String[] args);}      // 1
interface G {synchronized void main(String[] args);}    // 2
interface H {final void main(String[] args);}          // 3
interface I {native void main(String[] args);}         // 4
```

Which interface declaration does not generate a compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

Question 21

```
interface A {String s1 = "A"; String m1();}
interface B implements A {String s1 = "B"; String m1();}
class C implements B {
    public String m1() {return s1;}
    public static void main(String[] args) {
        A a = new C(); System.out.print(a.m1());
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Compile-time error
- d. Run-time error
- e. None of the above

Question 22

```
interface A {int i = 1; int m1();}  
interface B extends A {int i = 10; int m1();}  
class C implements B {  
    public int m1() {return ++i;}  
    public static void main(String[] args) {  
        System.out.print(new C().m1());  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 2
- b. Prints: 11
- c. Compile-time error
- d. Run-time error
- e. None of the above

Question 23

```
interface Z {void m1();} // 1  
class A implements Z {void m1() {}} // 2  
class B implements Z {public void m1() {}} // 3  
abstract class C implements Z {public abstract void m1();} // 4
```

A Compile-time error is generated at which line?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

Question 24

```
interface Z {void m1();} // 1  
class D implements Z {public final void m1() {}} // 2  
class E implements Z {public synchronized void m1() {}} // 3  
class G implements Z {public native void m1();} // 4
```

A Compile-time error is generated at which line?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

Question 25

```
interface I10 {String name = "I10"; String s10 = "I10.s10";}  
interface I20 {String name = "I20"; String s20 = "I20.s20";}  
class C10 implements I10, I20 { // 1  
    public static void main(String[] args) {  
        System.out.print(s10+","); // 2  
        System.out.print(s20+","); // 3  
        System.out.print(name); // 4  
    }  
}
```

What is the result of attempting to compile and run the program?

- a. Prints: I10.s10,I20.s20,I10
- b. Prints: I10.s10,I20.s20,I20
- c. Prints: I10.s10,I20.s20,
- d. Prints: I10.s10,I20.s20,null
- e. Compile-time error at line 1
- f. Compile-time error at line 2
- g. Compile-time error at line 3
- h. Compile-time error at line 4
- i. Run-time error
- j. None of the above

Question 26

```
interface I10 {String name = "I10"; String s10 = "I10.s10";}
interface I20 {String name = "I20"; String s20 = "I20.s20";}
class C20 implements I10, I20 {                      // 1
    public static void main(String[] args) {
        System.out.print(I10.s10+",");           // 2
        System.out.print(I20.s20+",");           // 3
        System.out.print(I20.name);              // 4
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: I10.s10,I20.s20,I10
- b. Prints: I10.s10,I20.s20,I20
- c. Prints: I10.s10,I20.s20,
- d. Prints: I10.s10,I20.s20,null
- e. Compile-time error at line 1
- f. Compile-time error at line 2
- g. Compile-time error at line 3
- h. Compile-time error at line 4
- i. Run-time error
- j. None of the above

No. Answer Remark

1 c d 3 4 Methods declared within an interface are implicitly public. If no access modifier is included in the method declaration; then, the declaration is implicitly public. An attempt to declare the method using a weaker access privilege, private or protected, results in a compile-time error.

2 d If an interface is named in the implements clause of a class, then the class must implement all of the methods declared within the interface. This question asks which answer option is not true. Some true statements are as follows. An interface can be declared within an enclosing class or interface. The members of an interface can be constants, abstract method declarations, class declarations or interface declarations. If an interface is named in the implements clause of a class, then the class must implement all of the methods declared within the interface or the class must be declared abstract. The untrue answer option did not mention that an abstract class is not required to implement any of the methods declared in an interface that is named in the implements clause of the class declaration.

3 a e An interface declaration can be a member of an interface. An abstract method declaration can be a member of an interface. An interface can be declared within an enclosing class

or interface. The members of an interface can be constants, abstract method declarations, class declarations, or interface declarations. The body of a method declared within an interface is a semicolon. An interface can extend another interface, but can not implement an interface. An abstract class that has an interface, I1, in its implements clause is not required to implement any of the methods declared within I1.

4 a f abstract public All interfaces are implicitly abstract. The explicit application of the abstract modifier to an interface declaration is redundant and is strongly discouraged. The declaration of an interface within the body of an enclosing class or interface is called a member type declaration. Every member type declaration appearing within the body of a directly enclosing interface is implicitly static and public. Use of the access modifiers, private or protected, is contradictory and results in a compile-time error. In contrast, the modifiers, private and protected, are applicable to a member type declaration appearing within the body of a directly enclosing class. The modifier, final, is never applicable to an interface. The keyword, implements, is not a modifier.

5 a d e f abstract private protected public All interfaces are implicitly abstract. The explicit application of the modifier, abstract, to an interface is redundant and is strongly discouraged. The declaration of an interface within the body of an enclosing class or interface is called a member type declaration. The private, protected and static modifiers are applicable to a member type declaration that appears in the body of a directly enclosing class. In contrast, the modifiers, private and protected, are not applicable to a member type declaration appearing within the body of a directly enclosing interface. The modifier, final, is never applicable to an interface. The keyword, extends, is not a modifier.

6 a static A member interface is always implicitly static. The modifier, static, can not be applied to an interface that is not a member interface. The modifier, synchronized, is applicable to a concrete implementation of a method, but is not applicable to any interface. The modifiers, volatile and transient, are only applicable to variables that are members of a class. The keyword, implements, is not a modifier.

7 a e abstract public The modifier, abstract, is applicable to an interface declaration, but its use is strongly discouraged; because every interface is implicitly abstract. An interface can not be final. The modifiers, private and protected, are applicable only to an interface declaration that is a member of a directly enclosing class declaration. If an interface is not a member of a directly enclosing class, or if the interface is a member of a directly enclosing interface; then, the modifiers, private and protected, are not applicable. If an interface is declare public, then the compiler will generate an error if the class is not stored in a file that has the same name as the interface plus the extension .java.

8 a b abstract public The modifier, abstract, is applicable to an interface declaration, but its use is strongly discouraged; because every interface is implicitly abstract. If an interface is declare public, then the compiler will generate an error if the class is not stored in a file that has the same name as the interface plus the extension .java. The modifier, static, is applicable to a member interface, but not to an interface that is not nested. The modifier,

synchronized, is applicable only to concrete implementations of methods. The modifiers, transient and volatile, are applicable only to variables.

9 c f final public The modifier, abstract, is not applicable to a variable. All field declarations within an interface are implicitly public, static and final. Use of those modifiers is redundant but legal. Although const is a Java keyword, it is not currently used by the Java programming language. An interface member can never be private or protected.

10 a static All field declarations within an interface are implicitly public, static and final. Use of these modifiers is redundant but legal. A field that is declared final can not also be declared volatile; so a field of an interface can not be declared volatile. The modifier, synchronized, is never applicable to a field.

11 a d e f public abstract static final A class that is declared within an enclosing interface is implicitly public and static; so the access modifiers, protected and private, are not applicable.

12 e None of the above All field declarations within an interface are implicitly public, static and final. Use of these modifiers is redundant but legal. No other modifiers can be applied to a field declaration within an interface.

13 e 5 All field declarations within an interface are implicitly public, static and final. Use of these modifiers is redundant but legal. No other modifiers can be applied to a field declaration within an interface.

14 a e abstract public All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. An abstract method can not also be declared private, static, final, native or synchronized; so the same restriction applies to methods declared within an interface.

15 f None of the above All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. An abstract method can not also be declared private, static, final, native or synchronized; so the same restriction applies to methods declared within an interface. Transient and volatile are not method modifiers.

16 c d e 3 4 5 Methods declared within an interface are implicitly public even if the modifier, public, is omitted from the declaration. Within the body of a class declaration, an attempt to implement the method using a weaker access privilege, private, protected or package access, results in a compile-time error.

17 c d 3 4 All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. Since all methods declared within an interface are implicitly public, a weaker access level can not be declared.

18 a b c 1 2 3 All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. The final, synchronized and native modifiers can not appear in the declaration of an abstract method,

and can not be applied to an abstract method declared within an interface.

19 c d e 3 4 5 All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. Since all methods declared within an interface are implicitly public, a weaker access level can not be declared.

20 a 1 All methods declared within an interface are implicitly abstract. The final, synchronized and native modifiers can not appear in the declaration of an abstract method, and can not be applied to an abstract method declared within an interface.

21 c Compile-time error In the declaration of interface B, the keyword, extends, has been replaced by the keyword, implements.

22 c Compile-time error Fields declared within an interface are implicitly public, final, and static. A compile-time error is generated in response to the attempt to increment the value of i.

23 b 2 All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. Methods declared within an interface are implicitly public even if the modifier, public, is omitted from the declaration. Within the body of a class declaration, an attempt to implement the method using a weaker access privilege, private, protected or package access, results in a compile-time error. An abstract class that implements an interface is free to override any of the inherited method declarations with another abstract method declaration.

24 e None of the above All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration within an interface, the usage is redundant and is discouraged. The modifiers, final, synchronized and native, can not appear in the declaration of an abstract method, but they can be added to an implementation of an abstract method.

25 h Compile-time error at line 4 Class C10 inherits ambiguous declarations of the name field. As long as the field is not referenced as a member of class C10; then, no compile-time error occurs. Line 4 generates the compile-time error, because it is the first to access the name field as a member of class C10.

26 b Prints: I10.s10,I20.s20,I20 Class C20 inherits ambiguous declarations of the name field. As long as the field is not referenced as a member of class C20; then, no compile-time error occurs. Although line 4 may appear to generate the compile-time error it does not, because name is accessed directly as a member of interface I20. Therefore, the compiler does not encounter an ambiguity.

Inheritance

Question 1

Which of the following statements are true?

- a. A constructor can invoke another constructor of the same class using the alternate constructor invocation, "this(argumentListopt);".
- b. A constructor can invoke itself using the alternate constructor invocation, "this(argumentListopt);".
- c. The alternate constructor invocation, "this(argumentListopt);", can legally appear anywhere in the constructor body.
- d. A constructor can invoke the constructor of the direct superclass using the superclass constructor invocation, "super(argumentListopt);".
- e. The number of constructor invocations that may appear in any constructor body can equal but not exceed the number of alternate constructors declared in the same class.
- f. A constructor is not permitted to throw an exception.

Question 2

Suppose that the superclass constructor invocation, "super(argumentListopt);", appears explicitly in a subclass constructor. If a compile-time error is to be avoided then the arguments for the superclass constructor invocation, "super(argumentListopt);", can not refer to which of the following?

- a. Static variables declared in this class or any superclass.
- b. Instance variables declared in this class or any superclass.
- c. Static methods declared in this class or any superclass.
- d. Instance methods declared in this class or any superclass.
- e. The keyword this.
- f. The keyword super.

Question 3

```
class A {void m1(String s1) {}}
class B extends A {
    void m1(String s1) {} // 1
    void m1(boolean b) {} // 2
    void m1(byte b) throws Exception {} // 3
    String m1(short s) {return new String();} //4
    private void m1(char c) {} // 5
    protected void m1(int i) {} // 6
}
```

What is the result of attempting to compile the program?

- a. Compile-time error at line 1
- b. Compile-time error at line 2
- c. Compile-time error at line 3
- d. Compile-time error at line 4
- e. Compile-time error at line 5
- f. Compile-time error at line 6
- g. None of the above

Question 4

```
class A {void m1() {System.out.print("A.m1");}}
class B extends A {
    void m1() {System.out.print("B.m1");}
    static void m1(String s) {System.out.print(s+",");}
}
class C {
    public static void main (String[] args) {B.m1("main"); new B().m1();}}
```

}

What is the result of attempting to compile and run the program?

- a. Prints: main,B.m1
- b. Compile-time error
- c. Run-time error
- d. None of the above

Question 5

Which of the following are true statements?

- a. The relationship between a class and its superclass is an example of a "has-a" relationship.
- b. The relationship between a class and its superclass is an example of an "is-a" relationship.
- c. The relationship between a class and an object referenced by a field within the class is an example of a "has-a" relationship.
- d. The relationship between a class and an object referenced by a field within the class is an example of an "is-a" relationship.

Question 6

```
class A {String s1 = "A.s1"; String s2 = "A.s2";}
class B extends A {
    String s1 = "B.s1";
    public static void main(String args[]) {
        B x = new B(); A y = (A)x;
        System.out.println(x.s1+" "+x.s2+" "+y.s1+" "+y.s2);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: B.s1 A.s2 B.s1 A.s2
- b. Prints: B.s1 A.s2 A.s1 A.s2
- c. Prints: A.s1 A.s2 B.s1 A.s2
- d. Prints: A.s1 A.s2 A.s1 A.s2
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 7

```
class C {
    void printS1() {System.out.print("C.printS1 ");}
    static void printS2() {System.out.print("C.printS2 ");}
}
class D extends C {
    void printS1(){System.out.print("D.printS1 ");}
    void printS2() {System.out.print("D.printS2 ");}
    public static void main (String args[]) {
        C c = new D(); c.printS1(); c.printS2();
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: C.printS1 C.printS2
- b. Prints: C.printS1 D.printS2
- c. Prints: D.printS1 C.printS2
- d. Prints: D.printS1 D.printS2

- e. Run-time error
 - f. Compile-time error
 - g. None of the above
- Question 8

```
class E {
    void printS1(){System.out.print("E.printS1 ");}
    static void printS2() {System.out.print("E.printS2");}
}
class F extends E {
    void printS1(){System.out.print("F.printS1 ");}
    static void printS2() {System.out.print("F.printS2");}
    public static void main (String args[]) {
        E x = new F(); x.printS1(); x.printS2();
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: E.printS1 E.printS2
- b. Prints: E.printS1 F.printS2
- c. Prints: F.printS1 E.printS2
- d. Prints: F.printS1 F.printS2
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 9

```
class P {
    static void printS1(){System.out.print("P.printS1 ");}
    void printS2() {System.out.print("P.printS2 ");}
    void printS1S2(){printS1();printS2();}
}
class Q extends P {
    static void printS1(){System.out.print("Q.printS1 ");}
    void printS2(){System.out.print("Q.printS2 ");}
    public static void main(String[] args) {
        new Q().printS1S2();
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: P.printS1 P.printS2
- b. Prints: P.printS1 Q.printS2
- c. Prints: Q.printS1 P.printS2
- d. Prints: Q.printS1 Q.printS2
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 10

```
class R {
    private void printS1(){System.out.print("R.printS1 ");}
    protected void printS2() {System.out.print("R.printS2 ");}
    protected void printS1S2(){printS1();printS2();}
}
class S extends R {
```

```

private void printS1(){System.out.print("S.printS1 ");}
protected void printS2(){System.out.print("S.printS2 ");}
public static void main(String[] args) {
    new S().printS1S2();
}
}

```

What is the result of attempting to compile and run the program?

- a. Prints: R.printS1 R.printS2
- b. Prints: R.printS1 S.printS2
- c. Prints: S.printS1 R.printS2
- d. Prints: S.printS1 S.printS2
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 11

```

class T {
    private int i1, i2;
    void printI1I2() {System.out.print("T, i1="+i1+", i2="+i2);}
    T(int i1, int i2) {this.i1=i1; this.i2=i2;}
}
class U extends T {
    private int i1, i2;
    void printI1I2() {System.out.print("U, i1="+i1+", i2="+i2);}
    U(int i1, int i2) {this.i1=i1; this.i2=i2;}
    public static void main(String[] args) {
        T t = new U(1,2); t.printI1I2();
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: U, i1=1, i2=2
- b. Prints: T, i1=1, i2=2
- c. Prints: U, i1=null, i2=null
- d. Prints: T, i1=null, i2=null
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 12

```

interface I {String s1 = "I";}
class A implements I {String s1 = "A";}
class B extends A {String s1 = "B";}
class C extends B {
    String s1 = "C";
    void printIt() {
        System.out.print(((A)this).s1 + ((B)this).s1 +
                        ((C)this).s1 + ((I)this).s1);
    }
    public static void main (String[] args) {new C().printIt();}
}

```

What is the result of attempting to compile and run the program?

- a. Prints: ABCI
- b. Run-time error

- c. Compile-time error
- d. None of the above

Question 13

```
abstract class D {String s1 = "D"; String getS1() {return s1;}}
```

```
class E extends D {String s1 = "E"; String getS1() {return s1;}}
```

```
class F {
```

```
    public static void main (String[] s) {
```

```
        D x = new E(); System.out.print(x.s1 + x.getS1());
```

```
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: DD
- b. Prints: DE
- c. Prints: ED
- d. Prints: EE
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 14

```
class A {static void m() {System.out.print("A");}}
```

```
class B extends A {static void m() {System.out.print("B");}}
```

```
class C extends B {static void m() {System.out.print("C");}}
```

```
class D {
```

```
    public static void main(String[] args) {
```

```
        C c = new C(); c.m(); B b = c; b.m(); A a = b; a.m();
```

```
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CBA
- d. Prints: CCC
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 15

```
class A {String s1 = "A";}
```

```
class B extends A {String s1 = "B";}
```

```
class C extends B {String s1 = "C";}
```

```
class D {
```

```
    static void m1(A x) {System.out.print(x.s1);}
```

```
    static void m1(B x) {System.out.print(x.s1);}
```

```
    static void m1(C x) {System.out.print(x.s1);}
```

```
    public static void main(String[] args) {
```

```
        A a; B b; C c; a = b = c = new C(); m1(a); m1(b); m1(c);
```

```
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC

- c. Prints: CBA
 - d. Prints: CCC
 - e. Compile-time error
 - f. Run-time error
 - g. None of the above
- Question 16

```

class Leg{}
class Fur{}
abstract class Pet {
    public abstract void eat();
    public abstract void sleep();
}
class Dog extends Pet {
    Leg leftFront = new Leg(), rightFront = new Leg();
    Leg leftRear = new Leg(), rightRear = new Leg();
    Fur fur = new Fur();
    public Fur shed() {return fur;}
    public void eat() {}
    public void sleep() {}
}
class Cat extends Dog {
    public void ignoreOwner() {}
    public void climbTree() {}
}

```

Which of the following statements is not a true statement?

- a. A Cat object inherits an instance of Fur and four instances of Leg from the Dog superclass.
- b. A Cat object is able to sleep and eat.
- c. A Cat object is able to climb a tree.
- d. The relationship between Dog and Pet is an example of an appropriate use of inheritance.
- e. The relationship between Cat and Dog is an example of an appropriate use of inheritance.
- f. None of the above.

Question 17

```

class A {
    A() {System.out.print("CA ");}
    static {System.out.print("SA ");}
}
class B extends A {
    B() {System.out.print("CB ");}
    static {System.out.print("SB ");}
    public static void main (String[] args) {B b = new B();}
}

```

What is the result of attempting to compile and run the above program?

- a. Prints: SA CA SB CB
- b. Prints: SA SB CA CB
- c. Prints: SB SA CA CB
- d. Prints: SB CB SA CA
- e. Runtime Exception

- f. Compiler Error
 - g. None of the above
- Question 18

```
class A {String s1="A";}
class B extends A {String s1="B";}
class C extends B {String s1="C";}
class D extends C {
    String s1="D";
    void m1() {
        System.out.print(this.s1 + ",");      // 1
        System.out.print(((C)this).s1 + ","); // 2
        System.out.print(((B)this).s1 + ","); // 3
        System.out.print(((A)this).s1);       // 4
    }
    public static void main (String[] args) {
        new D().m1(); // 5
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: D,D,D,D
- b. Prints: D,C,B,A
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.
- f. Compile-time error at 4.
- g. Compile-time error at 5.
- h. Run-time error
- i. None of the above

Question 19

```
class SuperA {String s1="SuperA";}
class SuperB {String s1="SuperB";}
class A extends SuperA {
    String s1="A";
    class B extends SuperB { // 1
        String s1="B";
        void m1() {
            System.out.print(this.s1 + ","); // 2
            System.out.print(super.s1 + ","); // 3
            System.out.print(A.this.s1 + ","); // 4
            System.out.print(A.super.s1); // 5
        }
    }
    public static void main (String[] args) {
        new A().new B().m1(); // 6
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: B,SuperB,B,SuperB
- b. Prints: B,SuperB,A,SuperA
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.

- f. Compile-time error at 4.
- g. Compile-time error at 5.
- h. Compile-time error at 6.
- i. Run-time error
- j. None of the above

Question 20

```
class A {void m1() {System.out.print("A");}}
class B extends A {void m1(){System.out.print("B");}}
class C extends B {void m1() {System.out.print("C");}}
class D extends C {
    void m1() {System.out.print("D");}
    void m2() {
        m1();
        ((C)this).m1(); // 1
        ((B)this).m1(); // 2
        ((A)this).m1(); // 3
    }
    public static void main (String[] args) {
        new D().m2(); // 4
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: DCBA
- b. Prints: DDDD
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.
- f. Compile-time error at 4.
- g. Run-time error
- h. None of the above

Question 21

```
class A {public void m1() {System.out.print("A1");}}
class B extends A {
    public void m1() {System.out.print("B1");}
    public void m2() {System.out.print("B2");}
}
class C {
    public static void main(String[] args) {
        A a1 = new B();
        a1.m1();      // 1
        a1.m2();      // 2
        ((B)a1).m1(); // 3
        ((B)a1).m2(); // 4
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A1B2B1B2
- b. Prints: B1B2B1B2
- c. Compile-time error at 1
- d. Compile-time error at 2
- e. Compile-time error at 3
- f. Compile-time error at 4

- g. Run-time error
- h. None of the above

No.	Answer	Remark
1	a d	A constructor can invoke another constructor of the same class using the alternate constructor invocation, "this(argumentListOpt);". A constructor can invoke the constructor of the direct superclass using the superclass constructor invocation, "super(argumentListOpt);". If an alternate constructor invocation appears in the body of the constructor, then it must be the first statement. The same is true for a superclass constructor invocation. A compile-time error is generated if a constructor attempts to invoke itself either directly or indirectly.
2	b d e f	Instance variables declared in this class or any superclass. Instance methods declared in this class or any superclass. The keyword this. The keyword super. If the superclass constructor invocation, "super(argumentListOpt);", appears explicitly or implicitly, then it must be the first statement in the body of the constructor. Until the superclass constructor invocation runs to completion, no other statements are processed within the body of the constructor. The same is true of the constructors of any superclass. (Note: The primordial class, Object, does not have a superclass, so the constructors do not include a superclass constructor invocation statement.) Suppose class B is a subclass of A. The process of creating and initializing an instance of B includes the creation and initialization of an instance of the superclass A and an instance of the superclass Object. The superclass constructor invocation statement appearing in a constructor of B is invoked before the completion of the constructors of the superclasses A and Object. A superclass constructor invocation statement appearing in B can not refer to the non-static members of the superclasses, because the process of initializing those non-static superclass members is not complete when the superclass constructor invocation occurs in B. The same is true of the non-static members of B.
3	g	None of the above If a superclass method is not private and is accessible to code in a subclass, then any subclass method that has the same signature as the superclass method must also have the same return type. In other words, if a superclass method is overridden or hidden in a subclass, then the overriding or hiding subclass method must have the same return type as the superclass method. No such restriction applies to method overloading. If two methods share an overloaded method name but not the same parameter list, then the two methods need not have the same return type. Class A declares one method: The name is m1 and the single parameter is of type String. Class B extends A and declares six methods that overload the name m1. The method, B.m1(String s1), overrides the superclass method, A.m1(String s1). The overriding subclass method must have the same return type as the superclass method, but the methods that overload the name m1 are free to have different return types. An overriding subclass method is not permitted to throw any checked exception that is not listed or is not a subclass of any of those listed in the throws clause of the superclass method. No such restriction applies to method overloading. If two methods share an overloaded method name but not the same parameter list, then the two methods are free to have differing throws clauses.

4 a Prints: main,B.m1 Suppose a superclass method is not private and is accessible to code in a subclass. If the superclass method is declared static, then any subclass method sharing the same signature must also be declared static. Similarly, if the superclass method is not declared static, then any subclass method sharing the same signature must not be declared static. The rules governing method overloading are different. If a superclass method is declared static, then any subclass method that overloads the superclass method is free to be declared static or non-static. Similarly, if a method is declared non-static, then any overloading method is free to be declared static or non-static. Method B.m1() shares the same signature as the non-static superclass method A.m1(), so B.m1() must also be non-static. The method B.m1(String s) overloads the method name m1, but does not share the same signature with any superclass method; therefore, B.m1(String s) can be declared static even though the other methods of the same name are non-static.

5 b c The relationship between a class and its superclass is an example of an "is-a" relationship. The relationship between a class and an object referenced by a field within the class is an example of a "has-a" relationship. Inheritance is an example of an "is-a" relationship, because the subclass "is-a" specialized type of the superclass. The relationship between a class and an object referenced by a field declared within the class is an example of a "has-a" relationship, because the class "has-a" object.

6 b Prints: B.s1 A.s2 A.s1 A.s2 The variables of a subclass can hide the variables of a superclass or interface. The variable that is accessed is determined at compile-time based on the type of the reference--not the run-time type of the object. The two references x and y refer to the same instance of type B. The name x.s1 uses a reference of type B; so it refers to the variable s1 declared in class B. The name y.s1 uses a reference of type A; so it refers to the variable s1 declared in class A.

7 f Compile-time error Suppose a superclass method is not private and is accessible to code in a subclass. If the superclass method is declared static, then any subclass method sharing the same signature must also be declared static. Similarly, if the superclass method is declared non-static, then any subclass method sharing the same signature must also be declared non-static. The attempted declaration of the non-static method D.printS2 generates a compile-time error; because the superclass method, C.printS2, is static.

8 c Prints: F.printS1 E.printS2 A static method is selected based on the compile-time type of the reference--not the run-time type of the object. A non-static method is selected based on the run-time type of the object--not the compile-time type of the reference. Both method invocation expressions, x.printS1() and x.printS2(), use a reference of the superclass type, E, but the object is of the subclass type, F. The first of the two expressions invokes an instance method on an object of the subclass type; so the overriding subclass method is selected. The second invokes a static method using a reference of the superclass type; so the superclass method is selected.

9 b Prints: P.printS1 Q.printS2 Suppose a method m1 is invoked using the method invocation expression m1(). If m1 is a static member of the class where the invocation expression occurs, then that is the implementation of the method that is invoked at run-time regardless of the run-time type of the object. If m1 is non-static, then the

selected implementation is determined at run-time based on the run-time type of the object. The program invokes method printS1S2 on an instance of class Q. The body of method printS1S2 contains two method invocation expressions, printS1() and printS2(). Since method printS1 is static, the implementation declared in class P is invoked. Since printS2 is non-static and the run-time type of the object is Q, the invoked method is the one declared in class Q.

10 b Prints: R.printS1 S.printS2 A private method of a superclass is not inherited by a subclass. Even if a subclass method has the same signature as a superclass method, the subclass method does not override the superclass method. Suppose a non-static method m1 is invoked using the method invocation expression m1(). If m1 is a private member of the class T where the invocation expression occurs, then the implementation in class T is selected at run-time regardless of the run-time type of the object. If the non-static method m1 is not private, then the selected implementation is determined at run-time based on the run-time type of the object. The program invokes the non-static method printS1S2 on an instance of class S, so the run-time type is S. The body of method R.printS1S2 contains two method invocation expressions, printS1() and printS2(). Since class R contains a private implementation of the instance method printS1, it is the implementation that is selected regardless of the run-time type of the object. Since printS2 is not private and not static, the selected implementation of printS2 depends on the run-time type of the object. The method printS1S2 is invoked on an instance of class S; so the run-time type of the object is S, and the implementation of printS2 declared in class S is selected.

11 f Compile-time error The two-parameter constructor of U does not explicitly invoke the two-parameter constructor of T; therefore, the constructor of U will try to invoke a no-parameter constructor of T, but none exists.

12 a Prints: ABCI Suppose that a class extends a superclass, X, or implements an interface, X. The field access expression ((X)this).hiddenField is used to access the hidden field, hiddenField, that is accessible within the superclass or interface, X.

13 b Prints: DE At run-time, the actual field that is accessed depends on the compile-time type of the reference--not the run-time type of the object. The compile-time type of the reference x in the name x.s1 is D; so the selected field is the one declared in class D. A non-static method is selected based on the run-time type of the object--not the compile-time type of the reference. The same reference variable x is used in the method invocation expression x.getS1(), and the compile-time type is still D. At run-time, the actual type of the object is E; so the selected method is the one declared in class E.

14 c Prints: CBA Class C extends B, and B extends A. The static method C.m hides method B.m, and B.m hides A.m. In the method invocation expression c.m(), the compile-time type of the reference c is C. A static method is invoked based on the compile-time type of the reference; so the method invocation expression c.m() invokes the method m declared in class C. The compile-time type of the reference b is B; so the method invocation expression b.m() invokes the method m declared in class B. The compile-time type of the reference a is A; so the method invocation expression a.m() invokes the method m declared in class A.

15 b Prints: ABC In all three cases, the object passed to method m1 is an instance of class C; however, the type of the reference

is different for each method. Since fields are accessed based on the type of the reference, the value printed by each method is different even though the same instance is used for each method invocation.

16 e The relationship between Cat and Dog is an example of an appropriate use of inheritance. An appropriate inheritance relationship includes a subclass that "is-a" special kind of the superclass. The relationship between the Dog subclass and the Pet superclass is an example of an appropriate inheritance relationship, because a Dog "is-a" Pet. The relationship between the Cat subclass and the Dog superclass is not an example of an appropriate use of inheritance, because a Cat is not a special kind of a Dog. The goal of the OO paradigm is to develop software models that are accurate and reusable. If the software model is not accurate, then it probably is not reusable and the goals of the OO paradigm are not achieved. Code reuse and maintenance becomes increasingly difficult when inheritance is used to model inappropriate relationships. For example, suppose that somebody implements a herdSheep method in the Dog class. The Cat subclass would inherit the method and suddenly each instance of Cat would acquire the unwanted capability to make an attempt to herd sheep. It is difficult to imagine that a Cat would perform well in that role, so additional maintenance would be required to resolve the problem.

17 b Prints: SA SB CA CB The static initializer of the super class runs before the static initializer of the subclass. The body of the superclass constructor runs to completion before the body of the subclass constructor runs to completion.

18 b Prints: D,C,B,A A field of a superclass can be inherited by a subclass if the superclass field is not private and not hidden by a field declaration in the subclass and is accessible to code in the subclass. The field D.s1 hides C.s1, and C.s1 hides B.s1, and B.s1 hides A.s1. The keyword this serves as a reference to the object on which a method has been invoked. In the field access expression this.s1 appearing on line 1, the keyword this denotes a reference to the object of type D on which method m1 has been invoked. In the field access expression ((C)this).s1 appearing on line 2, the reference denoted by the keyword this is cast from type D to type C. The field that is accessed at run-time depends on the compile-time type of the reference; so the field access expression ((C)this).s1 refers to the variable s1 declared in class C.

19 b Prints: B,SuperB,A,SuperA The expression A.this.s1 is an example of a qualified this expression. It accesses the variable s1 declared in class A. The expression A.super.s1 is equivalent to ((SuperA)A.this).s1. It accesses the variable s1 declared in class SuperA.

20 b Prints: DDDD The instance method that is invoked depends on the run-time type of the object--not the compile-time type of the reference. In each case, the method m1 is invoked on an object of type D; so the implementation of m1 in type D is selected each time.

21 d Compile-time error at 2 The method invocation expression a1.m2() generates a compile-time error, because the named method, m2, is declared in class B, but the reference is of the superclass type, A. The reference a1 is of type A; so a1 is able to access only those methods that are declared in class A and subclass methods that override those of class A. Only one method, m1, is declared in A; so a reference of type A can be used to invoke A.m1 or an overriding implementation of m1 that is

declared in a subclass of A. Class B extends A and overrides method m1. A reference of type A can be used to invoke method m1 on an instance of type B. Class B declares an additional method, m2, that does not override a method of class A; so a reference of type A can not invoke B.m2.

Method Overloading

Question 1

```
class A {void m1(A a) {System.out.print("A");}}
```

```
class B extends A {void m1(B b) {System.out.print("B");}}
```

```
class C extends B {void m1(C c) {System.out.print("C");}}
```

```
class D extends C {
    void m1(D d) {System.out.print("D");}
    public static void main(String[] args) {
        A a1 = new A(); B b1 = new B(); C c1 = new C(); D d1 = new D();
        d1.m1(a1); d1.m1(b1); d1.m1(c1);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: DDD
- d. Prints: ABCD
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 2

```
class GFC215 {
    static String m(float i) {return "float";}
    static String m(double i) {return "double";}
    public static void main (String[] args) {
        int a1 = 1; long b1 = 2; System.out.print(m(a1)+","+ m(b1));
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: float,float
- b. Prints: float,double
- c. Prints: double,float
- d. Prints: double,double
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 3

```
class A {} class B extends A {} class C extends B {}
class D {
    void m1(A a) {System.out.print("A");}
    void m1(B b) {System.out.print("B");}
    void m1(C c) {System.out.print("C");}
    public static void main(String[] args) {
        A c1 = new C(); B c2 = new C(); C c3 = new C(); D d1 = new D();
        d1.m1(c1); d1.m1(c2); d1.m1(c3);
    }
}
```

}}

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CCC
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 4

```
class GFC216 {  
    static String m(float i) {return "float";}  
    static String m(double i) {return "double";}  
    public static void main (String[] args) {  
        char a1 = 1; long b1 = 2; System.out.print(m(a1)+", "+ m(b1));  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: float,float
- b. Prints: float,double
- c. Prints: double,float
- d. Prints: double,double
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 5

```
class A {void m1(A a) {System.out.print("A");}}  
class B extends A {void m1(B b) {System.out.print("B");}}  
class C extends B {void m1(C c) {System.out.print("C");}}  
class D {  
    public static void main(String[] args) {  
        A c1 = new C(); B c2 = new C(); C c3 = new C(); C c4 = new C();  
        c4.m1(c1); c4.m1(c2); c4.m1(c3);  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CCC
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 6

```
class GFC217 {  
    static String m(int i) {return "int";}  
    static String m(float i) {return "float";}  
    public static void main (String[] args) {  
        long a1 = 1; double b1 = 2; System.out.print(m(a1)+", "+ m(b1));  
    } }
```

What is the result of attempting to compile and run the program?

- a. Prints: float,float
- b. Prints: float,double
- c. Prints: double,float
- d. Prints: double,double
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 7

```
class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
    public static void main(String[] args) {
        A c1 = new C(); C c2 = new C(); c1.m1(c2);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Prints: C
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 8

```
class GFC218 {
    static void m(Object x) {System.out.print("Object");}
    static void m(String x) {System.out.print("String");}
    public static void main(String[] args) {m(null);}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: Object
- b. Prints: String
- c. Compile-time error
- d. Run-time error
- e. None of the above

Question 9

```
class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
    public static void main(String[] args) {
        A a1 = new A(); B b1 = new B(); C c1 = new C(); C c4 = new C();
        a1.m1(c4); b1.m1(c4); c1.m1(c4);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CCC
- d. Compile-time error

- e. Run-time error
 - f. None of the above
- Question 10

```
class GFC200 {}
class GFC201 {
    static void m(Object x) {System.out.print("Object");}
    static void m(String x) {System.out.print("String");}
    static void m(GFC200 x) {System.out.print("GFC200");}
    public static void main(String[] args) {m(null);}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: Object
- b. Prints: String
- c. Prints: GFC200
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 11

```
class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
    public static void main(String[] args) {
        A a1 = new A(); B b1 = new B(); C c1 = new C(); A c2 = new C();
        c2.m1(a1); c2.m1(b1); c2.m1(c1);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CCC
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 12

```
class GFC202 {} class GFC203 extends GFC202 {}
class GFC204 {
    static void m(GFC202 x) {System.out.print("GFC202");}
    static void m(GFC203 x) {System.out.print("GFC203");}
    public static void main(String[] args) {m(null);}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: GFC202
- b. Prints: GFC203
- c. Compile-time error
- d. Run-time error
- e. None of the above

Question 13

```

class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
    public static void main(String[] args) {
        A a1 = new A(); B b1 = new A(); C c1 = new A(); C c2 = new C();
        c2.m1(a1); c2.m1(b1); c2.m1(c1);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CCC
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 14

```

class GFC205 {} class GFC206 extends GFC205 {}
class GFC207 extends GFC206 {
    static void m(GFC205 x, GFC205 y) {System.out.print("GFC205,GFC205");}
    static void m(GFC205 x, GFC206 y) {System.out.print("GFC205,GFC206");}
    static void m(GFC206 x, GFC205 y) {System.out.print("GFC206,GFC205");}
    static void m(GFC206 x, GFC206 y) {System.out.print("GFC206,GFC206");}
    public static void main(String[] args) {
        GFC207 gfc207 = new GFC207(); m(gfc207, gfc207);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: GFC205,GFC205
- b. Prints: GFC205,GFC206
- c. Prints: GFC206,GFC205
- d. Prints: GFC206,GFC206
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 15

```

class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
    public static void main(String[] args) {
        A a1 = new A(); B b1 = new B(); C c1 = new C(); C c2 = new A();
        c2.m1(a1); c2.m1(b1); c2.m1(c1);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CCC
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 16

```
class GFC211 {} class GFC212 extends GFC211 {}
class GFC213 extends GFC212 {
    static void m(GFC211 x, GFC211 y) {System.out.print("GFC211,GFC211");}
    static void m(GFC211 x, GFC212 y) {System.out.print("GFC211,GFC212");}
    static void m(GFC212 x, GFC211 y) {System.out.print("GFC212,GFC211");}
    static void m(GFC212 x, GFC212 y) {System.out.print("GFC212,GFC212");}
    static void m(GFC211 x, GFC213 y) {System.out.print("GFC211,GFC213");}
    public static void main(String[] args) {
        GFC213 gfc213 = new GFC213(); m(gfc213, gfc213);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: GFC211,GFC211
- b. Prints: GFC211,GFC212
- c. Prints: GFC212,GFC211
- d. Prints: GFC212,GFC212
- e. Prints: GFC211,GFC213
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 17

```
class GFC214 {
    static void m1(boolean b1) {System.out.print("boolean ");}
    static void m1(byte b1) {System.out.print("byte ");}
    static void m1(short s1) {System.out.print("short ");}
    static void m1(char c1) {System.out.print("char ");}
    static void m1(int i1) {System.out.print("int ");}
    public static void main(String[] args) {
        byte b1; m1(b1 = 1); m1(b1); m1(b1 == 1);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: byte byte byte
- b. Prints: byte byte boolean
- c. Prints: int int int
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 18

```
class A {} class B extends A {}
class C extends B {
    static void m(A x, A y) {System.out.print("AA");}
    static void m(A x, B y) {System.out.print("AB");}
    static void m(B x, A y) {System.out.print("BA");}
    static void m(B x, B y) {System.out.print("BB");}
    public static void main(String[] args) {
        A a1; B b1; m(null,null); m(a1=null,b1=null); m(b1, a1);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: BBABAB
- b. Prints: BBABBA
- c. Prints: BBBBAB
- d. Prints: BBBBBA
- e. Prints: BBBBBB
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 19

```
class A {} class B extends A {}
class C extends B {
    static void m1(A x) {System.out.print("m1A");}
    static void m2(B x) {System.out.print("m2B"); m1(x);}
    static void m2(A x) {System.out.print("m2A"); m1(x);}
    static void m3(C x) {System.out.print("m3C"); m2(x);}
    static void m3(B x) {System.out.print("m3B"); m2(x);}
    static void m3(A x) {System.out.print("m3A"); m2(x);}
    public static void main(String[] args) {m3(new C());}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: m3Am2Am1A
- b. Prints: m3Bm2Bm1A
- c. Prints: m3Cm2Bm1A
- d. Prints: m3Cm2Am1A
- e. Compile-time error
- f. Run-time error
- g. None of the above

No. Answer Remark

1 b Prints: ABC The method invocation expression d1.m1(a1) uses reference d1 of type D to invoke method m1. Since the reference d1 is of type D, the class D is searched for an applicable implementation of m1. The methods inherited from the superclasses, C, B and A, are included in the search. The argument, a1, is a variable declared with the type A; so method A.m1(A a) is invoked.

2 a Prints: float,float A method invocation conversion can widen an argument of type float to match a method parameter of type double, so any argument that can be passed to m(float i) can also be passed to m(double i) without generating a compile-time type error. For that reason, we can say that m(float i) is more specific than m(double i). Since both methods are applicable, the more specific of the two, m(float i), is chosen over the less specific, m(double i). The arguments of the method invocation expressions, m(a1) and m(b1), are of types int and long respectively. A method invocation conversion can widen an argument of type int or long to match either of the two method parameter types float or double; so both methods, m(float i) and m(double i), are applicable to the two method invocation expressions. Since both methods are applicable, the more specific of the two, m(float i) is chosen rather than the less specific, m(double i).

3 b Prints: ABC Three methods overload the method name m1. Each has a single parameter of type A or B or C. For any method invocation expression of the form m1(referenceArgument), the method is

selected based on the declared type of the variable referenceArgument--not the run-time type of the referenced object. The method invocation expression d1.m1(c1) uses reference d1 of type D to invoke method m1 on an instance of type D. The argument, c1, is a reference of type A and the run-time type of the referenced object is C. The argument type is determined by the declared type of the reference variable c1--not the run-time type of the object referenced by c1. The declared type of c1 is type A; so the method m1(A a) is selected. The declared type of c2 is type B; so the method invocation expression d1.m1(c2) invokes method m1(B b). The declared type of c3 is type C; so the method invocation expression d1.m1(c3) invokes method m1(C c).

4 a Prints: float,float A method invocation conversion can widen an argument of type float to match a method parameter of type double, so any argument that can be passed to m(float i) without generating a compile-time type error can also be passed to m(double i). For that reason, we can say that m(float i) is more specific than m(double i). The arguments of the method invocation expressions, m(a1) and m(b1), are of types char and long respectively. A method invocation conversion can widen an argument of type char or long to match either of the two method parameter types float or double; so both methods, m(float i) and m(double i), are applicable to the two method invocation expressions. Since both methods are applicable, the more specific of the two, m(float i) is chosen rather than the less specific, m(double i).

5 b Prints: ABC Three methods overload the method name m1. Each has a single parameter of type A or B or C. For any method invocation expression of the form m1(referenceArgument), the method is selected based on the declared type of the variable referenceArgument--not the run-time type of the referenced object. The method invocation expression c4.m1(c1) uses reference c4 of type C to invoke method m1 on an instance of type C. The argument, c1, is a reference of type A and the run-time type of the referenced object is C. The argument type is determined by the declared type of the reference variable c1--not the run-time type of the object referenced by c1. The declared type of c1 is type A; so the method A.m1(A a) is selected. The declared type of c2 is type B; so the method invocation expression c4.m1(c2) invokes method B.m1(B b). The declared type of c3 is type C; so the method invocation expression c4.m1(c3) invokes method C.m1(C c).

6 e Compile-time error The method invocation expression, m(b1), contains an argument of type double. A method invocation conversion will not implicitly narrow the argument to match the parameter type of the method, m(float i). The method invocation expression, m(a1), contains an argument of type long. A method invocation conversion will widen the argument to match the parameter type of the method, m(float i).

7 a Prints: A The reference c1 is of the superclass type, A; so it can be used to invoke only the method m1 declared in class A. The methods that overload the method name m1 in the subclasses, B and C, can not be invoked using the reference c1. A method invocation conversion promotes the argument referenced by c2 from type C to type A, and the method declared in class A is executed. Class A declares only one method, m1. The single parameter is of type A. Class B inherits the method declared in class A and overloads the method name with a new method that has a single parameter of type B. Both methods sharing the overloaded name, m1, can be invoked using a reference of type B; however, a

reference of type A can be used to invoke only the method declared in class A. Class C inherits the methods declared in classes A and B and overloads the method name with a new method that has a single parameter of type C. All three methods sharing the overloaded name, m1, can be invoked using a reference of type C; however, a reference of type B can be used to invoke only the method declared in class B and the method declared in the superclass A. The method invocation expression c1.m1(c2) uses reference c1 of type A to invoke method m1. Since the reference c1 is of type A, the search for an applicable implementation of m1 is limited to class A. The subclasses, B and C, will not be searched; so the overloading methods declared in the subclasses can not be invoked using a reference of the superclass type.

8 b Prints: String A method invocation conversion can widen an argument of type String to match a method parameter of type Object, so any argument that can be passed to m(String x) without generating a compile-time type error can also be passed to m(Object x). For that reason, we can say that m(String x) is more specific than m(Object x). The argument of the method invocation expression, m(null), is of type null and can be converted to either type String or Object by method invocation conversion, so both methods, m(String x) and m(Object x), are applicable. The more specific of the two, m(String x), is chosen over the less specific, m(Object x).

9 a Prints: AAA The declared type of the reference variables, a1, b1 and c1, is the superclass type, A; so the three reference variables can be used to invoke only the method m1(A a) that is declared in the superclass, A. The methods that overload the method name m1 in the subclasses, B and C, can not be invoked using a reference variable of the superclass type, A. A method invocation conversion promotes the argument referenced by c4 from type C to type A, and the method declared in class A is executed.

10 d Compile-time error The type of the argument is null and could be converted to any of the types Object, String or GFC200, by method invocation conversion. All three methods are applicable, but none of the three is more specific than both of the other two. The ambiguity results in a compile-time type error. If type GFC200 were a subclass of type String; then any argument that could be pass to m(GFC200 x) could also be passed to m(String x) without causing a compile-time type error, and we could say that m(GFC200 x) is more specific than m(String x). Since GFC200 is not a subclass of type String, a method invocation conversion is not able to widen an argument of type GFC200 to match a method parameter of type String, so m(GFC200 x) is not more specific than m(String x).

11 a Prints: AAA The reference c2 is of the superclass type, A; so it can be used to invoke only the method, m1, declared in class A. The methods that overload the method name m1 in the subclasses, B and C, can not be invoked using the reference c2.

12 b Prints: GFC203 The type of the argument is null and could be converted to either type GFC202 or GFC203 by method invocation conversion; so both methods are applicable. The more specific of the two, m(GFC203 x), is chosen. Type GFC203 is a subclass of type GFC202; so any argument that can be passed to m(GFC203 x) can also be passed to method m(GFC202 x) without causing a compile-time type error; therefore, we can say that method m(GFC203 x) is more specific than m(GFC202 x).

13 d Compile-time error The declarations of b1 and c1 cause compile-time errors, because a reference of a subclass type can not refer to an instance of the superclass type.

14 d Prints: GFC206,GFC206 Type GFC207 is a subclass of types GFC206 and GFC205, so any of the four methods are applicable to the method invocation expression, m(gfc207, gfc207). The most specific of the four, m(GFC206 x, GFC206 y), is chosen. Type GFC206 is a subclass of type GFC205, and method m(GFC206 x, GFC206 y) is more specific than the other three, because any invocation of m(GFC206 x, GFC206 y) could also be handled by any of the other three without causing a compile-time type error.

15 d Compile-time error The declaration of c2 causes a compile-time error, because a reference of a subclass type can not refer to an instance of the superclass class.

16 f Compile-time error The method invocation expression, m(gfc213, gfc213), is ambiguous; because, no applicable method is more specific than all of the others. Method m(GFC212 x, GFC212 y) is more specific than m(GFC212 x, GFC211 y), because any invocation of m(GFC212 x, GFC212 y) could also be handled by m(GFC212 x, GFC211 y) without causing a compile-time type error. However, some invocations of m(GFC212 x, GFC212 y) can not be handled by m(GFC211 x, GFC213 y), so m(GFC212 x, GFC212 y) is not more specific than m(GFC211 x, GFC213 y). Furthermore, not all invocations of m(GFC211 x, GFC213 y) could be handled by m(GFC212 x, GFC212 y).

17 b Prints: byte byte boolean Variable b1 was initialized by the first method invocation statement, so the second method invocation statement does not result in a compile-time error. The assignment expression, b1 = 1, initializes variable b1 with the value 1, and the same value is passed as an argument to method m1(byte b1). The method invocation expression, m1(b1), invokes the same method, m1(byte b1). The argument of the third method invocation expression, m1(b1 == 1), is the result of the equality expression, b1 == 1. The type of the result and the argument is boolean, so the invoked method is m1(boolean b1).

18 b Prints: BBABBA Type B is a subclass of type A, and method m(B x, B y) is more specific than the other three; because any invocation of it could be handled by any of the other three without causing a compile-time type error. All four methods are applicable to the first method invocation expression, m(null,null). The most specific method, m(B x, B y), is chosen; and both arguments are converted to type B. In the second method invocation expression, m(a1=null,b1=null), simple assignment expressions initialize the local variables a1 and b1 with null references of types A and B respectively. The invoked method is m(A x, B y). In the third method invocation expression, the positions of the arguments are reversed relative to the previous invocation: The type of the first argument is now B and the second is A. The invoked method is m(B x, A y).

19 c Prints: m3Cm2Bm1A The method invocation expression, m3(new C()), invokes method m3(C x), because the argument type matches the parameter type of the method declaration exactly. Method m3 uses the parameter as the argument of the next invocation expression, m2(x). Of the two overloaded versions of m2, the most specific is invoked, m2(B x). Type B is a subclass of A, so any invocation of m2(B x) could be handled by m2(A x) without causing a compile-time type error. For that reason, m2(B x) is more specific than m2(A x).

Encapsulation

Question 1

Which of the following are true statements?

- a. Encapsulation is a form of data hiding.
- b. A tightly encapsulated class is always immutable.
- c. Encapsulation is always used to make programs run faster.
- d. Encapsulation helps to protect data from corruption.
- e. Encapsulation allows for changes to the internal design of a class while the public interface remains unchanged.

Question 2

Which of the following are true statements?

- a. A top-level class can not be called "tightly encapsulated" unless it is declared private.
- b. Encapsulation enhances the maintainability of the code.
- c. A tightly encapsulated class allows fast public access to member fields.
- d. A tightly encapsulated class allows access to data only through accessor and mutator methods.
- e. Encapsulation usually reduces the size of the code.
- f. A tightly encapsulated class might have mutator methods that validate data before it is loaded into the internal data model.

Question 3

A class can not be called "tightly encapsulated" unless which of the following is true?

- a. The class is declared final.
- b. All local variables are declared private.
- c. All method parameters are declared final.
- d. No method returns a reference to any object that is referenced by an internal data member.
- e. None of the above

Question 4

A class can not be called "tightly encapsulated" unless which of the following is true?

- a. All of the methods are declared private.
- b. All of the methods are synchronized.
- c. All local variables are declared final.
- d. The class is a direct subclass of Object.
- e. Accessor methods are used to prevent fields from being set with invalid data.
- f. None of the above

Question 5

A class can not be called "tightly encapsulated" unless which of the following are true?

- a. The data members can not be directly manipulated by external code.
- b. The class is declared final.
- c. It has no public mutator methods.

d. The superclass is tightly encapsulated.

Question 6

A class can not be called "tightly encapsulated" unless which of the following is true?

- a. The class is a nested class.
- b. The constructors are declared private.
- c. The mutator methods are declared private.
- d. The class implements the Encapsulated interface.
- e. None of the above

Question 7

A class can not be called "tightly encapsulated" unless which of the following is true?

- a. All member fields are declared final.
- b. The class is not anonymous.
- c. The internal data model can be read and modified only through accessor and mutator methods.
- d. The class is an inner class.
- e. None of the above

Question 8

```
class GFC500 {private String name;}  
class GFC501 {  
    private String name;  
    private void setName(String name) {this.name = name;}  
    private String getName() {return name;}  
}  
class GFC502 {  
    private String name;  
    public void setName(String name) {this.name = name;}  
    public String getName() {return name;}  
}
```

Which class is not tightly encapsulated?

- a. GFC501
- b. GFC502
- c. GFC503
- d. None of the above

Question 9

```
class GFC505 extends GFC504 {  
    public void setName(String name) {this.name = name;}  
    public String getName() {return name;}  
}  
class GFC504 extends GFC503 {  
    private void setName(String name) {this.name = name;}  
    private String getName() {return name;}  
}  
class GFC503 {String name;}
```

Which class is tightly encapsulated?

- a. GFC503
- b. GFC504

- c. GFC505
 - d. None of the above
- Question 10

```
class GFC506 {private String name;}  
class GFC507 extends GFC506 {  
    String name;  
    public void setName(String name) {this.name = name;}  
    public String getName() {return name;}  
}  
class GFC508 extends GFC506 {  
    private String name;  
    public GFC508(String name) {setName(name);}  
    public void setName(String name) {this.name = name;}  
    public String getName() {return name;}  
}
```

Which class is not tightly encapsulated?

- a. GFC506
- b. GFC507
- c. GFC508
- d. None of the above

No. Answer Remark

1 a d e Encapsulation is a form of data hiding. Encapsulation helps to protect data from corruption. Encapsulation allows for changes to the internal design of a class while the public interface remains unchanged. A tightly encapsulated class does not allow direct public access to the internal data model. Instead, access is permitted only through accessor (i.e. get) and mutator (i.e. set) methods. The additional time required to work through the accessor and mutator methods typically slows execution speed. Encapsulation is a form of data hiding. A tightly encapsulated class does not allow public access to any data member that can be changed in any way; so encapsulation helps to protect internal data from the possibility of corruption from external influences. The mutator methods can impose constraints on the argument values. If an argument falls outside of the acceptable range, then a mutator method could throw an `IllegalArgumentException`. The internal design of a tightly encapsulated class can change while the public interface remains unchanged. An immutable class is always tightly encapsulated, but not every tightly encapsulation class is immutable.

2 b d f Encapsulation enhances the maintainability of the code. A tightly encapsulated class allows access to data only through accessor and mutator methods. A tightly encapsulated class might have mutator methods that validate data before it is loaded into the internal data model. The data members of a tightly encapsulated class are declared private; so changes to the data model are less likely to impact external code. Access to internal data can be provided by public accessor (i.e. get) and mutator (i.e. set) methods. The mutator methods can be used to validate the data before it is loaded into the internal data model. The use of accessor and mutator methods is likely to increase the size of the code and slow execution speed.

3 e None of the above If a class A has a method that returns a reference to an internal, mutable object; then external code can use

the reference to modify the internal state of class A. Therefore, class A can not be considered tightly encapsulated. However, the methods of a tightly encapsulated class may return a reference to an immutable object or a reference to a copy or clone of an internal object.

4 f None of the above One answer option reads as follows: "Accessor methods are used to prevent fields from being set with invalid data." The answer would be correct if the word "Accessor" were replaced by the word "Mutator". Accessor methods are used to read data members; mutator methods are used to set data members. The mutator methods can validate the parameter values before the values are used to change the state of the internal data model.

5 a d The data members can not be directly manipulated by external code. The superclass is tightly encapsulated. If a class A is not tightly encapsulated, then no subclass of A is tightly encapsulated.

6 e None of the above A tightly encapsulated class may have public mutator methods.

7 c The internal data model can be read and modified only through accessor and mutator methods. A class is not tightly encapsulated if the internal data model can be read and/or modified without working through accessor (i.e. get) and mutator (i.e. set) methods.

8 d None of the above All three classes are tightly encapsulated, because the data members are private. A tightly encapsulated class can have public accessor and mutator methods, but it is not required to have those methods.

9 d None of the above Class GFC503 is not tightly encapsulated; so no subclass of GFC503 is tightly encapsulated.

10 b GFC507 Class GFC507 has a public field; so it is not tightly encapsulated.

Threads

Question 1

Which of the following methods are members of the Object class?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. sleep
- f. start
- g. yield
- h. wait

Question 2

Which of the following methods are static members of the Thread class?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. sleep
- f. start
- g. yield
- h. wait

Question 3

Which of the following methods are deprecated members of the Thread class?

- a. join
- b. notify
- c. notifyAll
- d. resume
- e. run
- f. sleep
- g. start
- h. stop
- i. suspend
- j. yield
- k. wait

Question 4

Which of the following methods name the InterruptedException in its throws clause?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. sleep
- f. start
- g. yield
- h. wait

Question 5

A timeout argument can be passed to which of the following methods?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. sleep
- f. start
- g. yield
- h. wait

Question 6

Which of the following instance methods should only be called by a thread that holds the lock of the instance on which the method is invoked?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. start
- f. wait

Question 7

Which of the following is a checked exception?

- a. IllegalMonitorStateException
- b. IllegalThreadStateException
- c. IllegalArgumentException

- d. InterruptedException
- e. None of the above

Question 8

Which kind of variable would you prefer to synchronize on?

- a. A member variable of a primitive type
- b. A member variable that is an object reference
- c. A method local variable that is a reference to an instance that is created within the method
- d. None of the above

Question 9

synchronized (expression) block

The synchronized statement has the form shown above. Which of the following are true statements?

- a. A compile-time error occurs if the expression produces a value of any reference type
- b. A compile-time error occurs if the expression produces a value of any primitive type
- c. A compile-time error does not occur if the expression is of type boolean
- d. The synchronized block may be processed normally if the expression is null
- e. If execution of the block completes normally, then the lock is released
- f. If execution of the block completes abruptly, then the lock is released
- g. A thread can hold more than one lock at a time
- h. Synchronized statements can be nested
- i. Synchronized statements with identical expressions can be nested

Question 10

Which of the following is a true statement?

- a. The process of executing a synchronized method requires the thread to acquire a lock
- b. Any overriding method of a synchronized method is implicitly synchronized
- c. If any method in a class is synchronized, then the class itself must also be declared using the synchronized modifier
- d. If a thread invokes a static synchronized method on an instance of class A, then the thread must acquire the lock of that instance of class A
- e. None of the above

Question 11

Which of the following thread state transitions model the lifecycle of a thread?

- a. The Dead state to the Ready state
- b. The Ready state to the Not-Runnable state
- c. The Ready state to the Running state
- d. The Running state to the Not-Runnable state
- e. The Running state to the Ready state
- f. The Not-Runnable state to the Ready state

g. The Not-Runnable state to the Running state

Question 12

Which of the following are true statements?

- a. The Thread.yield method might cause the thread to move to the Not-Runnable state
- b. The Thread.yield method might cause the thread to move to the Ready state
- c. The same thread might continue to run after calling the Thread.yield method
- d. The Thread.yield method is a static method
- e. The behavior of the Thread.yield method is consistent from one platform to the next
- f. The Thread.sleep method causes the thread to move to the Not-Runnable state
- g. The Thread.sleep method causes the thread to move to the Ready state

Question 13

Which of the following will not force a thread to move into the Not-Runnable state?

- a. Thread.yield method
- b. Thread.sleep method
- c. Thread.join method
- d. Object.wait method
- e. By blocking on I/O
- f. Unsuccessfully attempting to acquire the lock of an object
- g. None of the above

Question 14

Which of the following will cause a dead thread to restart?

- a. Thread.yield method
- b. Thread.join method
- c. Thread.start method
- d. Thread.resume method
- e. None of the above

Question 15

When a thread is created and started, what is its initial state?

- a. New
- b. Ready
- c. Not-Runnable
- d. Runnning
- e. Dead
- f. None of the above

Question 16

Which of the following are true statements?

- a. The Thread.run method is used to start a new thread running
- b. The Thread.start method causes a new thread to get ready to run at the discretion of the thread scheduler
- c. The Runnable interface declares the start method
- d. The Runnable interface declares the run method
- e. The Thread class implements the Runnable interface

- f. If an Object.notify method call appears in a synchronized block, then it must be the last method call in the block
- g. No restriction is placed on the number of threads that can enter a synchronized method
- h. Some implementations of the Thread.yield method will not yield to a thread of lower priority

Question 17

Which of the following are true statements?

- a. Thread.MAX_PRIORITY == 10
- b. Thread.MAX_PRIORITY == 5
- c. Thread.NORM_PRIORITY == 5
- d. Thread.NORM_PRIORITY == 3
- e. Thread.NORM_PRIORITY == 0
- f. Thread.MIN_PRIORITY == 1
- g. Thread.MIN_PRIORITY == 0
- h. Thread.MIN_PRIORITY == -5
- i. Thread.MIN_PRIORITY == -10

Question 18

Which of the following are true statements?

- a. A program will terminate only when all daemon threads stop running
- b. A program will terminate only when all user threads stop running
- c. A daemon thread always runs at Thread.MIN_PRIORITY
- d. A thread inherits its daemon status from the thread that created it
- e. The daemon status of a thread can be changed at any time using the Thread.setDaemon method
- f. The Thread.setDaemon method accepts one of two argument values defined by the constants Thread.DAEMON and Thread.USER

Question 19

```
class A extends Thread {
    public A(Runnable r) {super(r);}
    public void run() {System.out.print("A");}
}
class B implements Runnable {
    public void run() {System.out.print("B");}
}
class C {
    public static void main(String[] args) {
        new A(new B()).start();
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Prints: AB
- d. Prints: BA
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 20

```
class A implements Runnable {
```

```

    public void run() {System.out.print(Thread.currentThread().getName());}
}
class B implements Runnable {
    public void run() {
        new A().run();
        new Thread(new A(),"T2").run();
        new Thread(new A(),"T3").start();
    }
}
class C {
    public static void main (String[] args) {
        new Thread(new B(),"T1").start();
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: T1T1T1
- b. Prints: T1T1T2
- c. Prints: T1T2T2
- d. Prints: T1T2T3
- e. Prints: T1T1T3
- f. Prints: T1T3T3
- g. Compile-time error
- h. Run-time error
- i. None of the above

Question 21

```

class AnException extends Exception {}
class A extends Thread {
    public void run() throws AnException {
        System.out.print("A"); throw new AnException();
    }
}
class B {
    public static void main (String[] args) {
        A a = new A(); a.start(); System.out.print("B");
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Prints: AB
- d. Prints: BA
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 22

```

class A extends Thread {
    public void run() {System.out.print("A");}
}
class B {
    public static void main (String[] args) {
        A a = new A();
        a.start();
        a.start(); // 1
    }
}

```

}

What is the result of attempting to compile and run the program?

- a. The program compiles and runs without error
- b. The second attempt to start thread t1 is successful
- c. The second attempt to start thread t1 is ignored
- d. Compile-time error at marker 1
- e. An IllegalThreadStateException is thrown at run-time
- f. None of the above

No. Answer Remark

1 b c h notify notifyAll wait

2 e g sleep yield

3 d h i resume stop suspend For the purposes of the exam, you don't need to memorize the deprecated methods of the Thread class. Even though a question such as this will not be on the exam, every Java programmer should know that the deprecated methods should not be used in new programs.

4 a e h join sleep wait

5 a e h join sleep wait

6 b c f notify notifyAll wait

7 d InterruptedException The methods Object.wait, Thread.join and Thread.sleep name InterruptedException in their throws clauses.

8 b A member variable that is an object reference Primitives don't have locks; therefore, they can not be used to synchronize threads. A method local variable that is a reference to an instance that is created within the method should not be used to synchronize threads, because each thread has its own instance of the object and lock.

Synchronization on an instance that is created locally makes about as much sense as placing on your doorstep a box full of keys to the door. Each person that comes to your door would have their own copy of the key; so the lock would provide no security.

9 b e f g h i A compile-time error occurs if the expression produces a value of any primitive type If execution of the block completes normally, then the lock is released If execution of the block completes abruptly, then the lock is released A thread can hold more than one lock at a time Synchronized statements can be nested Synchronized statements with identical expressions can be nested

10 a The process of executing a synchronized method requires the thread to acquire a lock The synchronized modifier can not be applied to a class. A method that overrides a synchronized method does not have to be synchronized. If a thread invokes a synchronized instance method on an instance of class A, then the thread must acquire the lock of that instance of class A. The same is not true for synchronized static methods. A synchronized static method is synchronized on the lock for the Class object that represents the class for which the method is a member.

11 c d e f The Ready state to the Running state The Running state to the Not-Runnable state The Running state to the Ready state The Not-Runnable state to the Ready state A dead thread can not be restarted.

12 b c d f The Thread.yield method might cause the thread to move to the Ready state The same thread might continue to run after calling the Thread.yield method The Thread.yield method is a static method The Thread.sleep method causes the thread to move to the Not-

Runnable state The `Thread.yield` method is intended to cause the currently executing thread to move from the Running state to the Ready state and offer the thread scheduler an opportunity to allow a different thread to execute based on the discretion of the thread scheduler. The thread scheduler may select the same thread to run immediately, or it may allow a different thread to run. The `Thread.yield` method is a native method; so the behavior is not guaranteed to be the same on every platform. However, at least some implementations of the `yield` method will not yield to a thread that has a lower priority.

13 a `Thread.yield` method The `Thread.yield` method may cause a thread to move into the Ready state, but that state transition is not guaranteed. The JLS states that the `Thread.yield` method provides a hint to the thread scheduler, but the scheduler is free to interpret--or ignore--the hint as it sees fit. Nothing in the JLS suggests that the thread might move to the Not-Runnable state.

14 e None of the above A dead thread can not be restarted.

15 b Ready

16 b d e h The `Thread.start` method causes a new thread to get ready to run at the discretion of the thread scheduler. The `Runnable` interface declares the `run` method. The `Thread` class implements the `Runnable` interface. Some implementations of the `Thread.yield` method will not yield to a thread of lower priority. The `Object.notify` method can only be called by the thread that holds the lock of the object on which the method is invoked. Suppose that thread T1 enters a block that is synchronized on an object, A. Within the block, thread T1 holds the lock of A. Even if thread T1 calls the `notify` method immediately after entering the synchronized block, no other thread can grab the lock of object A until T1 leaves the synchronized block. For that reason, the transfer of control from thread T1 to any waiting thread can not be accelerated by moving the `notify` method to an earlier point in the synchronized block. The behavior of `Thread.yield` is platform specific. However, at least some implementations of the `yield` method will not yield to a thread that has a lower priority. Invoking the `Thread.yield` method is like offering a suggestion to the JVM to allow another thread to run. The response to the suggestion is platform specific.

17 a c f `Thread.MAX_PRIORITY == 10` `Thread.NORM_PRIORITY == 5`
`Thread.MIN_PRIORITY == 1`

18 b d A program will terminate only when all user threads stop running. A thread inherits its daemon status from the thread that created it.

19 a Prints: A If a `Runnable` target object is passed to the constructor of the `Thread` class, then the `Thread.run` method will invoke the `run` method of the `Runnable` target. In this case, the `Thread.run` method is overridden by A.`run`. The A.`run` method does nothing more than print the letter A. The invocation of the A.`start` method inside the main method results in the invocation of A.`run`, and the letter A is printed. The B.`run` method is never invoked.

20 e Prints: T1T1T3 The `Thread.currentThread` method returns a reference to the currently executing thread. When the `run` method is invoked directly it does not start a new thread; so T1 is printed twice.

21 e Compile-time error The `Runnable.run` method does not have a throws clause; so any implementation of `run` can not throw a checked exception.

22 e An IllegalThreadStateException is thrown at run-time For the purposes of the exam, invoking the start method on a thread that has already been started will generate an IllegalThreadStateException. The actual behavior of the method might be different. If the start method is invoked on a thread that is already running, then an IllegalThreadStateException will probably be thrown. However, if the thread is already dead then the second attempt to start the thread will probably be ignored, and no exception will be thrown. For the purposes of the exam, the exception is always thrown in response to the second invocation of the start method. This is a case where the exam tests your knowledge of the specification and ignores the actual behavior of the 1.4 version of the JVM.

Collection

Question 1

Which implementation of the List interface produces the slowest access to an element in the middle of the list by means of an index?

- a. Vector
- b. ArrayList
- c. LinkedList
- d. None of the above

Question 2

```
import java.util.*;
class GFC100 {
    public static void main (String args[]) {
        Object a1 = new LinkedList(), b1 = new TreeSet();
        Object c1 = new TreeMap();
        System.out.print((a1 instanceof Collection)+",");
        System.out.print((b1 instanceof Collection)+",");
        System.out.print(c1 instanceof Collection);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 3

- Each element must be unique.
- Contains no duplicate elements.
- Elements are not key/value pairs.
- Accessing an element can be almost as fast as performing a similar operation on an array.

Which of these classes provides the specified features?

- a. LinkedList
- b. TreeMap
- c. TreeSet
- d. HashMap
- e. HashSet
- f. LinkedHashMap
- g. Hashtable
- h. None of the above

Question 4

```
import java.util.*;
class GFC101 {
    public static void main (String args[]) {
        Object a1 = new HashMap(), b1 = new ArrayList();
        Object c1 = new HashSet();
        System.out.print((a1 instanceof Collection)+",");
        System.out.print((b1 instanceof Collection)+",");
        System.out.print(c1 instanceof Collection);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 5

- Entries are organized as key/value pairs.
- Duplicate entries replace old entries.

Which interface of the java.util package offers the specified behavior?

- a. List
- b. Map
- c. Set
- d. None of the above

Question 6

```
import java.util.*;
class GFC102 {
    public static void main (String args[]) {
        Object a = new HashSet();
        System.out.print((a instanceof Set)+",");
        System.out.print(a instanceof SortedSet);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true

e. None of the above

Question 7

Which implementation of the List interface provides for the fastest insertion of a new element into the middle of the list?

- a. Vector
- b. ArrayList
- c. LinkedList
- d. None of the above

Question 8

```
import java.util.*;
class GFC103 {
    public static void main (String args[]) {
        Object a1 = new TreeSet();
        System.out.print((a1 instanceof Set)+",");
        System.out.print(a1 instanceof SortedSet);
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. None of the above

Question 9

- Stores key/value pairs.
- Duplicate entries replace old entries.
- Entries are sorted using a Comparator or the Comparable interface.

Which of these classes provides the specified features?

- a. LinkedList
- b. TreeMap
- c. TreeSet
- d. HashMap
- e. HashSet
- f. Hashtable
- g. None of the above

Question 10

```
import java.util.*;
class GFC104 {
    public static void main (String args[]) {
        LinkedList a1 = new LinkedList();
        ArrayList b1 = new ArrayList();
        Vector c1 = new Vector();
        System.out.print((a1 instanceof List)+",");
        System.out.print((b1 instanceof List)+",");
        System.out.print(c1 instanceof List);
    }}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false

- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 11

- Entries are not organized as key/value pairs.
- Duplicate entries are rejected.

Which interface of the java.util package offers the specified behavior?

- a. List
- b. Map
- c. Set
- d. None of the above

Question 12

```
import java.util.*;
class GFC105 {
    public static void main (String args[]) {
        Object a = new HashSet(), b = new HashMap();
        Object c = new Hashtable();
        System.out.print((a instanceof Collection)+",");
        System.out.print((b instanceof Collection)+",");
        System.out.print(c instanceof Collection);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 13

- Elements are not key/value pairs.
- Contains no duplicate elements.
- The entries can be sorted using the Comparable interface.

Which of these classes provides the specified features?

- a. LinkedList
- b. TreeMap
- c. TreeSet
- d. HashMap
- e. HashSet
- f. Hashtable
- g. None of the above

Question 14

```

import java.util.*;
class GFC106 {
    public static void main (String args[]) {
        Object a = new HashSet(), b = new HashMap();
        Object c = new Hashtable();
        System.out.print((a instanceof Map)+",");
        System.out.print((b instanceof Map)+",");
        System.out.print(c instanceof Map);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 15

- Stores key/value pairs.
- Allows null elements, keys, and values.
- Duplicate entries replace old entries.
- Entries are not sorted.

Which of these classes provides the specified features?

- a. LinkedList
- b. TreeMap
- c. TreeSet
- d. HashMap
- e. HashSet
- f. Hashtable
- g. None of the above

Question 16

```

import java.util.*;
class GFC107 {
    public static void main (String args[]) {
        Object a = new HashSet(), b = new HashMap();
        Object c = new Hashtable();
        System.out.print((a instanceof Cloneable)+",");
        System.out.print((b instanceof Cloneable)+",");
        System.out.print(c instanceof Cloneable);
    }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false

- h. Prints: true,true,true
- i. None of the above

Question 17

- Entries are not organized as key/value pairs.
- Generally accepts duplicate elements.
- Entries may be accessed by means of an index.

Which interface of the java.util package offers the specified behavior?

- a. List
- b. Map
- c. Set
- d. None of the above

Question 18

```
import java.util.*;
import java.io.Serializable;
class GFC108 {
    public static void main (String args[]) {
        HashMap a = new HashMap();
        boolean b1, b2, b3;
        b1 = (a instanceof Cloneable) & (a instanceof Serializable);
        b2 = a instanceof Map;
        b3 = a instanceof Collection;
        System.out.print(b1 + "," + b2 + "," + b3);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 19

Which of the following classes allow unsynchronized read operations by multiple threads?

- a. Vector
- b. Hashtable
- c. TreeMap
- d. TreeSet
- e. HashMap
- f. HashSet

Question 20

- Entries are organized as key/value pairs.
- Duplicate entries replace old entries.
- Entries are sorted using a Comparator or the Comparable interface.

Which interface of the java.util package offers the specified behavior?

- a. List

- b. Map
- c. Set
- d. SortedSet
- e. SortedMap
- f. None of the above

Question 21

```
import java.util.*;
class GFC110 {
    public static void main (String[] args) {
        Object m = new LinkedHashMap();
        System.out.print((m instanceof Collection)+",");
        System.out.print((m instanceof Map)+",");
        System.out.print(m instanceof List);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 22

```
import java.util.*;
class GFC109 {
    public static void main (String[] args) {
        Object v = new Vector();
        System.out.print((v instanceof Collections)+",");
        System.out.print((v instanceof Arrays)+",");
        System.out.print(v instanceof List);
    }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

No. Answer Remark

1 c LinkedList ArrayList and Vector both use an array to store the elements of the list; so access to any element using an index

is very fast. A `LinkedList` is implemented using a doubly linked list; so access to an element requires the list to be traversed using the links.

2 g Prints: true,true,false The List and Set interfaces extend the Collection interface; so both List and Set could be cast to type Collection without generating a `ClassCastException`. Therefore, the first two of the three relational expressions return true. The Map interface does not extend Collection. The reference variable `c1` refers to an instance of `TreeMap`; so the relational expression `c1 instanceof Collection` returns the value false.

3 e HashSet The elements of a Map are key/value pairs; so a Map is not a good choice. A List generally accepts duplicate elements. A Set stores a collection of unique elements. Any attempt to store a duplicate element in a Set is rejected. Adding and removing an element in a `TreeSet` involves walking the tree to determine the location of the element. A `HashSet` stores the elements in a hashtable; so elements in a `HashSet` can be accessed almost as quickly as elements in an array as long as the hash function disperses the elements properly. Although the `LinkedHashSet` is not among the answer options it could arguably satisfy the requirements. However, the put and remove methods of the `LinkedHashSet` are a little slower than the same methods of the `HashSet` due to the need to maintain the linked list through the elements of the `LinkedHashSet`.

4 d Prints: false,true,true The Map interface does not extend Collection. The reference variable `a1` refers to an instance of `HashMap`; so the relational expression `a1 instanceof Collection` returns the value false. The List and Set interfaces extend the Collection interface; so both List and Set could be cast to type Collection without generating a `ClassCastException`. Therefore, the second and third relational expressions return true.

5 b Map The List and Set interfaces do not support key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries.

6 c Prints: true,false `HashSet` implements the Set interface, but not the `SortedSet` interface.

7 c `LinkedList` `ArrayList` and `Vector` both use an array to store the elements of the list. When an element is inserted into the middle of the list the elements that follow the insertion point must be shifted to make room for the new element. The `LinkedList` is implemented using a doubly linked list; an insertion requires only the updating of the links at the point of insertion. Therefore, the `LinkedList` allows for fast insertions and deletions.

8 d Prints: true,true `TreeSet` implements the Set interface and the `SortedSet` interface.

9 b `TreeMap` The requirement to store key/value pairs is directly satisfied by a concrete implementation of the Map interface. The List and Set interfaces recognize objects, but do not recognize keys and values. `TreeMap` and `TreeSet` store elements in a sorted order based on the key, but the `TreeSet` does not support key/value pairs.

10 h Prints: true,true,true The 1.2 version of Java introduced the updated `Vector` class that implements the List interface.

11 c Set The Map interface organizes entries as key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries.

12 e Prints: true,false,false HashSet is a subclass of AbstractSet and AbstractCollection; therefore, it implements the Collection interface. HashMap and Hashtable do not implement the Collection interface. Instead, HashMap extends AbstractMap and implements the Map interface. Hashtable extends Dictionary and implements the Map interface.

13 c TreeSet The elements are not key/value pairs; so a Map is not a good choice. A List generally accepts duplicate elements. A Set stores a collection of unique objects; so any attempt to store a duplicate object is rejected. TreeSet stores elements in an order that is determined either by a Comparator or by the Comparable interface.

14 d Prints: false,true,true HashSet implements the Set interface, but not the Map interface. HashMap extends AbstractMap and implements the Map interface. Hashtable extends Dictionary and implements the Map interface.

15 d HashMap The requirement to store key/value pairs is directly satisfied by a concrete implementation of the Map interface. The List and Set interfaces recognize objects, but do not recognize keys and values. The requirement to allow null elements is not satisfied by a Hashtable. TreeMap and TreeSet store elements in a sorted order based on the key.

16 h Prints: true,true,true All three implement Cloneable.

17 a List The Map interface organizes entries as key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries. A List allows entries to be accessed using an index.

18 g Prints: true,true,false HashMap does not implement the Collection interface.

19 c d e f TreeMap TreeSet HashMap HashSet The Vector and Hashtable methods are synchronized and do not allow for simultaneous access by multiple threads. The concrete subclasses of the AbstractList, AbstractMap, and AbstractSet classes allow for unsynchronized read operations by multiple threads. Additionally, the synchronized wrapper methods of the Collections class allow for the instantiation of a Collection, List, Map, Set, SortedMap, or SortedSet with synchronized methods. If simultaneous read and write operations are necessary then a synchronized instance should be used.

20 e SortedMap The List and Set interfaces do not support key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries. A Map organizes the entries as key/value pairs. The SortedMap is similar to a Map except that the ordering of the elements is determined by a Comparator or the Comparable interface.

21 c Prints: false,true,false LinkedHashMap does not implement the Collection interface or the List interface.

22 b Prints: false,false,true The Collections class is not the same as the Collection interface. The Collections class contains a variety of methods used to work with collections. For example, Collections.shuffle is used to randomly shuffle the elements of a Collection. Similarly, the Arrays class provides utility methods for working with arrays.

SCJP MOCK QUESTIONS

SCJP 1.4 Mock Exam -1

Q1)What will be the output?

```
public class Test1{
    public static void main(String[] args){
        int arr[] = new int[3];
        for(int i = 0;i < arr.length;i++){
            System.out.println(arr[i]);
        }
    }
}
```

- A1 0 0 0
- A2 ArrayIndexOutOfBoundsException
- A3 NullPointerException
- A4 null null null

Q2)Which of the following are valid array declarations?

- A1 int arr[] = new int[];
- A2 float arr[10] = new fl
- A3 double []arr = new double[10];
- A4 None Of the Above.

Q3)What will be the output?

```
public class Test3{
    public void method(){
        System.out.println("Called");
    }
    public static void main(String[] args){
        method();
    }
}
```

- A1 "Called"
- A2 Compiler
- A3 Runtime
- A4 Nothing

Q4)What will be the output?

```
public class Test4{
    public static void method(){
        System.out.println("Called");
    }
    public static void main(String[] args){
        Test4 t4 = null;
        t4.method();
    }
}
```

- A1 "Called"
- A2 Compiler
- A3 Runtime Exception
- A4 Nothing is printed in screen

Q5)What will be the output?

```
public class Test5{
    public void Test5() {
```

```

        System.out.println("Constructor1");
    }
    public Test5() {
        System.out.println("Constructor2");
    }
    public static void main(String[] args) {
        Test5 t5 = new Test5();
    }
}

```

- A1 "Constructor1"
A2 "Constructor2"
A3 "Constructor1""Constructor2"
A4 Compiler Error

Q6)What will be the output?

```

public class Test6{
    public Test6(){
        this(4);
    }
    public Test6(byte var){
        System.out.println(var);
    }
    public static void main(String[] args){
        Test6 t6 = new Test6();
    }
}

```

- A1 4
A2 4 4
A3 Compiler Error
A4 Compiles and Runs without error

Q7)What will be the output?

```

public class Test7{
    public Test7(){}
    public Test7(Test7 ref){
        this (ref,"Hai");
    }
    public Test7(Test7 ref,String str){
        ref.Test7(str);
        System.out.println("Hi");
    }
    public void Test7(String str){
        System.out.println(str);
    }
    public static void main(String[] args){
        Test7 t = new Test7();
        Test7 t7 = new Test7(t);
    }
}

```

- A1 HI
A2 hai
A3 Hai Hi
A4 Hi Hai

Q8)Which of the following are valid Constructors?

- A1 public Test8(){}
A2 private void Test8(){}

A3 protected Test8(int k){}
A4 Test8(){}

Q9) Which of the following are valid method declarations?

A1 abstract method(){}
A2 abstrct void method(){}
A3 final void method(){}
A4 int method({})

Q10) Which of the following are valid top-level class declarations?

A1 class Test10
A2 public class Test10
A3 final class Test10
A4 abstract final class Test10

Q11) transient keyword can be used with?

A1 method
A2 variable
A3 class
A4 constructor

Q12) which of the following are valid combinations for class declaration?

A1 abstract final class Test12{}
A2 abstract static class Test12{}
A3 final static class Test12{}
A4 public final strictfp class Test12{}

Q13) which of the following are valid constructor signatures?

A1 public void className()
A2 public className()
A3 private className()
A4 static className()

Q14) Which of the following modifiers can be used with top class declaration?

A1 static
A2 privatr
A3 public
A4 final
A5 abstract

Q15) Which of the following are valid array declarations?

A1 int arr[] = new int[];
A2 int arr[][] = new int [10][10];
A3 float arr[][] = new float[][10];
A4 float arr[] = new float[10];

Q16) What will be the output of the following program?

```
public class Test1 {
    static{
        System.out.println("Static");
    }
}
```

```
        System.out.println("Instance");
    }
    public void Test1(){
        System.out.println("Constructor");
    }
    public static void main(String[] args) {
        Test1 t = null;
    }
}
```

- A1 Instance Static
- A2 Static Instance
- A3 Static
- A4 Static Instance Constructor

Q17)What will be the output of the following program?

```
class Sup{
    public Sup(String str){
        System.out.println("Super class");
    }
}

public class Test2 extends Sup{
    public Test2(){
        System.out.println("Sub class");
    }
    public static void main(String[] args) {
        Test2 t2 = new Test2();
    }
}
```

- A1 Super class,SubClass
- A2 Super class
- A3 Sub class
- A4 Compiler Error

Q18)What will be the output of the following program?

```
public class Test3 {
    public static void main(String[] args) {
        System.out.println("Main Method1");
    }
    public static void main(String args){
        System.out.println("Main Method2");
    }
}
```

- A1 Main Method1
- A2 Main Method1 Main Method2
- A3 Main Method2
- A4 Runtime Exception

Q19)What will be the output of the following program?

```
public class Test4 {
    public static void main(String args) {
        System.out.println("Sample Program");
    }
}
```

- A1 Sample Program
- A2 Compiler Error

- A3 Runtime Exception
A4 None

Q20)What will be the output of the following program?

```
class Sup1{
    public Sup1(){
        System.out.println("Hai");
    }
    private Sup1(String str){
        System.out.println(str);
    }
}

public class Test5 extends Sup1{
    private Test5(String str){
        System.out.println(str);
        super();
    }
    public static void main(String[] args) {
        Test5 t5 = new Test5("HI");
    }
}
```

- A1 Hai,Hi,Hi
A2 Hai,Hi
A3 Hi,Hi
A4 Compiler Error

Q21)Which of the following are not a wrapper class?

- A1 String
A2 Integer
A3 StringBuffer
A4 Boolean

Q22>Select the correct syntax for main method :

- A1 public void main(String args[])
A2 public static void main(String args)
A3 public static void Main(String args[])
A4 None of the Above

Q23)Which of the following are not a valid declarations?

- A1 float f = 1;
A2 float f = 1.2f;
A3 float f = 1.2;
A4 float f = (float)1.2;

Q24)String s1 = new String("hi");

```
String s2 = "hi";
System.out.println(s1 ==s2);
System.out.println(s1.equals(s2));
```

- A1 false true
A2 true false
A3 true true
A4 None of the above.

```
Q25)Integer i = new Integer(0);  
  
Float f = new Float(0);  
System.out.println(i==f);  
System.out.println(i.equals(f));
```

- A1 true false
- A2 false true
- A3 true true
- A4 Compiler error

SCJP 1.4 Mock Exam -1 Answers

1) A1 is correct

Local Array variables are initialized to their default values.

2) A3)double []arr = new double[10];

A1 is not valid because (new int[]) array size must be specified unless it is anonymous array declaration.

A2 is not valid because array declaration must not specify the size of the array.

3) A2) Compiler Error

non-static(instance) methods or variables are cannot be referenced from static context. Compiler will give the following error while compiling the above code: Test3.java:6: non-static method method() cannot be referenced from a static context method();

4) A1) "Called"

Static methods canbe called only by referance.Because its not binded with objects.So, in this case "null" value doesn't make sense."method" is called without any problem.

5) A2)"Constructor2"

Constructors cannot have return types. If its declared with return types then it is consider as methods. So, output wull be "Constructor2".

6) A3)Compiler Error

7) A3)Hai Hi

8) A1)public Test8(){}
A3)protected Test8(int k){}

9) A3)final void method(){}
A4)int method()

10) A1) class Test10
A2) public Test10
A3) final Test10

11) b)variable

12) c)final static class()
d)public final strictfp class{}

13) b)public className()
c)private className()

14) c)public
d)final
e)abstract

15) b)int arr[][] = new int [10][10];
d)float arr[] = new float[10];

16) c)Static

17) d)Compiler Error

18) a)Main Method1

19) c)Runtime Exception

20) d)Compiler Error

21) A1)String
A4)String Buffer

22) A4)None of the Above

23) A3)float f = 1.2;

24) A1) false true

25) A4)Compiler error

~~~~~

```
import java.util.*;  
public class Test1{  
    public static void main(String a[]){  
        Set s = new TreeSet();  
        s.add(new Person(20));  
        s.add(new Person(10));  
        System.out.println(s);  
    }  
}  
class Person{  
    Person(int i){}  
}  
What is the output?
```

- A1 10 20  
A2 ClassCastException  
A3 Compiler Error  
A4 Compiler Error

---

```
import java.util.*;  
public class Test2{  
    public static void main(String a[]){  
        Map s = new Hashtable();  
        s.put(null,null);  
        System.out.println(s);  
    }  
}  
What is the output?
```

- A1 [null = null]

- 
- A2 NullPointerException  
A3 null  
A4 []
- 

```
import java.util.*;  
public class Test3{  
    public static void main(String a[]){  
        Map s = new WeakHashMap(10);  
        s.put(null,null);  
Q3        System.out.println(s);  
    }  
}
```

What is the output?

- A1 [null=null]  
A2 NullPointerException  
A3 null  
A4 []
- 

```
import java.util.*;  
public class Test4{  
    public static void main(String a[]){  
        Map s = new LinkedHashMap();  
        s.put("1","one");  
        s.put("3","three");  
Q4        s.put("2","two");  
        System.out.println(s);  
    }  
}
```

What is the output?

- A1 [1=one,3=three,2=two]  
A2 NullPointerException  
A3 [1=one,2=two,3=three]  
A4 []
- 

```
import java.util.*;  
public class Test5{  
    public static void main(String a[]){  
        Map s = new HashMap();  
        s.put("1","one");  
Q5        s.put("3","three");  
        s.put("2","two");  
        System.out.println(s);  
    }  
}
```

What is the output?

- A1 [1=one,3=three,2=two]  
A2 [3=three,2=two,1=one]  
A3 cannot predict the order  
A4 []
- 

```
public class Test6{  
    public static void method(float f){  
        System.out.println("Float");  
    }  
    public static void method(double f){  
        System.out.println("Double");  
    }  
    public static void main(String a[]){  
        float f1 = 2.0f;  
        float f2 = 2.0f;
```

```
        method(1.0);
        method(1.0f);
        method(1.0f*2.0f);
        method(1.0f*2.0);
        method(f1*f2);
    }
}
```

What is the output?

- A1 Double Double Double Double
  - A2 Float Float Float Float
  - A3 Double Float Double Double
  - A4 Double Double Double Double
- 

```
public class Test7{
    public static void method(byte b){
        System.out.println("Byte");
    }
    public static void method(int i){
        System.out.println("Int");
    }
    public static void main(String a[]){
Q7        byte b = 1;
        method(1);
        method(128);
        method((byte)128);
        method(b);
    }
}
```

What is the output?

- A1 Byte Int Int Byte
  - A2 Byte Int Int Byte
  - A3 Byte Int Byte Byte
  - A4 Int Int Byte Byte
- 

```
public class Test8{
    public static void main(String a[]){
        byte b = 1;
        char c = 2;
        short s = 3;
        int i = 4;

Q8        c = b; // 1
        s = b; // 2
        i = b; // 3
        s = c * b; // 4
    }
}
```

Which of the following are correct?

- A1 Error at mark 1
  - A2 Error at mark 2
  - A3 Error at mark 3
  - A4 Error at mark 4
- 

```
public class Test9{
    public static void main(String a[]){
        final byte b = 1;
        char c = 2;
        short s = 3;
        int i = 4;

Q9        c = b; // 1
    }
```

```
s = b; // 2
i = b; //3
s = c * b; //4
}
}
Which of the following are correct?
```

- 
- A1 Error at mark 1
  - A2 Error at mark 2
  - A3 Error at mark 3
  - A4 Error at mark 4

---

```
public class Test10{
    public static void main(String a[]){
        String s1 = "Sun";
        String s2 = "Java";
        s1.concat(s2);
        System.out.println(s1);
    }
}
```

Q10 What is output?

- A1 Sun
- A2 Java
- A3 SunJava
- A4 JavaSun

---

```
public class Test11{
    public static void main(String a[]){
        Integer i1 = new Integer(127);
        Integer i2 = new Integer(127);
        Long l = new Long(127);
        System.out.println(i1 == i2);
        System.out.println(i1.equals(i2));
        System.out.println(i1.equals(l));
    }
}
```

Q11 What is output?

- A1 false true true
- A2 true true true
- A3 false true false
- A4 Compiler Error

---

```
public class Test12{
    public static void main(String a[]){
        byte b = 100;
        Byte b1= new Byte(100);
        Byte b2 = new Byte(b);
        System.out.println(b1 == b2);
        System.out.println(b1.equals(b2));
    }
}
```

Q12 What is output?

- A1 false true
- A2 true false

A3 true true

A4 Compiler Error

---

```
public class Test13{
    public static void method(String s){
        System.out.println("String Version");
    }
    public static void method(StringBuffer sb){
        System.out.println("String Buffer Version");
    }
Q13     public static void main(String a[]){
        method(null);
    }
}
```

What is output?

A1 String Version

A2 String Buffer Version

A3 Runtime Exception

A4 Compiler Error

---

```
public class Test14{
    static String s ="Instance";
    public static void method(String s){
        s+="Add";
    }
    public static void main(String a[]){
        Test14 t = new Test14();
        s = "New Instance";
        String s = "Local";
        method(s);
        System.out.println(s);
        System.out.println(t.s);
    }
}
```

What is output?

A1 Local Instance

A2 Local New Instance

A3 Loca Add New Instance

A4 Compiler Error

---

```
public class Test15{
    public static void method(StringBuffer sb) {
        sb.append(" Added");
        sb = new StringBuffer("Hai");
    }
    public static void main(String a[]){
        StringBuffer sb = new StringBuffer("String Buffer");
Q15      method(sb);
        System.out.println(sb);
    }
}
```

What is output?

A1 String Buffer

A2 String Buffer Added

A3 Hai

A4 Compiler Error

---

```
public class Test16{
    public static void method(StringBuffer sb) {
        sb = new StringBuffer("Hai");
        sb.append(" Added");
    }
    public static void main(String a[]){
        StringBuffer sb = new StringBuffer("String Buffer");
        Q16      method(sb);
        System.out.println(sb);
    }
}
What is output?
```

- 
- A1 String Buffer
  - A2 String Buffer Added
  - A3 Hai
  - A4 Compiler Error

```
import java.util.*;
public class Test17{
    public static void main(String a[]){
        Map m = new Hashtable(10,0.75f);
        System.out.println(m.size());
    }
}
What is output?
```

- 
- A1
  - A2 10
  - A3 7
  - A4 cOMPILE R eRROR

---

Q18 What is the default capacity of java.util.Hashtable?

- A1 10
  - A2 16
  - A3 11
  - A4 20
- 

Q19 What is the default capacity of java.util.HashMap?

- A1 10
  - A2 16
  - A3 11
  - A4 20
- 

Q20 Which of the following classes has synchronized methods?

- A1 ArrayList
  - A2 Vector
  - A3 HashTable
  - A4 WeakHashMap
- 

```
public class Test21{
    public static void main(String a[]){
        String s1 = new String("Hai");
        String s2 = "Hai";
        String s3 = new String("Hai").intern();
        System.out.println(s1 == s2);
        System.out.println(s1 == s3);
    }
}
What is output?
```

```
        System.out.println(s2 == s3);
    }
}
What is output?
```

- A1 false false true
  - A2 true false true
  - A3 false false false
  - A4 false true true
- 

```
public class Test22{
    public static void main(String a[]){
        String s1 = "SunMicroSystems";
        System.out.println(s1.substring(0));
        System.out.println(s1.substring(1, 4));
        System.out.println(s1.substring(8));
Q22     }
}
What is output?
```

- A1 SunMicrosystems sun oSystem
  - A2 SunMicrosystems unM Systems
  - A3 StringIndexOutOfBoundsException
  - A4 None Of the above
- 

```
public class Test23{
    public static void main(String a[]){
        String s1 = "Sun";
Q23     System.out.println(s1.substring(5));
    }
}
What is output?
```

- A1 -1
  - A2 0
  - A3 StringIndexOutOfBoundsException
  - A4 ArrayIndexOutOfBoundsException
- 

Q24 Which of the following are static methods in java.lang.String class?

- A1 valueOf
  - A2 length
  - A3 indexOf
  - A4 All the above.
- 

```
public class Test25{
    public static void main(String a[]){
        StringBuffer sb = new StringBuffer(8);
        sb.append("TestString");
        System.out.println(sb.capacity());
        System.out.println(sb.length());
Q25     }
}
```

What is output?

- A1 8 10
- A2 10 10
- A3 18 10
- A4 18 18

## Answers

1 ClassCastException  
2 NullPointerException  
3 [null = null]  
4 [1=one,3=three,2=two]  
5 cannot predict the order.  
6 Double Float Double Float  
7 Int Int Byte Byte  
8 Error at mark 1  
9 Error at mark 4  
10 Sun  
11 false true false  
12 Compiler Error  
13 Compiler Error  
14 Local New Instance  
15 String Buffer Added  
16 String Buffer  
17 0  
18 11  
19 16  
20 Vector  
HashTable  
21 false false true  
22 SunMicrosystems unM Systems  
23 StringIndexOutOfBoundsException  
24 valueOf  
25 18 10

---

## Mock Exam - 3

```
public class Test1{
    public static void main(String args[]){
        System.out.println(method());
    }
    public static int method(){
        try{
            return 1;
        }
        Q1      catch(Exception e){
            return 2;
        }
        finally{
            return 3;
        }
    }
}
```

What will be the output?

- A1 1  
A2 2  
A3 3  
A4 0
- 

Q2 public class Test2{
 public static void main(String args[]){

```
        System.out.println(method());
    }
    public static int method(){
        try{
            throw new Exception();
            return 1;
        }
        catch(Exception e){
            return 2;
        }
        finally{
            return 3;
        }
    }
}
```

What will be the output?

- A1 1
  - A2 2
  - A3 3
  - A4 4
  - A4 Compiler error
- 

```
public class Test3{
    public static void main(String args[]){
        System.out.println(method());
    }
    public static int method(){
        try{
            throw new Exception();
        }
        catch(Exception e){
            throw new Exception();
        }
        finally{
            return 3;
        }
    }
}
```

What will be the output?

- A1 3
  - A2 0
  - A3 Runtime Exception
  - A4 Compiler error
- 

```
public class Test4{
    public static void main(String args[]){
        System.out.println(method());
    }
    public static int method(){
        return;
    }
}
```

What will be the output?

- A1 null
  - A2 0
  - A3 Runtime exception
  - A4 Compiler error
- 

```
import java.io.IOException;
public class Test5{
    public static void main(String args[]){
        try{
            throw new IOException();
        }
    }
}
```

```
        }
    catch(Exception e){
        System.out.println("Exception");
    }
    catch(IOException e){
        System.out.println("IOException");
    }
}
}
```

What will be the output?

- A1 Exception
  - A2 IOException
  - A3 Exception IOException
  - A4 Compilers error
- 

```
public class Test6{
    public static void main(String args[]) throws Exception{
        try{
            throw new Exception();
        }
        finally{
            System.out.println("No Error");
        }
    }
}
```

What will be the output?

- A1 No Error followed by java.lang.Exception
  - A2 java.lang.Exception followed by No Error
  - A3 No Error
  - A4 Compiler Error
- 

```
public class Test7{
    public static void main(String args[]) throws Exception{
        Test7 t = new Test7();
        t.method();
        System.out.println("Print");
    }
    public void method()throws Exception{
        throw new Exception();
    }
}
```

What will be the output?

- A1 Print
  - A2 Exception thrown at runtime
  - A3 no output
  - A4 Compiler Error
- 

```
public class Test8{
    public static void main(String args[]) throws Exception{
        Test8 t = new Test8();
        t.method();
        System.out.println("Print");
    }
    public void method(){
        try{
            throw new Exception();
        }catch(Exception e){}
    }
}
```

What will be the output?

- A1 Print
- A2 Exception thrown at runtime
- A3 no output
- A4 Compiler Error

---

```
public class Test9 extends A{
    public static void main(String args[]) throws Exception{
        Test9 t = new Test9();
    }
}
Q9 class A{
    A() throws Exception{
        System.out.println("A Class");
    }
}
What will be the output?
```

- A1 A Class
  - A2 Runtimxception
  - A3 no output
  - A4 Compiler Error
- 

```
public class Test10 extends A{
    Test10()throws Exception{
        System.out.println("Test10 Class");
    }
    public static void main(String args[]) throws Exception{
        Test10 t = new Test10();
    }
}
Q10 class A{
    A() throws Exception{
        System.out.println("A Class");
    }
}
What will be the output?
```

- A1 A Class Test10 Class
  - A2 Runtimxception
  - A3 no output
  - A4 Compiler Error
- 

```
public class Test11 extends A{
    Test11()throws Exception{
        System.out.println("Test10 Class");
    }
    Test11(int i){}
    public static void main(String args[]) throws Exception{
        Test11 t = new Test11();
    }
}
Q11 class A{
    A() throws Exception{
        System.out.println("A Class");
    }
}
What will be the output?
```

- A1 A Class Test10 Class
  - A2 Runtimxception
  - A3 no output
  - A4 Compiler Error
- 

```
import java.io.IOException;
public class Test12 extends A{
    public void method() throws Exception{
        System.out.println("Subclass");
    }
}
Q12 public static void main(String args[]) throws Exception{
    A a = new A();
    a.method();
    a = new Test12();
    a.method();
}
```

```

        }
    }
class A{
    public void method() throws IOException{
        System.out.println("Superclass");
    }
}

```

What will be the output?

- A1 Subclass Superclass
  - A2 Runtimxception
  - A3 Superclass Superclass
  - A4 Compiler Error
- 

Q13 What are the legal arguments types for switch?

- A1 int
  - A2 byte
  - A3 char
  - A4 All the above.
- 

Q14 Which of the following are valif if constructs?

- A1 if(2>3){}
  - A2 if(false){}
  - A3 if(false){}
  - A4 if(true)
- 

```

public class Test15{
    public static void main(String args[]) throws Exception{
        for (int i = 0;i < 3;i++){
            for (int j = 0;j < 3;j++){
                System.out.print(i);
                System.out.print(j+",");
                break;
            }
        }
    }
}

```

What will be the output?

- A1 00,
  - A2 00,10,20,
  - A3 000102
  - A4 None of the above
- 

```

public class Test16 extends A{
    Test16(){
        System.out.println("Sub");
    }
    public static void main(String args[]){
        Test16 t = new test16();
    }
.....

```

#### Mock Exam - 4

```

public class Test1{
    static void method(Object obj){
        System.out.println("Object");
    }
    static void method(String str){
        System.out.println("String");
    }
    public static void main(String args[]){
        method(null);
    }
}

```

```
}
```

What will be the output?

- A1 String
  - A2 Object
  - A3 null
  - A4 Compiler Error
- 

```
public class Test2{  
    static void method(StringBuffer obj){  
        System.out.println("StringBuffer");  
    }  
    static void method(String str){  
        System.out.println("String");  
    }  
Q2     public static void main(String args[]){  
        method(null);  
    }  
}
```

What will be the output?

- A1 String
  - A2 Object
  - A3 null
  - A4 Compiler Error
- 

```
class Test{}  
public class Test3{  
    static void method(Object obj){  
        System.out.println("StringBuffer");  
    }  
    static void method(String str){  
        System.out.println("String");  
    }  
Q3     static void method(Test t){  
        System.out.println("Test");  
    }  
    public static void main(String args[]){  
        method(null);  
    }  
}
```

What will be the output?

- A1 String
  - A2 Object
  - A3 Test
  - A4 Compiler Error
- 

```
public class Test4{  
    public static void main(String args[]){  
        I i1 = new A();  
        I i2 = new B();  
        A a = new A();  
        System.out.println(i1 instanceof I);  
        System.out.println(i2 instanceof B);  
Q4      System.out.println(a instanceof I);  
    }  
}  
interface I{}  
class A implements I{}  
class B implements I{}  
What will be the output?
```

- A1 true true true

A2 true false true

A3 true false false

A4 Compiler Error

---

```
public class Test5{
    public static void main(String args[]){
        System.out.println(I.k);
    }
}
interface I{
    int k;
}
What will be the output?
```

A1 true true true

A2 true false true

A3 true false false

A4 Compiler Error

---

```
public class Test6 implements I{
    int k = 1;
    public static void main(String args[]){
        System.out.println(k);
    }
}
interface I{
    int k = 0;
}
What will be the output?
```

A1 0

A2 1

A3 null

A4 Compiler Error

---

```
public class Test7 implements I{
    int k = 1;
    static{
        k = k * 2;
    }
    {
        k = k * 2;
    }
}
public static void main(String args[]){
    Test7 t1 = new Test7();
    Test7 t2 = new Test7();
    System.out.println(t1.k);
    System.out.println(t2.k);
    System.out.println(k);
}
}
What will be the output?
```

A1 0

A2 1

A3 null

A4 Compiler Error

---

```
public class Test8{
    static int k = 1;
    static{
        k = k * 2;
    }
    {
        k = k * 2;
    }
}
public static void main(String args[]){
```

```
        System.out.println(k);
    }
}
```

What will be the output?

- A1 1
  - A2 2
  - A3 4
  - A4 Compiler Error
- 

```
public class Test9{
    static int k = 1;

    {
        k = k * 2;
    }
}

Q9 public static void main(String args[]){
    System.out.println(k);
}
```

What will be the output?

- A1 1
  - A2 2
  - A3 4
  - A4 Compiler Error
- 

```
public class Test10{
    final static int k;
    static{
        k = 0;
    }
}

Q10 public static void main(String args[]){
    System.out.println(k);
}
```

What will be the output?

- A1 0
- A2 1
- A3 null
- A4 Compiler Error

#### Answers

- 1 String
- 2 Compiler Error
- 3 Compiler Error
- 4 true true true
- 5 true true true
- 6 Compiler Error
- 7 Compiler Error
- 8 2
- 9 1
- 10 0



Q1 Which of the following are the correct form of documentation

**comments?**

- A1 //some text here
  - A2 /\*some text here\*/
  - A3 /\*\*some text here\*/
  - A4 all the above
- 

**Q2 State the correct formula for minimum/maximum values for integer primitives where no\_of\_bits is the size of the type in bits.**

- A1  $2^{(no\_of\_bits-1)} / 2^{(no\_of\_bits-1)+1}$
  - A2  $2^{(no\_of\_bits+1)} / 2^{(no\_of\_bits+1)+1}$
  - A3  $2^{(no\_of\_bits-1)} / 2^{(no\_of\_bits-1)-1}$
  - A4 all the above
- 

**Q3 Which of the following initializes boolean primitive?**

- A1 Boolean flag=true;
  - A2 boolean flag=true;
  - A3 boolean flag=TRUE;
  - A4 Boolean flag=TRUE;
- 

**Q4 which of the following is the correct way to define a class that will be in the default package**

package default;

A1 import java.util.\*;

- import java.util.\*;
- A2 package default;
  - A3 import java.util.\*;
  - A4 all the above
- 

**Q5 Which of the following main method in a java application is correct?**

- A1 public static void main(String[] args)
  - A2 public void main(String args[]])
  - A3 public static void main (string[] args)
  - A4 final public static void main (String[] args)
  - A5 static public void main(String x[]])
  - A6 static void main (string[] args)
  - A7 a and e only.
  - A8 g and d
- 

**Q6 Which of the following is default integer primitive**

- A1 short
  - A2 int
  - A3 byte
  - A4 char
  - A5 long
-

**Q7 Which of the following is not a reserved word in java**

- A1 import
  - A2 finally
  - A3 friend
  - A4 goto
- 

**When writing a utility class, someclass, which extends mainclass**

**Q8 class and will be used by several other classes in a large project.**  
**These other classes will be in different packages.Pick the correct class declaration**

- A1 class someclass extends mainclass
  - A2 protected class someclass extends mainclass
  - A3 public class someclass extends mainclass
  - A4 none
- 

**Q9 Which of the following variable names are invalid?**

- A1 example
  - A2 2sumup
  - A3 its4u
  - A4 \$money
- 

**Take a look at the following code:**

```
public static void main (String[] args){  
    System.out.println(args[1]);  
}
```

**Q10 }**

**The above is compiled and then executed by the following command line.**

**java test one two three four**

**choose the correct output**

- A1 one
- A2 two
- A3 three
- A4 four
- A5 none.

**Answers**

- 1 c  
c Substitute no\_of\_bits = ( 8 for byte , 16 for short, 16 for char, 32 for int, 64 for long, 32 for float, 64 for double).We get  $(2^7) / (2^7)$  - 1 for int and so on for other types
- 2 b primitive boolean keyword is 'boolean' and boolean can be only 'true' or 'false'
- 3
- 4 c. there is nothing like explicit declaration for default package. The class is added to default package if there is no package statement.
- 5 h Valid declaration for the main() method must be public and static, have void as return type and take a single array of String objects as

arguments. The order of public and static keywords is irrelevant.  
Also declaring the method final does not effect the method's potential to be used as a main() method.

- 6 b  
7 c . There are no friend functions as in C++.  
8 c  
9 b  
10 b Array index start from 0. So somearray[0] points to the first element in the array.

---

Mock 6

- 1. Please select signed integrals**
  - A. char, byte, and short
  - B. byte, short, int, and long
  - C. char, short, and long
- 2. Java characters are ...**
  - A. ASCII code
  - B. Binary code
  - C. Unicode
  - D. ANSI code
- 3. Please select the size of an int type**
  - A. 32 bytes
  - B. 16 bits
  - C. 32 bits
  - D. 16 bytes
- 4. Select default value of boolean type**
  - A. true
  - B. false
  - C. 0
  - D. 1
- 5. Consider the following line of code:**  
`char x[] = new char[10];`  
**After execution, what is value of x[0]?**
  - A. 0
  - B. '\u0000'
  - C. null
- 6. A package statement MUST exist in all classes**
  - A. True
  - B. False
- 7. Please choose invalid identifiers**
  - A. temp
  - B. BIGint
  - C. 25\_length
  - D. !length
- 8. Please select floating point types**
  - A. byte

- B. int
  - C. double
  - D. short
  - E. long
  - F. float
9. All operands are evaluated left to right
- A. true
  - B. false
10. Consider the following line of code:
- ```
byte x=64;  
byte y=5;  
byte z= (byte)(x*y);
```
- After execution what is value of z?
- A. 320
  - B. 0
  - C. 645
  - D. 64
11. Consider the following line of code:
- ```
int x=7;  
int y=4;  
int z=7/4;
```
- After execution what is value of z?
- A. 1.75
  - B. 0
  - C. 1
  - D. 2
12. Please select the true statement for ! operator
- A. The ! operator inverts the value of a boolean expression
  - B. The ! operator inverts the bit pattern of an integral expression.
  - C. Both A and B
  - D. None of the above
13. Please select arithmetic operations which can result in ArithmeticException
- A. Multiplication: \*
  - B. Division: /
  - C. Modulo: %
  - D. Addition: +
  - E. Subtraction: -
14. Please select operators which perform bit shifts of the binary representation of the integral types
- A. <<
  - B. >>
  - C. >>>
  - D. ~
  - E. &
  - F. ^
  - G. |
15. A protected method may be overridden by ...
- A. A private method
  - B. A protected method
  - C. A public method
  - D. All of the above

16. **A public method may not be overridden by ...**
- A. A private method
  - B. A protected method
  - C. A public method
  - D. All of the above
17. **The private modifier can be applied to ...**
- A. A variable
  - B. A method
  - C. A top level class
  - D. All of the above
18. **The abstract modifier can NOT be applied to ...**
- A. A variable
  - B. A method
  - C. A class
  - D. A constructor
19. **A class variable is declared using following modifiers**
- A. protected
  - B. private
  - C. public
  - D. static
20. **An unary operator operates on a single value**
- A. True
  - B. False
21. **The following types of loop are guaranteed to be executed at least once**
- A. The do loop
  - B. The while loop
  - C. The for loop
  - D. All of the above
22. **The switch() construct is used to make a choice based upon**  
...  
  - A. A char value
  - B. An int value
  - C. A String value
  - D. None of the above
23. **The circumstances that can prevent execution of the code in a finally block are**
- A. The death of the thread
  - B. The use of System.exit()
  - C. It is always guaranteed to be executed.
24. **Select correct statement(s)**
- A. The continue statement abandons the loop altogether
  - B. The break statement causes the current iteration of the loop to be abandoned.
  - C. The break statement abandons the loop altogether
25. **How can you declare a overloaded method?**
- A. Reusing the method name with different arguments and same return type
  - B. Reusing the method name with different arguments and different return type
  - C. Reusing the name with identical arguments and return type

- D. None of the above
26. **How can you declare a overriding method?**
- A. Using the same method name with identical arguments and return type
  - B. Using the same method name with identical arguments and different return type
  - C. Using the same method name with different arguments and same return type
  - D. All of the above
27. **When a default constructor is provided by the compiler?**
- A. If you define no constructor at all in a class
  - B. When you define at least one constructor
  - C. It is always provided whether you define a constructor or not
  - D. It is never provided
28. **A static inner class can access ...**
- A. instance variables of the enclosing class
  - B. static variables of the enclosing class
  - C. Both A and B
  - D. None of the above
29. **An inner class created inside a method can access**
- A. Any local variables of a method that contain an inner class.
  - B. Any instance variables of the enclosing class
  - C. Any final variables of the enclosing class or a method that contain an inner class.
  - D. None of the above
30. **Please select true statement(s) for an anonymous inner class**
- A. An anonymous class can not extend to a superclass
  - B. An anonymous class can not implement an interface
  - C. An anonymous class can extend to a superclass and implement an interface both.
  - D. An anonymous class can extend to a superclass or implement an interface
31. **Please select true statement(s) for a member inner class**
- A. An inner class in class scope can have any accessibility of the top level class, including private.
  - B. An Inner class defined local to a block may be static
  - C. An anonymous inner class can not declare a constructor.
  - D. An inner class can not have same name as enclosing class.
  - E. All of the above
32. **Please select invalid statement(s) for a thread**
- A. You can restart a dead thread
  - B. You can't call the methods of a dead thread
  - C. Both of the above
  - D. None of the above
33. **Select correct statements for a java.lang.String**
- A. Strings are sequences of 16 bit Unicode characters.
  - B. Strings are immutable.

- C. Both of the above
  - D. None of the above
34. **Select correct statements for == operator.**
- A. It compare object reference of two objects
  - B. It compare the value of two objects
  - C. Both of the above
  - D. None of the above
35. **Please select collection(s) that do not reject duplicates**
- A. java.util.List
  - B. java.util.Set
  - C. java.util.Map
  - D. All of the above
36. **Please select a default layout manager of the java.awt.Panel**
- A. java.awtFlowLayout with left justified
  - B. java.awtFlowLayout with center justified
  - C. java.awtFlowLayout with right justified
  - D. None of the above
37. **Please select a default layout manager for the java.awt.Frame**
- A. java.awt.FlowLayout
  - B. java.awt.BorderLayout
  - C. java.awt.GridBagLayout
  - D. None of the above
38. **Please select true statement for prefix operator(++x/--x).**
- A. The prefix operator(++x/--x) evaluates the value of the operand after increment/decrement operation.
  - B. The prefix operator(++x/--x) evaluates the value of the operand before increment/decrement operation.
  - C. Both A and B
  - D. None of the above
39. **Please select true statement(s) for shift operators.**
- A. >>> operator always produces a positive number.
  - B. >> always produces a positive number for a positive number.
  - C. >> always produces a negative number for a negative number.
  - D. None of the above
40. **Please select true statement(s) for shift operators.**
- A. Shifting is not allowed in long integral type
  - B. Shifting is not allowed in float integral type
  - C. Shifting is not allowed in double integral type
  - D. Shifting is not allowed in int integral type.
41. **Please identify correct assignment for boolean type.**
- A. boolean javaExam=true;
  - B. boolean javaExam=True;
  - C. boolean javaExam=1;
  - D. All of the above
42. **Bitwise operator can be applied to ...**
- A. Integral types
  - B. Float types
  - C. Both of the above
  - D. None of the above
43. **instanceof operator can be used with ...**
- A. interfaces
  - B. Arrays
  - C. Classes

- D. All of the above
44. **Please select true statement(s)**
- A. The equals method compares content of two objects
  - B. The == operator tests reference of two objects
  - C. The equals method compares reference of two objects.
  - D. The == operator compares content of two objects
45. **Please identify invalid use of comparison operator for the following code:**
- ```
String s=new String("S");
String t=new String("T");
int x=83;
int y=84;
```
- A. s == t
  - B. x!=y
  - C. x==s
  - D. s!=t
46. **Please select true statement(s) for instanceof operator:**
- A. The instanceof operator can be applied to object reference
  - B. The instanceof operator can be applied to an array.
  - C. Both of the above
  - D. None of the above
47. **please select true statement(s) for static modifier.**
- A. A static method can access non-static variables of the class.
  - B. A static method can call non-static methods.
  - C. A static method can be overridden by non-static method.
  - D. None of the above
48. **Please, select true statement(s) for thread.**
- A. Invoking a constructor of a Thread registers the thread with thread scheduler.
  - B. Calling a start() method of thread registers the thread with thread scheduler.
  - C. Calling a run() method of thread registers the thread with thread scheduler.
  - D. All of the above.
49. **Invoking yield() on a running thread cause following effect(s):**
- A. A running thread will be placed in suspended state.
  - B. A running thread will be placed in sleeping state.
  - C. A running thread will be placed in ready state.
  - D. Neither of the above.
50. **The default priority of the thread is**
- A. 0
  - B. 1
  - C. 5
  - D. 10
51. **Which of the following methods are NOT static of Thread class?**
- A. start()
  - B. sleep(long l)
  - C. run()
  - D. yield()
52. **Select true statement(s) about an inner class declared inside a method, also known as local inner class**
- A. The local inner class can access any local variables

- declared in the method and the parameters passed to the method.
- B. The local inner class can access only public variables declared in enclosing class
  - C. The local inner class can access public, private, and protected variables declared in enclosing class.
  - D. The local inner class can access only final declared variables in the method.

**53. How can you prevent class JavaExam from being extended?**

declare class static JavaExam

declare class synchronized JavaExam

declare class final JavaExam

None of the above

**54. Assume that following methods are declared in one class.**

- 1. public void addElement(Object javaExam)
- 2. public void addElement( Object [] javaExam)
- 3. public Object addElement ( int index, Object javaExam)

**Please select true statement(s) for above methods.**

- . All methods are example of overloading method

All methods are example of overriding method

Method # 1 and method # 2 are example of overloading method, whereas method # 3 is an example of overriding method.

None of the above

**55. When can't you override a method?**

- . when method is declared abstract

When method is declared final

when method is declared private

when method is declared static

**56. Please select invalid declaration of the top level class**

- . public abstract final class JavaExam

public final class JavaExam implement Runnable

protected static class JavaExam

public class JavaExam extend Thread

**57. Please select invalid types for a switch construct**

- . float
  - long
  - String
- All of the above

**58. Please select invalid java keywords**

- . include
- ifdef
- sizeof
- goto

**59. Which of the following statements are NOT true about Java garbage collection Thread?**

- . The garbage collection thread has a low priority
- The garbage collection thread can be invoked at any time

A call to System.gc() is guaranteed to invoke garbage collection thread immediately

A call to Runtime.getRuntime().gc() is guaranteed to invoke garbage collection thread immediately

**60. An inner class can not declare\_\_\_\_\_ variables.**

- . static
- protected
- final
- transient

**61. Which of the following types can be added to java.util.Vector?**

- . reference
- null
- int

All of the above

62. Please select a true statement about `delete()` method of `java.io.File`.

- . It can delete a file

It can delete an empty directory

Both of the above

Neither of the above

63. The `continue` statement causes the current iteration of the loop to be skipped and the next iteration starts.

- . True

False

64. The return type of constructor is `void`.

- . True

False

65. 'null' is valid java type.

- . True

False

66. Invoking a constructor of `java.io.File` class creates a file on the file system.

True

False

67. Select true statement(s) about native modifier.

native modifier can be applied to a class

native modifier can be applied to a method

native modifier can be applied to an instance variable

native variable can be applied to a class variable

68. What method(s) needed to be declared by a class implementing Runnable interface?

public void start()

public void run()

public boolean start()

public boolean run()

**69. The priority of a newly created thread is always Thread.NORM\_PRIORITY.**

True

False

**70. What methods are declared in java.lang.Thread class?**

public static void sleep(long millis, int nanos)

public static native void sleep(long millis, int nanos)

public static native void sleep(long millis)

public static void sleep(long millis)

**71. A yield method of the Thread class suspends the operation of the current thread.**

True

False

**72. What methods are NOT synchronized in java.util.Vector class?**

size()

add(int index, Object element)

capacity()

get(int index)

**73. Please select unchecked exception(s)?**

java.lang.NullPointerException

java.lang.ClassNotFoundException

java.lang.ClassCastException

java.awt.AWTException

**74. Which of the following declarations are valid to throw an exception?**

throw new java.lang.Exception();

throws new java.lang.Exception();

Both of the above

None of the above

**75. Which of the following classes are immutable?**

java.lang.String

java.lang.StringBuffer

java.util.Vector

java.lang.Integer

**76. Which of the following classes store and retrieve values based on a key?**

java.util.Hashtable

java.util.Vector

java.util.LinkedList

java.util.HashMap

**77. Which of the following classes can store null value for a key?**

java.util.Hashtable

java.util.HashMap

java.util.Properties

All of the above

**78. java.util.Vector uses synchronized methods to store and retrieve values.**

True

False

**79. `java.util.Hashtable` uses synchronized methods to store and retrieve values.**

True

False

**80. `java.util.HashMap` uses synchronized methods to store and retrieve values.**

True

False

**81. Which of the following collections maintain object references in the order they were added?**

`java.util.Vector`

`java.util.Hashtable`

`java.util.HashMap`

`java.util.ArrayList`

**82. `java.util.Hashtable` implements which of the following interfaces?**

`java.util.Dictionary`

`java.util.Map`

`java.util.HashMap`

`java.util.Hashmap`

**Answers:**

1. B

2. C

3. C

4. B

5. B

6. B

7. C and D

8. C and F

9. A

10. D. **Comment:** A byte value can represent a range of -128 to +127. Arithmetically answer is 320, but when you store this result to a byte variable you will get a value 64 since result is out of the range (-128 to +127).

11. C

12. A

13. B and C are correct

14. A, B, and C

15. B and C
16. A and B
17. A and B
18. A and D
19. D
20. A. **Comment:** + and - operator can take two values
21. A
22. A and B
23. A and B
24. C
25. A and B
26. A
27. A
28. B
29. C
30. D
31. A , C, and D
32. C. Thanks Chris Pereira for your feedback in our discussion area.
33. C
34. A
35. A
36. B
37. B
38. A
39. B, and C.
40. B and C
41. A
42. A
43. D
44. A and B
45. C
46. C
47. D
48. B
49. C
50. C
51. A and C
52. C and D
53. C
54. A
55. B
56. A, B, C, and D
57. D
58. A, B, and C
59. B, C, and D
60. A
61. A and B
62. C
63. A
64. B
65. A
66. B
67. B
68. B
69. B **Comment:** A newly created thread inherits the priority of the Thread that creates it.
70. A and C
71. A
72. A, B, and C
73. A and C

74. A  
75. A and D  
76. A and D  
77. B  
78. A  
79. A  
80. B  
81. A and D  
82. B

~~~~~  
MOCK 7

1. **Consider the following line of code:**

```
public class Test{  
    public void main(){  
        System.out.println("Hi");  
    }  
    public static void main (String [] args)  
    {  
        Test t=new Test();  
        t.main();  
    }  
}
```

What will be happen if you compile and run above program?

- A. It will not compile
- B. It will not run
- C. It will compile but will not run
- D. It will output "Hi"

2. **After execution of the code fragment below, what are the value of the variables x1, y1, and z1?**

```
int x=10; int y=10; int z=10; int x1, y1, z1;  
x1=++y;  
y1=z++;  
z1=z;  
A. x1 = 10 , y1 = 10 and z1=10  
B. x1 = 10, y1 = 11, and z1=10  
C. x1=10, y1 = 10, and z1=11  
D. x1=11, y1=10, and z1=11
```

3. **Consider the following application:**

```
Class Test{  
    public int addTest(int x, int y)  
    {  
        x=x+1; y=y+1;  
        int z=(x+y);  
        return z;  
    }  
    public static void main(String [] args)  
    {  
        int x=10; int y=10; int z=0;  
        Test t=new Test();  
        z= t.addTest(x,y);  
        System.out.println("x="+x+", y="+y+", z="+z);  
    }  
}
```

What will be output of the above program?

- A. x=10, y=10, z=22
B. x=11, y=11, z=22
C. x=10, y=10, z=20
D. x=11, y=11, z=20
4. Consider the following application. Assume that **MyList** class is declared in **MyList.java** and **ListManager** class is declared in **ListManager.java** file.
- ```
public class MyList{
 int size=1;
 public static void main(String [] args)
 {
 MyList list=new MyList();
 list.size=10;
 ListManager lm=new ListManager();
 lm.expandList(list);
 System.out.println("list.size="+list.size);
 }
} //end of MyList
public class ListManager{
 public void expandList(MyList l)
 {
 l.size=l.size+10;
 }
}//end of ListManager
```
- What will be output of the above program?

. . . list.size=0

- A. list.size=10  
B. list.size=11  
C. list.size=20
5. If int x = -1 then which of the following expression results in a positive value in x?

. . . x=x>>>32

- A. x=x>>>5  
B. x=x>>5  
C. x=~x

6. Which of the following lines of code would print "Equal" when you run it?

. . . int x=1; float y=1.0F; if(x==y){ System.out.println("Equal");}

- A. int x=1; Integer y= new Integer(1); if(x==y) { System.out.println("Equal");}  
B. Integer x=new Integer(1); Integer y=new Intger(1); if(x==y){ System.out.println("Equal");}  
C. String x="1"; String y="1"; if (x==y) { System.out.println("Equal");}

**7. Which of the following declarations are correct for the top level class?**

. . . public synchronized class MyTest extends Thread

- A. private class MyTest extends Thread
- B. public abstract class MyTest extends Thread
- C. class MyTest extends Thread

**8. Consider the following lines of code**

```
class Test{
String x;
public void testDemo(int n)
{
 String y;
 if (n>0) {
 y="Hello";
 }
 System.out.println(x+y);
}
public static void main(String [] args)
{
 Test test=new Test();
 test.testDemo(2);
}
```

What will happen if you try to compile and run above program?

. . . It will produce compiler warning that variable y may not have been initialized

- A. It will produce compiler warning that variable x may not have been initialized
- B. It will output "Hello"
- C. It will output "nullHello"

**9. Consider that Parent and Child classes are defined in two different files as below:**

```
class Parent{
public Parent(){
System.out.println("I am Parent");
}
}
class Child extends Parent{
public Child(int x){
System.out.println("I am Child");
}
public static void main(String [] args){
Child c=new Child(10);
}
```

What will be output if you try to compile and run above program?

. . . It will not compile.

- A. It will compile successfully. It will output "I am Parent" and then "I am Child."
- B. It will compile successfully. It will output "I am Child" and then "I am Parent."
- C. It will compile successfully, but will not run.

**10. Consider following code:**

```
public class MyList{
 static int size;
 public expandList(int newSize){
 ListExpander lexp=new ListExpander();
 Vector expandedList=lexp.expand();
 class ListExpander{
 public Vector expand(){
 Vector v=new Vector(this.size+newSize);
 return v;
 }
 }
 }
}
```

What will happen if you attempt to compile above code?

- . compiler error, "cannot refer inside an inner class to a static variable."
- A. compiler error, "cannot refer inside an inner class to a non-final variable newSize defined in a different method."
- B. Both of the above
- C. None of the above

**11. Consider following code:**

```
public class Parent{
 public int size=0;
 static class InnerClass{
 public void incrementParentSize(){
 XXX=XXX+10;
 }
 }
}
```

In above example, how can you access 'size' variable (of outer class Parent) inside innerclass at the place of 'XXX'

.

super.size

- A. this.size
- B. Parent.size
- C. Can not access it

**12. Assume that Parent and Child classes are in different files:**

```
public class Parent{
 public Parent(int x, int y){
 System.out.println("Created Parent");
 }
}//end of Parent class
public class Child extends Parent{
 public Child(int x, int y){
 //
 }
 public Child(int x, int y, int z){
 System.out.println("Creating child");
 this(x,y);
 }
 public static void main(String [] args){
 Child c=new Child(1,2,3);
 }
}
```

```
}
```

What will happen if you try to compile and run above program?

. It will compile successfully. It will output "Created Parent" and then "Creating child"

- A. It will compile successfully. It will output "Creating child" and then "Created Parent"
- B. It will not compile giving warning, "Explicit constructor invocation must be first statement in constructor."
- C. It will not compile giving warning, "Expression is not a valid block statement."

**13. Consider following code:**

```
public class OuterClass{
 class InnerClass{
 }
 public void innerClassDemo(){
 //Explicit instance of InnerClass
 }
}
```

In above example, how can you explicitly create an instance of InnerClass?

. InnerClass i=InnerClass();

- A. InnerClass i=OuterClass.InnerClass();
- B. InnerClass i=new OuterClass ().new InnerClass();
- C. OuterClass.InnerClass i=new OuterClass.InnerClass();

**14. Please select valid array declaration(s):**

. int x[20];

- A. int []x=new int[20];
- B. int [][] x=new int [20][];
- C. int [][][] x=new int[20][20][20];
- D. int [] x={1,2,3};

**15. Consider following code:**

```
public class Test{
 protected void demo() throws NumberFormatException, ArrayIndexOutOfBoundsException {
 //something here
 }
 public void demo(String s){
 //something here
 }
}//end of Test class
```

Please select true statement(s) for demo method

. It is an example of overloading method

- A. It is an example of overriding method
- B. Both of the above
- C. None of the above

**16. For the following code, please consider that super class is defined in question #15:**

```
public class MyTest extends Test{
 private void demo() throws IndexOutOfBoundsException, ClassNotFoundException
 {
 //something here
 }
}//end of MyTest class
```

What will happen if you try to compile above class?

. . . It will compile successfully.

- A. Compiler error: Exception java.lang.ClassNotFoundException in throws clause of void MyTest.demo() is not compatible with void Test.demo().
- B. Compiler error: Cannot reduce visibility of the inherited method from Test.
- C. Both B and C

**17. Consider the following code:**

```
public class Test{
 public void demo(String [] list){
 try{
 String s=list[list.length+1];
 System.out.println(s);
 }catch(ArrayIndexOutOfBoundsException e){
 return;
 }finally{
 System.out.println("Finally here.");
 }
 }
 public static void main(String [] args){
 Test t=new Test();
 String [] list={"one","two"};
 t.demo(list);
 System.out.println("Done!");
 }
}//end of Test class
```

What happen if you try compile and run above code?

. . . It will not compile.

- A. It will output "null" and then "Finally here."
- B. It will output "Done!"
- C. It will output "Finally here" and then "Done!"

**18. Please consider following code:**

```
public class Test{
 public static void demo(String s)
 {
 debug("In demo:"+s);
 }
 private void debug(String s){
 System.out.println(s);
 }
 public static void main(String [] args){
 Test.demo("Hello");
 }
}
```

What will happen if you try to compile and run above program?

. . . It will compile successfully, but will not run.

- A. It will compile successfully, and outputs "In demo:Hello."
- B. It will not compile with error message "Can not make a static reference to the instance method named."
- C. None of the above

19. **Consider the following code:**

```
/** File Drawable.java */
public interface Drawable{
 public void draw();
 public void fill();
} /** End of file Drawable.java */

/** File Circle.java */
public class Circle implements Drawable{
 int center=0;
 public void draw(){
 System.out.println("Drawing circle");
 }
 public static void main(String [] args){
 Circle c=new Circle();
 c.draw();
 }
} /** End of file Circle.java */
```

If you attempt to compile and run Circle class what will be output?

It will compile successfully, and outputs "Drawing circle."

- A. It will not compile, and reports error: "class Circle must implement inherited abstract method void Drawable.fill."
- B. It will not compile, and reports error: "Method Drawable.fill requires a body instead of a semicolon."
- C. None of the above

20. **Consider the following code:**

```
int x=2; int y=3; int z=4;
if(x>2){
 System.out.println("Tested x");
}if(y<3){
 System.out.println("Tested y");
}if(z<=3){
 System.out.println("Tested z");
}
```

Which line would be part of the output?

Tested x.

- A. Tested y.
- B. Tested z.
- C. None of the above.

21. **Consider the following code:**

```
for(int i=0;i<2;i++)
{
 for(int j=i;j<3; j++)
 {
 if (i==j)
 {
 continue;
 }
 }
}
```

```
 System.out.println("i=" + i + " j=" + j);
 }
}
```

Which lines would be part of the output?

. . . i = 0 j = 1

- A. i = 0 j = 2
- B. i = 1 j = 2
- C. None of the above

**22. Consider the following code:**

```
int j=0;
for(int i=0;i<2;i++)
{
 for (j=i; j<3; j++)
 {
 continue;
 }
 System.out.println("i = " + i + " j = " + j);
}
```

Which lines would be part of the output?

. . . i = 0 j = 0

- A. i = 1 j = 1
- B. i = 0 j = 3
- C. i = 1 j = 3

**23. Consider the following code:**

```
int i=0; int j=0;
for(i=0;i<2;i++)
{
 for (j=i; j<3; j++)
 {
 break;
 }
 System.out.println("i = " + i + " j = " + j);
}
```

Which lines would be part of the output?

. . . i = 0 j = 0

- A. i = 1 j = 1
- B. i = 0 j = 3
- C. i = 1 j = 3

**24. Consider the following code:**

```
int i, j=0;
outer:
for(i=0;i<2;i++)
{
 for (j=i; j<3; j++)
 {
 continue outer;
 }
}
```

```
 System.out.println("i = "+i+" j = "+j);
}
```

Which lines would be part of the output?

. . . i = 0 j = 0

- A. i = 1 j = 1
- B. i = 0 j = 3
- C. None of the above

**25. Consider the following code:**

```
int i, j=0;

for(i=0;i<2;i++)
{
 inner:
 for (j=i; j<3; j++)
 {
 break inner;
 }
 System.out.println("i = "+i+" j = "+j);
}
```

Which lines would be part of the output?

. . . i = 0 j = 0

- A. i = 1 j = 1
- B. i = 0 j = 3
- C. None of the above

**26. Consider following lines of code:**

```
Thread currentThread=Thread.currentThread();
int priority = currentThread.getPriority();
Thread t1=new Thread();
t1.setPriority(9);
ThreadGroup tgrp=new ThreadGroup();
tgrp.setMaxPriority(10);
Thread t2=new Thread(tgrp,"t2");
System.out.println("Priority of t1="+t1.getPriority());
System.out.println("Priority of t2="+t2.getPriority());
What will be output of the above code?
```

. . . Priority of t1=5 and Priority of t2=10

- A. Priority of t1=9 and Priority of t2=10
- B. Priority of t1=9 and Priority of t2=5
- C. Neither of above

**27. Consider the following code:**

```
/** File Thread1.java */
class Thread1 implements Runnable{
 public void run(){
 System.out.println("Running Thread1");
 } } /** End of file Thread1.java */
```

```

/** Thread2.java */
class Thread2 extends Thread{
public void run(){
System.out.println("Running Thread2");
}
public static void main(String [] args){
Thread t1= new Thread1();
Thread t2=new Thread2(t1);
t1.start();
t2.start();
}
}
/** End of Thread2.java*/

```

If you try to compile and run above code what will be result?

"Running thread1" following "Running thread2"

- A. "Running thread2" following "Running thread1"
- B. It will not compile because in Thread1 and Thread2 start() is not defined .
- C. It will not compile because constructor invoked to create Thread2 with arguments (Thread1) is not defined

**28. Consider the following code:**

```

class MyThread extends Thread{
public void run(){
System.out.println("Done");
}
public void demo(){
System.out.println("Demo");
}
public static void main(String args[]){
MyThread th=new MyThread();
th.run();
th.stop();
th.demo();
}
}

```

What will happen if you try to compile and run above code:

. It will throw an exception at th.run() because run() was called before calling start().

- A. It will throw an exception at th.demo() because Thread variable th was already stopped calling stop().
- B. It will output "Done" following "Demo"
- C. Neither of the above.

**29. Please consider following code:**

```

String s1=" 5 + 5 = 10 ";
s1.trim();
s1.replace('+', '-');

```

How many String objects will be created after executing above lines?

. 1

- A. 2
- B. 3
- C. 4

30. String s="Hi";  
StringBuffer sb=new StringBuffer(s);  
String s1=new String("There");  
StringBuffer sb1=new StringBuffer(s1);  
if(s==sb){  
System.out.println("s==sb");  
}if(s.equals(sb)){  
System.out.println("s.equals(sb)");  
}if(s1.equals(sb1)){  
System.out.println("s1.equals(sb1)");  
}  
Please, select which of the following will be part of output?

. . . It will not compile at if(s==sb) because operands on both side are not compatible

- A. It will print s1.equals(sb1)
- B. It will print s.equals(sb)
- C. It will compile successfully, but it will not output anything

31. Consider that following code is declared in BussyThread.java file

```
public class BussyThread extends Thread{
 public void run(){
 for(int i=0;i<10; i++){
 i=i-1;
 }//end of for loop
 }//end of run()
public static void main(String args[]){
 BussyThread b1=new BussyThread();
 BussyThread b2=new BussyThread();
 b1.start();
 b2.start();
}
```

//end of class  
Above class will start two threads b1 and b2. Select True statements for above class.

. . . Only b1 thread will get chance to run

- A. Only b2 thread will get chance to run
- B. Both thread will get chance to run sharing CPU time
- C. Neither of the thread will be able to run.

32. **What changes in run() method of BussyThread will enable both threads to run?**

. . . adding yield() into run method

- A. adding try{sleep(1000);}catch (InterruptedException e){} into run method
- B. adding wait(1000) into run method
- C. Neither of the above

33. **Consider the following classes are in MyThread.java, YourThread.java, and Driver.java files:**  
public class MyThread implements Runnable{

```

public void run(){
 System.out.println("Running MyThread");
}
}//end of MyThread
public class YourThread extends Thread{
public YourThread(Runnable r){
super(r);
}
public void run(){
System.out.println("Running YourThread");
}
}//end of YourThread
public class Driver{

public static void main(String args[]){
MyThread t1= new MyThread();
YourThread t2 = new YourThread(t1);
t2.start();
}
}//end of class

```

If you try to run Driver class what will be result?

- . It will output "Running MyThread."
- A. It will output "Running YourThread."
- B. It will output both "Running MyThread," and "Running YourThread."
- C. It will not run.

34. Consider following code:

```

35. String s=null;
36. String t="null";
37. if (s==t)
38. {
39. System.out.println("s equal to t");
40. }else
41. {
42. System.out.println("s not equal to t");
43. }

```

what will result if you try to compile and run above code?

- . it compiles successfully, but throws NullpointerException at if (s==t)
- A. It will not compile.
- B. It compiles successfully and output "s equal to t"
- C. It compiles successfully and output "s not equal to t"

44. Consider the following code:

```

45. public void demo() {
46. String s[];
47. if (s.equals(null))
48. {
49. System.out.println("s is null");
50. } else
51. {
52. System.out.println("s is not equal");
53. }
}

```

What will be result if you try to compile and run above code?

- . Compile error produced, "variable s may not have been initialized."

- A. It compile successfully, but throws NullpointerException at if ( s.equals(null) )
- B. It compile successfully, and outputs "s is null."
- C. It compile successfully, and outputs "s is not null."

**54. Consider the following code:**

```

public class MyList
{
 private static final int MAX_SIZE = 10;
 private Object [] list = new Object[MAX_SIZE];
 public void add(Object obj)
 {
 int size=list.length;
 if(size >= MAX_SIZE)
 {

 class ListExpander
 {
 public void expand()
 {
 Object temp [] = list;
 list = new Object[size+MAX_SIZE];
 for (i=0;i<temp.length; i++)
 {
 list[i]=temp[i];
 }
 }
 }//end of public void expand()
 } end of class ListExpander
 ListExpander listEx = new ListExpander();
 listExp.expand();
 list[size] = obj;
 }//end of if
}//end of add
}//end of class MyList

```

**What will be result if you try to compile and run the above code:**

- . Compiler error reported, "Cannot refer inside an inner class to a non-final variable 'size' defined in a different method."

- A. Compiler error reported, "Cannot refer inside an inner class to a private member variable 'list' defined in enclosing class MyList."
- B. Compiler error reported, "Cannot refer inside an inner class to a static member variable MAX\_SIZE defined in enclosing class MyList."
- C. It compiles and runs successfully.

55. **Consider following example of an inner class**

```
public class MyTest{
 public String publicVariable = "a";
 private String privateVariable = "b";
 public static int SIZE = 0;
 private static int MAX_SIZE = 0;
 public static DemoHelper{
 public demo{
 System.out.println("Demo = "+XXX);
 }
 }
}//end of inner class
}
```

**which variable of the MyTest class will be able to use in place of XXX?**

. . . publicVariable

- A. privateVariable
- B. SIZE
- C. MAX\_SIZE

56. **What will be result if you try to compile and run following code?**

```
public class Record extends String{}
```

. . . Compiler error reported, "Can not extend a final class."

- A. Compiler error reported, "Must implement method int compareTo(Object)."
- B. Compile and run successfully.
- C. None of the above.

57. **Consider the following two classes:**

```
public class Parent{
 protected void demo() throws Exception{}
} // end of Parent class
public class Child extends Parent{
 private void demo() {}
}
```

**What will be result if you try to compile above two classes?**

- . Compiler object for the method of a Child class, "Can not reduce the visibility of the inherited method."

- A. Compiler object for demo() method of a Child class, "Inherited method is not compatible with void Parent.demo() throws Exception."
- B. Compile successfully.
- C. None of the above

**58. Consider the following two classes:**

```
public class Parent{
protected void demo() {}
} // end of Parent class
public class Child extends Parent{
public void demo() throws Exception{}
}
```

**What will be result if you try to compile above two classes?**

- . Compiler object for the method of a Child class, "Can not widen the visibility of the inherited method."
- A. Compiler object for demo() method of a Child class, "Exception java.lang.Exception in throws clause of void Child.demo() is not compatible with void Parent.demo()."
- B. Compile successfully
- C. None of the above

**59. Consider the following two classes:**

```
public class Parent{
protected void demo() {}
} // end of Parent class
public class Child extends Parent{
public int demo()
{return 0;}
}
```

**What will be result if you try to compile above two classes?**

- . Compiler object for the method of a Child class, "Can not widen the visibility of the inherited method."
- A. Compiler object for the method of a Child class, "Return type is not compatible with void Parent.demo()."
- B. Compile successfully.
- C. None of the above

**60. Consider the following two classes:**

```
public class Parent{
protected static void demo() {}
} // end of Parent class
public class Child extends Parent{
public void demo() {}
}
```

**What will be result if you try to compile above two classes?**

- . Compiler object for the method of a Child class, "Can not widen the visibility of the inherited method."

- A. Compiler object for the method of a Child class, "inherited method void Child.demo() is not compatible with void Parent.demo()."
- B. Compiler object for the method of a Child class, "The instance method can not override the static method from Parent."
- C. Compile successfully.

**61. Consider that class Employee and Salesman are in different file called Employee.java and Salesman.java:**

```
/** Employee.java file*/
public class Employee{
 int salary=1000;
 public int getSalary(){
 return salary;
 }
}
/** End of Employee.java file*/
/** Salesman.java file*/
public class Salesman extends Employee{
 int commission =100;
 public int getSalary(){
 return salary+commission;
 }
 public static void main(String [] args){
 Salesman sm = new Salesman();
 Employee em = sm;
 System.out.println(em.getSalary());
 }
}
/** End of Salesman.java file*/
```

**What will be result if you try to compile and run above code?**

- . Compiler error reported , "Type mismatch: Cannot convert from Salesman to Employee."

  - A. It compile successfully and outputs 1000.
  - B. It compiles successfully and outputs 1100.
  - C. None of the above

**62. Considering following code what will be the result if you try to compile the following code:**

```
public abstract class Test{
 public void demo(){
 System.out.println("demo");
 }
}
```

- . It will compile successfully.

  - A. Compiler error reported, "An abstract method must be defined."
  - B. Compiler error reported, "Invalid declaration of class."
  - C. None of the above

**63. Considering following code what will be the result if you try to compile the following code:**

```
public class Test{
 public abstract void demo();
}
```

- . Compiler error reported, "Method requires a body instead of semicolon."
- A. Compiler error reported, "Abstract methods are only defined by abstract classes."
- B. Compile successfully.
- C. None of the above.

**64. The GenericList has the following method:**

```
public void addItem(Object item)
```

You are writing a class GroceryList that extends GenericList. Which of the following would be legal declarations of overloading methods?

- . public void additem(Vector item)
- A. public void addItem(Object [] items) throws Exception
- B. protected void additem(Object item)
- C. All of the above

**65. What will be result if you try to compile the following code?**

```
public class Parent{
 String name=null;
 public Parent(String n){
 name=n;
 }
}
public class Child extends Parent{
 String type="X";
}
```

- . Compile successfully.
- A. Compiler error reported, because Parent class did not declare constructor with arguments ().
- B. Compiler error reported, because Child class did not declare a constructor.
- C. Both of the above B and C

**66. What will be legal statement in the following method?**

```
public void demo(int x){
 XXX y=10;
}
```

- . public int

- A. int
- B. final int
- C. static int

**67. What will be result if you try to compile and run following code fragment?**

```
public void demo (String [] args){
 int i=1;
 for(int i=0;i<args.length;i++)
 {
 System.out.println(args[i]);
 }
}
```

- . Compile successfully, but throws IndexOutOfBoundsException during runtime.
- A. Compile error reported, "Local name i is already defined."
- B. Throws NullPointerException during runtime
- C. None of the above

**Answers:**

- 4. D
- 5. D
- 6. A
- 7. D
- 8. B and D
- 9. A and D
- 10. C and D
- 11. A
- 12. B
- 13. B
- 14. D
- 15. C and D
- 16. C and D (Corrected)
- 17. B, C, D and E

18. A

19. D Note: This is an example of overriding method.

20. D

21. C

22. B

23. D

24. A, B, and C

25. C and D

26. A and B

27. D

28. A and B

29. C

30. D

31. C

32. C

33. A

34. C

35. A and B

36. B

37. D

38. A

39. A

40. C and D

41. A

42. A

43. B

44. B

45. C

46. C

47. A

48. B

49. A and B

50. B

51. B and C

52. B

---

**1.What is the result when you compile and run the following code?**

```
class Top {

 static void myTop() {
 System.out.println("Testing myTop method in Top class");
 }
}
public class Down extends Top {

 void myTop() {
 System.out.println("Testing myTop method in Down class");
 }
 public static void main(String [] args) {
 Top t = new Down();
 t.myTop();
 }
}
```

- A) Compile Time error
- B) Runtime error
- C) Prints Testing myTop method in Top class on the console
- D) Prints Testing myTop method in Down class on the screen

**2. Which of the code fragments will throw an "ArrayOutOfBoundsException" ?**

- A) for (int i = 0; i < args.length; i ++ ) {  
 System.out.print( i );  
}
- B) System.out.print(args.length);
- C) for ( int i = 0; i < 1; i++ ) {  
 System.out.print(args[i]);  
}
- D) None of the above

**3. What is the result of the following program, when you compile and run?**

```

public class MyTest {

 final int x;
 public MyTest() {
 System.out.println(x + 10);
 }
 public static void main(String args[]) {
 MyTest mt = new MyTest();
 }
}

```

- A) Compile time error
- B) Runtime error
- C) Prints on the screen 10
- D) Throws an exception

**4. What is the output when you compile and run the following code fragment?**

```

class MyTest {

 public void myTest() {
 System.out.println("Printing myTest in MyTest class");
 }

 public static void myStat() {
 System.out.println("Printing myStat in MyTest class");
 }
}

public class Test extends MyTest {

 public void myTest() {
 System.out.println("Printing myTest in Test class");
 }

 public static void myStat() {
 System.out.println("Printing myStat in Test class");
 }

 public static void main (String args[]) {

 MyTest mt = new Test();
 mt.myTest();
 mt.myStat();
 }
}

```

- A) Printing myTest in MyTest class followed by Printing myStat in MyTest class
- B) Printing myTest in Test class followed by Printing myStat in MyTest class
- C) Printing myTest in MyTest class followed by Printing myStat in MyTest class
- D) Printing myStat in MyTest class followed by Printing myStat in MyTest class

**5. Select all the exceptions thrown by wait() method of an Object class, which you can replace in the place of xxx legally?**

```

class T implements Runnable {

 public void run() {
 System.out.println("Executing run() method");

```

```

 myTest();
 }

public synchronized void myTest() {
 try {
 wait(-1000);
 System.out.println("Executing the myTest() method") ;
 } XXX
}
}

public class MyTest {

```

- A) catch ( InterruptedException ie) {}
- B) catch ( IllegalArgumentException il ) {}
- C) catch ( IllegalMonitorStateException im ) {}
- D) Only catch ( InterruptedException e ) {} exception

**6. Which of the following are examples of immutable classes , select all correct answer(s)?**

- A) String
- B) StringBuffer
- C) Double
- D) Integer

**7. Select the correct answer for the code fragment given below?**

```

public class TestBuffer {

 public void myBuf(StringBuffer s, StringBuffer s1) {
 s.append(" how are you") ;
 s = s1;
 }

 public static void main (String args[]) {
 TestBuffer tb = new TestBuffer();
 StringBuffer s = new StringBuffer("Hello");
 StringBuffer s1 = new StringBuffer("doing");
 tb.myBuf(s, s1);
 System.out.print(s);
 }
}

```

- A) Prints Hello how are you
- B) Prints Hello
- C) Prints Hello how are you doing
- D) Compile time error

**8. What is the result when you compile and run the following code?**

```
public class MyTest {
```

```

public void myTest(int[] increment) {
 increment[1]++;
}

public static void main (String args[]) {
 int myArray[] = new int[1];
 MyTest mt = new MyTest();
 mt.myTest(myArray);
 System.out.println(myArray[1]);
}
}

```

- A) Compile time error
- B) Runtime error
- C) ArrayOutOfBoundsException
- D) Prints 1 on the screen

**9. Choose all valid identifiers?**

- A) int100
- B) byte
- C) aString
- D) a-Big-Integer
- E) Boolean
- F) strictfp

**10. Select the equivalent answer for the code given below?**

```

boolean b = true;
if (b) {
 x = y;
} else {
 x = z;
}

```

- A) x = b ? x = y : x = z ;
- B) x = b ? y : z ;
- C) b = x ? y : z ;
- D) b = x ? x = y : x = z ;

**11. Choose all correct answers?**

- A) int a [][] = new int [20][20];
- B) int [] a [] = new int [20][];
- C) int [][] a = new int [10][];
- D) int [][] a = new int [] [10];

**12. Consider the following code and select the correct answer?**

```

class Vehicle {

 String str ;
 public Vehicle() {
 }
 public Vehicle (String s) {
 str = s;
 }
}

public class Car extends Vehicle {

```

```

public static void main (String args[]) {
 final Vehicle v = new Vehicle (" Hello");
 v = new Vehicle (" How are you");
 v.str = "How is going";
 System.out.println("Greeting is : " + v.str);
}
}

```

- A) Compiler error while subclassing the Vehicle
- B) Compiler error , you cannot assign a value to final variable
- B) Prints Hello
- C) Prints How is going

**13. Java source files are concerned which of the following are true ?**

- A) Java source files can have more than one package statements.
- B) Contains any number of non-public classes and only one public class
- C) Contains any number of non-public classes and any number of public classes
- D) import statements can appear anywhere in the class
- E) Package statements should appear only in the first line or before any import statements of source file

**14. Select all correct answers from the following?**

```

int a = -1;
int b = -1;
a = a >>> 31;
b = b >> 31;

```

- A) a = 1, b =1
- B) a = -1, b -1
- C) a = 1, b = 0
- D) a = 1, b = -1

**15. What is the value of a , when you compile and run the following code?**

```

public class MyTest {

 public static void main (String args[]) {

 int a = 10;
 int b = 9;
 int c = 7;
 a = a ^ b ^ c;
 System.out.println (a);

 }
}

```

- A) 10
- B) 9
- C) 7
- D) 4

**16. The following code has some errors, select all the correct answers from the following?**

```

public class MyTest {

```

```

public void myTest(int i) {
 for (int x = 0; x < i; x++) {
 System.out.println(x);
 }
}
public abstract void Test() {
 myTest(10);
}
}

```

- A) At class declaration
- B) myTest() method declaration
- C) Test() method declaration
- D) No errors, compiles successfully

**17. At what point the following code shows compile time error?**

```

class A {
 A() {
 System.out.println("Class A constructor");
 }
}

class B extends A {
 B() {
 System.out.println("Class B constructor");
 }
}

public class C extends A {
 C() {
 System.out.println("Class C constructor");
 }
}

public static void main (String args[]) {
 A a = new A(); // Line 1
 A a1 = new B(); // Line 2
 A a2 = new C(); // Line 3
 B b = new C(); // Line 4
}
}

```

- A) A a = new A(); // Line 1
- B) A a1 = new B(); // Line 2
- C) A a2 = new C(); // Line 3
- D) B b = new C(); // Line 4

**18. Which of the following statements would return false? if given the following statements.**

```

String s = new String ("New year");
String s1 = new String("new Year");

```

- A) s == s1
- B) s.equals(s1);
- C) s = s1;
- D) None of the above

**19. Select all correct answers about what is the definition of an interface?**

- A) It is a blue print
- B) A new data type
- C) Nothing but a class definition
- D) To provide multiple inheritance

**20. Select all correct answers from the following code snippets?**

- A) // Comments  
import java.awt.\*;  
package com;
- B) import java.awt.\*;  
// Comments  
package com;
- C) package com;  
import java.awt.\*;  
// Comments
- D) // Comments  
package com;  
import java.awt.\*;  
public class MyTest {}

**21. What is the result when you compile and run the following code?**

```
public class MyError {

 public static void main (String args[]) {
 int x = 0;
 for (int i = 0; i < 10; i++) {
 x = new Math(i);
 new System.out.println(x);
 }
 }
}
```

- A) Prints 0 to 9 in sequence
- B) No output
- C) Runtime error
- D) Compile time error

**22. There are two computers are connected to internet, one computer is trying to open a socket connection to read the home page of another computer, what are the possible exceptions thrown while connection and reading InputStream?.**

- A) IOException
- B) MalformedURLException
- C) NetworkException
- D) ConnectException

**23. What is the result from the following code when you run?**

```
import java.io.*;

class A {

 A() throws Exception {
 System.out.println ("Executing class A constructor");
 }
}
```

```

 throw new IOException();
 }

}

public class B extends A {
 B() {
 System.out.println ("Executing class B constructor");
 }

 public static void main (String args[]) {
 try {
 A a = new B();
 } catch (Exception e) {
 System.out.println(e.getMessage());
 }
 }
}

```

- A) Executing class A constructor
- B) Executing class B constructor
- C) Runtime error
- D) Compile time error

**24. What is the result from the following code when you run?**

```

import java.io.*;

class A {

 A() {
 System.out.println ("Executing class A constructor");
 }

 A(int a) throws Exception {
 System.out.println ("Executing class A constructor");
 throw new IOException();
 }

}

public class B extends A {
 B() {
 System.out.println ("Executing class B constructor");
 }

 public static void main (String args[]) {
 try {
 A a = new B();
 } catch (Exception e) {
 System.out.println(e.getMessage());
 }
 }
}

```

- A) Executing class A constructor followed by Executing class B constructor
- B) No output
- C) Runtime error
- D) Compile time error

**25. What is the result when you compile and run the following code?**

```
byte Byte = 10;
byte Double = 12;
byte Integer = Byte * Double;
```

- A) 120;
- B) Compile time error while declaring variables
- C) Compile time error while multiplication
- D) None of the above

**26. Select all valid methods for Component class?**

- A) setBounds(), setVisible(), setFont()
- B) add(), remove()
- C) setEnabled(), setVisible()
- D) addComponent()

**27. Which subclasses of the Component class will display the MenuBar?**

- A) Window, Applet
- B) Applet, Panel
- C) Frame
- D) Menu, Dialog

**28. Select all correct answers from the following statements?**

- A) Frame's default layout manager is BorderLayout
- B) CheckBox, List are examples of non visual components
- C) Applets are used to draw custom drawings
- D) Canvas has no default behavior or appearance

**29. Select all the methods of java.awt.List?**

- A) void addItem(String s), int getRows()
- B) void addItem(String s, int index), void getRows()
- C) int[] getSelectedIndexes(), int getItemCount()
- D) int[] getSelectedIndexes(), String[] getSelectedItems()

**30. Please select all correct answers?**

- A) java.awt.TextArea.SCROLLBARS\_NONE
- B) java.awt.TextArea does not generate Key events
- C) java.awt.TextField generates Key events and Action events
- D) java.awt.TextArea can be scrolled using the <-- and --> keys.

**31. What is the result if you try to compile and run the following code ?**

```
public class MyTest {

 public static void myTest() {
 System.out.println("Printing myTest() method");
 }
 public void myMethod() {
 System.out.println("Printing myMethod() method");

 }
 public static void main(String[] args) {
 myTest();
 myMethod();
 }
}
```

}

- A) Compile time error
- B) Compiles successfully
- C) Error in main method declaration
- D) Prints on the screen Printing myTest() method followed by Printing myMethod() method

### **32. What line of a given program will throw FileNotFoundException?**

```
import java.io.*;

public class MyReader {

 public static void main (String args[]) {
 try {
 FileReader fileReader = new FileReader("MyFile.java");
 BufferedReader bufferedReader = new BufferedReader(fileReader);
 String strString;
 fileReader.close();

 while ((strString = bufferedReader.readLine()) != null) {
 System.out.println (strString);
 }

 } catch (IOException ie) {
 System.out.println (ie.getMessage());
 }
 }
}
```

- A) This program never throws FileNotFoundException
- B) The line fileReader.close() throws FileNotFoundException
- C) At instantiation of FileReader object.
- D) While constructing the BufferedReader object

### **33. When the following program will throw an IOException?**

```
import java.io.*;

class FileWrite {
 public static void main(String args[]) {
 try {
 String strString = "Now is the time to take Sun Certification";
 char buffer[] = new char[strString.length()];
 strString.getChars(0, strString.length(), buffer, 0);
 FileWriter f = new FileWriter("MyFile1.txt");
 FileWriter f1 = f;
 f1.close();
 for (int i=0; i < buffer.length; i += 2) {
 f.write(buffer[i]);
 }
 f.close();

 FileWriter f2 = new FileWriter("MyFile2.txt");
 f2.write(buffer);
 f2.close();
 } catch (IOException ie) {
 System.out.println(ie.getMessage());
 }
 }
}
```

```
 }
}
```

- A) This program never throws IOException
- B) The line f1.close() throws IOException
- C) While writing to the stream f.write(buffer[i]) throws an IOExcption
- D) While constructing the FileWriter object

**34. Which line of the program could be throwing an exception, if the program is as listed below. Assume that "MyFile2.txt" is a read only file.**

**Note:** MyFile2.txt is read only file..

```
import java.io.*;

class FileWrite {
 public static void main(String args[]) {
 try {
 String strString = "Updating the critical data section"
 char buffer[] = new char[strString.length()];
 strString.getChars(0, strString.length(), buffer, 0);
 FileWriter f = new FileWriter("MyFile1.txt");
 FileWriter f1 = f;
 for (int i=0; i < buffer.length; i += 2) {
 f1.write(buffer[i]);
 }
 f1.close();

 FileWriter f2 = new FileWriter("MyFile2.txt");
 f2.write(buffer);
 f2.close();
 } catch (IOException ie) {
 System.out.println(ie.getMessage());
 }
 }
}
```

- A) This program never throws IOException
- B) The line f1.close() throws IOException
- C) While writing to the stream f1.write(buffer[i]) throws an IOException
- D) While constructing the FileWriter f2 = new FileWriter("MyFile2.txt");

**35. Select all the correct answers about File Class?**

- A) A File class can be used to create files and directories
- B) A File class has a method mkdir() to create directories
- C) A File class has a method mkdirs() to create directory and its parent directories
- D) A File cannot be used to create directories

**36. Using File class, you can navigate the different directories and list all the files in the those directories?**

- A) True
- B) False

**37. Select all the constructor definitions of "FileOutputStream"?**

- A) FileOutputStream(FileDescriptor fd)
- B) FileOutputStream(String fileName, boolean append)

- C) FileOutputStream(RandomAccessFile raFile)
- D) FileOutputStream( String dirName, String filename)

**38. Select all correct answers for Font class?**

- A) new Font ( Font.BOLD, 18, 16)
- B) new Font ( Font.SERIF, 24, 18)
- C) new Font ( "Serif", Font.PLAIN, 24);
- D) new Font ( "SanSerif", Font.ITALIC, 24);
- E) new Font ( "SanSerif", Font.BOLD+Font.ITALIC, 24);

**39. In an applet programming the requirement is that , what ever the changes you do in the applets graphics context need to be accumulated to the previous drawn information. Select all the correct code snippets?**

- A) public void update ( Graphics g ) {  
    paint( g ) ;  
}
- B) public void update ( Graphics g ) {  
    update( g ) ;  
}
- C) public void update ( Graphics g ) {  
    repaint( g ) ;  
}
- D) public void update ( Graphics g ) {  
    print( g ) ;  
}

**40. How can you load the image from the same server where you are loading the applet, select the correct answer form the following?**

- A) public void init() {  
    Image i = getImage ( getDocumentBase(), "Logo.jpeg");  
}
- B) public void init() {  
    Image i = getImage ( "Logo.jpeg");  
}
- C) public void init() {  
    Image i = new Image ( "Logo.jpeg");  
}
- D) public void init() {  
    Image i = getImage ( new Image( "Logo.jpeg") );  
}

**41. Which of the following answers can be legally placed in the place of XXX?**

```
class Check {
 Check() { }
}

public class ICheck extends Check {
 public static void main (String[] args) {
 Object o = new ICheck();
```

```

Check i = new ICheck();
Check c = new Check();

if (o instanceof XXX) {
 System.out.println("True");
}
}

}

```

- A) Object, ICheck only
- B) Check , ICheck only
- C) Object only
- D) Object, Check, ICheck

**42. There are 20 threads are waiting in the waiting pool with same priority, how can you invoke 15th thread from the waiting pool?**

- A) By calling resume() method
- B) By calling interrupt() method
- C) Calling call() method
- D) By calling notify(15) method on the thread instance
- E) None of the above

**43. Select all the correct answers regarding thread synchronization ?**

- A) You can synchronize entire method
- B) A class can be synchronized
- C) Block of code can be synchronized
- D) The notify() and notifyAll() methods are called only within a synchronized code

**44. The thread run() method has the following code, what is the result when the thread runs?**

```

try {
 sleep(200);
 System.out.println("Printing from thread run() method");
} catch (IOException ie) { }

```

- A) Compile time error
- B) Prints on the console Printing from thread run() method
- C) At line 2 the thread will be stop running and resumes after 200 milliseconds and prints  
Printing from thread run() method
- D) At line 2 the thread will be stop running and resumes exactly 200 milliseconds elapsed

**45. What is the result when you compile and run the following code?**

```

import java.awt.*;

public class TestBorder extends Frame {
 public static void main(String args[]) {
 Button b = new Button("Hello");
 TestBorder t = new TestBorder();
 t.setSize(150,150);
 t.add(b);
 }
}

```

- A) A Frame with one big button named Hello
- B) A small button Hello in the center of the frame
- C) A small button Hello in the right corner of the frame
- D) Frame does not visible

**46. Select all correct answers from the following?**

- A) public abstract void Test() { }
- B) public void abstract Test();
- C) public abstract void Test();
- D) native void doSomthing( int i );

**47. Please select all correct statements from the following?**

- A) `toString()` method is defined in `Object` class.
- B) `toString()` method is defined in `Class` class.
- C) `wait()`, `notify()`, `notifyAll()` methods are defined in `Object` class and used for Thread communication.
- D) `toString()` method provides string representation of an Object state.

**48. From the following declarations select all correct variables/method declarations?**

```
Button bt = new Button ("Hello");
```

- A) public transient int val;
- B) public synchronized void Test() ;
- C) bt.addActionListener( new ActionListener () );
- D) synchronized ( this ) {  
 // Assume that "this" is an arbitrary object instance.  
}

**49. Which of the following classes will throw "NumberFormatException"?**

- A) Double
- B) Boolean
- C) Integer
- D) Byte

**50. Fill all the blanks from the following ?**

- A) `Math.abs(3.0)` returns 3.0  
`Math.abs(-3.4)` returns -----
- B) `Math.ceil(3.4)` returns -----  
`Math.ceil(-3.4)` returns -3.0
- C) `Math.floor(3.4)` returns -----  
`Math.ceil(-3.4)` returns -4.0
- D) `Math.round(3.4)` returns 3  
`Math.round(-3.4)` returns -----

**51. Select from the following which is legal to put in the place of XXX?**

```
public class OuterClass {
 private String s = "I am outer class member variable";
 class InnerClass {
 private String s1 = "I am inner class variable";
```

```

 public void innerMethod() {
 System.out.println(s);
 System.out.println(s1);
 }
 }
 public static void outerMethod() {
 // XXX legal code here
 inner.innerMethod();
 }
}

```

- A) OuterClass.InnerClass inner = new OuterClass().new InnerClass();
- B) InnerClass inner = new InnerClass();
- C) new InnerClass();
- D) None of the above

**52. If you save and compile the following code, it gives compile time error. How do you correct the compile time error?**

```

public class OuterClass {
 final String s = "I am outer class member variable";
 public void Method() {
 String s1 = "I am inner class variable";
 class InnerClass {
 public void innerMethod() {
 int xyz = 20;
 System.out.println(s);
 System.out.println("Integer value is" + xyz);
 System.out.println(s1); // Illegal, compiler error
 }
 }
 }
}

```

- A) By making s1 as static variable
- B) By making s1 as public variable
- C) By making s1 as final variable
- D) By making InnerClass as static

**53. What is the reason using \$ in inner class representation?**

- A) Because the inner classes are defined inside of any class
- B) Due to the reason that inner classes can be defined inside any method
- C) This is convention adopted by Sun, to insure that there is no ambiguity between packages and inner classes.
- D) Because if use getClass().getName() will gives you the error

**54. What is the result when you compile and run the following code?**

```

import java.util.*;
public class MyVector {

 public Vector myVector () {
 Vector v = new Vector();
 return v.addElement("Adding element to vector");
 }

 public static void main (String [] args) {
 MyVector mv = new MyVector();
 }
}

```

```
 System.out.println(mv.myVector());
 }
}
```

- A) Prints Adding element to vector
- B) Compile time error
- C) Runtime error
- D) Compiles and runs successfully

**55. What is the output on the screen when compile and run the following code?**

```
public class TestComp {

 public static void main(String args[]) {
 int x = 1;
 System.out.println("The value of x is " + (~x >> x));
 }
}
```

- A) 1
- B) 2
- C) -1
- D) -2

**56. The method getWhen() is defined in which of the following class?**

- A) AWTEvent
- B) EventObject
- C) InputEvent
- D) MouseEvent

**57. Select all correct answers from the following?**

- A) getSource() method is defined in java.awt.AWTEvent class
- B) getSource() method is defined in java.util.EventObject class
- C) getID() method is defined in java.awt.AWTEvent class
- D) getID() method is defined in java.util.EventObject class

**58. Which of the following are correct answers?**

- A) A listener object is an instance of a class that implements a listener interface.
- B) An event source is an object , which can register listener objects and sends notifications whenever event occurs.
- C) Event sources fires the events.
- D) Event listeners fires events.

**59. What are possible ways to implement LinkedList class?**

- A) As a HashMap
- B) As a Queue
- C) As a TreeSet
- D) As a Stack

**60. Please select the correct answer from the following?**

```
public class ThrowsDemo {
 static void throwMethod() throws Exception {
 System.out.println("Inside throwMethod.");
 }
}
```

```

 throw new IllegalAccessException("demo");
 }
 public static void main(String args[]) {
 try {
 throwMethod();
 } catch (IllegalAccessException e) {
 System.out.println("Caught " + e);
 }
 }
}

```

- A) Compilation error
- B) Runtime error
- C) Compile successfully, nothing is printed.
- D) inside throwMethod. followed by caught: java.lang.IllegalAccessException: demo

### **Answers**

---

#### **Answer 1:**

- A) Compile Time error

#### **Explanation:**

Generally static methods can be overridden by static methods only .. Static methods may not be overridden by non static methods..

#### **Answer 2:**

- C) for ( int i = 0; i < 1; i++ ) {
 System.out.print(args[i]);
 }

#### **Explanation:**

Answer C) will cause an "ArrayOutOfBoundsException" if you do not pass the command line arguments to the Java Program. A) and B) will work without any problem.

#### **Answer 3:**

- A) Compile time error

#### **Explanation:**

The final variables behaves like constants, so the final variables must be initialized before accessing them. They can be initialized where they are declared or in every "constructor" if the class. ( even if class has one or more constructors defined ).

#### **Answer 4:**

- B) Printing myTest in Test class followed by Printing myStat in MyTest class

#### **Explanation:**

Static methods are determined at compile time but the non static ( instance methods ) methods are identified at runtime using Run Time Type Identification ( RTTI).

#### **Answer 5:**

- A) catch ( InterruptedException ie ) {}
- B) catch ( IllegalArgumentException il ) {}
- C) catch ( IllegalMonitorStateException im ) {}

**Explanation:**

The wait() method of an Object class throws InterruptedException when the thread moving from running state to wait state. If the value of timeout is negative or the value of nanos is not in the range 0-999999 then wait() method throws IllegalArgumentException exception at runtime. If the current thread is not the owner of this object's monitor then it throws IllegalMonitorStateException exception. Click [here](#) for more information from Java Documentation.

**Answer 6:**

- A) String
- C) Double
- D) Integer

**Explanation:**

String, Integer, Double are immutable classes, once assign a values it cannot be changed. Please refer the wrapper classes for more information on Integer, and Double.

**Answer 7:**

- A) Prints Hello how are you

**Explanation:**

Assigning or interchanging the object references does not change the values, but if you change the values through object references , changes the values .

**Answer 8:**

- B) Runtime error
- C) ArrayOutOfBoundsException

**Explanation:**

This piece of code throws an ArrayOutOfBoundsException at runtime . If you modify the code int myArray[] = new int[1]; to int myArray[] = new int[2]; , it prints 1 on the screen. The changes you made on the array subscript seen by the caller.

**Answer 9:**

- A) int 100
- C) aString
- E) Boolean

**Explanation:**

The byte, strictfp are Java keywords and cannot be defined as identifiers, the a-Big-Integer has "-" which is not a valid identifier. The identifiers must starts with letters, \$, or \_ ( underscore), subsequent characters may be letters, dollar signs, underscores or digits, any other combination will gives you the compiler error.

**Answer 10:**

- B) x = b ? y : z ;

**Explanation**

If b is true the value of x is y, else the value is z. This is "ternary" operator provides the way to simple conditions into a single expression. If the b is true, the left of ( :) is assigned to x else right of the ( :) is assigned to x. Both side of the ( :) must have the data type.

**Answer 11:**

- A) int a [][] = new int [20][20];
- B) int [] a [] = new int [20][];
- C) int [][] a = new int [10][];

**Explanation:**

Multidimensional arrays in Java are just arrays within arrays. Multidimensional arrays are defined as rows and columns. The outer array must be initialized. If you look at the answers the outer arrays are initialized.

**Answer 12:**

- B) Compiler error , you cannot assign a value to final variable

**Explanation:**

In Java final variables are treated as constants ( comparing to other languages like Visual Basic and etc.) , once it is initialized you cannot change the values of primitive, if final variables are object references then you cannot assign any other references.

**Answer 13:**

- B) Contains any number of non-public classes and only one public class
- E) Package statements should appear only in the first line or before any import statements of source file

**Explanation:**

The source files always contains only one package statement, you cannot define multiple package statements and these statements must be before the import statements. At any point of time Java source files can have any number of non-public class definitions and only one public definition class. If you have any import statements those should be defined before class definition and after the package definition.

**Answer 14:**

- D) a = 1, b = -1

**Explanation:**

The operator `>>>` is unsigned right shift, the new bits are set to zero, so the `-1` is shifted 31 times and became `1` ( because `a` is defined as integer ). The operator `>>` is signed right shift operator, the new bits take the value of the MSB ( Most Significant Bit ) . The operator `<<` will behave like same as `>>>` operator. The shifting operation is applicable to only integer data types.

**Answer 15:**

- D) 4

**Explanation:**

The operator is bitwise XOR operator. The values of `a`, `b`, `c` are first converted to binary equivalents and calculated using `^` operator and the results are converted back to original format.

**Answer 16:**

- C) Test() method declaration

**Explanation:**

The abstract methods cannot have body. In any class if one method is defined as abstract the class should be defined as abstract class. So in our example the `Test()` method must be redefined.

**Answer 17:**

D) B b = new C(); // Line 4

**Explanation:**

According to the inheritance rules, a parent class references can appear to the child class objects in the inheritance chain from top to bottom. But in our example class B, and class C are in the same level of hierarchy and also these two classes does not have parent and child relationship which violates the inheritance rules.

**Answer 18:**

- A) s == s1
- B) s.equals(s1);

**Explanation:**

The string objects can be compared for equality using == or the equals() method ( even though these two have different meaning ). In our example the string objects have same wording but both are different in case. So the string object object comparison is case sensitive.

**Answer 19:**

- A) It is a blue print
- B) A new data type
- C) To provide multiple inheritance

**Explanation:**

One of the major fundamental change in Java comparing with C++ is interfaces. In Java the interfaces will provide multiple inheritance functionality. In Java always a class can be derived from only one parent, but in C++ a class can derive from multiple parents.

**Answer 20:**

- C) package com;  
import java.awt.\*;  
// Comments
- D) // Comments  
package com;  
import java.awt.\*;  
public class MyTest {}

**Explanation**

In a given Java source file, the package statement should be defined before all the import statement or the first line in the .java file provided if you do not have any comments or Javadoc definitions. The sequence of definitions are:

// Comments ( if any)  
Package definition  
Multiple imports  
Class definition

**Answer 21:**

- D) Compile time error

**Explanation:**

The code fails at the time Math class instantiation. The java.lang.Math class is final class and the

default constructor defined as private. If any class has private constructors , we cannot instantiate them from out the class ( except from another constructor ).

**Answer 22:**

- A) IOException
- B) MalformedURLException

**Explanation:**

In Java the URL class will throw "MalformedURLException" while constructing the URL, and while reading incoming stream of data they will throw IOException..

**Answer 23:**

- D) Compile time error

**Explanation:**

In Java the constructors can throw exceptions. If parent class default constructor is throwing an exception, the derived class default constructor should handle the exception thrown by the parent.

**Answer 24:**

- A) Executing class A constructor followed by Executing class B constructor

**Explanation:**

In Java the constructors can throw exceptions. According to the Java language exceptions, if any piece of code throwing an exception it is callers worry is to handle the exceptions thrown by the piece of code. If parent class default constructor is throwing an exception, the derived class default constructor should handle the exception thrown by the parent. But in our example the non default constructor is throwing an exception if some one calls that constructor they have to handle the exception.

**Answer 25:**

- C) Compile time error while multiplication

**Explanation:**

This does not compile because according to the arithmetic promotion rules, the \* ( multiplication ) represents binary operator. There are four rules apply for binary operators. If one operand is float,double,long then other operand is converted to float,double,long else the both operands are converted to int data type. So in our example we are trying put integer into byte which is illegal.

**Answer 26:**

- A) setBounds(), setVisible(), setFont()
- B) add(), remove()
- C) setEnabled(), setVisible()

**Explanation:**

The component class is the parent class of all AWT components like Button, List, Label and etc. Using these methods you can set the properties of components. The add(), remove() methods are used to add PopMenu and to remove MenuComponent.

**Answer 27:**

- C) Frame

**Explanation:**

Java supports two kinds of menus, pull-down and pop-up menus. Pull-down menus are accessed via a menu bar. Menu bars are only added to Frames.

**Answer 28:**

- A) Frame's default layout manager is BorderLayout
- D) Canvas has no default behavior or appearance

**Explanation:**

In Java AWT each container has its own default layout manager implemented as part of its implementation. For example Frame has default layout manager is BorderLayout , Applet has FlowLayout and etc. The Canvas is kind of component where you can draw custom drawings. The Canvas generates Mouse, MouseMotion, and Key events .

**Answer 29:**

- A) void addItem(String s), int getRows()
- C) int[] getSelectedIndexes(), int getItemCount()
- D) int[] getSelectedIndexes(), String[] getSelectedItems()

**Explanation:**

The java.awt.List has methods to select , count the visible rows.

|                                 |                                                            |
|---------------------------------|------------------------------------------------------------|
| void addItem(String s) -->      | adds an item to the bottom of the list                     |
| int getRows() -->               | returns the number of visible lines in the list            |
| int[] getSelectedIndexes() -->  | returns array of indexes currently selected items          |
| int getItemCount() -->          | returns the number of items in the list                    |
| String[] getSelectedItems() --> | returns array of string values of currently selected items |

**Answer 30:**

- A) java.awt.TextArea.SCROLLBARS\_NONE
- C) java.awt.TextField generates Key events and Action events
- D) java.awt.TextArea can be scrolled using the <-- and --> keys.

**Explanation:**

The TextArea and TextField are the subclasses of TextComponent class. The TextArea has static fields to give you the functionality of horizontal and vertical scroll bars. These are the following fields:

java.awt.TextArea.SCROLLBARS\_BOTH  
java.awt.TextArea.SCROLLBARS\_NONE  
java.awt.TextArea.SCROLLBARS\_HORIZONTAL\_ONLY  
java.awt.TextArea.SCROLLBARS\_VERTICAL\_ONLY

The TextArea and TextField will generate Key events and TextField will generate Action events apart from the Key events.

**Answer 31:**

- A) Compile time error

**Explanation:**

In Java there are two types of methods , static and non static methods. Static methods are belong to class and non static methods are belongs to instances. So from a non static method you can call static as well as static methods, but from a static method you cannot call non static methods ( unless create a instance of a class ) but you can call static methods.

**Answer 32:**

- C) At instantiation of FileReader object.

**Explanation:**

While constructing the FileReader object, if the file is not found in the file system the "FileNotFoundException" is thrown. If the input stream is closed before reading the stream throws IOException.

**Answer 33:**

C) While writing to the stream f.write(buffer[i]) throws an IOException

**Explanation:**

While writing to a IO stream if the stream is closed before writing throws an IOException. In our example the f ( stream ) is closed via f1 reference variable before writing to it.

**Answer 34:**

D) While constructing the FileWriter f2 = new FileWriter("MyFile2.txt");

**Explanation:**

Constructing the FileWriter object, if the file already exists it overrides it (unless explicitly specified to append to the file). FileWriter will create the file before opening it for output when you create the object. In the case of read-only files, if you try to open and IOException will be thrown.

**Answer 35:**

A) A File class can be used to create files and directories

B) A File class has a method mkdir() to create directories

C) A File class has a method.mkdirs() to create directory and its parent directories.

**Explanation:**

The File class has three methods to create empty files, those are createNewFile(), createTempFile(String prefix, String suffix) and createTempFile(String prefix, String suffix, File directory).

File class has two utility methods mkdir() and mkdirs() to create directories. The mkdir() method creates the directory and returns either true or false. Returning false indicates that either directory already exists or directory cannot be created because the entire path does not exists. In the situation when the path does not exists use the mkdirs() method to create directory as well as parent directories as necessary.

**Answer 36:**

A) True

**Explanation:**

File class can be used to navigate the directories in the underlying file system. But in the File class there is no way you change the directory . Constructing the File class instance will always point to only one particular directory. To go to another directory you may have to create another instance of a File class.

**Answer 37:**

A) FileOutputStream(FileDescriptor fd)

B) FileOutputStream(String fileName, boolean append)

**Explanation:**

The valid FileOutputStream constructors are:

- FileOutputStream(String fileName)
- FileOutputStream(File file)
- FileOutputStream(FileDescriptor fd)

- FileOutputStream(String fileName, boolean append)

**Answer 38:**

- C) new Font ("Serif", Font.PLAIN, 24);  
 D) new Font ("SanSerif", Font.ITALIC, 24);  
 E) new Font ("SanSerif", Font.BOLD+Font.ITALIC, 24);

**Explanation:**

The Font class gives you to set the font of a graphics context. While constructing the Font object you pass font name, style, and size of the font. The font availability is dependent on platform. The Font class has three types of font names called "Serif", "SanSerif", "Monospaced" these are called in JDK 1.1 and after "Times Roman", Helvetica" and "Courier".

**Answer 39:**

- A) public void update ( Graphics g ) {  
     paint( g ) ;  
 }

**Explanation:**

If you want accumulate the previous information on the graphics context override the update() and inside the method call the paint() method by passing the graphics object as an argument. The repaint() method always calls update() method

**Answer 40:**

- A) public void init() {  
     Image i = getImage ( getDocumentBase(), "Logo.jpeg");  
 }

**Explanation:**

The Applet and Toolkit classes has a method getImage() , which has two forms:

- getImage(URL file)
- getImage(URL dir, String file)

These are two ways to refer an image in the server . The Applet class getDocumentBase() methods returns the URL object which is your url to the server where you came from or where your image resides.

**Answer 41:**

- D) Object, Check, ICheck

**Explanation:**

The instanceof operator checks the class of an object at runtime. In our example o refers to Object class and Check and ICheck refers to the subclasses of Object class. Due to the inheritance hierarchy Check and ICheck returns true.

**Answer 42:**

- E) None of the above

**Explanation:**

There is no way to call a particular thread from a waiting pool. The methods notify() will calls thread from waiting pool, but there is no guaranty which thread is invoked. The method notifyAll() method puts all the waiting threads from the waiting pool in ready state.

**Answer 43:**

- A) You can synchronize entire method
- C) Block of code can be synchronized
- D) The notify() and notifyAll() methods are called only within a synchronized code

**Explanation:**

The keyword controls accessing the single resource from multiple threads at the same time. A method or a piece of code can be synchronized, but there is no way to synchronize a calls. To synchronize a method use synchronized keyword before method definition. To synchronize block of code use the synchronized keyword and the arbitrary instance.

**Answer 44:**

- A) Compile time error

**Explanation:**

The IOException never thrown here. The exception is thrown is InterruptedException. To correct instead of catching IOException use InterruptedException.

**Answer 45:**

- D) Frame does not visible

**Explanation:**

The Frame is not going to be visible unless you call setVisible(true) method on the Frame's instance. But the frame instance is available in computers memory. If do not set the size of the Frame you see default size of the frame ( i.e.. in minimized mode)

**Answer 46:**

- C) public abstract void Test();
- D) native void doSomthing( int i );

**Explanation:**

The abstract methods does not have method bodies. In any given class if one method is defined as abstract the class must defined as abstract class.

**Answer 47:**

- A) `toString()` method is defined in Object class.
- C) `wait()`, `notify()`, `notifyAll()` methods are defined in Object class and used for Thread communication.
- D) `toString()` method provides string representation of an Object state.

**Explanation:**

The `toString()` is defined in Object class the parent all classes which will gives you the string representation of the object's state. This more useful for debugging purpose. The `wait()`, `notify()`, `notifyAll()` methods are also defined in Object class are very helpful for Thread communication. These methods are called only in synchronized methods.

**Answer 48:**

- A) `public transient int val;`
- D) `synchronized ( this ) {`  
    `// Assume that "this" is an arbitrary object instance.`  
    `}`

**Explanation:**

To define transient variables just include "transient" keyword in the variable definition. The transient variables are not written out anywhere, this is the way when you do object serialization not to write the critical data to a disk or to a database.

The "synchronized" keyword controls the single resource not to access by multiple threads at the same time. The synchronized keyword can be applied to a method or to a block of code by passing the arbitrary object instance name as an argument.

**Answer 49:**

- A) Double
- C) Integer
- D) Byte

**Explanation:**

In Java all the primitive data types have wrapper classes to represent in object format and will throw "NumberFormatException". The Boolean does not throw "NumberFormatException" because while constructing the wrapper class for Boolean which accepts string also as an argument.

**Answer 50:**

- A) Math.abs(3.0) returns 3.0  
Math.abs(-3.4) returns 3.4
- B) Math.ceil(3.4) returns 4.0  
Math.ceil(-3.4) returns -3.0
- C) Math.floor(3.4) returns 3.0  
Math.floor(-3.4) returns -4.0
- D) Math.round(3.4) returns 3  
Math.round(-3.4) returns -3

**Explanation:**

The Math class abs() method returns the absolute values, for negative values it strips off the negation and returns positive absolute value. This method returns always double value.

The method ceil(), returns double value not less than the integer (in our case 3). The other ways to say this method returns max integer value. (All the decimals are rounded to 1 and is added to integer value). For negative values it behaves exactly opposite.

The method floor() is exactly reverse process of what ceil() method does.

The round() method just rounds to closest integer value.

**Answer 51:**

- A) OuterClass.InnerClass inner = new OuterClass().new InnerClass();

**Explanation:**

The static methods are class level methods to execute those you do not need a class instance. If you try to execute any non static method or variables from static methods you need to have instance of a class. In our example we need to have OuterClass reference to execute InnerClass method.

**Answer 52:**

C) By making s1 as final variable

**Explanation:**

In Java it is possible to declare a class inside a method. If you do this there are certain rules will be applied to access the variables of enclosing class and enclosing methods. The classes defined inside any method can access only final variables of enclosing class.

**Answer 53:**

C) This is convention adopted by Sun , to insure that there is no ambiguity between packages and inner classes.

**Explanation:**

This is convention adopted to distinguish between packages and inner classes. If you try to use Class.forName() method the call will fail instead use getClass().getName() on an instance of inner class.

**Answer 54:**

B) Compile time error

**Explanation:**

The method in Vector class , addElement() returns type of void which you cannot return in our example. The myVector() method in our MyVector class returns only type of Vector.

**Answer 55:**

C) -1

**Explanation:**

Internally the x value first gets inverted ( two's compliment ) and then shifted 1 times. First when it is inverted it becomes negative value and shifted by one bit.

**Answer 56:**

C) InputEvent

**Explanation:**

The InputEvent class has method getWhen() which returns the time when the event took place and the return type is long.

**Answer 57:**

B) getSource() method is defined in java.util.EventObject class  
C) getID() method is defined in java.awt.AWTEvent class

**Explanation:**

The super class of all event handling is java.util.EventObject which has a method called getSource() , which returns the object that originated the event.

The subclass of EventObject is AWTEvent has a method getID() , which returns the ID of the event which specifies the nature of the event.

**Answer 58:**

A) A listener object is an instance of a class that implements a listener interface.  
B) An event source is an object , which can register listener objects and sends notifications

whenever event occurs.  
C) Event sources fires the events.

**Explanation:**

The event listeners are instance of the class that implements listener interface . An event source is an instance of class like Button or TextField that can register listener objects and sends notifications whenever event occurs.

**Answer 59:**

- B) As a Queue
- D) As a Stack

**Explanation:**

This implements java.util.List interface and uses linked list for storage. A linked list allows elements to be added, removed from the collection at any location in the container by ordering the elements. With this implementation you can only access the elements in sequentially. You can easily treat the LinkedList as a stack, queue and etc., by using the LinkedList methods.

**Answer 60:**

- A) Compilation error

**Explanation:**

The method throwMethod() is throwing and type Exception class instance, while catching the exception you are catching the subclass of Exception class.

## operators

### Question 1

```
class EBH019 {
 public static void main (String args[]) {
 int i1 = 0xffffffff, i2 = i1 << 1;
 int i3 = i1 >> 1, i4 = i1 >>> 1;
 System.out.print(Integer.toHexString(i2) + ", ");
 System.out.print(Integer.toHexString(i3) + ", ");
 System.out.print(Integer.toHexString(i4));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: ffffffff,fffffff,fffffff
- b. Prints: ffffffff,fffffff,7fffffff
- c. Prints: ffffffff,7fffffff,fffffff
- d. Prints: ffffffff,7fffffe,7fffffe
- e. Prints: ffffffe,fffffff,fffffff
- f. ANS Prints: ffffffe,fffffff,7fffffe
- g. Prints: ffffffe,7fffffff,fffffff
- h. Prints: ffffffe,7fffffff,7fffffff
- i. Run-time error
- j. Compile-time error
- k. None of the above

### Question 2

```
class EBH201 {
 public static void main (String[] args) {
 int a = 1 || 2 ^ 3 && 5;
 int b = ((1 || 2) ^ 3) && 5;
 int c = 1 || (2 ^ (3 && 5));
 System.out.print(a + "," + b + "," + c);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0
- b. Prints: 0,0,3
- c. Prints: 0,3,0
- d. Prints: 0,3,3
- e. Prints: 3,0,0
- f. Prints: 3,0,3
- g. Prints: 3,3,0
- h. Prints: 3,3,3
- i. Run-time error
- j. ANS Compile-time error
- k. None of the above

### Question 3

```
cl
ass EBH202 {
 static boolean a, b, c;
 public static void main (String[] args) {
 boolean x = (a = true) || (b = true) && (c = true);
 System.out.print(a + "," + b + "," + c);
```

}}

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. ANS Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 4

```
class EBH203 {
 static boolean a, b, c;
 public static void main (String[] args) {
 boolean x = a || (b = true) && (c = true);
 System.out.print(a + "," + b + "," + c);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. ANS Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 5

```
class EBH011 {
 public static void main (String[] args) {
 float a = Float.POSITIVE_INFINITY;
 double b = Double.POSITIVE_INFINITY;
 double c = Double.NaN;
 System.out.print((a == b)+", "+(c == c)+", "+(c != c));
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. ANS Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true

- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 6

```
class EBH012 {
 public static void main (String[] args) {
 byte x = 3, y = 5;
 System.out.print((-x == ~x + 1)+", "+(-y == ~y + 1));
 }}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. ANS Prints: true,true
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 7

```
class EBH013 {
 public static void main (String[] args) {
 byte x = 3, y = 5;
 System.out.print((~x == -x - 1)+", "+(~y == -y - 1));
 }}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. ANS Prints: true,true
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 8

```
class EBH014 {
 public static void main (String[] args) {
 byte x = 3, y = 5;
 System.out.print((y % x) + ",");
 System.out.print(y == ((y/x)*x + (y%x)));
 }}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,true
- b. ANS Prints: 2,true
- c. Prints: 1,false
- d. Prints: 2,false
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 9

```

class Color {}

class Red extends Color {}
class Blue extends Color {}
class A {
 public static void main (String[] args) {
 Color color1 = new Red(); Red color2 = new Red();
 boolean b1 = color1 instanceof Color;
 boolean b2 = color1 instanceof Blue;
 boolean b3 = color2 instanceof Blue;
 System.out.print(b1+", "+b2+", "+b3);
 }
}

```

What is the result of attempting to compile and run the program?

- a. false,false,false
- b. false,false,true
- c. false,true,false
- d. false,true,true
- e. true,false,false
- f. true,false,true
- g. true,true,false
- h. true,true,true
- i. Run-time error
- j.ANS Compile-time error
- k. None of the above

Question 10

```

class EBH020 {
 public static void main (String[] args) {
 int a = 1 | 2 ^ 3 & 5;
 int b = ((1 | 2) ^ 3) & 5;
 int c = 1 | (2 ^ (3 & 5));
 System.out.print(a + "," + b + "," + c);
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0
- b. Prints: 0,0,3
- c. Prints: 0,3,0
- d. Prints: 0,3,3
- e. Prints: 3,0,0
- f.ANS Prints: 3,0,3
- g. Prints: 3,3,0
- h. Prints: 3,3,3
- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 11

```

class EBH025 {
 public static void main (String args[]) {
 int i1 = 0xffffffff, i2 = i1 << 33;
 int i3 = i1 << (33 & 0x1f);
 System.out.print(Integer.toHexString(i2) + ", ");
 }
}

```

```
 System.out.print(Integer.toHexString(i3));
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0
- b. Prints: 0,ffffffe
- c. Prints: 0,fffffff
- d. Prints: ffffffff,fffffff
- e. Prints: ffffffff,ffffffe
- f. Prints: ffffffe,fffffff
- g.ANS Prints: ffffffe,ffffffe
- h. Run-time error
- i. Compile-time error
- j. None of the above

Question 12

```
class EBH001 {
 static int m(int i) {System.out.print(i + " , "); return i;}
 public static void main(String s[]) {
 m(m(1) - m(2) + m(3) * m(4));
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1, 2, 3, 4, 8,
- b.ANS Prints: 1, 2, 3, 4, 11,
- c. Prints: 3, 4, 1, 2, 11,
- d. Run-time error
- e. Compile-time error
- f. None of the above

Question 13

```
class EBH002 {
 static int m(int i) {System.out.print(i + " , "); return i;}
 public static void main(String s[]) {
 m(m(1) + m(2) % m(3) * m(4));
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1, 2, 3, 4, 0,
- b. Prints: 1, 2, 3, 4, 3,
- c.ANS Prints: 1, 2, 3, 4, 9,
- d. Prints: 1, 2, 3, 4, 12,
- e. Prints: 2, 3, 4, 1, 9,
- f. Prints: 2, 3, 4, 1, 3,
- g. Run-time error
- h. Compile-time error
- i. None of the above

Question 14

```
class EBH005 {
 public static void main (String[] s) {
 byte b = 127; b <= 2;System.out.println(b);
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: -4
- b. Prints: -3
- c. Prints: -2
- d. Prints: 0
- e. Prints: 1
- f. Prints: 127
- g. Prints: 508
- h. Run-time error
- i. Compile-time error
- j. None of the above

Question 15

```
class EBH007{
 public static void main (String[] s) {
 byte b = 5; System.out.println(b<<33);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: -1
- b. Prints: 0
- c. Prints: 1
- d. Prints: 5
- e. Prints: 10
- f. Run-time error
- g. Compile-time error
- h. None of the above

Question 16

```
class EBH015 {
 public static void main (String[] args) {
 System.out.print((new Object() instanceof Object)+",");
 System.out.print((new Object() instanceof String)+",");
 System.out.print((new String() instanceof Object));
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 17

```
class EBH101 {
 static int m(int i) {System.out.print(i + " , "); return i;}
 public static void main(String s[]) {
 int i = 1; m(m(++i) + m(i++) + m(-i) + m(i++));
 } }
```

}}

What is the result of attempting to compile and run the above program?

- a. Prints: 1, 2, 3, 4, 10,
- b. Prints: 1, 2, -3, 4, 4,
- c. Prints: 2, 2, -3, -3, -2,
- d. Prints: 2, 2, -3, 3, 4,
- e. Prints: 2, 3, -3, -2, 0,
- f. Prints: 2, 3, -3, 4, 6,
- g. Prints: 2, 3, 4, 5, 14,
- h. Run-time error
- i. Compile-time error
- j. None of the above

Question 18

```
class EBH102 {
 static int m(int i) {System.out.print(i + ","); return i;}
 public static void main(String s[]) {
 int i = 1, j = m(i++) + m(i++) * m(i++) + m(i++);
 System.out.print(j % 5);
 } }
```

What is the result of attempting to compile and run the above program?

- a. Prints: 1,2,3,4,0
- b. Prints: 1,2,3,4,1
- c. Prints: 1,2,3,4,2
- d. Prints: 1,2,3,4,3
- e. Prints: 1,2,3,4,4
- f. Prints: 1,2,3,4,5
- g. Run-time error
- h. Compile-time error
- i. None of the above

Question 19

```
class EBH103 {
 public static void main (String[] args) {
 byte a = 1, b = 2, c = (byte)a++, d = (byte)++b, e = (byte)a + b;
 System.out.print(c + d + e);
 } }
```

What is the result of attempting to compile and run the above program?

- a. Prints: 1 2 3
- b. Prints: 6
- c. Prints: 2 3 5
- d. Prints: 10
- e. Prints: 1 3 4
- f. Prints: 8
- g. Prints: 1 3 5
- h. Prints: 9
- i. Run-time error.
- j. Compile-time error.
- k. None of the above

Question 20

```

class EBH204 {
 static boolean m1(String s, boolean b) {
 System.out.print(s + (b ? "T" : "F"));
 return b;
 }
 public static void main(String[] args) {
 m1("A",m1("B",false) || m1("C",true) || m1("D",false));
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: ATBFCT
- b. Prints: ATBFCTDF
- c. Prints: BFCTAT
- d. Prints: BTCTDFAT
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 21

```

class EBH104 {
 static int m(int i) {System.out.print(i + ", "); return i;}
 public static void main(String s[]) {
 int i = 1; m(m(++i) - m(i++) + m(-i) * m(~i));
 }
}

```

What is the result of attempting to compile and run the above program?

- a. Prints: 2, 2, -3, -4, 8,
- b. Prints: 2, 2, -3, -4, 12,
- c. Prints: 2, 3, -3, -4, 7,
- d. Prints: 1, 1, 1, 1, 0,
- e. Prints: 2, 2, -2, -2, 4,
- f. Prints: 2, 3, -2, -2, 3,
- g. Prints: -1, -2, 2, 2, 0,
- h. Run-time error
- i. Compile-time error
- j. None of the above

Question 22

```

class EBH105 {
 static int m(int i) {System.out.print(i + ","); return 0;}
 public static void main (String[] args) {
 int i = 0; i = i++ + m(i); System.out.print(i);
 }
}

```

What is the result of attempting to compile and run the above program?

- a. Prints: 0,0
- b. Prints: 1,0
- c. Prints: 0,1
- d. Prints: 1,1
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 23

```
class EBH106 {
 public static void main(String args[]) {
 int a = 1; a += ++a + a++; System.out.print(a);
 }
}
```

What is the result of attempting to compile and run the above program?

- a. Prints: 3
- b. Prints: 4
- c. Prints: 5
- d. Prints: 6
- e. Prints: 7
- f. Run-time error
- g. Compile-time error
- h. None of the above

Question 24

```
class EBH107 {
 static int m(int i) {System.out.print(i + ","); return i;}
 public static void main(String s[]) {
 int i=0, j = m(++i) + m(++i) * m(++i) % m(++i) + m(++i);
 System.out.print(j%5);
 }
}
```

What is the result of attempting to compile and run the above program?

- a. Prints: 1,2,3,4,5,0
- b. Prints: 1,2,3,4,5,1
- c. Prints: 1,2,3,4,5,2
- d. Prints: 1,2,3,4,5,3
- e. Prints: 1,2,3,4,5,4
- f. Prints: 1,2,3,4,5,5
- g. Run-time error
- h. Compile-time error
- i. None of the above

Question 25

```
class EBH108 {
 public static void main(String s[]) {
 int i=0, j = ++i + ((++i * ++i) % ++i) + ++i;
 System.out.print(j%5);
 }
}
```

What is the result of attempting to compile and run the above program?

- a. Prints: 1
- b. Prints: 2
- c. Prints: 3
- d. Prints: 4
- e. Prints: 5
- f. Run-time error
- g. Compile-time error
- h. None of the above

Question 26

```
class EBH023 {
 static String m1(boolean b){return b?"T":"F";}
```

```

public static void main(String [] args) {
 boolean b1 = false?false:true?false:true?false:true;
 boolean b2 = false?false:(true?false:(true?false:true));
 boolean b3 = ((false?false:true)?false:true)?false:true;
 System.out.println(m1(b1) + m1(b2) + m1(b3));
}

```

What is the result of attempting to compile and run the program?

- a. Prints: FFF
- b. Prints: FFT
- c. Prints: FTF
- d. Prints: FTT
- e. Prints: TFF
- f. Prints: TFT
- g. Prints: TTF
- h. Prints: TTT
- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 27

```

class EBH024 {
 public static void main(String[] args) {
 int i1 = 15;
 String b1 = (i1>30)?"Red":(i1>20)?"Green":(i1>10)?"Blue":"Violet";
 String b2 =
(i1>30)?"Red":((i1>20)?"Green":((i1>10)?"Blue":"Violet"));
 System.out.println(b1 + "," + b2);
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: Red,Red
- b. Prints: Green,Green
- c. Prints: Blue,Blue
- d. Prints: Violet,Violet
- e. Prints: Blue,Violet
- f. Prints: Violet,Blue
- g. Prints: Blue,Green
- h. Prints: Green,Blue
- i. Run-time error
- j. Compile-time error
- k. None of the above

Question 28

```

class EBH003 {
 static int m(int i) {System.out.print(i + " , "); return i;}
 public static void main(String s[]) {
 m(m(~1) + m(1|2) + m(1&2) + m(1^3) + m(1<<1));
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: -2, 3, 0, 3, 0, 6,
- b. Prints: -2, 3, 0, 2, 1, 4,

- c. Prints: -2, 3, 0, 2, 2, 5,
- d. Prints: -2, 3, 0, 3, 2, 6,
- e. Prints: -1, 3, 0, 3, 2, 7,
- f. Prints: -2, 0, 3, 3, 0, 6,
- g. Prints: -1, 0, 3, 2, 1, 4,
- h. Prints: -2, 0, 3, 2, 2, 5,
- i. Prints: -2, 0, 3, 3, 2, 6,
- j. Prints: -1, 0, 3, 3, 2, 7,
- k. Run-time error
- l. Compile-time error
- m. None of the above

Question 29

```
class EBH021 {
 public static void main(String[] args) {
 System.out.print((-1 & 0x1f) + "," + (8 << -1));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0
- b. Prints: -1,4
- c. Prints: 0x1f,8
- d. Prints: 31,16
- e. Run-time error
- f. Compile-time error
- g. None of the above

Answers:      Answer      Remark

1      f      Prints: fffffffe, fffffff, 7fffffff      If the left-hand operand of a shift operator, `<<`, `>>` and `>>>`, is of type int, then the shift distance is always within the range of 0 to 31, inclusive; and is specified by the least significant 5 bits of the right hand operand. Similarly, if the left-hand operand of a shift operator is of type long, then the shift distance is always within the range of 0 to 63, inclusive; and is specified by the least significant 6 bits of the right hand operand. The left shift operator, `<<`, shifts each bit of the left operand to the left a distance specified by the shift distance. A number of bits that is equal to the shift distance are shifted out of the left-most bit position and are lost. A number of bits that is equal to the shift distance are shifted in at the right. The signed right shift operator, `>>`, shifts each bit of the left operand to the right a distance specified by the shift distance. A number of bits that is equal to the shift distance are shifted out of the right-most bit position and are lost. A number of bits that is equal to the shift distance are shifted in at the left. The value of each bit that is shifted in at the left is equal to the value of the sign bit. The signed right shift operator maintains the sign of the left operand. The unsigned right shift operator, `>>>`, is similar to the signed right shift operator except for the fact that each bit shifted in at the left is zero.

2      j      Compile-time error      Both operands of the conditional and operator and the conditional or operator must be of type boolean.

3 e Prints: true,false,false The conditional and expression is not evaluated, because the left hand operand of the conditional or expression is true. The original expression is as follows:  $x = (a = \text{true}) \mid\mid (b = \text{true}) \&\& (c = \text{true})$ . The left hand operand of the conditional or expression is the result of the assignment expression,  $(a = \text{true})$ . Since the left hand operand of the conditional or expression is true, the right hand operand will not be evaluated. In this case, the right hand operand is the conditional and expression. Consequently, neither operand of the conditional and expression is evaluated and the variables, b and c, maintain their default values of false.

4 d Prints: false,true,true The right hand operand of the conditional or operator is evaluated only if the left hand operand is false. The right hand operand of the conditional and operator is only evaluated if the left hand operand is true. In this case, the left hand operand of the conditional or operator is false, so the right hand operand must also be evaluated. The left hand operand of the conditional and operator is the result of the conditional or expression, true, so the right hand operand is evaluated.

5 f Prints: true,false,true The positive infinity of type float is promoted to the positive infinity of type double. NaN is not equal to anything including itself.

6 d Prints: true,true The sign of an integral numeric type is changed by inverting all of the bits and by adding one.

7 d Prints: true,true The bitwise complement operator produces the same result as changing the sign and subtracting one. Please note that the operand must be an integral type. The bitwise complement operator can not be applied to a floating-point value.

8 b Prints: 2,true Suppose the left operand were divided by the right operand. The remainder operator returns the remainder of the division operation. For integral types, the identity,  $(y == ((y/x)*x+(y%x)))$ , is always true.

9 j Compile-time error The type of the reference color2 is Red. Since Red is not a subclass or a superclass of Blue, the expression color2 instanceof Blue is rejected at compile-time. Please note: The expression, x instanceof T, produces a compile-time error whenever the cast expression (T)x produces a compile-time error. If the program had been able to compile and run, the expression color1 instanceof Color would evaluate to true at run-time. The reference color1 refers to an instance of type Red. Since Red is a subclass of Color, the expression color1 instanceof Color would evaluate to true at run-time. The expression, color1 instanceof Blue would evaluate to false at run-time. The reference, color1, is of type Color. Since Color is a superclass of Blue, the expression, color1 instanceof Blue, is accepted at compile-time. The type of the object instance referenced by color1 is Red. Since Red is not Blue or a subclass of Blue, the expression, color1 instanceof Blue, would be false at run-time.

10 f Prints: 3,0,3 Java evaluates operands from left to right while respecting operator precedence. The order of operator precedence starting with the lowest is as follows: |, ^, &. Although complete memorization of the operator precedence chart is not necessary, it is a good idea to memorize the three levels that appear in this question.

11 g Prints: ffffffe,ffffffe For each of the three shift operators, <<, >> and >>>, the shift distance is specified by the right hand operand. If the left operand is of type int, then the shift distance

is always within the range 0 to 31, inclusive; and the following expression is always true: `(int1 << shift) == (int1 << (shift & 0x1f))`. The hexadecimal representation of decimal 31 is 0x1f and the binary representation is 0001 1111. The hexadecimal representation of decimal 33 is 0x21 and the binary representation is 0010 0001. The expression `i1 << (33 & 0x1f)` is equivalent to `(0xffffffff << (0x21 & 0x1f))`. Evaluation of the right hand operand of the shift operator produces `(0xffffffff << 1)`. The final result is 0xffffffe. Similarly, if the left operand is of type long, then the shift distance is always within the range 0 to 63, inclusive; and the following expression is always true: `(long1 << shift) == (long1 << (shift & 0x3f))`.

12 b Prints: 1, 2, 3, 4, 11, The expression can be simplified as follows:  $j = 1 - 2 + 3 * 4 = 11$ . The original expression is as follows: `m(m(1) - m(2) + m(3) * m(4))`. Simplification step one. Evaluate each operand from left to right: `m(1 - 2 + 3 * 4)`. Step two. Add parentheses to indicate operator precedence: `m(1 - 2 + (3 * 4))`. Step three. Evaluate the inner-most parentheses: `m(1 - 2 + 12)`. Step four: Work through the expression from left to right.  $j = 11$ .

13 c Prints: 1, 2, 3, 4, 9, The expression can be simplified as follows:  $1 + 2 \% 3 * 4 = 9$ . The original expression is as follows: `m(m(1) + m(2) \% m(3) * m(4))`. Simplification step one. Evaluate each operand from left to right: `m(1 + 2 \% 3 * 4)`. Step two. Add parentheses to indicate operator precedence and associativity: `m(1 + ((2 \% 3) * 4))`. Step three. Evaluate the inner-most parentheses: `m(1 + (2 * 4))`. Step four. Evaluate the inner-most parentheses: `m(1 + 8)`. The result is 9.

14 a Prints: -4 If the left-hand operand of the shift operator is of type byte, short, or char then the left operand is promoted to a 32 bit int and all four bytes are shifted. When a variable of type int with a value of 127 is shifted two bits to the left, the result is 508. The compound assignment operator includes an implicit cast to the type of the left-hand operand. The expression, `E1 op= E2`, is equivalent to `E1=(T)((E1) op (E2))`, where T is the type of the left hand operand. Therefore, when 508 is cast to an eight bit byte, the three most significant bytes (24 bits) are discarded leaving only the least significant byte (8 bits). The result is the binary value, 11111100, which is the two's complement representation of negative four.

15 e Prints: 10 If the left-hand operand of the shift operator is of type byte, short, or char then the left operand is promoted to a 32 bit int and all four bytes are shifted. If the promoted type of the left-hand operand is of type int, then the shift distance is always within the range of 0 to 31, inclusive; and is specified by the least significant 5 bits of the right-hand operand. In this case, the shift distance is 33, and the five least significant bits are 00001; so the shift distance is one bit. Note: If the type of the left hand operand is long, then the least significant six bits of the right hand operand are used.

16 f Prints: true,false,true The left operand of the instanceof operator must be null or a reference to an instance of an Object or a subclass of Object. The right operand of the instanceof operator must be a class type, interface type or array type. If the left operand is a reference to an instance of the type specified by the right operand or if the left operand is a reference to an instance of a subclass of the type specified by the right operand, then instanceof returns true.

17 d Prints: 2, 2, -3, 3, 4, The expression can be simplified as follows:  $j = 2 + 2 + -3 + 3 = 4$ . The original expression is as follows:  $j = ++i + i++ + -i + i++$ . Simplification step one. Evaluate the unary expressions from left to right:  $j = 2 + 2 + -3 + 3$ . Step two.

Complete the evaluation of the simplified expression:  $j = 4$ .

18 b Prints: 1,2,3,4,1 The expression can be simplified as follows:  $j = 1 + (2 * 3) + 4 = 11$ . The original expression is as follows:  $j = m(i++) + m(i++) * m(i++) + m(i++)$ . The method, m, prints and then returns the value of the parameter, so the original expression is equivalent to the following:  $j = i++ + i++ * i++ + i++$ . Step one. Work through the expression from left to right to evaluate the unary expressions:  $j = 1 + 2 * 3 + 4$ . Step two. Add parentheses to indicate operator precedence:  $j = 1 + (2 * 3) + 4$ . Step three. Work through the simplified expression:  $j = 1 + 6 + 4 = 11$ . Step four. Evaluate the expression that is the argument of the print method:  $j \% 5 = 11 \% 5 = 1$ .

19 j Compile-time error. The precedence of the cast operator is higher than the precedence of the addition operator, so the cast applies only to variable a and not to the result of the addition. Binary numeric promotion causes the byte variables a and b to be promoted to type int before the addition operation, and the result of the addition is also of type int. The attempt to assign the int result to the byte variable e generates a possible loss of precision error.

20 c Prints: BFCTAT The right operand of the conditional or operator is evaluated only if the left hand operand is false. In this case, the left operand of the first conditional or operator is false so the right hand operand is evaluated. No further evaluation of the expression is necessary so the right hand operand of the second conditional or operator is not evaluated.

21 b Prints: 2, 2, -3, -4, 12, The original expression is as follows:  $m(m(++i) - m(i++) + m(-i) * m(~i))$ . The method, m, prints and then returns the value of the parameter, so the original expression is equivalent to the following:  $++i - i++ + -i * ~i$ . We can use a simplification process that evaluates the expression as follows. Step one. Work through the expression from left to right to evaluate the unary expressions:  $2 - 2 + -3 * -4$ . Step two. Add the parentheses to indicate operator precedence:  $2 - 2 + (-3 * -4)$ . Step three. Evaluate the innermost parentheses:  $2 - 2 + 12$ . Step four. Evaluate the simplified expression to produce the result, 12. The bitwise complement operator,  $\sim$ , inverts each bit of the operand. To avoid working in binary the same result can be obtained by changing the sign of the operand and then subtracting 1. The following identity is always true  $\sim x == -x - 1$ .

22 b Prints: 1,0 The expression,  $i = i++ + m(i)$ , can be reduced to,  $i = 0 + 0$ . The left operand of the addition expression is found to be zero, and the value of variable i is then incremented to 1. The right hand operand of the addition operation is evaluated next. The value, 1, is passed to method m. After printing the argument value, method m returns the value zero. The two operands of the addition operation are zero as is the result of the addition. The zero value serves as the right hand operand of the assignment statement, so the value, zero, is assigned to variable i.

23 c Prints: 5 The two statements,  $int a=1$  followed by  $a += ++a + a++$ , can be rewritten as the single statement,  $a=(int)((1)+(++a + a++))$ . Further evaluation produces  $a=(int)((1)+(2 + 2))$ . Generally speaking, a

compound assignment expression of the form E1 op= E2 can be rewritten as E1=(T)((E1)op(E2)) where T is the type of E1.

24 d Prints: 1,2,3,4,5,3 The expression can be simplified as follows:  $j = 1 + ((2 * 3) \% 4) + 5 = 8$ . The original expression is as follows:  $j = ++i + ++i * ++i \% ++i + ++i$ . Step one. Evaluate the unary expressions from left to right:  $j = 1 + 2 * 3 \% 4 + 5$ . Step two. Add parentheses to indicate operator precedence:  $j = 1 + ((2 * 3) \% 4) + 5$ . Step three. Evaluate the inner most parentheses:  $j = 1 + (6 \% 4) + 5$ . Repeat step three:  $j = 1 + 2 + 5$ . Repeat step three:  $j = 8$ . The argument of the print expression is:  $j \% 5$ . The result is:  $8 \% 5 = 3$ .

25 c Prints: 3 The expression can be simplified as follows:  $j = 1 + ((2 * 3) \% 4) + 5 = 8$ . The original expression is as follows:  $j = ++i + ((++i * ++i) \% ++i) + ++i$ . Step one. Evaluate the unary expressions from left to right:  $j = 1 + ((2 * 3) \% 4) + 5$ . Step two. Evaluate the inner-most parentheses:  $j = 1 + (6 \% 4) + 5$ . Step three: Evaluate the inner-most parentheses.  $j = 1 + 2 + 5$ . Step four: Work through the expression from left to right.  $j = 8$ . The argument of the print expression is:  $j \% 5$ . The result is:  $8 \% 5 = 3$ .

26 b Prints: FFT The expression used to assign variable b1 is equivalent to the expression used to assign variable b2. The results demonstrate that the conditional operator (?:) groups from right-to-left.

27 c Prints: Blue,Blue The expression used to assign variable b1 is equivalent to the expression used to assign variable b2. The results demonstrate that the conditional operator (?:) groups from right-to-left.

28 c Prints: -2, 3, 0, 2, 2, 5, The expression can be simplified as follows:  $-2+3+0+2+2=5$ . The original expression is as follows:  $m(m(\sim 1) + m(1|2) + m(1&2) + m(1^3) + m(1<<1))$ . Expr 1:  $\sim 1 = -1 - 1 = -2$ . Expr 2:  $1|2 = 0001 | 0010 = 0011 = 3$ . Expr 3:  $1&2 = 0001 \& 0010 = 0000$ . Expr 4:  $1^3 = 0001 ^ 0011 = 0010 = 2$ . Expr 5:  $1<<1 = 0001 << 1 = 0010 = 2$ . Note: The bitwise expressions were demonstrated using only four bits of the 32 bit int type values.

29 g None of the above Prints 31,0. The expression  $(-1 \& 0x1f)$  is equal to  $(0xffffffff \& 0x1f)$ , and both are equal to the hex value 0x1f or decimal 31. The expression  $(8 << -1)$  is equivalent to  $(8 << 0xffffffff)$ . If the left hand operand of a shift expression is of type int, then the right hand operand is implicitly masked with the value 0x1f. In other words, the expression  $(8 << -1)$  is equivalent to  $(8 << (-1 \& 0x1f))$ . By replacing -1 with the hexadecimal representation we have  $(8 << (0xffffffff \& 0x1f))$ . By evaluating the right hand operand we have  $(8 << 31)$ . When 8 is shifted 31 bits to the left, the result is zero since the only non-zero bit is lost as it is shifted beyond the most significant bit of the int data type.

## Arguments

### Question 1

```
class GFC401 {
 static int m1(int x) {return ++x;}
 public static void main (String[] args) {
 int x = 1;
 int y = m1(x);
```

```
 System.out.println(x + "," + y);
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1
- b. Prints: 1,2
- c. Prints: 2,1
- d. Prints: 2,2
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 2

```
class GRC10 {
 public static void main (String[] s) {
 System.out.print(s[1] + s[2] + s[3]);
 }
}
java GRC10 A B C D E F
```

What is the result of attempting to compile and run the program using the specified command line?

- a. Prints: ABC
- b. Prints: BCD
- c. Prints: CDE
- d. Prints: A B C
- e. Prints: B C D
- f. Prints: C D E
- g. Compile-time error
- h. Run-time error
- i. None of the above

Question 3

```
class GFC402 {
 static int x=1;
 void m1(int i) {x++; i++;}
 public static void main (String[] args) {
 int y=3; m1(y);
 System.out.println(x + "," + y);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,3
- b. Prints: 2,3
- c. Prints: 1,4
- d. Prints: 2,4
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 4

```
class GFC403 {
 private static int x=1;
 static void m1(int i) {x++; i++;}
 public static void main (String[] args) {
```

```
 int y=3; m1(y);
 System.out.println(x + "," + y);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,3
- b. Prints: 2,3
- c. Prints: 1,4
- d. Prints: 2,4
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 5

```
class GFC404 {
 private static int x=1;
 static void m1(int x, int y) {x++; y++;}
 public static void main (String[] args) {
 int y=3; m1(x, y);
 System.out.println(x + "," + y);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,3
- b. Prints: 2,3
- c. Prints: 1,4
- d. Prints: 2,4
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 6

```
class GFC301 {
 private String name;
 public GFC301(String name) {this.name = name;}
 public void setName(String name) {this.name = name;}
 public String getName() {return name;}
 public static void m1(GFC301 r1, GFC301 r2) {
 r1.setName("Bird");
 r2 = r1;
 }
 public static void main (String[] args) {
 GFC301 pet1 = new GFC301("Dog");
 GFC301 pet2 = new GFC301("Cat");
 m1(pet1,pet2);
 System.out.println(pet1.getName() + "," + pet2.getName());
 }
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: Dog,Cat
- b. Prints: Dog,Bird
- c. Prints: Bird,Cat
- d. Prints: Bird,Bird
- e. Run-time error

- f. Compile-time error
  - g. None of the above
- Question 7

```
class GFC303 {
 private String name;
 public GFC303(String name) {this.name = name;}
 public void setName(String name) {this.name = name;}
 public String getName() {return name;}
 public static void m1(GFC303 pet1, GFC303 pet2) {
 pet1 = new GFC303("Fish");
 pet2 = null;
 }
 public static void main (String[] args) {
 GFC303 pet1 = new GFC303("Dog");
 GFC303 pet2 = new GFC303("Cat");
 m1(pet1,pet2);
 System.out.println(pet1.getName() + "," + pet2.getName());
 }
}
```

- What is the result of attempting to compile and run the program?
- a. Prints: Dog,Cat
  - b. Prints: Dog,Fish
  - c. Prints: Fish,Cat
  - d. Prints: Fish,Fish
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- Question 8

```
class GFC304 {
 static void m1(int[] i1, int[] i2) {
 int[] i3 = i1; i1 = i2; i2 = i3;
 }
 public static void main (String[] args) {
 int[] i1 = {1}, i2 = {3}; m1(i1, i2);
 System.out.print(i1[0] + "," + i2[0]);
 }
}
```

- What is the result of attempting to compile and run the program?
- a. Prints: 1,1
  - b. Prints: 1,3
  - c. Prints: 3,1
  - d. Prints: 3,3
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- Question 9

```
class GFC305 {
 static void m1(int[] i1, int[] i2) {
 int i = i1[0]; i1[0] = i2[0]; i2[0] = i;
 }
 public static void main (String[] args) {
```

```
 int[] i1 = {1}, i2 = {3}; m1(i1, i2);
 System.out.print(i1[0] + "," + i2[0]);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1
- b. Prints: 1,3
- c. Prints: 3,1
- d. Prints: 3,3
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 10

```
class GFC306 {
 static int[] i1 = {1}, i2 = {3};
 static void m1(int[] i1) {
 int[] i3 = i1; i1 = i2; i2 = i3;
 }
 public static void main (String[] args) {
 m1(i1);
 System.out.print(i1[0] + "," + i2[0]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1
- b. Prints: 1,3
- c. Prints: 3,1
- d. Prints: 3,3
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 11

```
class GFC307 {
 static void m1(int[] i1, int[] i2) {
 i1 = i2 = null;
 }
 public static void main (String[] args) {
 int[] i1 = {1}, i2 = {3}; m1(i1, i2);
 System.out.print(i1[0] + "," + i2[0]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0
- b. Prints: 1,1
- c. Prints: 1,3
- d. Prints: 3,1
- e. Prints: null,null
- f. Run-time error
- g. Compile-time error
- h. None of the above

Question 12

```

class GFC308 {
 int[] i1 = {1}, i2 = {3};
 void m1() {
 m2(i1, i2);
 System.out.print(i1[0] + "," + i2[0]);
 }
 void m2(int[] i1, int[] i2) {
 int[] i3 = i1;
 this.i1 = i2;
 this.i2 = i3;
 }
 public static void main (String[] args) {
 new GFC308().m1();
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0
- b. Prints: 1,1
- c. Prints: 1,3
- d. Prints: 3,1
- e. Prints: null,null
- f. Run-time error
- g. Compile-time error
- h. None of the above

No. Answer Remark

1 b Prints: 1,2 Primitive arguments are passed by value. The method m1 increments the parameter x, and the result is returned. The local variable x of main remains unchanged.

2 b Prints: BCD The index for the first element of an array is zero so the first argument printed by this program is the second argument on the command line following the name of the class.

3 f Compile-time error Method m1 is an instance method, and must be invoked with reference to an instance of type GFC402. Method m1 can not be invoked from a static context.

4 b Prints: 2,3 Variables of primitive type are passed to methods by value: Only a copy of the value of the variable is passed to the method. While the method works with a local copy of the variable, the original variable remains unchanged by any actions performed on the method parameter. For that reason, method m1 does not change the value of the variable y in the main method. However, method m1 does have direct access to the class variable x and the content of the class variable is modified by method m1.

5 a Prints: 1,3 Variables of primitive type are passed to methods by value: Only a copy of the value of the variable is passed to the method. While the method works with a local copy of the variable, the original variable remains unchanged by any actions performed on the method parameter. For that reason, method m1 does not change the contents of the variable y in the main method or the class variable x.

6 c Prints: Bird,Cat The method m1 is invoked by the method invocation expression m1(pet1,pet2). The value of the reference variable denoted by the argument pet1 is used to initialize the method parameter r1. Inside of method m1, the method invocation expression r1.setName("Bird") uses the copy of the value of the argument pet1 to

assign a new name to the instance of GFC301 that is referenced by the local variable pet1 in the main method. Generally speaking, a reference parameter can be used to invoke methods on the referenced object and change the state of the object to the extent provided by the object's methods. The method invocation expression m1(pet1,pet2) has a second argument pet2, and the value of pet2 is used to initialize the method parameter r2. Inside of method m1, the assignment expression r2 = r1 changes the value of the method parameter r2; but the local variable of the main method denoted by the argument pet2 appearing in the method invocation expression m1(pet1,pet2) remains unchanged.

7 a Prints: Dog,Cat The method m1 is invoked by the method invocation expression m1(pet1,pet2). A copy of the reference argument pet1 is assigned to the method parameter pet1. Inside the body of method m1, the assignment expression pet1 = new GFC303("Fish") assigns a reference to a new instance of GFC303 to the method parameter pet1; but the argument pet1 that appears in the method invocation expression m1(pet1,pet2) and the local variable pet1 that is declared in the main method remain unchanged. The method invocation expression m1(pet1,pet2) has a second argument pet2, and a copy of pet2 is assigned to the method parameter pet2. Inside of method m1, the assignment expression pet2 = null changes the value of the method parameter pet2; but the argument pet2 appearing in the method invocation expression remains unchanged in the main method.

8 b Prints: 1,3 The method m1 is invoked by the method invocation expression m1(i1, i2). The argument i1 denotes a local variable of type int[] that is declared in the main method. The value of the argument is a reference to the array, and the argument value is used to initialize the method parameter i1 of method m1. Inside the body of m1, the expression i1 = i2 sets the value of parameter i1 to the value of parameter i2, but the change in the value of the parameter i1 does not change the original argument value or the local variable i1 of the main method that the argument denotes. Similarly, the assignment expression i2 = i3 in method m1 does not change the value of the local variable i1 declared in the main method.

9 c Prints: 3,1 Method m1 is not able to change the value of the local variables that are declared in the main method and serve as the arguments in the method invocation expression. However, method m1 is able to modify the contents of the arrays that are referenced by the method parameters.

10 a Prints: 1,1 The method m1 is invoked by the method invocation expression m1(i1). The argument i1 denotes the static member variable i1. Inside the declaration of method m1, the method parameter i1 shadows the static member variable i1. The assignment expression i1 = i2 assigns the value of the member variable i2 to the method parameter i1, but the member variable i1 remains unchanged. Inside of method m1, the member variable i2 is not shadowed; so the assignment expression i2 = i3 assigns the reference value contained by the method local variable i3 to the member variable i2. This question demonstrates that argument values are passed to method parameters by value, and the method parameter is only a copy of the argument value. A change made to the method parameter does not change the value of any variable that is shadowed by the parameter and does not change the value of the argument appearing in the method invocation expression.

11 c Prints: 1,3 Although the reference parameters i1 and i2 are reassigned inside of m1, the change has no impact outside of m1. Array references are passed by value: the invoked method gets a copy of the array reference.

12 d Prints: 3,1 Inside of method m2, the local variables i1 and i2 remain unchanged while the shadowed instance variables are changed.

#### Exception Handling

Question 1

```
class A {
 public static void main (String[] args) {
 Error error = new Error();
 Exception exception = new Exception();
 System.out.print((exception instanceof Throwable) + ",");
 System.out.print(error instanceof Throwable);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 2

```
class A {A() throws Exception {}} // 1
class B extends A {B() throws Exception {}} // 2
class C extends A {C() {}} // 3
```

Which of the following statements are true?

- a. class A extends Object.
- b. Compile-time error at 1.
- c. Compile-time error at 2.
- d. Compile-time error at 3.

Question 3

```
class A {
 public static void main (String[] args) {
 Object error = new Error();
 Object runtimeException = new RuntimeException();
 System.out.print((error instanceof Exception) + ",");
 System.out.print(runtimeException instanceof Exception);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compile-time error

- f. Run-time error
  - g. None of the above
- Question 4

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
 public static void main(String args[]) {
 int a,b,c,d,f,g,x;
 a = b = c = d = f = g = 0;
 x = 1;
 try {
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 } a++;
 }
 catch (Level2Exception e) {b++;}
 finally {c++;}
 }
 catch (Level1Exception e) { d++;}
 catch (Exception e) {f++;}
 finally {g++;}
 System.out.print(a+" "+b+" "+c+" "+d+" "+f+" "+g);
 }
}
```

- What is the result of attempting to compile and run the program?
- a. Prints: 0,0,0,1,0,0
  - b. Prints: 0,0,1,1,0,1
  - c. Prints: 0,1,1,1,0,1
  - d. Prints: 1,0,1,1,0,1
  - e. Prints: 1,1,1,1,0,1
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above

Question 5

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
 public static void main(String args[]) {
 int a,b,c,d,f,g,x;
 a = b = c = d = f = g = 0;
 x = 2;
 try {
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 } a++;
 }
 }
```

```

 catch (Level2Exception e) {b++;}
 finally {c++;}
 }
 catch (Level1Exception e) { d++;}
 catch (Exception e) {f++;}
 finally {g++;}
 System.out.print(a+" , "+b+" , "+c+" , "+d+" , "+f+" , "+g);
}
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,1,0,0,1
- b. Prints: 0,1,0,0,0,0
- c. Prints: 0,1,1,0,0,1
- d. Prints: 0,1,0,0,0,1
- e. Prints: 1,1,1,0,0,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 6

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
 public static void main(String args[]) {
 int a,b,c,d,f,g,x;
 a = b = c = d = f = g = 0;
 x = 3;
 try {
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 } a++; }
 catch (Level2Exception e) {b++;}
 finally {c++;}
 }
 catch (Level1Exception e) { d++;}
 catch (Exception e) {f++;}
 finally {g++;}
 System.out.print(a+" , "+b+" , "+c+" , "+d+" , "+f+" , "+g);
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1,1,0,0,1
- b. Prints: 0,1,1,0,0,1
- c. Prints: 0,1,0,0,0,0
- d. Prints: 0,1,0,0,0,1
- e. Prints: 0,0,1,0,0,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 7

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
 public static void main(String args[]) {
 int a,b,c,d,f,g,x;
 a = b = c = d = f = g = 0;
 x = 4;
 try {
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 case 4: throw new Exception();
 } a++;
 }
 catch (Level2Exception e) {b++;}
 finally{c++;}
 }
 catch (Level1Exception e) { d++;}
 catch (Exception e) {f++;}
 finally {g++;}
 System.out.print(a+" , "+b+" , "+c+" , "+d+" , "+f+" , "+g);
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,0,0,1
- b. Prints: 0,0,0,0,1,0
- c. Prints: 0,0,1,0,0,1
- d. Prints: 0,0,1,0,1,1
- e. Prints: 0,1,1,1,1,1
- f. Prints: 1,1,1,1,1,1
- g. Compile-time error
- h. Run-time error
- i. None of the above

Question 8

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
 public static void main(String args[]) {
 int a,b,c,d,f,g,x;
 a = b = c = d = f = g = 0;
 x = 5;
 try {
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 case 4: throw new Exception();
 } a++;
 }

```

```

 catch (Level2Exception e) {b++;}
 finally {c++;}
 }
 catch (Level1Exception e) { d++;}
 catch (Exception e) {f++;}
 finally {g++;}
 System.out.print(a+", "+b+", "+c+", "+d+", "+f+", "+g);
}}

```

What is the result of attempting to compile and run the program?

- a. Prints: 1,0,0,0,0,0
- b. Prints: 1,0,1,0,0,1
- c. Prints: 0,0,1,0,0,1
- d. Prints: 1,1,1,1,1,1
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 9

```

class ColorException extends Exception {}
class WhiteException extends ColorException {}
class White {
 void m1() throws ColorException {throw new WhiteException();}
 void m2() throws WhiteException {}
 public static void main (String[] args) {
 White white = new White();
 int a,b,d,f; a = b = d = f = 0;
 try {white.m1(); a++;} catch (ColorException e) {b++;}
 try {white.m2(); d++;} catch (WhiteException e) {f++;}
 System.out.print(a+", "+b+", "+d+", "+f+");
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,0
- c. Prints: 0,1,1,0
- d. Prints: 1,1,1,0
- e. Prints: 1,1,1,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 10

```

class ColorException extends Exception {}
class WhiteException extends ColorException {}
class White {
 void m1() throws ColorException {throw new ColorException();}
 void m2() throws WhiteException {throw new ColorException();}
 public static void main (String[] args) {
 White white = new White();
 int a,b,d,f; a = b = d = f = 0;
 try {white.m1(); a++;} catch (ColorException e) {b++;}
 try {white.m2(); d++;} catch (WhiteException e) {f++;}
 System.out.print(a+", "+b+", "+d+", "+f+");
 }
}

```

}}

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,1
- c. Prints: 0,1,0,1
- d. Prints: 0,1,1,1
- e. Prints: 1,1,1,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 11

```
class ColorException extends Exception {}
class WhiteException extends ColorException {}
class White {
 void m1() throws ColorException {throw new ColorException();}
 void m2() throws WhiteException {throw new WhiteException();}
 public static void main (String[] args) {
 White white = new White();
 int a,b,d,f; a = b = d = f = 0;
 try {white.m1(); a++;} catch (WhiteException e) {b++;}
 try {white.m2(); d++;} catch (WhiteException e) {f++;}
 System.out.print(a+","+b+","+d+","+f);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,1
- c. Prints: 0,1,0,1
- d. Prints: 0,1,1,1
- e. Prints: 1,1,1,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 12

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Brown {
 public static void main(String args[]) {
 int a, b, c, d, f; a = b = c = d = f = 0;
 int x = 1;
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 } a++; }
 catch (Level3Exception e) {b++;}
 catch (Level2Exception e) {c++;}
 catch (Level1Exception e) {d++;}
 finally {f++;}
```

```
 System.out.print(a+" "+b+" "+c+" "+d+" "+f);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,1
- b. Prints: 0,0,1,1,1
- c. Prints: 0,1,1,1,1
- d. Prints: 1,1,1,1,1
- e. Prints: 0,0,1,0,1
- f. Prints: 0,1,0,0,1
- g. Prints: 1,0,0,0,1
- h. Compile-time error
- i. Run-time error
- j. None of the above

Question 13

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Brown {
 public static void main(String args[]) {
 int a, b, c, d, f; a = b = c = d = f = 0;
 int x = 2;
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 } a++; }
 catch (Level3Exception e) {b++;}
 catch (Level2Exception e) {c++;}
 catch (Level1Exception e) {d++;}
 finally {f++;}
 System.out.print(a+" "+b+" "+c+" "+d+" "+f);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,1
- b. Prints: 0,0,1,1,1
- c. Prints: 0,1,1,1,1
- d. Prints: 1,1,1,1,1
- e. Prints: 0,0,1,0,1
- f. Prints: 0,1,0,0,1
- g. Prints: 1,0,0,0,1
- h. Compile-time error
- i. Run-time error
- j. None of the above

Question 14

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Brown {
 public static void main(String args[]) {
```

```

int a, b, c, d, f; a = b = c = d = f = 0;
int x = 4;
try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 } a++;
}
catch (Level3Exception e) {b++;}
catch (Level2Exception e) {c++;}
catch (Level1Exception e) {d++;}
finally {f++;}
System.out.print(a+, "+b+, "+c+, "+d+, "+f);
}
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,1
- b. Prints: 0,0,1,1,1
- c. Prints: 0,1,1,1,1
- d. Prints: 1,1,1,1,1
- e. Prints: 0,0,1,0,1
- f. Prints: 0,1,0,0,1
- g. Prints: 1,0,0,0,1
- h. Compile-time error
- i. Run-time error
- j. None of the above

Question 15

```

class ColorException extends Exception {}
class WhiteException extends ColorException {}
abstract class Color {
 abstract void m1() throws ColorException;
}
class White extends Color {
 void m1() throws WhiteException {throw new WhiteException();}
 public static void main (String[] args) {
 White white = new White();
 int a,b,c; a = b = c = 0;
 try {white.m1(); a++;}
 catch (WhiteException e) {b++;}
 finally {c++;}
 System.out.print(a+, "+b+, "+c);
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0
- b. Prints: 0,0,1
- c. Prints: 0,1,0
- d. Prints: 0,1,1
- e. Prints: 1,0,0
- f. Prints: 1,0,1
- g. Prints: 1,1,0
- h. Prints: 1,1,1
- i. Compile-time error

- j. Run-time error
  - k. None of the above
- Question 16

```
class RedException extends Exception {}
class BlueException extends Exception {}
class White {
 void m1() throws RedException {throw new RedException();}
 public static void main (String[] args) {
 White white = new White();
 int a,b,c,d; a = b = c = d = 0;
 try {white.m1(); a++;}
 catch (RedException e) {b++;}
 catch (BlueException e) {c++;}
 finally {d++;}
 System.out.print(a+"," +b+"," +c+"," +d);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,1
- c. Prints: 0,1,0,1
- d. Prints: 0,1,1,1
- e. Prints: 1,1,1,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 17

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
 public static void main(String args[]) {
 int a,b,c,d,f,g,x;
 a = b = c = d = f = g = 0;
 x = 1;
 try {
 throw new Level1Exception();
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 } a++; }
 catch (Level2Exception e) {b++;}
 finally {c++;}
 }
 catch (Level1Exception e) { d++;}
 catch (Exception e) {f++;}
 finally {g++;}
 System.out.print(a+"," +b+"," +c+"," +d+"," +f+"," +g);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1,1,0,0,1
- b. Prints: 0,1,1,0,0,1
- c. Prints: 0,1,0,0,0,0
- d. Prints: 0,1,0,0,0,1
- e. Prints: 0,0,1,0,0,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

No. Answer Remark

1 d Prints: true,true Both Error and Exception are subclasses of Throwable.

2 a d class A extends Object. Compile-time error at 3. The constructors for class B and class C both invoke the constructor for A. The constructor for class A declares Exception in the throws clause. Since the constructors for B and C invoke the constructor for A, it is necessary to declare Exception in the throws clauses of B and C. A compile-time error is generated at marker 3, because the constructor does not declare Exception in the throws clause.

3 b Prints: false,true Error is a direct subclass of Throwable. RuntimeException is a direct subclass of Exception.

4 b Prints: 0,0,1,1,0,1 The nested catch clause is able to catch a Level2Exception or any subclass of it. The switch statement throws a Level1Exception that can not be caught by the nested catch clause; so the nested finally block is executed as control passes to the first of the two outer catch clauses. The outer finally block is executed as control passes out of the try statement.

5 c Prints: 0,1,1,0,0,1 The nested catch block is able to catch a Level2Exception or any subclass of it causing b to be incremented. Both of the finally blocks are then executed.

6 b Prints: 0,1,1,0,0,1 The nested catch block is able to catch a Level2Exception or any subclass of it causing b to be incremented. Both of the finally blocks are then executed.

7 d Prints: 0,0,1,0,1,1 The nested catch clause is able to catch a Level2Exception or any subclass of it. The switch statement throws an Exception that can not be caught by the nested catch clause; so the nested finally block is executed as control passes to the second of the two outer catch clauses. The outer finally block is executed as control passes out of the try statement.

8 b Prints: 1,0,1,0,0,1 The switch statement does not throw an exception; so the switch completes normally. The subsequent statement increments the variable, a; and the try block completes normally. Both of the finally blocks are then executed.

9 c Prints: 0,1,1,0 The first try block contains two statements. The first invokes method m1, and the subsequent statement contains a post increment expression with the variable, a, as the operand. Method m1 throws a WhiteException exception, so variable a is not incremented as control passes to the catch block where b is incremented. The throws clause of m1 declares a ColorException, so the body may throw a ColorException or any subclass of ColorException. The second try block also contains two statements. The first invokes method m2, and the subsequent statement contains a post increment expression with the

variable, d, as the operand. Method m2 does not throw an exception, so d is incremented, and the try block completes normally. Although the throws clause of m2 declares a WhiteException, there is no requirement to throw any exception.

10 f Compile-time error The throws clause of White.m2 declares a WhiteException, so the body of m2 may throw a WhiteException or any subclass of WhiteException. Instead, the body of m2 throws a superclass of WhiteException. The result is a compile-time error.

11 f Compile-time error The throws clause of White.m1 declares a ColorException, but the catch clause in the main method catches only a subclass of ColorException. The result is a compile-time error.

12 a Prints: 0,0,0,1,1 The first catch clause has a parameter e of type Level3Exception, so the first catch clause is able to catch any exception type that is assignable to type Level3Exception. Since Level2Exception is the superclass of Level3Exception, an instance of Level2Exception is not assignable to a catch clause parameter of type Level3Exception. Similarly, Level1Exception is also a superclass of Level3Exception, so an instance of Level1Exception is not assignable to a catch clause parameter of type Level3Exception. The only exception type that can be caught by the first catch clause is a Level3Exception. The second catch clause has a parameter e of type Level2Exception, so the second catch clause is able to catch a Level2Exception. The Level1Exception is the superclass of Level2Exception. An instance of Level1Exception is not assignable to a catch clause parameter of type Level2Exception, so the second catch clause can not catch a Level1Exception. Since a Level3Exception is a subclass of Level2Exception an exception of type Level3Exception is assignable to a catch clause parameter type Level2Exception. All exceptions of type Level3Exception will be caught by the first catch clause, so the second catch clause in this program will not have an opportunity to catch a Level3Exception. The third catch clause has a parameter e of type Level1Exception, so the third catch clause is able to catch a Level1Exception. The exceptions of type Level2Exception and Level3Exception are assignable to the catch clause parameter of the third catch clause, but the exceptions of those subclass types will be caught by the first two catch clauses. The switch statement throws a Level1Exception. The try block completes abruptly as control passes to the third catch block where d is incremented. The finally block is also executed, so f is incremented.

13 e Prints: 0,0,1,0,1 The first catch block is able to catch a Level3Exception or any subclass of Level3Exception. The second catch block is able to catch a Level2Exception or any subclass of Level2Exception. The switch statement throws a Level2Exception. The try block completes abruptly as control passes to the second catch block where c is incremented. The finally block is also executed, so f is incremented.

14 g Prints: 1,0,0,0,1 The switch statement does not throw an exception; so the switch completes normally. The subsequent statement increments the variable, a; and the try block completes normally. The finally block is also executed, so f is incremented.

15 d Prints: 0,1,1 The try block contains two statements. The first invokes method m1, and the subsequent statement contains a post increment expression with the variable, a, as the operand. Method m1 throws a WhiteException exception, so variable a is not incremented as control passes to the catch block where b is incremented. Although

Color.m1 declares a ColorException in the throws clause, a subclass of Color is free to declare only a subclass of ColorException in the throws clause of the overriding method.

16 f Compile-time error A compile-time error is generated, because the second catch clause attempts to catch an exception that is never thrown in the try block.

17 f Compile-time error A throw statement is the first statement in the outer try block. A throw statement appearing in a try block causes control to pass out of the block. Consequently, statements can not be reached if they appear in the block after the throw statement. The switch statement that appears after the throw statement is unreachable and results in a compile-time error.

## Interfaces

### Question 1

```
interface A {
 void m1(); // 1
 public void m2(); // 2
 protected void m3(); // 3
 private void m4(); // 4
}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4

### Question 2

Which of the following statements is not true?

- a. An interface that is declared within the body of a class or interface is known as a nested interface.
- b. A constant can be a member of an interface.
- c. A class declaration can be a member of an interface.
- d. If an interface is named in the implements clause of a class, then the class must implement all of the methods declared within the interface.
- e. None of the above.

### Question 3

Which of the following statements are true?

- a. An interface declaration can be a member of an interface.
- b. A method declared within an interface must have a body represented by empty curly braces.
- c. An interface can implement another interface.
- d. An abstract class that implements an interface must implement all abstract methods declared within the interface.
- e. An abstract method declaration can be a member of an interface.

### Question 4

Which of the following are modifiers that can be applied to an interface that is a member of a directly enclosing interface?

- a. abstract
- b. implements
- c. final
- d. private
- e. protected
- f. public

Question 5

Which of the following are modifiers that can be applied to an interface that is a member of a directly enclosing class?

- a. abstract
- b. extends
- c. final
- d. private
- e. protected
- f. public

Question 6

Which of the following is a modifier that can be applied to an interface that is a member of a directly enclosing class or interface?

- a. static
- b. synchronized
- c. transient
- d. volatile
- e. implements
- f. None of the above.

Question 7

Suppose that an interface, I1, is not a member of an enclosing class or interface. Which of the following modifiers can be applied to interface I1?

- a. abstract
- b. final
- c. private
- d. protected
- e. public

Question 8

Suppose that an interface, I1, is not a member of an enclosing class or interface. Which of the following modifiers can be applied to interface I1?

- a. abstract
- b. public
- c. static
- d. synchronized
- e. transient
- f. volatile

Question 9

Which of the following are modifiers that can be applied to a field declaration within an interface?

- a. abstract
- b. const
- c. final

- d. private
  - e. protected
  - f. public
- Question 10

Which of the following is a modifier that can be applied to a field declaration within an interface?

- a. static
- b. synchronized
- c. transient
- d. volatile
- e. None of the above.

Question 11

Which of the following modifiers can be applied to a class that is declared within an enclosing interface?

- a. public
- b. protected
- c. private
- d. abstract
- e. static
- f. final

Question 12

```
interface A {
 int a = 1; // 1
 public int b = 2; // 2
 public static int c = 3; // 3
 public static final int d = 4; // 4
}
```

Which field declaration results in a compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

Question 13

```
interface A {
 protected int e = 5; // 1
 private int f = 6; // 2
 volatile int g = 7; // 3
 transient int h = 8; // 4
 public static final int d = 9; // 5
}
```

Which of the field declarations does not result in a compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5
- f. None of the above

Question 14

Which of the following are modifiers that can be applied to a method declaration within an interface?

- a. abstract
- b. final
- c. private
- d. protected
- e. public

Question 15

Which of the following is a modifier that can be applied to a method declaration within an interface?

- a. static
- b. synchronized
- c. transient
- d. volatile
- e. native
- f. None of the above

Question 16

```
interface A {void m1();} // 1
class B implements A {public void m1() {} } // 2
class C implements A {protected void m1() {} } // 3
class D implements A {private void m1() {} } // 4
class E implements A {void m1() {} } // 5
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5

Question 17

```
interface A {
 void m1(); // 1
 public void m2(); // 2
 protected void m3(); // 3
 private void m4(); // 4
 abstract void m5(); // 5
}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5

Question 18

```
interface A {
 final void m1(); // 1
 synchronized void m2(); // 2
```

```
 native void m3(); // 3
 abstract void m4(); // 4
 public void m5(); // 5
}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5

Question 19

```
interface A {void main(String[] args);} // 1
interface B {public void main(String[] args);} // 2
interface C {public static void main(String[] args);} // 3
interface D {protected void main(String[] args);} // 4
interface E {private void main(String[] args);} // 5
```

Which interface declarations generate a Compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5

Question 20

```
interface F {abstract void main(String[] args);} // 1
interface G {synchronized void main(String[] args);} // 2
interface H {final void main(String[] args);} // 3
interface I {native void main(String[] args);} // 4
```

Which interface declaration does not generate a compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

Question 21

```
interface A {String s1 = "A"; String m1();}
interface B implements A {String s1 = "B"; String m1();}
class C implements B {
 public String m1() {return s1;}
 public static void main(String[] args) {
 A a = new C(); System.out.print(a.m1());
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Compile-time error
- d. Run-time error
- e. None of the above

Question 22

```
interface A {int i = 1; int m1();}
interface B extends A {int i = 10; int m1();}
class C implements B {
 public int m1() {return ++i;}
 public static void main(String[] args) {
 System.out.print(new C().m1());
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 2
- b. Prints: 11
- c. Compile-time error
- d. Run-time error
- e. None of the above

Question 23

```
interface Z {void m1();} // 1
class A implements Z {void m1() {}} // 2
class B implements Z {public void m1() {}} // 3
abstract class C implements Z {public abstract void m1();} // 4
```

A Compile-time error is generated at which line?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

Question 24

```
interface Z {void m1();} // 1
class D implements Z {public final void m1() {}} // 2
class E implements Z {public synchronized void m1() {}} // 3
class G implements Z {public native void m1();} // 4
```

A Compile-time error is generated at which line?

- a. 1
- b. 2
- c. 3
- d. 4
- e. None of the above

Question 25

```
interface I10 {String name = "I10"; String s10 = "I10.s10";}
interface I20 {String name = "I20"; String s20 = "I20.s20";}
class C10 implements I10, I20 { // 1
 public static void main(String[] args) {
 System.out.print(s10+","); // 2
 System.out.print(s20+","); // 3
 System.out.print(name); // 4
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: I10.s10,I20.s20,I10
- b. Prints: I10.s10,I20.s20,I20
- c. Prints: I10.s10,I20.s20,
- d. Prints: I10.s10,I20.s20,null
- e. Compile-time error at line 1
- f. Compile-time error at line 2
- g. Compile-time error at line 3
- h. Compile-time error at line 4
- i. Run-time error
- j. None of the above

Question 26

```
interface I10 {String name = "I10"; String s10 = "I10.s10";}
interface I20 {String name = "I20"; String s20 = "I20.s20";}
class C20 implements I10, I20 { // 1
 public static void main(String[] args) {
 System.out.print(I10.s10+","); // 2
 System.out.print(I20.s20+","); // 3
 System.out.print(I20.name); // 4
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: I10.s10,I20.s20,I10
- b. Prints: I10.s10,I20.s20,I20
- c. Prints: I10.s10,I20.s20,
- d. Prints: I10.s10,I20.s20,null
- e. Compile-time error at line 1
- f. Compile-time error at line 2
- g. Compile-time error at line 3
- h. Compile-time error at line 4
- i. Run-time error
- j. None of the above

No. Answer Remark

1 c d 3 4 Methods declared within an interface are implicitly public. If no access modifier is included in the method declaration; then, the declaration is implicitly public. An attempt to declare the method using a weaker access privilege, private or protected, results in a compile-time error.

2 d If an interface is named in the implements clause of a class, then the class must implement all of the methods declared within the interface. This question asks which answer option is not true. Some true statements are as follows. An interface can be declared within an enclosing class or interface. The members of an interface can be constants, abstract method declarations, class declarations or interface declarations. If an interface is named in the implements clause of a class, then the class must implement all of the methods declared within the interface or the class must be declared abstract. The untrue answer option did not mention that an abstract class is not required to implement any of the methods declared in an interface that is named in the implements clause of the class declaration.

3 a e An interface declaration can be a member of an interface. An abstract method declaration can be a member of an interface. An interface can be declared within an enclosing class

or interface. The members of an interface can be constants, abstract method declarations, class declarations, or interface declarations. The body of a method declared within an interface is a semicolon. An interface can extend another interface, but can not implement an interface. An abstract class that has an interface, I1, in its implements clause is not required to implement any of the methods declared within I1.

4 a f abstract public All interfaces are implicitly abstract. The explicit application of the abstract modifier to an interface declaration is redundant and is strongly discouraged. The declaration of an interface within the body of an enclosing class or interface is called a member type declaration. Every member type declaration appearing within the body of a directly enclosing interface is implicitly static and public. Use of the access modifiers, private or protected, is contradictory and results in a compile-time error. In contrast, the modifiers, private and protected, are applicable to a member type declaration appearing within the body of a directly enclosing class. The modifier, final, is never applicable to an interface. The keyword, implements, is not a modifier.

5 a d e f abstract private protected public All interfaces are implicitly abstract. The explicit application of the modifier, abstract, to an interface is redundant and is strongly discouraged. The declaration of an interface within the body of an enclosing class or interface is called a member type declaration. The private, protected and static modifiers are applicable to a member type declaration that appears in the body of a directly enclosing class. In contrast, the modifiers, private and protected, are not applicable to a member type declaration appearing within the body of a directly enclosing interface. The modifier, final, is never applicable to an interface. The keyword, extends, is not a modifier.

6 a static A member interface is always implicitly static. The modifier, static, can not be applied to an interface that is not a member interface. The modifier, synchronized, is applicable to a concrete implementation of a method, but is not applicable to any interface. The modifiers, volatile and transient, are only applicable to variables that are members of a class. The keyword, implements, is not a modifier.

7 a e abstract public The modifier, abstract, is applicable to an interface declaration, but its use is strongly discouraged; because every interface is implicitly abstract. An interface can not be final. The modifiers, private and protected, are applicable only to an interface declaration that is a member of a directly enclosing class declaration. If an interface is not a member of a directly enclosing class, or if the interface is a member of a directly enclosing interface; then, the modifiers, private and protected, are not applicable. If an interface is declare public, then the compiler will generate an error if the class is not stored in a file that has the same name as the interface plus the extension .java.

8 a b abstract public The modifier, abstract, is applicable to an interface declaration, but its use is strongly discouraged; because every interface is implicitly abstract. If an interface is declare public, then the compiler will generate an error if the class is not stored in a file that has the same name as the interface plus the extension .java. The modifier, static, is applicable to a member interface, but not to an interface that is not nested. The modifier,

synchronized, is applicable only to concrete implementations of methods. The modifiers, transient and volatile, are applicable only to variables.

9 c f final public The modifier, abstract, is not applicable to a variable. All field declarations within an interface are implicitly public, static and final. Use of those modifiers is redundant but legal. Although const is a Java keyword, it is not currently used by the Java programming language. An interface member can never be private or protected.

10 a static All field declarations within an interface are implicitly public, static and final. Use of these modifiers is redundant but legal. A field that is declared final can not also be declared volatile; so a field of an interface can not be declared volatile. The modifier, synchronized, is never applicable to a field.

11 a d e f public abstract static final A class that is declared within an enclosing interface is implicitly public and static; so the access modifiers, protected and private, are not applicable.

12 e None of the above All field declarations within an interface are implicitly public, static and final. Use of these modifiers is redundant but legal. No other modifiers can be applied to a field declaration within an interface.

13 e 5 All field declarations within an interface are implicitly public, static and final. Use of these modifiers is redundant but legal. No other modifiers can be applied to a field declaration within an interface.

14 a e abstract public All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. An abstract method can not also be declared private, static, final, native or synchronized; so the same restriction applies to methods declared within an interface.

15 f None of the above All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. An abstract method can not also be declared private, static, final, native or synchronized; so the same restriction applies to methods declared within an interface. Transient and volatile are not method modifiers.

16 c d e 3 4 5 Methods declared within an interface are implicitly public even if the modifier, public, is omitted from the declaration. Within the body of a class declaration, an attempt to implement the method using a weaker access privilege, private, protected or package access, results in a compile-time error.

17 c d 3 4 All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. Since all methods declared within an interface are implicitly public, a weaker access level can not be declared.

18 a b c 1 2 3 All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. The final, synchronized and native modifiers can not appear in the declaration of an abstract method,

and can not be applied to an abstract method declared within an interface.

19 c d e 3 4 5 All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. Since all methods declared within an interface are implicitly public, a weaker access level can not be declared.

20 a 1 All methods declared within an interface are implicitly abstract. The final, synchronized and native modifiers can not appear in the declaration of an abstract method, and can not be applied to an abstract method declared within an interface.

21 c Compile-time error In the declaration of interface B, the keyword, extends, has been replaced by the keyword, implements.

22 c Compile-time error Fields declared within an interface are implicitly public, final, and static. A compile-time error is generated in response to the attempt to increment the value of i.

23 b 2 All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. Methods declared within an interface are implicitly public even if the modifier, public, is omitted from the declaration. Within the body of a class declaration, an attempt to implement the method using a weaker access privilege, private, protected or package access, results in a compile-time error. An abstract class that implements an interface is free to override any of the inherited method declarations with another abstract method declaration.

24 e None of the above All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration within an interface, the usage is redundant and is discouraged. The modifiers, final, synchronized and native, can not appear in the declaration of an abstract method, but they can be added to an implementation of an abstract method.

25 h Compile-time error at line 4 Class C10 inherits ambiguous declarations of the name field. As long as the field is not referenced as a member of class C10; then, no compile-time error occurs. Line 4 generates the compile-time error, because it is the first to access the name field as a member of class C10.

26 b Prints: I10.s10,I20.s20,I20 Class C20 inherits ambiguous declarations of the name field. As long as the field is not referenced as a member of class C20; then, no compile-time error occurs. Although line 4 may appear to generate the compile-time error it does not, because name is accessed directly as a member of interface I20. Therefore, the compiler does not encounter an ambiguity.

## Inheritance

### Question 1

Which of the following statements are true?

- a. A constructor can invoke another constructor of the same class using the alternate constructor invocation, "this(argumentListopt);".
- b. A constructor can invoke itself using the alternate constructor invocation, "this(argumentListopt);".
- c. The alternate constructor invocation, "this(argumentListopt);", can legally appear anywhere in the constructor body.
- d. A constructor can invoke the constructor of the direct superclass using the superclass constructor invocation, "super(argumentListopt);".
- e. The number of constructor invocations that may appear in any constructor body can equal but not exceed the number of alternate constructors declared in the same class.
- f. A constructor is not permitted to throw an exception.

Question 2

Suppose that the superclass constructor invocation, "super(argumentListopt);", appears explicitly in a subclass constructor. If a compile-time error is to be avoided then the arguments for the superclass constructor invocation, "super(argumentListopt);", can not refer to which of the following?

- a. Static variables declared in this class or any superclass.
- b. Instance variables declared in this class or any superclass.
- c. Static methods declared in this class or any superclass.
- d. Instance methods declared in this class or any superclass.
- e. The keyword this.
- f. The keyword super.

Question 3

```
class A {void m1(String s1) {}}
class B extends A {
 void m1(String s1) {} // 1
 void m1(boolean b) {} // 2
 void m1(byte b) throws Exception {} // 3
 String m1(short s) {return new String();} //4
 private void m1(char c) {} // 5
 protected void m1(int i) {} // 6
}
```

What is the result of attempting to compile the program?

- a. Compile-time error at line 1
- b. Compile-time error at line 2
- c. Compile-time error at line 3
- d. Compile-time error at line 4
- e. Compile-time error at line 5
- f. Compile-time error at line 6
- g. None of the above

Question 4

```
class A {void m1() {System.out.print("A.m1");}}
class B extends A {
 void m1() {System.out.print("B.m1");}
 static void m1(String s) {System.out.print(s+",");}
}
class C {
 public static void main (String[] args) {B.m1("main"); new B().m1();}}
```

}

What is the result of attempting to compile and run the program?

- a. Prints: main,B.m1
- b. Compile-time error
- c. Run-time error
- d. None of the above

Question 5

Which of the following are true statements?

- a. The relationship between a class and its superclass is an example of a "has-a" relationship.
- b. The relationship between a class and its superclass is an example of an "is-a" relationship.
- c. The relationship between a class and an object referenced by a field within the class is an example of a "has-a" relationship.
- d. The relationship between a class and an object referenced by a field within the class is an example of an "is-a" relationship.

Question 6

```
class A {String s1 = "A.s1"; String s2 = "A.s2";}
class B extends A {
 String s1 = "B.s1";
 public static void main(String args[]) {
 B x = new B(); A y = (A)x;
 System.out.println(x.s1+" "+x.s2+" "+y.s1+" "+y.s2);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: B.s1 A.s2 B.s1 A.s2
- b. Prints: B.s1 A.s2 A.s1 A.s2
- c. Prints: A.s1 A.s2 B.s1 A.s2
- d. Prints: A.s1 A.s2 A.s1 A.s2
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 7

```
class C {
 void printS1() {System.out.print("C.printS1 ");}
 static void printS2() {System.out.print("C.printS2 ");}
}
class D extends C {
 void printS1(){System.out.print("D.printS1 ");}
 void printS2() {System.out.print("D.printS2 ");}
 public static void main (String args[]) {
 C c = new D(); c.printS1(); c.printS2();
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: C.printS1 C.printS2
- b. Prints: C.printS1 D.printS2
- c. Prints: D.printS1 C.printS2
- d. Prints: D.printS1 D.printS2

- e. Run-time error
  - f. Compile-time error
  - g. None of the above
- Question 8

```
class E {
 void printS1(){System.out.print("E.printS1 ");}
 static void printS2() {System.out.print("E.printS2");}
}
class F extends E {
 void printS1(){System.out.print("F.printS1 ");}
 static void printS2() {System.out.print("F.printS2");}
 public static void main (String args[]) {
 E x = new F(); x.printS1(); x.printS2();
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: E.printS1 E.printS2
- b. Prints: E.printS1 F.printS2
- c. Prints: F.printS1 E.printS2
- d. Prints: F.printS1 F.printS2
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 9

```
class P {
 static void printS1(){System.out.print("P.printS1 ");}
 void printS2() {System.out.print("P.printS2 ");}
 void printS1S2(){printS1();printS2();}
}
class Q extends P {
 static void printS1(){System.out.print("Q.printS1 ");}
 void printS2(){System.out.print("Q.printS2 ");}
 public static void main(String[] args) {
 new Q().printS1S2();
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: P.printS1 P.printS2
- b. Prints: P.printS1 Q.printS2
- c. Prints: Q.printS1 P.printS2
- d. Prints: Q.printS1 Q.printS2
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 10

```
class R {
 private void printS1(){System.out.print("R.printS1 ");}
 protected void printS2() {System.out.print("R.printS2 ");}
 protected void printS1S2(){printS1();printS2();}
}
class S extends R {
```

```

private void printS1(){System.out.print("S.printS1 ");}
protected void printS2(){System.out.print("S.printS2 ");}
public static void main(String[] args) {
 new S().printS1S2();
}
}

```

What is the result of attempting to compile and run the program?

- a. Prints: R.printS1 R.printS2
- b. Prints: R.printS1 S.printS2
- c. Prints: S.printS1 R.printS2
- d. Prints: S.printS1 S.printS2
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 11

```

class T {
 private int i1, i2;
 void printI1I2() {System.out.print("T, i1="+i1+", i2="+i2);}
 T(int i1, int i2) {this.i1=i1; this.i2=i2;}
}
class U extends T {
 private int i1, i2;
 void printI1I2() {System.out.print("U, i1="+i1+", i2="+i2);}
 U(int i1, int i2) {this.i1=i1; this.i2=i2;}
 public static void main(String[] args) {
 T t = new U(1,2); t.printI1I2();
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: U, i1=1, i2=2
- b. Prints: T, i1=1, i2=2
- c. Prints: U, i1=null, i2=null
- d. Prints: T, i1=null, i2=null
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 12

```

interface I {String s1 = "I";}
class A implements I {String s1 = "A";}
class B extends A {String s1 = "B";}
class C extends B {
 String s1 = "C";
 void printIt() {
 System.out.print(((A)this).s1 + ((B)this).s1 +
 ((C)this).s1 + ((I)this).s1);
 }
 public static void main (String[] args) {new C().printIt();}
}

```

What is the result of attempting to compile and run the program?

- a. Prints: ABCI
- b. Run-time error

- c. Compile-time error
- d. None of the above

Question 13

```
abstract class D {String s1 = "D"; String getS1() {return s1;}}
```

```
class E extends D {String s1 = "E"; String getS1() {return s1;}}
```

```
class F {
```

```
 public static void main (String[] s) {
```

```
 D x = new E(); System.out.print(x.s1 + x.getS1());
```

```
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: DD
- b. Prints: DE
- c. Prints: ED
- d. Prints: EE
- e. Run-time error
- f. Compile-time error
- g. None of the above

Question 14

```
class A {static void m() {System.out.print("A");}}
```

```
class B extends A {static void m() {System.out.print("B");}}
```

```
class C extends B {static void m() {System.out.print("C");}}
```

```
class D {
```

```
 public static void main(String[] args) {
```

```
 C c = new C(); c.m(); B b = c; b.m(); A a = b; a.m();
```

```
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CBA
- d. Prints: CCC
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 15

```
class A {String s1 = "A";}
```

```
class B extends A {String s1 = "B";}
```

```
class C extends B {String s1 = "C";}
```

```
class D {
```

```
 static void m1(A x) {System.out.print(x.s1);}
```

```
 static void m1(B x) {System.out.print(x.s1);}
```

```
 static void m1(C x) {System.out.print(x.s1);}
```

```
 public static void main(String[] args) {
```

```
 A a; B b; C c; a = b = c = new C(); m1(a); m1(b); m1(c);
```

```
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC

- c. Prints: CBA
  - d. Prints: CCC
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- Question 16

```

class Leg{}
class Fur{}
abstract class Pet {
 public abstract void eat();
 public abstract void sleep();
}
class Dog extends Pet {
 Leg leftFront = new Leg(), rightFront = new Leg();
 Leg leftRear = new Leg(), rightRear = new Leg();
 Fur fur = new Fur();
 public Fur shed() {return fur;}
 public void eat() {}
 public void sleep() {}
}
class Cat extends Dog {
 public void ignoreOwner() {}
 public void climbTree() {}
}

```

Which of the following statements is not a true statement?

- a. A Cat object inherits an instance of Fur and four instances of Leg from the Dog superclass.
- b. A Cat object is able to sleep and eat.
- c. A Cat object is able to climb a tree.
- d. The relationship between Dog and Pet is an example of an appropriate use of inheritance.
- e. The relationship between Cat and Dog is an example of an appropriate use of inheritance.
- f. None of the above.

Question 17

```

class A {
 A() {System.out.print("CA ");}
 static {System.out.print("SA ");}
}
class B extends A {
 B() {System.out.print("CB ");}
 static {System.out.print("SB ");}
 public static void main (String[] args) {B b = new B();}
}

```

What is the result of attempting to compile and run the above program?

- a. Prints: SA CA SB CB
- b. Prints: SA SB CA CB
- c. Prints: SB SA CA CB
- d. Prints: SB CB SA CA
- e. Runtime Exception

- f. Compiler Error
  - g. None of the above
- Question 18

```
class A {String s1="A";}
class B extends A {String s1="B";}
class C extends B {String s1="C";}
class D extends C {
 String s1="D";
 void m1() {
 System.out.print(this.s1 + ","); // 1
 System.out.print(((C)this).s1 + ","); // 2
 System.out.print(((B)this).s1 + ","); // 3
 System.out.print(((A)this).s1); // 4
 }
 public static void main (String[] args) {
 new D().m1(); // 5
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: D,D,D,D
- b. Prints: D,C,B,A
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.
- f. Compile-time error at 4.
- g. Compile-time error at 5.
- h. Run-time error
- i. None of the above

Question 19

```
class SuperA {String s1="SuperA";}
class SuperB {String s1="SuperB";}
class A extends SuperA {
 String s1="A";
 class B extends SuperB { // 1
 String s1="B";
 void m1() {
 System.out.print(this.s1 + ","); // 2
 System.out.print(super.s1 + ","); // 3
 System.out.print(A.this.s1 + ","); // 4
 System.out.print(A.super.s1); // 5
 }
 }
 public static void main (String[] args) {
 new A().new B().m1(); // 6
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: B,SuperB,B,SuperB
- b. Prints: B,SuperB,A,SuperA
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.

- f. Compile-time error at 4.
- g. Compile-time error at 5.
- h. Compile-time error at 6.
- i. Run-time error
- j. None of the above

Question 20

```
class A {void m1() {System.out.print("A");}}
class B extends A {void m1(){System.out.print("B");}}
class C extends B {void m1() {System.out.print("C");}}
class D extends C {
 void m1() {System.out.print("D");}
 void m2() {
 m1();
 ((C)this).m1(); // 1
 ((B)this).m1(); // 2
 ((A)this).m1(); // 3
 }
 public static void main (String[] args) {
 new D().m2(); // 4
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: DCBA
- b. Prints: DDDD
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.
- f. Compile-time error at 4.
- g. Run-time error
- h. None of the above

Question 21

```
class A {public void m1() {System.out.print("A1");}}
class B extends A {
 public void m1() {System.out.print("B1");}
 public void m2() {System.out.print("B2");}
}
class C {
 public static void main(String[] args) {
 A a1 = new B();
 a1.m1(); // 1
 a1.m2(); // 2
 ((B)a1).m1(); // 3
 ((B)a1).m2(); // 4
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A1B2B1B2
- b. Prints: B1B2B1B2
- c. Compile-time error at 1
- d. Compile-time error at 2
- e. Compile-time error at 3
- f. Compile-time error at 4

- g. Run-time error
- h. None of the above

| No. | Answer  | Remark                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | a d     | A constructor can invoke another constructor of the same class using the alternate constructor invocation, "this(argumentListOpt);". A constructor can invoke the constructor of the direct superclass using the superclass constructor invocation, "super(argumentListOpt);". If an alternate constructor invocation appears in the body of the constructor, then it must be the first statement. The same is true for a superclass constructor invocation. A compile-time error is generated if a constructor attempts to invoke itself either directly or indirectly.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 2   | b d e f | Instance variables declared in this class or any superclass. Instance methods declared in this class or any superclass. The keyword this. The keyword super. If the superclass constructor invocation, "super(argumentListOpt);", appears explicitly or implicitly, then it must be the first statement in the body of the constructor. Until the superclass constructor invocation runs to completion, no other statements are processed within the body of the constructor. The same is true of the constructors of any superclass. (Note: The primordial class, Object, does not have a superclass, so the constructors do not include a superclass constructor invocation statement.) Suppose class B is a subclass of A. The process of creating and initializing an instance of B includes the creation and initialization of an instance of the superclass A and an instance of the superclass Object. The superclass constructor invocation statement appearing in a constructor of B is invoked before the completion of the constructors of the superclasses A and Object. A superclass constructor invocation statement appearing in B can not refer to the non-static members of the superclasses, because the process of initializing those non-static superclass members is not complete when the superclass constructor invocation occurs in B. The same is true of the non-static members of B.             |
| 3   | g       | None of the above If a superclass method is not private and is accessible to code in a subclass, then any subclass method that has the same signature as the superclass method must also have the same return type. In other words, if a superclass method is overridden or hidden in a subclass, then the overriding or hiding subclass method must have the same return type as the superclass method. No such restriction applies to method overloading. If two methods share an overloaded method name but not the same parameter list, then the two methods need not have the same return type. Class A declares one method: The name is m1 and the single parameter is of type String. Class B extends A and declares six methods that overload the name m1. The method, B.m1(String s1), overrides the superclass method, A.m1(String s1). The overriding subclass method must have the same return type as the superclass method, but the methods that overload the name m1 are free to have different return types. An overriding subclass method is not permitted to throw any checked exception that is not listed or is not a subclass of any of those listed in the throws clause of the superclass method. No such restriction applies to method overloading. If two methods share an overloaded method name but not the same parameter list, then the two methods are free to have differing throws clauses. |

4 a Prints: main,B.m1 Suppose a superclass method is not private and is accessible to code in a subclass. If the superclass method is declared static, then any subclass method sharing the same signature must also be declared static. Similarly, if the superclass method is not declared static, then any subclass method sharing the same signature must not be declared static. The rules governing method overloading are different. If a superclass method is declared static, then any subclass method that overloads the superclass method is free to be declared static or non-static. Similarly, if a method is declared non-static, then any overloading method is free to be declared static or non-static. Method B.m1() shares the same signature as the non-static superclass method A.m1(), so B.m1() must also be non-static. The method B.m1(String s) overloads the method name m1, but does not share the same signature with any superclass method; therefore, B.m1(String s) can be declared static even though the other methods of the same name are non-static.

5 b c The relationship between a class and its superclass is an example of an "is-a" relationship. The relationship between a class and an object referenced by a field within the class is an example of a "has-a" relationship. Inheritance is an example of an "is-a" relationship, because the subclass "is-a" specialized type of the superclass. The relationship between a class and an object referenced by a field declared within the class is an example of a "has-a" relationship, because the class "has-a" object.

6 b Prints: B.s1 A.s2 A.s1 A.s2 The variables of a subclass can hide the variables of a superclass or interface. The variable that is accessed is determined at compile-time based on the type of the reference--not the run-time type of the object. The two references x and y refer to the same instance of type B. The name x.s1 uses a reference of type B; so it refers to the variable s1 declared in class B. The name y.s1 uses a reference of type A; so it refers to the variable s1 declared in class A.

7 f Compile-time error Suppose a superclass method is not private and is accessible to code in a subclass. If the superclass method is declared static, then any subclass method sharing the same signature must also be declared static. Similarly, if the superclass method is declared non-static, then any subclass method sharing the same signature must also be declared non-static. The attempted declaration of the non-static method D.printS2 generates a compile-time error; because the superclass method, C.printS2, is static.

8 c Prints: F.printS1 E.printS2 A static method is selected based on the compile-time type of the reference--not the run-time type of the object. A non-static method is selected based on the run-time type of the object--not the compile-time type of the reference. Both method invocation expressions, x.printS1() and x.printS2(), use a reference of the superclass type, E, but the object is of the subclass type, F. The first of the two expressions invokes an instance method on an object of the subclass type; so the overriding subclass method is selected. The second invokes a static method using a reference of the superclass type; so the superclass method is selected.

9 b Prints: P.printS1 Q.printS2 Suppose a method m1 is invoked using the method invocation expression m1(). If m1 is a static member of the class where the invocation expression occurs, then that is the implementation of the method that is invoked at run-time regardless of the run-time type of the object. If m1 is non-static, then the

selected implementation is determined at run-time based on the run-time type of the object. The program invokes method printS1S2 on an instance of class Q. The body of method printS1S2 contains two method invocation expressions, printS1() and printS2(). Since method printS1 is static, the implementation declared in class P is invoked. Since printS2 is non-static and the run-time type of the object is Q, the invoked method is the one declared in class Q.

10 b Prints: R.printS1 S.printS2 A private method of a superclass is not inherited by a subclass. Even if a subclass method has the same signature as a superclass method, the subclass method does not override the superclass method. Suppose a non-static method m1 is invoked using the method invocation expression m1(). If m1 is a private member of the class T where the invocation expression occurs, then the implementation in class T is selected at run-time regardless of the run-time type of the object. If the non-static method m1 is not private, then the selected implementation is determined at run-time based on the run-time type of the object. The program invokes the non-static method printS1S2 on an instance of class S, so the run-time type is S. The body of method R.printS1S2 contains two method invocation expressions, printS1() and printS2(). Since class R contains a private implementation of the instance method printS1, it is the implementation that is selected regardless of the run-time type of the object. Since printS2 is not private and not static, the selected implementation of printS2 depends on the run-time type of the object. The method printS1S2 is invoked on an instance of class S; so the run-time type of the object is S, and the implementation of printS2 declared in class S is selected.

11 f Compile-time error The two-parameter constructor of U does not explicitly invoke the two-parameter constructor of T; therefore, the constructor of U will try to invoke a no-parameter constructor of T, but none exists.

12 a Prints: ABCI Suppose that a class extends a superclass, X, or implements an interface, X. The field access expression ((X)this).hiddenField is used to access the hidden field, hiddenField, that is accessible within the superclass or interface, X.

13 b Prints: DE At run-time, the actual field that is accessed depends on the compile-time type of the reference--not the run-time type of the object. The compile-time type of the reference x in the name x.s1 is D; so the selected field is the one declared in class D. A non-static method is selected based on the run-time type of the object--not the compile-time type of the reference. The same reference variable x is used in the method invocation expression x.getS1(), and the compile-time type is still D. At run-time, the actual type of the object is E; so the selected method is the one declared in class E.

14 c Prints: CBA Class C extends B, and B extends A. The static method C.m hides method B.m, and B.m hides A.m. In the method invocation expression c.m(), the compile-time type of the reference c is C. A static method is invoked based on the compile-time type of the reference; so the method invocation expression c.m() invokes the method m declared in class C. The compile-time type of the reference b is B; so the method invocation expression b.m() invokes the method m declared in class B. The compile-time type of the reference a is A; so the method invocation expression a.m() invokes the method m declared in class A.

15 b Prints: ABC In all three cases, the object passed to method m1 is an instance of class C; however, the type of the reference

is different for each method. Since fields are accessed based on the type of the reference, the value printed by each method is different even though the same instance is used for each method invocation.

16 e The relationship between Cat and Dog is an example of an appropriate use of inheritance. An appropriate inheritance relationship includes a subclass that "is-a" special kind of the superclass. The relationship between the Dog subclass and the Pet superclass is an example of an appropriate inheritance relationship, because a Dog "is-a" Pet. The relationship between the Cat subclass and the Dog superclass is not an example of an appropriate use of inheritance, because a Cat is not a special kind of a Dog. The goal of the OO paradigm is to develop software models that are accurate and reusable. If the software model is not accurate, then it probably is not reusable and the goals of the OO paradigm are not achieved. Code reuse and maintenance becomes increasingly difficult when inheritance is used to model inappropriate relationships. For example, suppose that somebody implements a herdSheep method in the Dog class. The Cat subclass would inherit the method and suddenly each instance of Cat would acquire the unwanted capability to make an attempt to herd sheep. It is difficult to imagine that a Cat would perform well in that role, so additional maintenance would be required to resolve the problem.

17 b Prints: SA SB CA CB The static initializer of the super class runs before the static initializer of the subclass. The body of the superclass constructor runs to completion before the body of the subclass constructor runs to completion.

18 b Prints: D,C,B,A A field of a superclass can be inherited by a subclass if the superclass field is not private and not hidden by a field declaration in the subclass and is accessible to code in the subclass. The field D.s1 hides C.s1, and C.s1 hides B.s1, and B.s1 hides A.s1. The keyword this serves as a reference to the object on which a method has been invoked. In the field access expression this.s1 appearing on line 1, the keyword this denotes a reference to the object of type D on which method m1 has been invoked. In the field access expression ((C)this).s1 appearing on line 2, the reference denoted by the keyword this is cast from type D to type C. The field that is accessed at run-time depends on the compile-time type of the reference; so the field access expression ((C)this).s1 refers to the variable s1 declared in class C.

19 b Prints: B,SuperB,A,SuperA The expression A.this.s1 is an example of a qualified this expression. It accesses the variable s1 declared in class A. The expression A.super.s1 is equivalent to ((SuperA)A.this).s1. It accesses the variable s1 declared in class SuperA.

20 b Prints: DDDD The instance method that is invoked depends on the run-time type of the object--not the compile-time type of the reference. In each case, the method m1 is invoked on an object of type D; so the implementation of m1 in type D is selected each time.

21 d Compile-time error at 2 The method invocation expression a1.m2() generates a compile-time error, because the named method, m2, is declared in class B, but the reference is of the superclass type, A. The reference a1 is of type A; so a1 is able to access only those methods that are declared in class A and subclass methods that override those of class A. Only one method, m1, is declared in A; so a reference of type A can be used to invoke A.m1 or an overriding implementation of m1 that is

declared in a subclass of A. Class B extends A and overrides method m1. A reference of type A can be used to invoke method m1 on an instance of type B. Class B declares an additional method, m2, that does not override a method of class A; so a reference of type A can not invoke B.m2.

### Method Overloading

#### Question 1

```
class A {void m1(A a) {System.out.print("A");}}
```

```
class B extends A {void m1(B b) {System.out.print("B");}}
```

```
class C extends B {void m1(C c) {System.out.print("C");}}
```

```
class D extends C {
 void m1(D d) {System.out.print("D");}
 public static void main(String[] args) {
 A a1 = new A(); B b1 = new B(); C c1 = new C(); D d1 = new D();
 d1.m1(a1); d1.m1(b1); d1.m1(c1);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: DDD
- d. Prints: ABCD
- e. Compile-time error
- f. Run-time error
- g. None of the above

#### Question 2

```
class GFC215 {
 static String m(float i) {return "float";}
 static String m(double i) {return "double";}
 public static void main (String[] args) {
 int a1 = 1; long b1 = 2; System.out.print(m(a1)+","+ m(b1));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: float,float
- b. Prints: float,double
- c. Prints: double,float
- d. Prints: double,double
- e. Compile-time error
- f. Run-time error
- g. None of the above

#### Question 3

```
class A {} class B extends A {} class C extends B {}
class D {
 void m1(A a) {System.out.print("A");}
 void m1(B b) {System.out.print("B");}
 void m1(C c) {System.out.print("C");}
 public static void main(String[] args) {
 A c1 = new C(); B c2 = new C(); C c3 = new C(); D d1 = new D();
 d1.m1(c1); d1.m1(c2); d1.m1(c3);
 }
}
```

}}

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CCC
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 4

```
class GFC216 {
 static String m(float i) {return "float";}
 static String m(double i) {return "double";}
 public static void main (String[] args) {
 char a1 = 1; long b1 = 2; System.out.print(m(a1)+", "+ m(b1));
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: float,float
- b. Prints: float,double
- c. Prints: double,float
- d. Prints: double,double
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 5

```
class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
 public static void main(String[] args) {
 A c1 = new C(); B c2 = new C(); C c3 = new C(); C c4 = new C();
 c4.m1(c1); c4.m1(c2); c4.m1(c3);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CCC
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 6

```
class GFC217 {
 static String m(int i) {return "int";}
 static String m(float i) {return "float";}
 public static void main (String[] args) {
 long a1 = 1; double b1 = 2; System.out.print(m(a1)+", "+ m(b1));
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: float,float
- b. Prints: float,double
- c. Prints: double,float
- d. Prints: double,double
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 7

```
class A {void m1(A a) {System.out.print("A");}}
```

```
class B extends A {void m1(B b) {System.out.print("B");}}
```

```
class C extends B {void m1(C c) {System.out.print("C");}}
```

```
class D {
```

```
 public static void main(String[] args) {
```

```
 A c1 = new C(); C c2 = new C(); c1.m1(c2);
```

```
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Prints: C
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 8

```
class GFC218 {
```

```
 static void m(Object x) {System.out.print("Object");}
```

```
 static void m(String x) {System.out.print("String");}
```

```
 public static void main(String[] args) {m(null);}
```

```
}
```

What is the result of attempting to compile and run the program?

- a. Prints: Object
- b. Prints: String
- c. Compile-time error
- d. Run-time error
- e. None of the above

Question 9

```
class A {void m1(A a) {System.out.print("A");}}
```

```
class B extends A {void m1(B b) {System.out.print("B");}}
```

```
class C extends B {void m1(C c) {System.out.print("C");}}
```

```
class D {
```

```
 public static void main(String[] args) {
```

```
 A a1 = new A(); B b1 = new B(); C c1 = new C(); C c4 = new C();
```

```
 a1.m1(c4); b1.m1(c4); c1.m1(c4);
```

```
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CCC
- d. Compile-time error

- e. Run-time error
  - f. None of the above
- Question 10

```
class GFC200 {}
class GFC201 {
 static void m(Object x) {System.out.print("Object");}
 static void m(String x) {System.out.print("String");}
 static void m(GFC200 x) {System.out.print("GFC200");}
 public static void main(String[] args) {m(null);}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: Object
- b. Prints: String
- c. Prints: GFC200
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 11

```
class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
 public static void main(String[] args) {
 A a1 = new A(); B b1 = new B(); C c1 = new C(); A c2 = new C();
 c2.m1(a1); c2.m1(b1); c2.m1(c1);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CCC
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 12

```
class GFC202 {} class GFC203 extends GFC202 {}
class GFC204 {
 static void m(GFC202 x) {System.out.print("GFC202");}
 static void m(GFC203 x) {System.out.print("GFC203");}
 public static void main(String[] args) {m(null);}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: GFC202
- b. Prints: GFC203
- c. Compile-time error
- d. Run-time error
- e. None of the above

Question 13

```

class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
 public static void main(String[] args) {
 A a1 = new A(); B b1 = new A(); C c1 = new A(); C c2 = new C();
 c2.m1(a1); c2.m1(b1); c2.m1(c1);
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CCC
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 14

```

class GFC205 {} class GFC206 extends GFC205 {}
class GFC207 extends GFC206 {
 static void m(GFC205 x, GFC205 y) {System.out.print("GFC205,GFC205");}
 static void m(GFC205 x, GFC206 y) {System.out.print("GFC205,GFC206");}
 static void m(GFC206 x, GFC205 y) {System.out.print("GFC206,GFC205");}
 static void m(GFC206 x, GFC206 y) {System.out.print("GFC206,GFC206");}
 public static void main(String[] args) {
 GFC207 gfc207 = new GFC207(); m(gfc207, gfc207);
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: GFC205,GFC205
- b. Prints: GFC205,GFC206
- c. Prints: GFC206,GFC205
- d. Prints: GFC206,GFC206
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 15

```

class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
 public static void main(String[] args) {
 A a1 = new A(); B b1 = new B(); C c1 = new C(); C c2 = new A();
 c2.m1(a1); c2.m1(b1); c2.m1(c1);
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CCC
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 16

```
class GFC211 {} class GFC212 extends GFC211 {}
class GFC213 extends GFC212 {
 static void m(GFC211 x, GFC211 y) {System.out.print("GFC211,GFC211");}
 static void m(GFC211 x, GFC212 y) {System.out.print("GFC211,GFC212");}
 static void m(GFC212 x, GFC211 y) {System.out.print("GFC212,GFC211");}
 static void m(GFC212 x, GFC212 y) {System.out.print("GFC212,GFC212");}
 static void m(GFC211 x, GFC213 y) {System.out.print("GFC211,GFC213");}
 public static void main(String[] args) {
 GFC213 gfc213 = new GFC213(); m(gfc213, gfc213);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: GFC211,GFC211
- b. Prints: GFC211,GFC212
- c. Prints: GFC212,GFC211
- d. Prints: GFC212,GFC212
- e. Prints: GFC211,GFC213
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 17

```
class GFC214 {
 static void m1(boolean b1) {System.out.print("boolean ");}
 static void m1(byte b1) {System.out.print("byte ");}
 static void m1(short s1) {System.out.print("short ");}
 static void m1(char c1) {System.out.print("char ");}
 static void m1(int i1) {System.out.print("int ");}
 public static void main(String[] args) {
 byte b1; m1(b1 = 1); m1(b1); m1(b1 == 1);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: byte byte byte
- b. Prints: byte byte boolean
- c. Prints: int int int
- d. Compile-time error
- e. Run-time error
- f. None of the above

Question 18

```
class A {} class B extends A {}
class C extends B {
 static void m(A x, A y) {System.out.print("AA");}
 static void m(A x, B y) {System.out.print("AB");}
 static void m(B x, A y) {System.out.print("BA");}
 static void m(B x, B y) {System.out.print("BB");}
 public static void main(String[] args) {
 A a1; B b1; m(null,null); m(a1=null,b1=null); m(b1, a1);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: BBABAB
- b. Prints: BBABBA
- c. Prints: BBBBAB
- d. Prints: BBBBBA
- e. Prints: BBBBBB
- f. Compile-time error
- g. Run-time error
- h. None of the above

Question 19

```
class A {} class B extends A {}
class C extends B {
 static void m1(A x) {System.out.print("m1A");}
 static void m2(B x) {System.out.print("m2B"); m1(x);}
 static void m2(A x) {System.out.print("m2A"); m1(x);}
 static void m3(C x) {System.out.print("m3C"); m2(x);}
 static void m3(B x) {System.out.print("m3B"); m2(x);}
 static void m3(A x) {System.out.print("m3A"); m2(x);}
 public static void main(String[] args) {m3(new C());}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: m3Am2Am1A
- b. Prints: m3Bm2Bm1A
- c. Prints: m3Cm2Bm1A
- d. Prints: m3Cm2Am1A
- e. Compile-time error
- f. Run-time error
- g. None of the above

No. Answer Remark

1 b Prints: ABC The method invocation expression d1.m1(a1) uses reference d1 of type D to invoke method m1. Since the reference d1 is of type D, the class D is searched for an applicable implementation of m1. The methods inherited from the superclasses, C, B and A, are included in the search. The argument, a1, is a variable declared with the type A; so method A.m1(A a) is invoked.

2 a Prints: float,float A method invocation conversion can widen an argument of type float to match a method parameter of type double, so any argument that can be passed to m(float i) can also be passed to m(double i) without generating a compile-time type error. For that reason, we can say that m(float i) is more specific than m(double i). Since both methods are applicable, the more specific of the two, m(float i), is chosen over the less specific, m(double i). The arguments of the method invocation expressions, m(a1) and m(b1), are of types int and long respectively. A method invocation conversion can widen an argument of type int or long to match either of the two method parameter types float or double; so both methods, m(float i) and m(double i), are applicable to the two method invocation expressions. Since both methods are applicable, the more specific of the two, m(float i) is chosen rather than the less specific, m(double i).

3 b Prints: ABC Three methods overload the method name m1. Each has a single parameter of type A or B or C. For any method invocation expression of the form m1(referenceArgument), the method is

selected based on the declared type of the variable referenceArgument--not the run-time type of the referenced object. The method invocation expression d1.m1(c1) uses reference d1 of type D to invoke method m1 on an instance of type D. The argument, c1, is a reference of type A and the run-time type of the referenced object is C. The argument type is determined by the declared type of the reference variable c1--not the run-time type of the object referenced by c1. The declared type of c1 is type A; so the method m1(A a) is selected. The declared type of c2 is type B; so the method invocation expression d1.m1(c2) invokes method m1(B b). The declared type of c3 is type C; so the method invocation expression d1.m1(c3) invokes method m1(C c).

4 a Prints: float,float A method invocation conversion can widen an argument of type float to match a method parameter of type double, so any argument that can be passed to m(float i) without generating a compile-time type error can also be passed to m(double i). For that reason, we can say that m(float i) is more specific than m(double i). The arguments of the method invocation expressions, m(a1) and m(b1), are of types char and long respectively. A method invocation conversion can widen an argument of type char or long to match either of the two method parameter types float or double; so both methods, m(float i) and m(double i), are applicable to the two method invocation expressions. Since both methods are applicable, the more specific of the two, m(float i) is chosen rather than the less specific, m(double i).

5 b Prints: ABC Three methods overload the method name m1. Each has a single parameter of type A or B or C. For any method invocation expression of the form m1(referenceArgument), the method is selected based on the declared type of the variable referenceArgument--not the run-time type of the referenced object. The method invocation expression c4.m1(c1) uses reference c4 of type C to invoke method m1 on an instance of type C. The argument, c1, is a reference of type A and the run-time type of the referenced object is C. The argument type is determined by the declared type of the reference variable c1--not the run-time type of the object referenced by c1. The declared type of c1 is type A; so the method A.m1(A a) is selected. The declared type of c2 is type B; so the method invocation expression c4.m1(c2) invokes method B.m1(B b). The declared type of c3 is type C; so the method invocation expression c4.m1(c3) invokes method C.m1(C c).

6 e Compile-time error The method invocation expression, m(b1), contains an argument of type double. A method invocation conversion will not implicitly narrow the argument to match the parameter type of the method, m(float i). The method invocation expression, m(a1), contains an argument of type long. A method invocation conversion will widen the argument to match the parameter type of the method, m(float i).

7 a Prints: A The reference c1 is of the superclass type, A; so it can be used to invoke only the method m1 declared in class A. The methods that overload the method name m1 in the subclasses, B and C, can not be invoked using the reference c1. A method invocation conversion promotes the argument referenced by c2 from type C to type A, and the method declared in class A is executed. Class A declares only one method, m1. The single parameter is of type A. Class B inherits the method declared in class A and overloads the method name with a new method that has a single parameter of type B. Both methods sharing the overloaded name, m1, can be invoked using a reference of type B; however, a

reference of type A can be used to invoke only the method declared in class A. Class C inherits the methods declared in classes A and B and overloads the method name with a new method that has a single parameter of type C. All three methods sharing the overloaded name, m1, can be invoked using a reference of type C; however, a reference of type B can be used to invoke only the method declared in class B and the method declared in the superclass A. The method invocation expression c1.m1(c2) uses reference c1 of type A to invoke method m1. Since the reference c1 is of type A, the search for an applicable implementation of m1 is limited to class A. The subclasses, B and C, will not be searched; so the overloading methods declared in the subclasses can not be invoked using a reference of the superclass type.

8 b Prints: String A method invocation conversion can widen an argument of type String to match a method parameter of type Object, so any argument that can be passed to m(String x) without generating a compile-time type error can also be passed to m(Object x). For that reason, we can say that m(String x) is more specific than m(Object x). The argument of the method invocation expression, m(null), is of type null and can be converted to either type String or Object by method invocation conversion, so both methods, m(String x) and m(Object x), are applicable. The more specific of the two, m(String x), is chosen over the less specific, m(Object x).

9 a Prints: AAA The declared type of the reference variables, a1, b1 and c1, is the superclass type, A; so the three reference variables can be used to invoke only the method m1(A a) that is declared in the superclass, A. The methods that overload the method name m1 in the subclasses, B and C, can not be invoked using a reference variable of the superclass type, A. A method invocation conversion promotes the argument referenced by c4 from type C to type A, and the method declared in class A is executed.

10 d Compile-time error The type of the argument is null and could be converted to any of the types Object, String or GFC200, by method invocation conversion. All three methods are applicable, but none of the three is more specific than both of the other two. The ambiguity results in a compile-time type error. If type GFC200 were a subclass of type String; then any argument that could be pass to m(GFC200 x) could also be passed to m(String x) without causing a compile-time type error, and we could say that m(GFC200 x) is more specific than m(String x). Since GFC200 is not a subclass of type String, a method invocation conversion is not able to widen an argument of type GFC200 to match a method parameter of type String, so m(GFC200 x) is not more specific than m(String x).

11 a Prints: AAA The reference c2 is of the superclass type, A; so it can be used to invoke only the method, m1, declared in class A. The methods that overload the method name m1 in the subclasses, B and C, can not be invoked using the reference c2.

12 b Prints: GFC203 The type of the argument is null and could be converted to either type GFC202 or GFC203 by method invocation conversion; so both methods are applicable. The more specific of the two, m(GFC203 x), is chosen. Type GFC203 is a subclass of type GFC202; so any argument that can be passed to m(GFC203 x) can also be passed to method m(GFC202 x) without causing a compile-time type error; therefore, we can say that method m(GFC203 x) is more specific than m(GFC202 x).

13 d Compile-time error The declarations of b1 and c1 cause compile-time errors, because a reference of a subclass type can not refer to an instance of the superclass type.

14 d Prints: GFC206,GFC206 Type GFC207 is a subclass of types GFC206 and GFC205, so any of the four methods are applicable to the method invocation expression, m(gfc207, gfc207). The most specific of the four, m(GFC206 x, GFC206 y), is chosen. Type GFC206 is a subclass of type GFC205, and method m(GFC206 x, GFC206 y) is more specific than the other three, because any invocation of m(GFC206 x, GFC206 y) could also be handled by any of the other three without causing a compile-time type error.

15 d Compile-time error The declaration of c2 causes a compile-time error, because a reference of a subclass type can not refer to an instance of the superclass class.

16 f Compile-time error The method invocation expression, m(gfc213, gfc213), is ambiguous; because, no applicable method is more specific than all of the others. Method m(GFC212 x, GFC212 y) is more specific than m(GFC212 x, GFC211 y), because any invocation of m(GFC212 x, GFC212 y) could also be handled by m(GFC212 x, GFC211 y) without causing a compile-time type error. However, some invocations of m(GFC212 x, GFC212 y) can not be handled by m(GFC211 x, GFC213 y), so m(GFC212 x, GFC212 y) is not more specific than m(GFC211 x, GFC213 y). Furthermore, not all invocations of m(GFC211 x, GFC213 y) could be handled by m(GFC212 x, GFC212 y).

17 b Prints: byte byte boolean Variable b1 was initialized by the first method invocation statement, so the second method invocation statement does not result in a compile-time error. The assignment expression, b1 = 1, initializes variable b1 with the value 1, and the same value is passed as an argument to method m1(byte b1). The method invocation expression, m1(b1), invokes the same method, m1(byte b1). The argument of the third method invocation expression, m1(b1 == 1), is the result of the equality expression, b1 == 1. The type of the result and the argument is boolean, so the invoked method is m1(boolean b1).

18 b Prints: BBABBA Type B is a subclass of type A, and method m(B x, B y) is more specific than the other three; because any invocation of it could be handled by any of the other three without causing a compile-time type error. All four methods are applicable to the first method invocation expression, m(null,null). The most specific method, m(B x, B y), is chosen; and both arguments are converted to type B. In the second method invocation expression, m(a1=null,b1=null), simple assignment expressions initialize the local variables a1 and b1 with null references of types A and B respectively. The invoked method is m(A x, B y). In the third method invocation expression, the positions of the arguments are reversed relative to the previous invocation: The type of the first argument is now B and the second is A. The invoked method is m(B x, A y).

19 c Prints: m3Cm2Bm1A The method invocation expression, m3(new C()), invokes method m3(C x), because the argument type matches the parameter type of the method declaration exactly. Method m3 uses the parameter as the argument of the next invocation expression, m2(x). Of the two overloaded versions of m2, the most specific is invoked, m2(B x). Type B is a subclass of A, so any invocation of m2(B x) could be handled by m2(A x) without causing a compile-time type error. For that reason, m2(B x) is more specific than m2(A x).

## Encapsulation

### Question 1

Which of the following are true statements?

- a. Encapsulation is a form of data hiding.
- b. A tightly encapsulated class is always immutable.
- c. Encapsulation is always used to make programs run faster.
- d. Encapsulation helps to protect data from corruption.
- e. Encapsulation allows for changes to the internal design of a class while the public interface remains unchanged.

### Question 2

Which of the following are true statements?

- a. A top-level class can not be called "tightly encapsulated" unless it is declared private.
- b. Encapsulation enhances the maintainability of the code.
- c. A tightly encapsulated class allows fast public access to member fields.
- d. A tightly encapsulated class allows access to data only through accessor and mutator methods.
- e. Encapsulation usually reduces the size of the code.
- f. A tightly encapsulated class might have mutator methods that validate data before it is loaded into the internal data model.

### Question 3

A class can not be called "tightly encapsulated" unless which of the following is true?

- a. The class is declared final.
- b. All local variables are declared private.
- c. All method parameters are declared final.
- d. No method returns a reference to any object that is referenced by an internal data member.
- e. None of the above

### Question 4

A class can not be called "tightly encapsulated" unless which of the following is true?

- a. All of the methods are declared private.
- b. All of the methods are synchronized.
- c. All local variables are declared final.
- d. The class is a direct subclass of Object.
- e. Accessor methods are used to prevent fields from being set with invalid data.
- f. None of the above

### Question 5

A class can not be called "tightly encapsulated" unless which of the following are true?

- a. The data members can not be directly manipulated by external code.
- b. The class is declared final.
- c. It has no public mutator methods.

d. The superclass is tightly encapsulated.

Question 6

A class can not be called "tightly encapsulated" unless which of the following is true?

- a. The class is a nested class.
- b. The constructors are declared private.
- c. The mutator methods are declared private.
- d. The class implements the Encapsulated interface.
- e. None of the above

Question 7

A class can not be called "tightly encapsulated" unless which of the following is true?

- a. All member fields are declared final.
- b. The class is not anonymous.
- c. The internal data model can be read and modified only through accessor and mutator methods.
- d. The class is an inner class.
- e. None of the above

Question 8

```
class GFC500 {private String name;}
class GFC501 {
 private String name;
 private void setName(String name) {this.name = name;}
 private String getName() {return name;}
}
class GFC502 {
 private String name;
 public void setName(String name) {this.name = name;}
 public String getName() {return name;}
}
```

Which class is not tightly encapsulated?

- a. GFC501
- b. GFC502
- c. GFC503
- d. None of the above

Question 9

```
class GFC505 extends GFC504 {
 public void setName(String name) {this.name = name;}
 public String getName() {return name;}
}
class GFC504 extends GFC503 {
 private void setName(String name) {this.name = name;}
 private String getName() {return name;}
}
class GFC503 {String name;}
```

Which class is tightly encapsulated?

- a. GFC503
- b. GFC504

- c. GFC505
  - d. None of the above
- Question 10

```
class GFC506 {private String name;}
class GFC507 extends GFC506 {
 String name;
 public void setName(String name) {this.name = name;}
 public String getName() {return name;}
}
class GFC508 extends GFC506 {
 private String name;
 public GFC508(String name) {setName(name);}
 public void setName(String name) {this.name = name;}
 public String getName() {return name;}
}
```

Which class is not tightly encapsulated?

- a. GFC506
- b. GFC507
- c. GFC508
- d. None of the above

No. Answer Remark

1 a d e Encapsulation is a form of data hiding. Encapsulation helps to protect data from corruption. Encapsulation allows for changes to the internal design of a class while the public interface remains unchanged. A tightly encapsulated class does not allow direct public access to the internal data model. Instead, access is permitted only through accessor (i.e. get) and mutator (i.e. set) methods. The additional time required to work through the accessor and mutator methods typically slows execution speed. Encapsulation is a form of data hiding. A tightly encapsulated class does not allow public access to any data member that can be changed in any way; so encapsulation helps to protect internal data from the possibility of corruption from external influences. The mutator methods can impose constraints on the argument values. If an argument falls outside of the acceptable range, then a mutator method could throw an `IllegalArgumentException`. The internal design of a tightly encapsulated class can change while the public interface remains unchanged. An immutable class is always tightly encapsulated, but not every tightly encapsulation class is immutable.

2 b d f Encapsulation enhances the maintainability of the code. A tightly encapsulated class allows access to data only through accessor and mutator methods. A tightly encapsulated class might have mutator methods that validate data before it is loaded into the internal data model. The data members of a tightly encapsulated class are declared private; so changes to the data model are less likely to impact external code. Access to internal data can be provided by public accessor (i.e. get) and mutator (i.e. set) methods. The mutator methods can be used to validate the data before it is loaded into the internal data model. The use of accessor and mutator methods is likely to increase the size of the code and slow execution speed.

3 e None of the above If a class A has a method that returns a reference to an internal, mutable object; then external code can use

the reference to modify the internal state of class A. Therefore, class A can not be considered tightly encapsulated. However, the methods of a tightly encapsulated class may return a reference to an immutable object or a reference to a copy or clone of an internal object.

4 f None of the above One answer option reads as follows: "Accessor methods are used to prevent fields from being set with invalid data." The answer would be correct if the word "Accessor" were replaced by the word "Mutator". Accessor methods are used to read data members; mutator methods are used to set data members. The mutator methods can validate the parameter values before the values are used to change the state of the internal data model.

5 a d The data members can not be directly manipulated by external code. The superclass is tightly encapsulated. If a class A is not tightly encapsulated, then no subclass of A is tightly encapsulated.

6 e None of the above A tightly encapsulated class may have public mutator methods.

7 c The internal data model can be read and modified only through accessor and mutator methods. A class is not tightly encapsulated if the internal data model can be read and/or modified without working through accessor (i.e. get) and mutator (i.e. set) methods.

8 d None of the above All three classes are tightly encapsulated, because the data members are private. A tightly encapsulated class can have public accessor and mutator methods, but it is not required to have those methods.

9 d None of the above Class GFC503 is not tightly encapsulated; so no subclass of GFC503 is tightly encapsulated.

10 b GFC507 Class GFC507 has a public field; so it is not tightly encapsulated.

## Threads

### Question 1

Which of the following methods are members of the Object class?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. sleep
- f. start
- g. yield
- h. wait

### Question 2

Which of the following methods are static members of the Thread class?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. sleep
- f. start
- g. yield
- h. wait

### Question 3

Which of the following methods are deprecated members of the Thread class?

- a. join
- b. notify
- c. notifyAll
- d. resume
- e. run
- f. sleep
- g. start
- h. stop
- i. suspend
- j. yield
- k. wait

### Question 4

Which of the following methods name the InterruptedException in its throws clause?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. sleep
- f. start
- g. yield
- h. wait

### Question 5

A timeout argument can be passed to which of the following methods?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. sleep
- f. start
- g. yield
- h. wait

### Question 6

Which of the following instance methods should only be called by a thread that holds the lock of the instance on which the method is invoked?

- a. join
- b. notify
- c. notifyAll
- d. run
- e. start
- f. wait

### Question 7

Which of the following is a checked exception?

- a. IllegalMonitorStateException
- b. IllegalThreadStateException
- c. IllegalArgumentException

- d. InterruptedException
- e. None of the above

Question 8

Which kind of variable would you prefer to synchronize on?

- a. A member variable of a primitive type
- b. A member variable that is an object reference
- c. A method local variable that is a reference to an instance that is created within the method
- d. None of the above

Question 9

synchronized (expression) block

The synchronized statement has the form shown above. Which of the following are true statements?

- a. A compile-time error occurs if the expression produces a value of any reference type
- b. A compile-time error occurs if the expression produces a value of any primitive type
- c. A compile-time error does not occur if the expression is of type boolean
- d. The synchronized block may be processed normally if the expression is null
- e. If execution of the block completes normally, then the lock is released
- f. If execution of the block completes abruptly, then the lock is released
- g. A thread can hold more than one lock at a time
- h. Synchronized statements can be nested
- i. Synchronized statements with identical expressions can be nested

Question 10

Which of the following is a true statement?

- a. The process of executing a synchronized method requires the thread to acquire a lock
- b. Any overriding method of a synchronized method is implicitly synchronized
- c. If any method in a class is synchronized, then the class itself must also be declared using the synchronized modifier
- d. If a thread invokes a static synchronized method on an instance of class A, then the thread must acquire the lock of that instance of class A
- e. None of the above

Question 11

Which of the following thread state transitions model the lifecycle of a thread?

- a. The Dead state to the Ready state
- b. The Ready state to the Not-Runnable state
- c. The Ready state to the Running state
- d. The Running state to the Not-Runnable state
- e. The Running state to the Ready state
- f. The Not-Runnable state to the Ready state

g. The Not-Runnable state to the Running state

Question 12

Which of the following are true statements?

- a. The Thread.yield method might cause the thread to move to the Not-Runnable state
- b. The Thread.yield method might cause the thread to move to the Ready state
- c. The same thread might continue to run after calling the Thread.yield method
- d. The Thread.yield method is a static method
- e. The behavior of the Thread.yield method is consistent from one platform to the next
- f. The Thread.sleep method causes the thread to move to the Not-Runnable state
- g. The Thread.sleep method causes the thread to move to the Ready state

Question 13

Which of the following will not force a thread to move into the Not-Runnable state?

- a. Thread.yield method
- b. Thread.sleep method
- c. Thread.join method
- d. Object.wait method
- e. By blocking on I/O
- f. Unsuccessfully attempting to acquire the lock of an object
- g. None of the above

Question 14

Which of the following will cause a dead thread to restart?

- a. Thread.yield method
- b. Thread.join method
- c. Thread.start method
- d. Thread.resume method
- e. None of the above

Question 15

When a thread is created and started, what is its initial state?

- a. New
- b. Ready
- c. Not-Runnable
- d. Runnning
- e. Dead
- f. None of the above

Question 16

Which of the following are true statements?

- a. The Thread.run method is used to start a new thread running
- b. The Thread.start method causes a new thread to get ready to run at the discretion of the thread scheduler
- c. The Runnable interface declares the start method
- d. The Runnable interface declares the run method
- e. The Thread class implements the Runnable interface

- f. If an Object.notify method call appears in a synchronized block, then it must be the last method call in the block
- g. No restriction is placed on the number of threads that can enter a synchronized method
- h. Some implementations of the Thread.yield method will not yield to a thread of lower priority

Question 17

Which of the following are true statements?

- a. Thread.MAX\_PRIORITY == 10
- b. Thread.MAX\_PRIORITY == 5
- c. Thread.NORM\_PRIORITY == 5
- d. Thread.NORM\_PRIORITY == 3
- e. Thread.NORM\_PRIORITY == 0
- f. Thread.MIN\_PRIORITY == 1
- g. Thread.MIN\_PRIORITY == 0
- h. Thread.MIN\_PRIORITY == -5
- i. Thread.MIN\_PRIORITY == -10

Question 18

Which of the following are true statements?

- a. A program will terminate only when all daemon threads stop running
- b. A program will terminate only when all user threads stop running
- c. A daemon thread always runs at Thread.MIN\_PRIORITY
- d. A thread inherits its daemon status from the thread that created it
- e. The daemon status of a thread can be changed at any time using the Thread.setDaemon method
- f. The Thread.setDaemon method accepts one of two argument values defined by the constants Thread.DAEMON and Thread.USER

Question 19

```
class A extends Thread {
 public A(Runnable r) {super(r);}
 public void run() {System.out.print("A");}
}
class B implements Runnable {
 public void run() {System.out.print("B");}
}
class C {
 public static void main(String[] args) {
 new A(new B()).start();
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Prints: AB
- d. Prints: BA
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 20

```
class A implements Runnable {
```

```

 public void run() {System.out.print(Thread.currentThread().getName());}
}
class B implements Runnable {
 public void run() {
 new A().run();
 new Thread(new A(),"T2").run();
 new Thread(new A(),"T3").start();
 }
}
class C {
 public static void main (String[] args) {
 new Thread(new B(),"T1").start();
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: T1T1T1
- b. Prints: T1T1T2
- c. Prints: T1T2T2
- d. Prints: T1T2T3
- e. Prints: T1T1T3
- f. Prints: T1T3T3
- g. Compile-time error
- h. Run-time error
- i. None of the above

Question 21

```

class AnException extends Exception {}
class A extends Thread {
 public void run() throws AnException {
 System.out.print("A"); throw new AnException();
 }
}
class B {
 public static void main (String[] args) {
 A a = new A(); a.start(); System.out.print("B");
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Prints: AB
- d. Prints: BA
- e. Compile-time error
- f. Run-time error
- g. None of the above

Question 22

```

class A extends Thread {
 public void run() {System.out.print("A");}
}
class B {
 public static void main (String[] args) {
 A a = new A();
 a.start();
 a.start(); // 1
 }
}

```

}

What is the result of attempting to compile and run the program?

- a. The program compiles and runs without error
- b. The second attempt to start thread t1 is successful
- c. The second attempt to start thread t1 is ignored
- d. Compile-time error at marker 1
- e. An IllegalThreadStateException is thrown at run-time
- f. None of the above

No. Answer Remark

1 b c h notify notifyAll wait

2 e g sleep yield

3 d h i resume stop suspend For the purposes of the exam, you don't need to memorize the deprecated methods of the Thread class. Even though a question such as this will not be on the exam, every Java programmer should know that the deprecated methods should not be used in new programs.

4 a e h join sleep wait

5 a e h join sleep wait

6 b c f notify notifyAll wait

7 d InterruptedException The methods Object.wait, Thread.join and Thread.sleep name InterruptedException in their throws clauses.

8 b A member variable that is an object reference Primitives don't have locks; therefore, they can not be used to synchronize threads. A method local variable that is a reference to an instance that is created within the method should not be used to synchronize threads, because each thread has its own instance of the object and lock.

Synchronization on an instance that is created locally makes about as much sense as placing on your doorstep a box full of keys to the door. Each person that comes to your door would have their own copy of the key; so the lock would provide no security.

9 b e f g h i A compile-time error occurs if the expression produces a value of any primitive type If execution of the block completes normally, then the lock is released If execution of the block completes abruptly, then the lock is released A thread can hold more than one lock at a time Synchronized statements can be nested Synchronized statements with identical expressions can be nested

10 a The process of executing a synchronized method requires the thread to acquire a lock The synchronized modifier can not be applied to a class. A method that overrides a synchronized method does not have to be synchronized. If a thread invokes a synchronized instance method on an instance of class A, then the thread must acquire the lock of that instance of class A. The same is not true for synchronized static methods. A synchronized static method is synchronized on the lock for the Class object that represents the class for which the method is a member.

11 c d e f The Ready state to the Running state The Running state to the Not-Runnable state The Running state to the Ready state The Not-Runnable state to the Ready state A dead thread can not be restarted.

12 b c d f The Thread.yield method might cause the thread to move to the Ready state The same thread might continue to run after calling the Thread.yield method The Thread.yield method is a static method The Thread.sleep method causes the thread to move to the Not-

Runnable state    The `Thread.yield` method is intended to cause the currently executing thread to move from the Running state to the Ready state and offer the thread scheduler an opportunity to allow a different thread to execute based on the discretion of the thread scheduler. The thread scheduler may select the same thread to run immediately, or it may allow a different thread to run. The `Thread.yield` method is a native method; so the behavior is not guaranteed to be the same on every platform. However, at least some implementations of the `yield` method will not yield to a thread that has a lower priority.

13    a    `Thread.yield` method    The `Thread.yield` method may cause a thread to move into the Ready state, but that state transition is not guaranteed. The JLS states that the `Thread.yield` method provides a hint to the thread scheduler, but the scheduler is free to interpret--or ignore--the hint as it sees fit. Nothing in the JLS suggests that the thread might move to the Not-Runnable state.

14    e    None of the above    A dead thread can not be restarted.

15    b    Ready

16    b d e h    The `Thread.start` method causes a new thread to get ready to run at the discretion of the thread scheduler. The `Runnable` interface declares the `run` method. The `Thread` class implements the `Runnable` interface. Some implementations of the `Thread.yield` method will not yield to a thread of lower priority. The `Object.notify` method can only be called by the thread that holds the lock of the object on which the method is invoked. Suppose that thread T1 enters a block that is synchronized on an object, A. Within the block, thread T1 holds the lock of A. Even if thread T1 calls the `notify` method immediately after entering the synchronized block, no other thread can grab the lock of object A until T1 leaves the synchronized block. For that reason, the transfer of control from thread T1 to any waiting thread can not be accelerated by moving the `notify` method to an earlier point in the synchronized block. The behavior of `Thread.yield` is platform specific. However, at least some implementations of the `yield` method will not yield to a thread that has a lower priority. Invoking the `Thread.yield` method is like offering a suggestion to the JVM to allow another thread to run. The response to the suggestion is platform specific.

17    a c f    `Thread.MAX_PRIORITY == 10`    `Thread.NORM_PRIORITY == 5`  
`Thread.MIN_PRIORITY == 1`

18    b d    A program will terminate only when all user threads stop running. A thread inherits its daemon status from the thread that created it.

19    a    Prints: A    If a `Runnable` target object is passed to the constructor of the `Thread` class, then the `Thread.run` method will invoke the `run` method of the `Runnable` target. In this case, the `Thread.run` method is overridden by A.`run`. The A.`run` method does nothing more than print the letter A. The invocation of the A.`start` method inside the main method results in the invocation of A.`run`, and the letter A is printed. The B.`run` method is never invoked.

20    e    Prints: T1T1T3    The `Thread.currentThread` method returns a reference to the currently executing thread. When the `run` method is invoked directly it does not start a new thread; so T1 is printed twice.

21    e    Compile-time error    The `Runnable.run` method does not have a throws clause; so any implementation of `run` can not throw a checked exception.

22 e An IllegalThreadStateException is thrown at run-time For the purposes of the exam, invoking the start method on a thread that has already been started will generate an IllegalThreadStateException. The actual behavior of the method might be different. If the start method is invoked on a thread that is already running, then an IllegalThreadStateException will probably be thrown. However, if the thread is already dead then the second attempt to start the thread will probably be ignored, and no exception will be thrown. For the purposes of the exam, the exception is always thrown in response to the second invocation of the start method. This is a case where the exam tests your knowledge of the specification and ignores the actual behavior of the 1.4 version of the JVM.

## Collection

### Question 1

Which implementation of the List interface produces the slowest access to an element in the middle of the list by means of an index?

- a. Vector
- b. ArrayList
- c. LinkedList
- d. None of the above

### Question 2

```
import java.util.*;
class GFC100 {
 public static void main (String args[]) {
 Object a1 = new LinkedList(), b1 = new TreeSet();
 Object c1 = new TreeMap();
 System.out.print((a1 instanceof Collection)+",");
 System.out.print((b1 instanceof Collection)+",");
 System.out.print(c1 instanceof Collection);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

### Question 3

- Each element must be unique.
- Contains no duplicate elements.
- Elements are not key/value pairs.
- Accessing an element can be almost as fast as performing a similar operation on an array.

Which of these classes provides the specified features?

- a. LinkedList
- b. TreeMap
- c. TreeSet
- d. HashMap
- e. HashSet
- f. LinkedHashMap
- g. Hashtable
- h. None of the above

Question 4

```
import java.util.*;
class GFC101 {
 public static void main (String args[]) {
 Object a1 = new HashMap(), b1 = new ArrayList();
 Object c1 = new HashSet();
 System.out.print((a1 instanceof Collection)+",");
 System.out.print((b1 instanceof Collection)+",");
 System.out.print(c1 instanceof Collection);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 5

- Entries are organized as key/value pairs.
- Duplicate entries replace old entries.

Which interface of the java.util package offers the specified behavior?

- a. List
- b. Map
- c. Set
- d. None of the above

Question 6

```
import java.util.*;
class GFC102 {
 public static void main (String args[]) {
 Object a = new HashSet();
 System.out.print((a instanceof Set)+",");
 System.out.print(a instanceof SortedSet);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true

e. None of the above

Question 7

Which implementation of the List interface provides for the fastest insertion of a new element into the middle of the list?

- a. Vector
- b. ArrayList
- c. LinkedList
- d. None of the above

Question 8

```
import java.util.*;
class GFC103 {
 public static void main (String args[]) {
 Object a1 = new TreeSet();
 System.out.print((a1 instanceof Set)+",");
 System.out.print(a1 instanceof SortedSet);
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. None of the above

Question 9

- Stores key/value pairs.
- Duplicate entries replace old entries.
- Entries are sorted using a Comparator or the Comparable interface.

Which of these classes provides the specified features?

- a. LinkedList
- b. TreeMap
- c. TreeSet
- d. HashMap
- e. HashSet
- f. Hashtable
- g. None of the above

Question 10

```
import java.util.*;
class GFC104 {
 public static void main (String args[]) {
 LinkedList a1 = new LinkedList();
 ArrayList b1 = new ArrayList();
 Vector c1 = new Vector();
 System.out.print((a1 instanceof List)+",");
 System.out.print((b1 instanceof List)+",");
 System.out.print(c1 instanceof List);
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false

- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 11

- Entries are not organized as key/value pairs.
- Duplicate entries are rejected.

Which interface of the java.util package offers the specified behavior?

- a. List
- b. Map
- c. Set
- d. None of the above

Question 12

```
import java.util.*;
class GFC105 {
 public static void main (String args[]) {
 Object a = new HashSet(), b = new HashMap();
 Object c = new Hashtable();
 System.out.print((a instanceof Collection)+",");
 System.out.print((b instanceof Collection)+",");
 System.out.print(c instanceof Collection);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 13

- Elements are not key/value pairs.
- Contains no duplicate elements.
- The entries can be sorted using the Comparable interface.

Which of these classes provides the specified features?

- a. LinkedList
- b. TreeMap
- c. TreeSet
- d. HashMap
- e. HashSet
- f. Hashtable
- g. None of the above

Question 14

```

import java.util.*;
class GFC106 {
 public static void main (String args[]) {
 Object a = new HashSet(), b = new HashMap();
 Object c = new Hashtable();
 System.out.print((a instanceof Map)+",");
 System.out.print((b instanceof Map)+",");
 System.out.print(c instanceof Map);
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 15

- Stores key/value pairs.
- Allows null elements, keys, and values.
- Duplicate entries replace old entries.
- Entries are not sorted.

Which of these classes provides the specified features?

- a. LinkedList
- b. TreeMap
- c. TreeSet
- d. HashMap
- e. HashSet
- f. Hashtable
- g. None of the above

Question 16

```

import java.util.*;
class GFC107 {
 public static void main (String args[]) {
 Object a = new HashSet(), b = new HashMap();
 Object c = new Hashtable();
 System.out.print((a instanceof Cloneable)+",");
 System.out.print((b instanceof Cloneable)+",");
 System.out.print(c instanceof Cloneable);
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false

- h. Prints: true,true,true
- i. None of the above

Question 17

- Entries are not organized as key/value pairs.
- Generally accepts duplicate elements.
- Entries may be accessed by means of an index.

Which interface of the java.util package offers the specified behavior?

- a. List
- b. Map
- c. Set
- d. None of the above

Question 18

```
import java.util.*;
import java.io.Serializable;
class GFC108 {
 public static void main (String args[]) {
 HashMap a = new HashMap();
 boolean b1, b2, b3;
 b1 = (a instanceof Cloneable) & (a instanceof Serializable);
 b2 = a instanceof Map;
 b3 = a instanceof Collection;
 System.out.print(b1 + "," + b2 + "," + b3);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 19

Which of the following classes allow unsynchronized read operations by multiple threads?

- a. Vector
- b. Hashtable
- c. TreeMap
- d. TreeSet
- e. HashMap
- f. HashSet

Question 20

- Entries are organized as key/value pairs.
- Duplicate entries replace old entries.
- Entries are sorted using a Comparator or the Comparable interface.

Which interface of the java.util package offers the specified behavior?

- a. List

- b. Map
- c. Set
- d. SortedSet
- e. SortedMap
- f. None of the above

Question 21

```
import java.util.*;
class GFC110 {
 public static void main (String[] args) {
 Object m = new LinkedHashMap();
 System.out.print((m instanceof Collection)+",");
 System.out.print((m instanceof Map)+",");
 System.out.print(m instanceof List);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 22

```
import java.util.*;
class GFC109 {
 public static void main (String[] args) {
 Object v = new Vector();
 System.out.print((v instanceof Collections)+",");
 System.out.print((v instanceof Arrays)+",");
 System.out.print(v instanceof List);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

No. Answer Remark

1 c LinkedList ArrayList and Vector both use an array to store the elements of the list; so access to any element using an index

is very fast. A `LinkedList` is implemented using a doubly linked list; so access to an element requires the list to be traversed using the links.

2    g    Prints: true,true,false    The List and Set interfaces extend the Collection interface; so both List and Set could be cast to type Collection without generating a `ClassCastException`. Therefore, the first two of the three relational expressions return true. The Map interface does not extend Collection. The reference variable `c1` refers to an instance of `TreeMap`; so the relational expression `c1 instanceof Collection` returns the value false.

3    e    HashSet    The elements of a Map are key/value pairs; so a Map is not a good choice. A List generally accepts duplicate elements. A Set stores a collection of unique elements. Any attempt to store a duplicate element in a Set is rejected. Adding and removing an element in a `TreeSet` involves walking the tree to determine the location of the element. A `HashSet` stores the elements in a hashtable; so elements in a `HashSet` can be accessed almost as quickly as elements in an array as long as the hash function disperses the elements properly. Although the `LinkedHashSet` is not among the answer options it could arguably satisfy the requirements. However, the put and remove methods of the `LinkedHashSet` are a little slower than the same methods of the `HashSet` due to the need to maintain the linked list through the elements of the `LinkedHashSet`.

4    d    Prints: false,true,true    The Map interface does not extend Collection. The reference variable `a1` refers to an instance of `HashMap`; so the relational expression `a1 instanceof Collection` returns the value false. The List and Set interfaces extend the Collection interface; so both List and Set could be cast to type Collection without generating a `ClassCastException`. Therefore, the second and third relational expressions return true.

5    b    Map    The List and Set interfaces do not support key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries.

6    c    Prints: true,false    `HashSet` implements the Set interface, but not the `SortedSet` interface.

7    c    `LinkedList`    `ArrayList` and `Vector` both use an array to store the elements of the list. When an element is inserted into the middle of the list the elements that follow the insertion point must be shifted to make room for the new element. The `LinkedList` is implemented using a doubly linked list; an insertion requires only the updating of the links at the point of insertion. Therefore, the `LinkedList` allows for fast insertions and deletions.

8    d    Prints: true,true    `TreeSet` implements the Set interface and the `SortedSet` interface.

9    b    `TreeMap`    The requirement to store key/value pairs is directly satisfied by a concrete implementation of the Map interface. The List and Set interfaces recognize objects, but do not recognize keys and values. `TreeMap` and `TreeSet` store elements in a sorted order based on the key, but the `TreeSet` does not support key/value pairs.

10    h    Prints: true,true,true    The 1.2 version of Java introduced the updated `Vector` class that implements the List interface.

11    c    Set    The Map interface organizes entries as key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries.

12 e Prints: true,false,false HashSet is a subclass of AbstractSet and AbstractCollection; therefore, it implements the Collection interface. HashMap and Hashtable do not implement the Collection interface. Instead, HashMap extends AbstractMap and implements the Map interface. Hashtable extends Dictionary and implements the Map interface.

13 c TreeSet The elements are not key/value pairs; so a Map is not a good choice. A List generally accepts duplicate elements. A Set stores a collection of unique objects; so any attempt to store a duplicate object is rejected. TreeSet stores elements in an order that is determined either by a Comparator or by the Comparable interface.

14 d Prints: false,true,true HashSet implements the Set interface, but not the Map interface. HashMap extends AbstractMap and implements the Map interface. Hashtable extends Dictionary and implements the Map interface.

15 d HashMap The requirement to store key/value pairs is directly satisfied by a concrete implementation of the Map interface. The List and Set interfaces recognize objects, but do not recognize keys and values. The requirement to allow null elements is not satisfied by a Hashtable. TreeMap and TreeSet store elements in a sorted order based on the key.

16 h Prints: true,true,true All three implement Cloneable.

17 a List The Map interface organizes entries as key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries. A List allows entries to be accessed using an index.

18 g Prints: true,true,false HashMap does not implement the Collection interface.

19 c d e f TreeMap TreeSet HashMap HashSet The Vector and Hashtable methods are synchronized and do not allow for simultaneous access by multiple threads. The concrete subclasses of the AbstractList, AbstractMap, and AbstractSet classes allow for unsynchronized read operations by multiple threads. Additionally, the synchronized wrapper methods of the Collections class allow for the instantiation of a Collection, List, Map, Set, SortedMap, or SortedSet with synchronized methods. If simultaneous read and write operations are necessary then a synchronized instance should be used.

20 e SortedMap The List and Set interfaces do not support key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries. A Map organizes the entries as key/value pairs. The SortedMap is similar to a Map except that the ordering of the elements is determined by a Comparator or the Comparable interface.

21 c Prints: false,true,false LinkedHashMap does not implement the Collection interface or the List interface.

22 b Prints: false,false,true The Collections class is not the same as the Collection interface. The Collections class contains a variety of methods used to work with collections. For example, Collections.shuffle is used to randomly shuffle the elements of a Collection. Similarly, the Arrays class provides utility methods for working with arrays.

## Arrays (1 Dimension)

### Question 1

```
class MWC101 {
 public static void main(String[] args) {
 int[] a1 = new int[]; // 1
 int a2[] = new int[5]; // 2
 int[] a3 = new int[]{1,2}; // 3
 int []a4 = {1,2}; // 4
 int[] a5 = new int[5]{1,2,3,4,5}; // 5
 } }
```

Compile-time errors are generated at which lines?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
- 

### Question 2

```
class MWC102 {
 public static void main (String[] args) {
 byte[] a = new byte[1]; long[] b = new long[1];
 float[] c = new float[1]; Object[] d = new Object[1];
 System.out.print(a[0]+"," +b[0]+"," +c[0]+"," +d[0]);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,null
- b. Prints: 0,0,0.0,null

- c. Compile-time error
  - d. Run-time error
  - e. None of the above
- 

### Question 3

```
class MWC103 {
 public static void main(String[] args) {
 int[] a1 = {1,2,3};
 int []a2 = {4,5,6};
 int a3[] = {7,8,9};
 System.out.print(a1[1]+","+a2[1]+","+a3[1]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 12
  - b. Prints: 15
  - c. Prints: 18
  - d. Prints: 1,4,7
  - e. **Prints: 2,5,8**
  - f. Prints: 3,6,9
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

### Question 4

```
class MWC104 {
 public static void main(String[] args) {
 int[5] a1; // 1
 int []a2; // 2
 }
}
```

```
int[]a3; // 3
int a4[]; // 4
}}
```

A compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
- 

### Question 5

```
class MWC105 {
 static boolean b1;
 public static void main(String[] args) {
 boolean[] array = new boolean[1];
 boolean b2;
 System.out.print(b1+",");
 System.out.print(array[0]+",");
 System.out.print(b2);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: true,true,true
- b. Prints: false,false,false
- c. Prints: null,null,null
- d. Prints: false,true,false
- e. Compile-time error
- f. Run-time error
- g. None of the above

1a e  
1 5

An array creation expression must have either a dimension expression or an initializer. If both are present, then a compile-time error is generated. Similarly, if neither is present, then a compile-time error is generated. If only the dimension expression is present, then an array with the specified dimension is created with all elements set to the default values. If only the initializer is present, then an array will be created that has the required dimensions to accommodate the values specified in the initializer. Java avoids the possibility of an incompatible dimension expression and initializer by not allowing both to appear in the same array creation expression. A compile-time error is generated by the array creation expression for a1, because it needs either a dimension expression or an initializer. A compile-time error is generated at 5, because either the dimension expression or the initializer must be removed.

2b Prints: 0,0,0.0,null

Each array contains the default value for its type. The default value of a primitive byte or a primitive long is printed as 0. The default value of a float primitive is printed as 0.0. The default value of an Object is null and is printed as null.

3e Prints: 2,5,8

Arrays a1, a2 and a3 all contain 3 integers. The first element of the array has an index of 0 so an index of one refers to the second element of each array.

4a 1

A compile-time error occurs at the line marked 1, because the array reference declaration can not be used to declare the number of components contained in the array. Instead, the dimension expression should be contained in an array creation expression such as new int[5].

5e Compile-time error

Variable b1 is initialized to false, because it is a class member. The array component array[0] is initialized to the default value, false, of the array type, boolean[], even though the array is declared locally. Local variable b2 is not initialized, because it is local. A compile-time error is generated by the statement that attempts to print the value of b2.

## Arrays (Multi-Dimensional)

### Question 1

```
class MWC201 {
 public static void main(String[] args) {
 int[][] a1 = {{1,2,3},{4,5,6},{7,8,9,10}};
 System.out.print(a1[0][2]+"," +a1[1][0]+"," +a1[2][1]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 3,4,8
  - b. Prints: 7,2,6
  - c. Compile-time error
  - d. Run-time error
  - e. None of the above
- 

## Question 2

```
class MWC202 {
 public static void main(String[] args) {
 int[] a1[] = {{1,2},{3,4,5},{6,7,8,9}};
 int []a2[] = {{1,2},{3,4,5},{6,7,8,9}};
 int a3[][] = {{1,2},{3,4,5},{6,7,8,9}};
 System.out.print(a1[0][1]+"," +a2[1][2]+"," +a3[2][3]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
  - b. Prints: 16
  - c. Prints: 1,5,9
  - d. Prints: 2,4,8
  - e. Prints: 2,5,9
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

## Question 3

```
class MWC207 {
```

```
public static void main(String[] args) {
 int[][] a1 = {{1;2};{3;4;5};{6;7;8;9}};
 System.out.print(a1[0][1]+","+a1[1][2]+","+a1[2][3]);
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
  - b. Prints: 16
  - c. Prints: 1,5,9
  - d. Prints: 2,4,8
  - e. Prints: 2,5,9
  - f. **Compile-time error**
  - g. Run-time error
  - h. None of the above
- 

#### Question 4

```
class MWC208 {
 public static void main(String[] args) {
 int[][] a1 = ((1,2),(3,4,5),(6,7,8,9));
 System.out.print(a1[0][1]+","+a1[1][2]+","+a1[2][3]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
- b. Prints: 16
- c. Prints: 1,5,9
- d. Prints: 2,4,8
- e. Prints: 2,5,9
- f. **Compile-time error**
- g. Run-time error

- h. None of the above
- 

### Question 5

```
class MWC209 {
 public static void main(String[] args) {
 int[][] a1 = [[1,2],[3,4,5],[6,7,8,9]];
 int[][] a2 = [[1,2],[3,4,5],[6,7,8,9]];
 int[][] a3 = [[1,2],[3,4,5],[6,7,8,9]];
 System.out.print(a1[0][1]+"," +a2[1][2]+"," +a3[2][3]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
  - b. Prints: 16
  - c. Prints: 1,5,9
  - d. Prints: 2,4,8
  - e. Prints: 2,5,9
  - f. **Compile-time error**
  - g. Run-time error
  - h. None of the above
- 

### Question 6

```
class MWC210 {
 public static void main(String[] args) {
 int[] a2 = {1,2}, a3 = {3,4,5}, a4 = {6,7,8,9}; // 1
 int[][] a1 = {a2,a3,a4}; // 2
 System.out.print(a1[0][1]+"," +a1[1][2]+"," +a1[2][3]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
  - b. Prints: 16
  - c. Prints: 1,5,9
  - d. Prints: 2,4,8
  - e. Prints: 2,5,9
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### Question 7

```
class MWC211 {
 public static void main(String[] args) {
 int a1[3]; // 1
 int []a2[]; // 2
 int[]a3; // 3
 int[] a4[]; // 4
 }}
A compile-time error is generated at which line?
```

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
- 

### Question 8

```
class MWC212 {
 public static void main(String[] args) {
 int[] a1[],a2[]; // 1
 }}
A compile-time error is generated at which line?
```

```
int []a3,[]a4; // 2
int []a5,a6[]; // 3
int[] a7,a8[]; // 4
}}
```

A compile-time error is generated at which line?

- a. 1
  - b. 2**
  - c. 3
  - d. 4
  - e. None of the above
- 

### Question 9

```
class MWC203 {
 public static void main(String[] args) {
 int[] a1[] = {new int[]{1,2},new int[]{3,4,5}};
 int []a2[] = new int[][]{{1,2},{3,4,5}};
 int a3[][] = {{1,2},new int[]{3,4,5}};
 System.out.print(a1[0][1]+","+a2[1][0]+","+a3[1][2]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
- b. Prints: 16
- c. Prints: 1,2,4
- d. Prints: 2,3,4
- e. Prints: 2,3,5**
- f. Prints: 2,4,5
- g. Compile-time error
- h. Run-time error

- i. None of the above
- 

### Question 10

```
class MWC204 {
 public static void main(String[] args) {
 int[][] a1 = {{1,2},{3,4,5},{6,7,8,9},{}};
 System.out.print(a1.length);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0
  - b. Prints: 3
  - c. Prints: 4
  - d. Prints: 9
  - e. Prints: 10
  - f. Prints: 11
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

### Question 11

```
class MWC205 {
 public static void main(String[] args) {
 int[][] a1 = {{1,2},{3,4,5},{6,7,8,9},{}};
 for (int i = 0; i < a1.length; i++) {
 System.out.print(a1[i].length+",");
 }
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 2,3,4,0,
  - b. Prints: 1,2,5,0,
  - c. Compile-time error
  - d. Run-time error
  - e. None of the above
- 

### Question 12

```
class MWC206 {
 public static void main (String[] args) {
 int[][] a1 = {{1,2,3},{4,5,6},{7,8,9}};
 for (int i = 0; i < 3; i++) {
 for (int j = 0; j < 3; j++) {
 System.out.print(a1[j][i]);
 }
 }
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 123456789
  - b. Prints: 147258369
  - c. Prints: 321654987
  - d. Prints: 369258147
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- 

### Question 13

```
class A11 {public String toString() {return "A11";}}
class A12 {
 public static void main(String[] arg) {
 A11[] a1 = new A11[1]; // 1
 }
}
```

```
A11[][] a2 = new A11[2][]; // 2
A11[][][] a3 = new A11[3][][]; // 3
a1[0] = new A11(); // 4
a2[0] = a2[1] = a1; // 5
a3[0] = a3[1] = a3[2] = a2; // 6
System.out.print(a3[2][1][0]); // 7
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: null
  - b. Prints: A11
  - c. Compile-time error at 1.
  - d. Compile-time error at 2.
  - e. Compile-time error at 3.
  - f. Compile-time error at 4.
  - g. Compile-time error at 5.
  - h. Compile-time error at 6.
  - i. Compile-time error at 7.
  - j. Run-time error
  - k. None of the above
- 

#### Question 14

```
class A13 {}
class A14 {
 public static void main(String[] arg) {
 A13[] a1 = new A13[1]; // 1
 A13[][] a2 = new A13[2][1]; // 2
 A13[][][] a3 = new A13[3][3][3]; // 3
 System.out.print(a3[2][2][2]); // 4
 a1[0] = new A13(); // 5
 a2[0] = a2[1] = a1; // 6
 }
}
```

```
a3[0] = a3[1] = a3[2] = a2; // 7
System.out.print(a3[2][2][2]); // 8
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: null
- b. Prints: nullnull
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.
- f. Compile-time error at 4.
- g. Compile-time error at 5.
- h. Compile-time error at 6.
- i. Compile-time error at 7.
- j. Compile-time error at 8.
- k. Run-time error

1 a Prints: 3,4,8 An array variable a1 is declared, and the declaration contains the initializer `{}{1,2,3},{4,5,6},{7,8,9,10}`. The initializer creates an array containing three components, and each is a reference to a subarray of type int[]. Each subarray contains components of type int, so the elements of the array referenced by a1 are of type int. The array access expression, `a1[0][2] = a1[1st subarray][third component] = 3`.

2 e Prints: 2,5,9 The declarations of the array variables, a1, a2 and a3, are different in terms of the position of the square brackets, but each is of type int[][] . The array access expression, `a1[0][1] = a[1st subarray][2nd component] = 2`. Each of the three array variable declarations has an array initializer. The same initializer, `{}{1,2},{3,4,5},{6,7,8,9}`, is used in each of the three cases. The initializer specifies three components of type int[], so each component is a reference to an array object containing components of type int. The size of each of the subarrays is different: The size of the first is 2, the second is 3, and the third is 4.

3 f Compile-time error The array initializer, `{}{1;2};{3;4;5};{6;7;8;9}` generates a compile-time error, because the commas have been replaced by semicolons. The array initializer should have been specified as follows: `{}{1,2},{3,4,5},{6,7,8,9}`.

4 f Compile-time error The array initializer, `((1,2),(3,4,5),(6,7,8,9))`, generates a compile-time error, because the curly braces have been replaced by parentheses. The array initializer should have been specified as follows: `{}{1,2},{3,4,5},{6,7,8,9}`.

5 f Compile-time error The array initializer, `[[1,2],[3,4,5],[6,7,8,9]]`, generates a compile-time error, because the curly braces have been replaced by square brackets. The array initializer should have been specified as follows: `{}{1,2},{3,4,5},{6,7,8,9}`.

6 e Prints: 2,5,9 The line marked 1 declares three array variables, a2, a3 and a4, and each references an array with components of type int. The line marked 2 declares an array variable, a1, where each component is initialized with a reference to one of the previously created arrays. The array access expression, a1[0][1], is evaluated as follows: a1[0][1] = a1[first subarray][second component] = 2.

7 a 1 A compile-time error occurs at the line marked 1, because the array variable declaration can not be used to specify the number of components contained in the array. Instead, the dimension expression should be contained in an array creation expression such as the following, new int[3].

8 b 2 An array variable is declared by placing brackets after the identifier or after the type name. A compile-time error occurs at the line marked 2, because the brackets appearing before the identifier for array variable a4 are not associated with the type or the identifier.

9 e Prints: 2,3,5 Each of the three array variable declarations, a1, a2 and a3, is different in terms of the position of the square brackets, but each declares a variable of type int[]{}. Each of the three declarations contains an array initializer. In each case, the initializer could be simplified as follows: {{1,2},{3,4,5}}. The initializer creates an array containing two components, and each is a reference to an array containing components of type int. The size of each of the subarrays is different: The size of the first is 2 and the second is 3. The array access expression, a1[0][1] = a[1st subarray][2nd component] = 2.

10 c Prints: 4 The array initializer, {{1,2},{3,4,5},{6,7,8,9},{}}, creates an array containing four components, and each is a reference to a subarray of type int[]. The size of the array referenced by variable a1, is 4, because a1 contains four components. The size of the first subarray is 2, the second is 3, the third is 4 and the fourth is zero.

11 a Prints: 2,3,4,0, The array initializer, {{1,2},{3,4,5},{6,7,8,9},{}}, creates an array containing four components, and each is a reference to an array of type int[]. The size of the first subarray is 2, the second is 3, the third is 4, and the fourth is zero. While the components of the array referenced by a1 are of type int[], the elements of the array referenced by a1 are of type int.

12 b Prints: 147258369 The array variable a1 is declared with the initializer, {{1,2,3},{4,5,6},{7,8,9}}. The array access expression, a1[0][1] = a1[first subarray][second element] = 2. If the argument of the print statement had been a1[i][j] then the output would have been 123456789. The tricky feature of this question is the reversal of i and j to produce the deceptive array access expression, a1[j][i]. The output is 147258369.

13 b Prints: A11 The declaration A11[] a1 = new A11[1] declares a variable a1 that references an array that contains one component of type A11. The declaration A11[][] a2 = new A11[2][] declares a variable a2 that references an array that contains two components of type A11[]. In other words, the array referenced by a2 contains two reference variables, and each is able to reference a subarray. The initial value of the subarray references is null. The size of the subarrays has not been specified. The declaration A11[][][] a3 = new A11[3][][] declares a variable a3 that references an array that contains three components of type A11[]{}. In other words, the array referenced by a3 contains three reference variables, and each is able to reference a subarray. The initial value of each subarray reference is null. The dimensions of the subarrays have not been specified. At line 5, a reference to the array referenced by a1 is assigned to each of the two components of the array referenced by a2, a2[0] = a2[1] = a1. At line 6, a reference to the array referenced by a2 is assigned to each of the three components of the array referenced by a3, a3[0] = a3[1] = a3[2] = a2. In other words, after line 6, each component of the array referenced by a3 is a reference to the array referenced by a2, and each element of the array referenced by a2 is a reference to the array referenced by a1, and the array referenced by a1 contains a reference to

an instance of class A11. Every element of the multi-dimensional array referenced by a3 contains a reference to a single instance of class A11. The print method invokes the `toString` method on the instance, and produces the output, A11.

14 a k Prints: null Run-time error    The declaration `A13[] a1 = new A13[1]` declares a variable a1 that references an array that contains one component of type A13. The declaration `A13[][] a2 = new A13[2][1]` declares a variable a2 that references an array that contains two components of type A13[]. In other words, the array referenced by a2 contains two references to two subarrays of type A13[]. The size of each subarray is 1. The declaration `A13[][][] a3 = new A13[3][3][3]` declares a variable a3 that references an array that contains three components of type A13[][]|. In other words, the array referenced by a3 contains three references to three subarrays of type A13[][]|. The dimensions of the subarrays are 3 X 3. The dimensions of the array referenced by a3 are 3 X 3 X 3. The number of elements in the array referenced by a3 is  $3 * 3 * 3 = 27$  elements. Each of the three subarrays contains three components, where each is a reference to a subarray of type A13[]|. Each of the 27 elements of the array referenced by a3 is a reference of type A13 that has been initialized to the default value of null. At line 7, a reference to the array referenced by a2 is assigned to each of the three components of the array referenced by a3, `a3[0] = a3[1] = a3[2] = a2`. Since the array referenced by a2 has dimensions 2 X 1, the array referenced by a3 now has dimensions 3 X 2 X 1. The number of elements is 6. An attempt to access any element beyond those 6 results in an exception at run-time.

## Arrays (Multi-Dimensional)

### Question 1

```
class MWC201 {
 public static void main(String[] args) {
 int[][] a1 = {{1,2,3},{4,5,6},{7,8,9,10}};
 System.out.print(a1[0][2]+"," +a1[1][0]+"," +a1[2][1]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 3,4,8
  - b. Prints: 7,2,6
  - c. Compile-time error
  - d. Run-time error
  - e. None of the above
-

## Question 2

```
class MWC202 {
 public static void main(String[] args) {
 int[] a1[] = {{1,2},{3,4,5},{6,7,8,9}};
 int []a2[] = {{1,2},{3,4,5},{6,7,8,9}};
 int a3[][] = {{1,2},{3,4,5},{6,7,8,9}};
 System.out.print(a1[0][1]+"," +a2[1][2]+"," +a3[2][3]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
  - b. Prints: 16
  - c. Prints: 1,5,9
  - d. Prints: 2,4,8
  - e. Prints: 2,5,9
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

## Question 3

```
class MWC207 {
 public static void main(String[] args) {
 int[][] a1 = {{1;2};{3;4;5};{6;7;8;9}};
 System.out.print(a1[0][1]+"," +a1[1][2]+"," +a1[2][3]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
- b. Prints: 16

- c. Prints: 1,5,9
  - d. Prints: 2,4,8
  - e. Prints: 2,5,9
  - f. **Compile-time error**
  - g. Run-time error
  - h. None of the above
- 

#### Question 4

```
class MWC208 {
 public static void main(String[] args) {
 int[][] a1 = ((1,2),(3,4,5),(6,7,8,9));
 System.out.print(a1[0][1]+"," +a1[1][2]+"," +a1[2][3]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
  - b. Prints: 16
  - c. Prints: 1,5,9
  - d. Prints: 2,4,8
  - e. Prints: 2,5,9
  - f. **Compile-time error**
  - g. Run-time error
  - h. None of the above
- 

#### Question 5

```
class MWC209 {
 public static void main(String[] args) {
 int[][] a1 = [[1,2],[3,4,5],[6,7,8,9]];
 int[][] a2 = [[1,2],[3,4,5],[6,7,8,9]];
 }
}
```

```
int[][] a3 = [[1,2],[3,4,5],[6,7,8,9]];
System.out.print(a1[0][1]+"," +a2[1][2]+"," +a3[2][3]);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
  - b. Prints: 16
  - c. Prints: 1,5,9
  - d. Prints: 2,4,8
  - e. Prints: 2,5,9
  - f. **Compile-time error**
  - g. Run-time error
  - h. None of the above
- 

### Question 6

```
class MWC210 {
 public static void main(String[] args) {
 int[] a2 = {1,2}, a3 = {3,4,5}, a4 = {6,7,8,9}; // 1
 int[][] a1 = {a2,a3,a4}; // 2
 System.out.print(a1[0][1]+"," +a1[1][2]+"," +a1[2][3]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
- b. Prints: 16
- c. Prints: 1,5,9
- d. Prints: 2,4,8
- e. **Prints: 2,5,9**
- f. Compile-time error
- g. Run-time error

- h. None of the above
- 

### Question 7

```
class MWC211 {
 public static void main(String[] args) {
 int a1[3]; // 1
 int []a2[]; // 2
 int[]a3; // 3
 int[] a4[]; // 4
 }
}
```

A compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
- 

### Question 8

```
class MWC212 {
 public static void main(String[] args) {
 int[] a1[],a2[]; // 1
 int []a3,[],a4; // 2
 int []a5,a6[]; // 3
 int[] a7,a8[]; // 4
 }
}
```

A compile-time error is generated at which line?

- a. 1

b. 2

- c. 3
  - d. 4
  - e. None of the above
- 

### Question 9

```
class MWC203 {
 public static void main(String[] args) {
 int[] a1[] = {new int[]{1,2},new int[]{3,4,5}};
 int []a2[] = new int[][]{{1,2},{3,4,5}};
 int a3[][] = {{1,2},new int[]{3,4,5}};
 System.out.print(a1[0][1]+"," +a2[1][0]+"," +a3[1][2]);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 14
  - b. Prints: 16
  - c. Prints: 1,2,4
  - d. Prints: 2,3,4
  - e. Prints: 2,3,5
  - f. Prints: 2,4,5
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

### Question 10

```
class MWC204 {
 public static void main(String[] args) {
 int[][] a1 = {{1,2},{3,4,5},{6,7,8,9},{}};
```

```
 System.out.print(a1.length);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0
  - b. Prints: 3
  - c. Prints: 4
  - d. Prints: 9
  - e. Prints: 10
  - f. Prints: 11
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

### Question 11

```
class MWC205 {
 public static void main(String[] args) {
 int[][] a1 = {{1,2},{3,4,5},{6,7,8,9},{}};
 for (int i = 0; i < a1.length; i++) {
 System.out.print(a1[i].length+",");
 }
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 2,3,4,0,
  - b. Prints: 1,2,5,0,
  - c. Compile-time error
  - d. Run-time error
  - e. None of the above
-

### Question 12

```
class MWC206 {
 public static void main (String[] args) {
 int[][] a1 = {{1,2,3},{4,5,6},{7,8,9}};
 for (int i = 0; i < 3; i++) {
 for (int j = 0; j < 3; j++) {
 System.out.print(a1[j][i]);
 }
 }
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 123456789
  - b. Prints: 147258369
  - c. Prints: 321654987
  - d. Prints: 369258147
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- 

### Question 13

```
class A11 {public String toString() {return "A11";}}
class A12 {
 public static void main(String[] arg) {
 A11[] a1 = new A11[1]; // 1
 A11[][] a2 = new A11[2][]; // 2
 A11[][][] a3 = new A11[3][][]; // 3
 a1[0] = new A11(); // 4
 a2[0] = a2[1] = a1; // 5
 a3[0] = a3[1] = a3[2] = a2; // 6
 System.out.print(a3[2][1][0]); // 7
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: null
  - b. Prints: A11
  - c. Compile-time error at 1.
  - d. Compile-time error at 2.
  - e. Compile-time error at 3.
  - f. Compile-time error at 4.
  - g. Compile-time error at 5.
  - h. Compile-time error at 6.
  - i. Compile-time error at 7.
  - j. Run-time error
  - k. None of the above
- 

#### Question 14

```
class A13 {}
class A14 {
 public static void main(String[] arg) {
 A13[] a1 = new A13[1]; // 1
 A13[][] a2 = new A13[2][1]; // 2
 A13[][][] a3 = new A13[3][3][3]; // 3
 System.out.print(a3[2][2][2]); // 4
 a1[0] = new A13(); // 5
 a2[0] = a2[1] = a1; // 6
 a3[0] = a3[1] = a3[2] = a2; // 7
 System.out.print(a3[2][2][2]); // 8
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: null

- b. Prints: nullnull
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.
- f. Compile-time error at 4.
- g. Compile-time error at 5.
- h. Compile-time error at 6.
- i. Compile-time error at 7.
- j. Compile-time error at 8.
- k. Run-time error

1a Prints: 3,4,8 An array variable a1 is declared, and the declaration contains the initializer  `{{1,2,3},{4,5,6},{7,8,9,10}}` . The initializer creates an array containing three components, and each is a reference to a subarray of type int[]. Each subarray contains components of type int, so the elements of the array referenced by a1 are of type int. The array access expression, `a1[0][2] = a1[1st subarray][third component] = 3`.

2e Prints: 2,5,9 The declarations of the array variables, a1, a2 and a3, are different in terms of the position of the square brackets, but each is of type int[][]. The array access expression, `a1[0][1] = a1[1st subarray][2nd component] = 2`. Each of the three array variable declarations has an array initializer. The same initializer,  `{{1,2},{3,4,5},{6,7,8,9}}` , is used in each of the three cases. The initializer specifies three components of type int[], so each component is a reference to an array object containing components of type int. The size of each of the subarrays is different: The size of the first is 2, the second is 3, and the third is 4.

3f Compile-time error The array initializer,  `{{1;2};{3;4;5};{6;7;8;9}}`  generates a compile-time error, because the commas have been replaced by semicolons. The array initializer should have been specified as follows:  `{{1,2},{3,4,5},{6,7,8,9}}` .

4f Compile-time error The array initializer,  `((1,2),(3,4,5),(6,7,8,9))` , generates a compile-time error, because the curly braces have been replaced by parentheses. The array initializer should have been specified as follows:  `{{1,2},{3,4,5},{6,7,8,9}}` .

5f Compile-time error The array initializer,  `[[1,2],[3,4,5],[6,7,8,9]]` , generates a compile-time error, because the curly braces have been replaced by square brackets. The array initializer should have been specified as follows:  `{{1,2},{3,4,5},{6,7,8,9}}` .

6e Prints: 2,5,9 The line marked 1 declares three array variables, a2, a3 and a4, and each references an array with components of type int. The line marked 2 declares an array variable, a1, where each component is initialized with a reference to one of the previously created arrays. The array access expression, `a1[0][1]`, is evaluated as follows: `a1[0][1] = a1[first subarray][second component] = 2`.

7a 1 A compile-time error occurs at the line marked 1, because the array variable declaration can not be used to specify the number of components contained in the array. Instead, the dimension expression should be contained in an array creation expression such as the following, new int[3].

8b 2 An array variable is declared by placing brackets after the identifier or after the type name. A compile-time error occurs at the line marked 2, because the brackets appearing before the identifier for array variable a4 are not associated with the type or the identifier.

9e Prints: 2,3,5 Each of the three array variable declarations, a1, a2 and a3, is different in terms of the position of the square brackets, but each declares a variable of type int[]]. Each of the three declarations contains an array initializer. In each case, the initializer could be simplified as follows: {{1,2},{3,4,5}}. The initializer creates an array containing two components, and each is a reference to an array containing components of type int. The size of each of the subarrays is different: The size of the first is 2 and the second is 3. The array access expression, a1[0][1] = a[1st subarray][2nd component] = 2.

10c Prints: 4 The array initializer, {{1,2},{3,4,5},{6,7,8,9},{}}, creates an array containing four components, and each is a reference to a subarray of type int[]. The size of the array referenced by variable a1, is 4, because a1 contains four components. The size of the first subarray is 2, the second is 3, the third is 4 and the fourth is zero.

11a

Prints: 2,3,4,0,

The array initializer, {{1,2},{3,4,5},{6,7,8,9},{}}, creates an array containing four components, and each is a reference to an array of type int[]. The size of the first subarray is 2, the second is 3, the third is 4, and the fourth is zero. While the components of the array referenced by a1 are of type int[], the elements of the array referenced by a1 are of type int.

12b Prints: 147258369 The array variable a1 is declared with the initializer, {{1,2,3},{4,5,6},{7,8,9}}. The array access expression, a1[0][1] = a1[first subarray][second element] = 2. If the argument of the print statement had been a1[i][j] then the output would have been 123456789. The tricky feature of this question is the reversal of i and j to produce the deceptive array access expression, a1[j][i]. The output is 147258369.

13b Prints: A11 The declaration A11[] a1 = new A11[1] declares a variable a1 that references an array that contains one component of type A11. The declaration A11[][] a2 = new A11[2][] declares a variable a2 that references an array that contains two components of type A11[]. In other words, the array referenced by a2 contains two reference variables, and each is able to reference a subarray. The initial value of the subarray references is null. The size of the subarrays has not been specified. The declaration A11[][][] a3 = new A11[3][][] declares a variable a3 that references an array that contains three components of type A11[]]. In other words, the array referenced by a3 contains three reference variables, and each is able to reference a subarray. The initial value of each subarray reference is null. The dimensions of the subarrays have not been specified. At line 5, a reference to the array referenced by a1 is assigned to each of the two components of the array referenced by a2, a2[0] = a2[1] = a1. At line 6, a reference to the array referenced by a2 is assigned to each of the three components of the array referenced by a3, a3[0] = a3[1] = a3[2] = a2. In other words, after line 6, each component of the array referenced by a3 is a reference to the array referenced by a2, and each element of the array referenced by a2 is a reference to the array referenced by a1, and the array referenced by a1 contains a reference to an instance of class A11. Every element of the multi-dimensional array referenced by a3 contains a reference to a single instance of class A11. The print method invokes the toString method on the instance, and produces the output, A11.

14a k Prints: null Run-time error The declaration A13[] a1 = new A13[1] declares a variable a1 that references an array that contains one component of type A13. The declaration A13[][] a2 = new A13[2][1] declares a variable a2 that references an array that contains two components of type A13[]]. In other words, the

array referenced by a2 contains two references to two subarrays of type A13[]. The size of each subarray is 1. The declaration A13[][][] a3 = new A13[3][3][3] declares a variable a3 that references an array that contains three components of type A13[][][]. In other words, the array referenced by a3 contains three references to three subarrays of type A13[][][]. The dimensions of the subarrays are 3 X 3. The dimensions of the array referenced by a3 are 3 X 3 X 3. The number of elements in the array referenced by a3 is  $3 * 3 * 3 = 27$  elements. Each of the three subarrays contains three components, where each is a reference to a subarray of type A13[]. Each of the 27 elements of the array referenced by a3 is a reference of type A13 that has been initialized to the default value of null. At line 7, a reference to the array referenced by a2 is assigned to each of the three components of the array referenced by a3, a3[0] = a3[1] = a3[2] = a2. Since the array referenced by a2 has dimensions 2 X 1, the array referenced by a3 now has dimensions 3 X 2 X 1. The number of elements is 6. An attempt to access any element beyond those 6 results in an exception at run-time.

## Class Declarations

### Question 1

```
public class Basics {} // 1 9036810446
class Basics1 {} // 2
protected class Basics2 {} // 3
private class Basics3 {} // 4
Class Basics4 {} // 5
```

Suppose these are top-level class declarations and not nested class declarations; and suppose that all of the declarations are contained in one file named Basics.java. Compile-time errors are generated at which lines?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
- 

### Question 2

```
public class Basics {} // 1
public class Basics2 {} // 2
public class Basics3 {} // 3
public class Basics4 {} // 4
```

Suppose these are top-level class declarations and not nested class declarations; and suppose that all of the declarations are contained in one file named Basics.java. A compile-time error is not generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
- 

### Question 3

Which of the following modifiers can be applied to a class that is not a nested class?

- a. Public
  - b. Protected
  - c. Private
  - d. Abstract
  - e. Static
  - f. Final
- 

### Question 4

Which of the follow statements is true.

- a. An anonymous class can be declared abstract.
  - b. A local class can be declared abstract.
  - c. An abstract class can be instantiated.
  - d. An abstract class is implicitly final.
  - e. An abstract class must declare at least one abstract method.
  - f. An abstract class can not extend a concrete class.
-

## Question 5

```
public class A {int i1; void m1() {}}
```

Which of the following statements are true?

- a. **class A extends Object.**
  - b. Field i1 is implicitly public, because class A is public.
  - c. Method m1 is implicitly public, because class A is public.
  - d. **The compiler will insert a default constructor implicitly.**
  - e. **The default constructor has no throws clause.**
  - f. The default constructor of class A has package access.
  - g. The default constructor accepts one parameter for each field in class A.
  - h. **The default constructor invokes the no-parameter constructor of the superclass.**
- 

## Question 6

```
abstract class A {} // 1
transient class G {} // 2
private class C {} // 3
static class E {} // 4
```

Suppose these are top-level class declarations and not nested class declarations. Which of these declarations would not produce a compile-time error?

- a. **1**
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
-

## Question 7

```
protected class D {} // 1
synchronized class F {} // 2
volatile class H {} // 3
final class B {} // 4
```

Suppose these are top-level class declarations and not nested class declarations. Which of these declarations would not produce a compile-time error?

- a. 1
- b. 2
- c. 3
- d. 4**
- e. None of the above

1 c d e 3 4 5 If a class C is declared as a member of an enclosing class then C may be declared using no access modifier or any of the three access modifiers, private, protected or public. However, if class C is not a local class, anonymous class or a member of an enclosing class or interface; then C may be declared with the public modifier or with package access (i.e. no modifier). The other two access modifiers, private and protected, are not applicable to any class that is not a member class. The class declaration, Class Basics4 {}, generates a compile-time error, because all of the letters of the reserved word class must be lower case.

2 a 1 Only one class in a source code file can be declared public. The other classes may not be public. Therefore, the declarations for classes Basics2, Basics3 and Basics4 generate compile-time errors.

3 a d f public abstract final The access modifiers, protected and private, can be applied to a class that is a member of an enclosing class, but can not be applied to a local class or a class that is not nested inside another class. The static modifier can be applied to a class that is a member of an enclosing class, but can not be applied to a local class or a class that is not nested inside another class. The public modifier can be applied to a top level class to allow the class to be accessed from outside of the package. The abstract modifier prevents the class from being instantiated. An abstract class may include zero, one or more abstract methods. The final modifier prevents a class from being extended.

4 b A local class can be declared abstract. An anonymous class can not be extended; therefore, an anonymous class can not be declared abstract. A local class can be abstract. An abstract class can not be instantiated. If a class declaration contains an abstract method, then the class must also be declared abstract. A class can be declared abstract even if it does not contain an abstract method. An abstract class can never be declared final.

5 a d e h class A extends Object. The compiler will insert a default constructor implicitly. The default constructor has no throws clause. The default constructor invokes the no-parameter constructor of the superclass. Field i1 and m1 both have package access. When no constructor is declared explicitly the compiler will insert one implicitly. The implicitly declared default constructor will have the same access privileges as the class. In this case, the class is public, so the default constructor is also public. The default constructor accepts no parameters and throws no exceptions.

6 a 1 The modifiers, private and static, can be applied to a nested class, but can not be applied to a class that is not nested. A class that is not nested can have public or package access, but not private. The transient modifier can not be applied to any class; because it is a field modifier.

7 d 4 The modifier, protected, can not be applied to a top level class, but can be applied to a nested class. The synchronized modifier can not be applied to any class, because it is a method modifier. The modifier, volatile, can not be applied to any class; because it is a field modifier.

## Nested Classes

### Question 1

Which of the follow are true statements.

- a. A nested class is any class that is declared within the body of another class or interface.
  - b. A nested class can not be declared within the body of an interface declaration.
  - c. An inner class is a nested class that is not static.
  - d. A nested class can not be declared static.
  - e. A named class is any class that is not anonymous.
- 

### Question 2

Which of the follow are true statements.

- a. A local class is declared within a method, constructor or block.
  - b. An anonymous class is always a local class.
  - c. A local class is a nested class.
  - d. A local class is a member class.
  - e. A local class is always a named class.
-

### Question 3

Which of the following are class modifiers? Select all that are applicable to a top-level class and all that are applicable to a nested class. It is not required that a modifier be applicable to both.

- a. **Abstract**
  - b. Extends
  - c. **final**
  - d. implements
  - e. **private**
  - f. **protected**
  - g. **public**
  - h. **static**
  - i. synchronized
  - j. Transient
  - k. Volatile
- 

### Question 4

Which of the following modifiers can be applied to a member class?

- a. **Abstract**
  - b. **final**
  - c. **public**
  - d. **protected**
  - e. **private**
  - f. **Static**
  - g. synchronized
  - h. transient
- 

### Question 5

Which of the following modifiers can be applied to a local class?

- a. public
  - b. protected
  - c. private
  - d. abstract
  - e. Static
  - f. Final
- 

#### Question 6

```
class Z {
 abstract class A {} // 1
 final class B {} // 2
 private class C {} // 3
 protected class D {} // 4
 public class E {} // 5
}
```

Which class declaration results in a compile-time error?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. None of the above**
- 

#### Question 7

```
class Z {
 static class F {} // 1
```

```
synchronized class G {} // 2
transient class H {} // 3
volatile class I {} // 4
}
```

Which class declaration does not result in a compile-time error?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
- 

### Question 8

```
class Z {
void m1() {
 abstract class A {} // 1
 final class B {} // 2
 private class C {} // 3
 protected class D {} // 4
 public class E {} // 5
}
```

Which class declarations result in compile-time errors?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
-

### Question 9

```
class Z {
 void m1() {
 static class F {} // 1
 synchronized class G {} // 2
 transient class H {} // 3
 volatile class I {} // 4
 abstract class A {} // 5
 final class B {} // 6
 }
}
```

Compile-time errors are generated at which lines?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. 6
- 

### Question 10

```
class A {
 A() {} // 1
 int A; // 2
 void A() {} // 3
 class A {} // 4
}
```

Which line results in a compile-time error?

- a. 1

- b. 2
  - c. 3
  - d. 4**
  - e. None of the above
- 

### Question 11

```
class A {
 int B; // 1
 void B() {} // 2
 class B {} // 3
}
```

Which line results in a compile-time error?

- a. 1
  - b. 2
  - c. 3
  - d. None of the above**
- 

### Question 12

```
abstract class A { // 1
 private abstract void m1(); // 2
 private abstract class B {} // 3
 private class C extends B {} // 4
}
```

Which line results in a compile-time error?

- a. 1
- b. 2**

- c. 3
  - d. 4
  - e. None of the above.
- 

### Question 13

```
abstract class A { // 1
 abstract final void m1(); // 2
 abstract final class B {} // 3
 class C extends B {} // 4
}
```

Which line does not result in a compile-time error?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
- 

### Question 14

```
abstract class A { // 1
 abstract synchronized void m1(); // 2
 abstract synchronized class B {} // 3
 synchronized class C extends B {} // 4
}
```

Which line does not result in a compile-time error?

- a. 1
- b. 2

c. 3

d. 4

e. None of the  
above

---

1 a c e

2 nested class is any class that is declared within the body of another class or interface. An inner class is a nested class that is not static. A named class is any class that is not anonymous.

Every class declared within the body of another class or interface is known as a nested

class. If the nested class does not have a name, then it is an anonymous class. If the nested class has a name, then it is not anonymous. If the nested class has a name and is not declared inside of a method, constructor or any block, then it is a member class. If a member class is not static, then it is an inner class. If a class is not nested, then it is a top level class. A nested class that is static is sometimes referred to as a top level class, but that usage of the term is confusing and should be avoided.

3 a c e

4 A local class is declared within a method, constructor or block. A local class is a nested class. A local class is always a named class. Every class declared within the body of another class is known as a nested class. If the nested class does not have a name, then it is an anonymous class. If the nested class has a name, then it is not anonymous. If the nested class has a name and is declared inside of a method, constructor or any block, then it is a local class. If a nested class does not have a name, then it can not be called a local class even if it is declared inside of a block. Therefore, an anonymous class is never called a local class. If the nested class has a name and is not declared inside of a method, constructor or any block, then it is a member class. If the class is not nested, then it is a top level class. Please note that this question is more rigorous than those that one might expect to find on the real exam. It has been included here as a convenient place to define some terms that are used to explain the answers to some of the other questions that appear in this mock exam.

3 a c e f g h abstract final private protected public static

The access modifiers, public, protected and private, can not be applied to a local class. The protected and private access modifiers can be applied to member classes, but not to a top level class. The public modifier can be applied to a top level class or a member class, but not a local class. The static modifier can be applied to a member class, but not to a local class or top level class. The keywords extends and implements are not modifiers. The synchronized modifier can be applied to a method, but not to a class. The modifiers, transient and volatile, can be applied to a field, but not to a class.

4 a b c d e f abstract final public protected private static

A nested class that has a name and is not a local class is a member class. A member class can be static or non-static. A non-static member class is also known as an inner class. All of the class modifiers may be applied to a member class. The modifiers, synchronized and transient, are not class modifiers.

5 d f abstract final

If a nested class has a name and is declared inside of a method, constructor or any block, then it is a local class. No access modifier can be applied to a local class declaration. A local class can not be static. A local class can be abstract, and can be final.

6 f None of the above

The following modifiers can be applied to a member class: abstract, private, protected, public, static and final.

7 a 1

Please note that this question asks which class declaration does **NOT** result in a compile-time error. The following modifiers can be applied to a member class: abstract, private, protected, public, static and final. The synchronized modifier can not be applied to any class, because it is a method modifier. The modifiers, transient and volatile, can not be applied to any class, because they are field modifiers.

8 c d e 3 4 5

The abstract and final modifiers can be applied to a method local class declaration.

9 a b c d 1 2 3 4

The modifiers, abstract and final, can be applied to a method local class declaration. The synchronized modifier can not be applied to a class, because it is a method modifier. The modifiers, transient and volatile, can not be applied to any class, because they are field modifiers.

10 d 4

A method and a field can share the same name, because they are used in different contexts and use different lookup procedures. They can even share the same name with the class in which they are declared. Please note that class names usually begin with an upper case letter while method and field names usually begin with a lower case letter. A nested class can not share the same name with its enclosing class.

11 d None of the above

A method, field, and a nested class can share the same name, because they are used in different contexts and use different lookup procedures. Please note that class names usually begin with an upper case letter while method and field names usually begin with a lower case letter. Also note that a nested class can not share the same name with its enclosing class; however, a method and field can share a name with the enclosing class. Even so, it is not a good idea to name a method with the name of the enclosing class, because it could be confused with a constructor.

12 b 2

An abstract method has no implementation, and is not useful until an extending class implements the method. Private methods are not inherited and can not be overridden. If an abstract method is declared private, then it can not be implemented in a subclass. Although a method may not be both private and abstract, a nested class can be; because another nested class can extend the abstract class and implement any abstract methods.

13 a 1

Please note that this question asks which line does **NOT** result in a compile-time error. An abstract method has no implementation and is not useful until an extending class implements the method. A final method can not be overridden by a subclass method. An abstract final method can not be implemented and is not legal. An abstract class may contain abstract method declarations and is assumed to be incomplete. A final class can not be extended. The implementation of an abstract final class could not be completed. The declaration of class C results in a compiler error, because a final class may not be listed in the extends clause.

14 a 1

Please note that this question asks which line does **NOT** result in a compile-time error. The modifier, synchronized, is a method modifier, but is not a class modifier. Any attempt to declare a synchronized class results in a compile-time error. Since the synchronized modifier specifies an implementation detail it makes no sense to use it with an abstract method that provides no implementation

## Anonymous Classes

### Question 1

Which of the following is a true statement?

- An anonymous class can extend only the Object class.

- b. An anonymous class can not implement an interface.
  - c. An anonymous class declaration can not have an implements clause.
  - d. An anonymous class declaration can name more than one interface in the implements clause.
  - e. The class instance creation expression for an anonymous class must never include arguments.
  - f. None of the above
- 

## Question 2

Which of the following are true statements?

- a. An anonymous class is implicitly abstract.
  - b. An anonymous class is implicitly final.
  - c. An anonymous class is implicitly static.
  - d. A static reference variable can reference an instance of an anonymous class.
  - e. An anonymous class declaration must have at least one explicit constructor declaration.
  - f. An anonymous class declaration can have more than one explicit constructor declaration.
- 

## Question 3

```
abstract class A {
 private int x = 4, y = 2;
 public int x() {return x;}
 public void x(int x) {this.x = x;}
 public int y() {return y;}
 public void y(int y) {this.y = y;}
 public abstract int math();
}
class B {
 static A a1 = new A(2,1) {
 public A(int i1, int i2) {x(i1);y(i2);};
 public int math() {return x() + y();}
 };
 public static void main(String[] args) {
```

```
System.out.print(a1.math());
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 8
  - b. Prints: 3122
  - c. **Compile-time error**
  - d. Run-time error
  - e. None of the above
- 

#### Question 4

```
class A {
 private static int f1 = 1;
 private int f2 = 2;
 void m1(int p1, final int p2) {
 int l1 = 5;
 final int l2 = 6;
 Object x = new Object() {
 int a = f1; // 1
 int b = f2; // 2
 int c = p1; // 3
 int d = p2; // 4
 int e = l1; // 5
 int f = l2; // 6
 };}
}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. **3**

d. 4

e. 5

f. 6

---

### Question 5

```
abstract class A {
 private int x = 1, y = 1;
 public A(int x, int y) {this.x = x; this.y = y;}
 public abstract int math();
}
class B {
 static A a1 = new A(2,1) {public int math() {return x + y;}};
 static A a2 = new A(2,1) {public int math() {return x - y;}};
 static A a3 = new A(2,1) {public int math() {return x * y;}};
 static A a4 = new A(2,1) {public int math() {return x / y;}};
 public static void main(String[] args) {
 System.out.print(" " + a1.math() + a2.math() +
 a3.math() + a4.math());
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 3122
  - b. Prints: 2011
  - c. **Compile-time error**
  - d. Run-time error
  - e. None of the above
- 

### Question 6

```
abstract class A {
```

```

private int x = 4, y = 2;
public int x() {return x;}
public void x(int x) {this.x = x;}
public int y() {return y;}
public void y(int y) {this.y = y;}
public abstract int math();
}
class B {
 static A a1 = new A() {public int math() {return x() + y();}}
 static A a2 = new A() {public int math() {return x() - y();}}
 static A a3 = new A() {public int math() {return x() * y();}}
 static A a4 = new A() {public int math() {return x() / y();}}
 public static void main(String[] args) {
 System.out.print(" " + a1.math() + a2.math() +
 a3.math() + a4.math());
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 18
  - b. Prints: 6282
  - c. **Compile-time error**
  - d. Run-time error
  - e. None of the above
- 

### Question 7

```

class A {String m1() {return "A.m1";}}
interface B {String m2();}
class C {
 static class D extends A implements B {
 public String m1() {return "D.m1";}
 public String m2() {return "D.m2";}
 }
}

```

```

static A a1 = new A() implements B {
 public String m1() {return "m1";}
 public String m2() {return "m2";}
};
public static void main(String[] args) {
 System.out.print(a1.m1() + "," + new C.D().m2());
}

```

What is the result of attempting to compile and run the program?

- a. Prints: m1,D.m2
  - b. Prints: A.m1,D.m2
  - c. **Compile-time error**
  - d. Run-time error
  - e. None of the above
- 

### Question 8

```

abstract class A {
 private int x = 4, y = 2;
 public A(int i1, int i2) {x=i1;y=i2;}
 public int x() {return x;}
 public void x(int x) {this.x = x;}
 public int y() {return y;}
 public void y(int y) {this.y = y;}
 public abstract int math();
}

class B {
 static A a1 = new A(2,1) {public int math() {return x() + y();}};
 static A a2 = new A(2,1) {public int math() {return x() - y();}};
 static A a3 = new A(2,1) {public int math() {return x() * y();}};
 static A a4 = new A(2,1) {public int math() {return x() / y();}};
 public static void main(String[] args) {
 System.out.print(" " + a1.math() + a2.math() +

```

```
a3.math() + a4.math());
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 8
- b. Prints: 3122
- c. Compile-time error
- d. Run-time error
- e. None of the above

1 c An anonymous class declaration can not have an implements clause. A class instance creation expression can create an instance of a named class or an anonymous class. For example, the class instance creation expression new Object() creates an instance of the class named Object. If a class body appears in the class instance creation expression, then an anonymous class is created. For example, the expression new Object() {void doNothing(){}} creates an instance of an anonymous class that extends Object and implements a method named doNothing. In other words, if a class name immediately follows the keyword new, then the anonymous class extends the named class. When a named class is being extended, then the class instance creation expression can contain an optional argument list. The arguments will be passed to the direct superclass constructor that has a matching parameter list. An anonymous class declaration can not have an implements clause or an extends clause.

2 b d An anonymous class is implicitly final. A static reference variable can reference an instance of an anonymous class.

An anonymous class can extend Object and implement an interface, or the anonymous class can extend a named class including Object. An anonymous class can not be extended; so it can not be abstract. An anonymous class declaration always creates an instance of a class; so it is not surprising that an anonymous class can not be declared static. Even so, a static reference variable can refer to an anonymous class. A constructor shares the same name as the class in which it is declared, but an anonymous class has no name. For that reason, it is not surprising that an anonymous class declaration can not contain an explicit constructor declaration. Instead, an anonymous class can contain an instance initializer.

3 c Compile-time error

An anonymous class declaration can not contain an explicit declaration of a constructor.

4 c e 3 5

Local method variables and method parameters are stored on the stack and go out of scope after the method is exited. Although a local reference variable is stored on the stack, the referenced object is stored on the heap; so the object can continue to exist long after the method runs to completion. An object that is instantiated within a method or block is not permitted to refer to a variable that is declared within the method or block unless the variable is declared final and the variable declaration precedes the creation of the object.

5 c Compile-time error

The instance variables x and y in class A are private, so they are not accessible to the anonymous subclasses. If you want to access the private variables of a superclass, then you will need to add accessor methods to the superclass such as getX and getY.

6 c Compile-time error When an anonymous class declaration is the last thing that appears in a statement, then a semicolon must follow the declaration. Anonymous class declarations provide an excellent opportunity for trick questions involving statements with missing semicolons.

7 c Compile-time error An anonymous class can extend Object and implement an interface, or the anonymous class can extend a named class including Object. An anonymous class declaration can not have an implements clause. In this case, the declaration of the anonymous class referenced by a1 generates a compile-time error as a result of the attempt to extend class A and implement interface B.

8 b Prints: 3122 The arguments that appear in the class instance creation expression of an anonymous class are passed to a constructor of the superclass.

## Constructors

### Question 1

```
class A {A(int i) {}} // 1
class B extends A {} // 2
```

Which of the following statements are true?

- a. The compiler attempts to create a default constructor for class A.
  - b. The compiler attempts to create a default constructor for class B.
  - c. Compile-time error at 1.
  - d. Compile-time error at 2.
- 

### Question 2

Which of the following modifiers can be applied to a constructor?

- a. private
  - b. abstract
  - c. final
  - d. Volatile
  - e. Native
  - f. None of the above.
- 

### Question 3

Which of the following techniques can be used to prevent the instantiation of a class by any code outside of the class?

- a. Do not declare any constructors.
  - b. Do not use a return statement in the constructor.
  - c. Declare all constructors using the keyword void to indicate that nothing is returned.
  - d. **Declare all constructors using the private access modifier.**
  - e. None of the above.
- 

#### Question 4

Which of the following modifiers can be applied to a constructor?

- a. **protected**
  - b. **public**
  - c. static
  - d. synchronized
  - e. Transient
- 

#### Question 5

Which of the following statements are true?

- a. **The compiler will create a default constructor if no other constructor is declared.**
  - b. **The default constructor takes no arguments.**
  - c. If a class A has a direct superclass, then the default constructor of class A invokes the no-argument constructor of the superclass.
  - d. The default constructor declares Exception in the throws clause.
  - e. The default constructor is always given the private access modifier.
  - f. The default constructor is always given the public modifier.
  - g. The default constructor is always given default package access.
-

## Question 6

Suppose that the compiler generates a default constructor for a class. If a compile-time error is to be avoided, which of the following must be true?

- a. The superclass must not have any constructor other than a default constructor.
  - b. The superclass must not have an accessible no-argument constructor.
  - c. The no-argument superclass constructor must not have a throws clause that includes a checked exception.
  - d. The no-argument superclass constructor must be declared private.
  - e. None of the above
- 

## Question 7

```
class A {A() throws Exception {}} // 1
class B extends A {B() throws Exception {}} // 2
class C extends A {} // 3
```

Which of the following statements is true?

- a. Compile-time error at 1.
- b. Compile-time error at 2.
- c. Compile-time error at 3.
- d. None of the above

1 b d

2 The compiler attempts to create a default constructor for class B. Compile-time error at 2. If no constructor is declared explicitly, then the compiler will implicitly create a default constructor that accepts no parameters, has no throws clause, and invokes its superclass constructor. Since class A has an explicitly declared constructor, the compiler will not create an implicit default constructor. Class B does not have an explicit constructor declaration, so the compiler attempts to create a default constructor. Since class A does not have a no-parameter constructor, the attempt by class B to invoke the no parameter constructor of A would fail. As a result, a compiler error is generated at marker 2.

2 a private

Constructors are not inherited and can not be overridden, so there is no need for the final modifier in a constructor declaration. Furthermore, an abstract constructor would be useless, since it could never be implemented. The volatile modifier can be applied to a field, but not to a constructor. Native constructors are not permitted, because it would be difficult for Java to verify that the native constructor properly invokes the superclass constructor.

3 d Declare all constructors using the private access modifier. If no constructors are declared explicitly; then the compiler will create one implicitly, and it will have the same access modifier as the class. The explicit declaration of any constructor will prevent the creation of a default constructor. If all

constructors are declared private, then code outside of the class will not have access to the constructors and will not have the ability to create an instance of the class. Constructors do not return a value and constructor declarations do not include a return type, so the keyword void is not applicable to a constructor declaration.

4 a b      protected public

Constructors can not be inherited, so an abstract constructor would be useless, since it could never be implemented. A static constructor would also be useless--or nearly so--since it would be unable to access the non-static members of the new instance. An object is not available to multiple threads during the construction process, so the synchronized modifier would not provide additional protection. The transient modifier can be applied to a field, but not a constructor.

5 a b c    The compiler will create a default constructor if no other constructor is declared. The default constructor takes no arguments. If a class A has a direct superclass, then the default constructor of class A invokes the no-argument constructor of the superclass. If no constructor is declared explicitly, then the compiler will implicitly insert a default constructor. The default constructor takes no arguments. The primordial class Object has no superclass; so the default constructor of type Object does not invoke a superclass constructor. If a class A has a direct superclass, then the default constructor of class A will invoke the no-argument superclass constructor. It is unlikely that the real exam would try to trick you with a question that requires you to know that the constructor of type Object does not invoke a superclass constructor. For the purposes of the real exam, it might be safer to overlook that particular unique feature of type Object. If a subclass constructor attempts to invoke the no-argument superclass constructor when none exists, then a compile-time error is generated. The access modifier implicitly assigned to the default constructor is the same as that assigned to the class. The default constructor does not have a throws clause. Consequently, a compile-time error is generated if the no-argument constructor of the superclass has a throws clause.

6 c           The no-argument superclass constructor must not have a throws clause that includes a checked exception.

The default constructor takes no arguments, and it invokes the superclass constructor with no arguments. If the superclass does not have an accessible no-argument constructor, then a compile-time error is generated. The default constructor does not have a throws clause. Consequently, a compile-time error is generated if the no-parameter constructor of the superclass has a throws clause.

7 c           Compile-time error at 3.

The compiler creates a constructor for class C implicitly. The implicitly created constructor accepts no parameters and has no throws clause. The constructors for class B and class C both invoke the constructor for A. The constructor for class A declares Exception in the throws clause. Since the constructors for B and C invoke the constructor for A implicitly, both B and C must declare Exception in their throws clause. A compile-time error is generated at marker 3, because the default constructor does not declare Exception in the throws clause.

## Field Declarations

### Question 1

```
class JSC102 {
 public static void main (String[] args) {
 private int x = 1; protected int y = 2; public int z = 3;
 System.out.println(x+y+z);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 123
  - b. Prints: 1 2 3
  - c. Prints: 6
  - d. Run-time error
  - e. **Compile-time error**
  - f. None of the above
- 

## Question 2

```
class JSC105 {
 private static int x; protected static int y; public static int z;
 public static void main (String[] args) {System.out.println(x+y+z);}
}
```

What is the result of attempting to compile and run the program?

- a. Prints nothing.
  - b. Prints an undefined value.
  - c. Prints: null
  - d. **Prints: 0**
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- 

## Question 3

```
class Red {
 public int a; public static int b;
 public static void main (String[] in) {
 Red r1 = new Red(), r2 = new Red(); r1.a++; r1.b++;
```

```
System.out.print(r1.a+, "+r1.b+, "+r2.a+, "+r2.b);
{}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0, 0, 0, 0
  - b. Prints: 0, 1, 1, 1
  - c. Prints: 1, 1, 1, 0
  - d. Prints: 1, 1, 0, 1
  - e. Prints: 1, 1, 0, 0
  - f. Prints: 1, 1, 1, 1
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

#### Question 4

```
class Basics {
 int x = 1, y = 2;
 public static void main (String[] args) {System.out.println(x+y);}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: x+y
  - b. Prints: 12
  - c. Prints: 3
  - d. Run-time error
  - e. Compile-time error
  - f. None of the above
- 

#### Question 5

Which of the following modifiers can be applied to the declaration of a field?

- a. abstract
  - b. final
  - c. private
  - d. Protected
  - e. Public
- 

### Question 6

Which of the following modifiers can be applied to the declaration of a field?

- a. Static
  - b. synchronized
  - c. Transient
  - d. Volatile
  - e. Native
- 

### Question 7

Which of the following is used to prevent the serialization of a non-static field?

- a. Final
  - b. Protected
  - c. Synchronized
  - d. Transient
  - e. Volatile
  - f. Native
- 

### Question 8

Which of the following statements are true?

- a. A value can not be assigned to a final field more than once.
  - b. A value can be assigned to a final field at any time or not at all.
  - c. Only static variables can be declared final.
  - d. A compile-time error is thrown if a blank final instance variable is not assigned a value before the end of each constructor.
  - e. A field can not be declared both final and volatile.
- 

### Question 9

```
class JSC101 {
 void m1() {
 public int a; // 1
 protected int b; // 2
 private int c; // 3
 static int d; // 4
 transient int e; // 5
 volatile int f; // 6
 final int g = 1; // 7
 }
}
```

Compile-time errors are generated at which lines?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. 6
  - g. 7
- 

### Question 10

```
class JSC103 {
 transient float e = 1; // 1
 volatile float f = 1; // 2
 abstract float j = 1; // 3
 final float g = 1; // 4
 private final float k = 1; // 5
 private transient float l = 1; // 6
 volatile final float m = 1; // 7
}
```

Compile-time errors are generated at which lines?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. 6
  - g. 7
- 

### Question 11

```
class JSC104{
 public float a; // 1
 protected float b; // 2
 private float c; // 3
 static float d; // 4
 synchronized float i; // 5
}
```

A compile-time error is generated at which line?

- a. 1
- b. 2

- c. 3
  - d. 4
  - e. 5**
  - f. None of the above.
- 

### Question 12

```
class Identifiers {
 int i1; // 1
 int _i2; // 2
 int i_3; // 3
 int #i4; // 4
 int $i5; // 5
 int %i6; // 6
 int i$7; // 7
 int 8i; // 8
}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4**
- e. 5
- f. 6**
- g. 7
- h. 8**

1e

Compile-time error

The access modifiers public, protected and private, can not be applied to variables declared inside methods.

2d

Prints: 0

Member variables are initialized automatically. Type int variables are initialized to zero.

3d

Prints: 1, 1, 0, 1

Both instances of class Red share a single copy of the static field b. Although field b is only incremented using the r1 reference, the change is visible in the r2 instance of class Red.

4e

Compile-time error

A static method can not access a non-static variable.

5b c d e

final private protected public

A field is a class member. A static field is sometimes called a class variable. A non-static field is sometimes called an instance variable. A variable declaration that is immediately contained by a block such as a method body is called a local variable. The access modifiers, private, protected and public, can be applied to a field. A final field can not have its value assigned more than once. The abstract modifier may be applied to methods but not to fields.

6a c d

static transient volatile

A transient field is not part of the persistent state of an object. Transient fields are not serialized. Fields that are shared between threads may be marked volatile to force each thread to reconcile its own working copy of the field with the master copy stored in the main memory. The synchronized modifier may be applied to methods but not to fields.

7d

transient

A transient field is not part of the persistent state of an object, so it is not serialized. A static field is also not part of the persistent state of an object, and also is not serialized.

8a d e

A value can not be assigned to a final field more than once. A compile-time error is thrown if a blank final instance variable is not assigned a value before the end of each constructor. A field can not be declared both final and volatile.

Static and non-static field variables may be declared final. All final fields must be definitely assigned a value once and only once. If the declaration of a final variable does not include an initializer then the variable is called a blank final. All blank, final, static variables must be assigned in a static initializer. All blank final non-static variables must be assigned by the end of the instance construction process. A field is sometimes shared between threads. The volatile modifier is

used to force threads to reconcile their own working copy of a field with the master copy in main memory. If a field is declared final then its value does not change and there is no need for threads to reconcile their own working copies of the variable with the master copy in main memory.

9a b c d e f  
1 2 3 4 5 6

A variable that is local to a method can not be accessed from outside of the class, so the access modifiers are not useful and not legal. A variable that is local to a method can not be part of the persistent state of an object, so the transient modifier is not useful and not legal. Local variables can not be shared between threads, so the volatile modifier is not useful and not legal. A local variable can be declared final to prevent its value from being assigned more than once. If the value of the variable needs to be accessed from a local class or an anonymous class, then the local variable or method parameter must be declared final.

10c g  
3 7

The abstract modifier can be applied to a class and a method but not a field. A field is sometimes shared between threads. The volatile modifier is used to force threads to reconcile their own working copy of a field with the master copy in main memory. If a field is declared final then its value does not change and there is no need for threads to reconcile their own working copies of the variable with the master copy in main memory.

11e  
5

The synchronized modifier is a method modifier and can not be applied to a field.

12d f h  
4 6 8

The first letter of an identifier can be any *Unicode JLS 3.1* character that is a *Java letter*. The first letter can not be a number. For historical reasons, the dollar sign \$ and underscore \_ are considered Java letters along with many currency symbols in use in the world today.

## Method Declarations

### Question 1

Which of the following modifiers can not be applied to a method?

- a. abstract
- b. private
- c. protected
- d. public
- e. Volatile

- 
- f. None of the above.
- 

### Question 2

Which of the following modifiers can not be applied to a method?

- a. Final
  - b. Static
  - c. synchronized
  - d. transient
  - e. native
  - f. None of the above.
- 

### Question 3

Which of the following modifiers can not be used with the abstract modifier in a method declaration?

- a. final
  - b. private
  - c. protected
  - d. public
  - e. Static
  - f. synchronized
  - g. native
- 

### Question 4

Which of the following statements is true?

- a. An abstract method can not be overridden by an abstract method.

- b. An instance method that is not abstract can not be overridden by an abstract method.
  - c. An abstract method declaration can not include a throws clause.
  - d. The body of an abstract method is represented by a set of empty brackets.
  - e. **None of the above.**
- 

### Question 5

Which of the following statements is not true?

- a. **A static method is also known as a class method.**
  - b. A class method is not associated with a particular instance of the class.
  - c. The keyword, this, can not be used inside the body of a static method.
  - d. **The keyword, super, may be used in the body of a static method.**
  - e. A method that is not static is known as an instance method.
  - f. None of the above.
- 

### Question 6

Which of the following statements are true?

- a. **A final method can not be overridden.**
  - b. **All methods declared in a final class are implicitly final.**
  - c. The methods declared in a final class must be explicitly declared final or a compile-time error occurs.
  - d. It is a compile-time error if a private method is declared final.
  - e. **A machine-code generator can inline the body of a final method.**
- 

### Question 7

Which of the following statements are true?

- a. If an accessible superclass method is static, then any method with the same signature in a subclass must also be static.
- b. If a superclass method is synchronized, then the overriding method must also be synchronized.
- c. If a superclass method is public, then the overriding method must also be public.
- d. If a superclass method is native, then the overriding method must also be native.
- e. If a superclass method is protected, then the overriding method must be protected or public.

1e

volatile

An abstract method declaration provides no method body. If one abstract method appears within a class declaration, then the entire class must be declared abstract and the class can not be instantiated. The access modifiers, private, protected and public, can be applied to a method. The field modifiers, transient and volatile, are not applicable to method declarations.

2d

transient

A final method can not be overridden. A static method is associated with a class, but not a particular instance of the class. A thread can not enter a synchronized method without first acquiring a lock. The field modifiers, transient and volatile, are not applicable to method declarations. A native method is implemented in platform-dependent code.

3a b e f g

final private static synchronized native

A final or private method can not be overridden, and can not be abstract. An abstract method declaration provides no implementation of the method, and all implementation details are left to the overriding method in the subclass. The synchronized modifier specifies an implementation detail that can be omitted from the declaration of an overriding method of a subclass, so it makes no sense to allow the use of the synchronized modifier in an abstract method declaration.

4e

None of the above.

An abstract method of a subclass can override by an abstract method of a superclass. The overriding abstract method declaration can be a good place to add comments. An abstract method of a subclass can override a concrete implementation of a method of a superclass. An abstract method declaration can have a throws clause. The body of an abstract method is a semicolon.

5d

The keyword, super, may be used in the body of a static method.

The keyword, this, refers to the instance on which the method has been invoked. A static method--also known as a *class method*--is not invoked on a particular instance of an object, but is instead invoked on the class. Since a static method is not associated with a particular instance, an attempt to use the keyword, this, within the body of a static method results in a Compile-time error. Similarly, the keyword, super, can not be used within the body of a static method.

6a b e

A final method can not be overridden. All methods declared in a final class are implicitly final. A machine-code generator can inline the body of a final method. All methods declared in a final class are implicitly final. It is permissible--but not required--that all such methods be explicitly declared final. All private methods are implicitly final. It is permissible--but not required--that all private methods be explicitly declared final. The body of an inline method is inserted directly into the code at the point where the method is invoked. If the method is invoked at 10 different points in the code, then the body can be copied to all 10 points. Inline code runs very quickly, because it removes the need to do the work associated with a method invocation. If the method is executed repeatedly in a loop, then inlining can improve performance significantly. The machine-code generator has the option to inline a final method.

7a c e

If an accessible superclass method is static, then any method with the same signature in a subclass must also be static. If a superclass method is public, then the overriding method must also be public. If a superclass method is protected, then the overriding method must be protected or public.

The signature of a method is the name of the method and the number and types of the method parameters. The return type is not part of the method signature. An instance method declared in a subclass overrides an accessible superclass method with the same signature. A static method of a subclass hides--but does not override--an accessible superclass method with the same signature. A compile-time error is generated if a subclass contains a declaration of an instance method that shares the same signature as an accessible static method of the superclass. The modifiers, synchronized, native and strictfp, specify implementation details that the programmer is free to change in a subclass. Similarly, a subclass can override a concrete implementation of a method with an abstract method declaration. A subclass method may not have weaker access than the overridden superclass method. For example, a public method may not be overridden by a private method. However, a subclass method can provide greater access than a superclass method. For example, a protected method can be overridden by a public method.

## Return Types

### Question 1

```
class JSC201 {
 static byte m1() {
 final char c1 = '\u0001';
 return c1; // 1
 }
 static byte m2(final char c2) {return c2;} // 2
 public static void main(String[] args) {
 char c3 = '\u0003';
 System.out.print(""+m1()+m2(c3)); // 3
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 13
  - b. Prints: 4
  - c. Compile-time error at 1
  - d. Compile-time error at 2
  - e. Run-time error
  - f. None of the above
- 

### Question 2

```
class JSC202 {
 static byte m1() {final short s1 = 2; return s1;} // 1
 static byte m2(final short s2) {return s2;} // 2
 public static void main(String[] args) {
 short s3 = 4;
 System.out.print(""+m1()+m2(s3)); // 3
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 24
  - b. Prints: 6
  - c. Compile-time error at 1.
  - d. **Compile-time error at 2.**
  - e. Run-time error
  - f. None of the above
- 

### Question 3

```
class JSC203 {
 static int m1(byte b) {return b;} // 1
 static int m2(char c) {return c;} // 2
 static int m3(long l) {return l;} // 3
```

```
public static void main(String[] args) {
 byte b = 1; char c = '\u0002'; long l = 4L;
 System.out.print(""+m1(b)+m2(c)+m3(l));
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 124
  - b. Prints: 7
  - c. Compile-time error at 1.
  - d. Compile-time error at 2.
  - e. **Compile-time error at 3.**
  - f. Run-time error
- 

#### Question 4

```
class JSC204 {
 static int m1(short s) {return s;} // 1
 static int m2(float f) {return f;} // 2
 public static void main(String[] args) {
 short s = 3; float f = 5.0f;
 System.out.print(""+m1(s)+m2(f));
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 35.0
  - b. Prints: 8.0
  - c. Compile-time error at 1.
  - d. **Compile-time error at 2.**
  - e. Run-time error
  - f. None of the above
-

## Question 5

```
class JSC205 {
 static int m1(int i) {return i;} // 1
 static void m2(int i) {return i;} // 2
 static int m3(int i) {return;} // 3
 public static void main(String[] args) {
 System.out.print(""+m1(1)+m2(2)+m3(3)); // 4
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 123
- b. Prints: 6
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.
- f. Compile-time error at 4.

1d

Compile-time error at 2

There is a compile-time error at 2. The char type variable c2 is not a compile-time constant, so it can not be assigned to type byte without an explicit cast. The method parameter c2 is declared final, so the value of c2 can not be changed within method m2. The value of method parameter c2 is set at run time to the value of the argument that is provided when m2 is invoked at line 3. For that reason, the method parameter c2 is not a compile-time constant. In method m2, the statement, "return c2;", is a return statement with an expression, c2. A compile-time error occurs if the type of the expression is not assignable to the declared result type of the method. The declared result type of method m2 is byte. The return statement attempts to return the value of the char type variable c2. If a char value is a compile-time constant, and if the value falls within the range of type byte, then the char value is assignable to type byte. In method m2, variable c2 is not a compile-time constant, because the value of c2 is not known at compile time. Instead, the value of c2 is assigned at run time to the value of the argument. Since the char type variable c2 is not a compile-time constant, the value of variable c2 is not assignable to the return type of method m2 without an explicit cast. While the declaration of method m2 produces a compile-time error, the declaration of method m1 does not. The local variable c1 is declared final and the value is set at compile time; so c1 is a compile-time constant. The value \u0001 falls within the range of type byte; so the value of the compile-time constant c1 is assignable to the return type of method m1 without an explicit cast.

2d

Compile-time error at 2.

There is a compile-time error at 2. The short type variable s2 is not a compile-time constant, so it can not be assigned to type byte without an explicit cast. The method parameter s2 is declared final, so the value of s2 can not be changed within method m2. The value of method parameter s2 is set at run time to the value of the argument that is provided when m2 is invoked at line 3. For that reason, the method parameter s2 is not a compile-time constant. In method m2, the statement,

"return s2; ", is a return statement with an expression, s2. A compile-time error occurs if the type of the expression is not assignable to the declared result type of the method. The declared result type of method m2 is byte. The return statement attempts to return the value of the short type variable s2. If a short value is a compile-time constant, and if the value falls within the range of type byte, then the short value is assignable to type byte without an explicit cast. In method m2, variable s2 is not a compile-time constant, because the value of s2 is not known at compile time. Instead, the value of s2 is assigned at run time to the value of the argument. Since the short type variable s2 is not a compile-time constant, the value of variable s2 is not assignable to the return type of method m2 without an explicit cast. While the declaration of method m2 produces a compile-time error, the declaration of method m1 does not. The local variable s1 is declared final and the value is set at compile time; so s1 is a compile-time constant. The value 2 falls within the range of type byte; so the value of the compile-time constant s1 is assignable to the return type of method m1 without an explicit cast.

3e

Compile-time error at 3.

There is a compile-time error at line 3. The long type variable, l, can not be assigned to type int without an explicit cast. The statement, "return l;", is a return statement with an expression, l. A compile-time error occurs if the type of the expression is not assignable to the declared result type of the method. The declared result type of the method, m3, is int. The type of the variable, l, is long, so an explicit cast is needed to perform the narrowing primitive conversion, "return (int)l;". The declarations of methods m1 and m2 do not generate compile-time errors, because the types of the expressions contained in the return statements are assignable to type int. Widening conversions from types byte, char, or short to type int do not require an explicit cast.

4d

Compile-time error at 2.

There is a compile-time error at 2, because a narrowing primitive conversion from type float to type int requires an explicit cast. There is no compile-time error at 1, because widening primitive conversions from types byte, char, or short to type int do not require an explicit cast.

5d e f

Compile-time error at 2. Compile-time error at 3. Compile-time error at 4.

At line 2, the statement, "return i;", contains the expression, i. The enclosing method, m2, is declared void. The return statement generates a compile-time error, because it contains an expression. At line 3, the statement, "return;", does not contain an expression. The enclosing method, m3, is declared with the result type, int. The return statement generates a compile-time error, because it does not contain an expression that produces a value that is assignable to the declared result type

.

## Control

### Question 1

```
class JMM102 {
 public static void main(String args[]) {
 for (int i = 0; i<5 ;i++) {
 switch(i) {
 case 0: System.out.print("v ");break;
 case 1: System.out.print("w ");
 case 2: System.out.print("x ");break;
 }
 }
 }
}
```

```
case 3: System.out.print("y ");
case 4: System.out.print("z ");break;
default: System.out.print("d ");
}}}}
```

What is the result of attempting to compile and run the program?

- a. Prints: v w x y z
  - b. Prints: v w x y z d
  - c. Prints: v w x x y z z
  - d. Prints: v w w x y y z d
  - e. Prints: d d d d d
  - f. Run-time error
  - g. Compile-time error
  - h. None of the above
- 

## Question 2

```
class JMM103 {
 public static void main(String args[]) {
 for (int i = 0; i < 5 ;i++) {
 switch(i) {
 0: System.out.print("v ");break;
 1: System.out.print("w ");
 2: System.out.print("x ");break;
 3: System.out.print("y ");
 4: System.out.print("z ");break;
 }}}}
```

What is the result of attempting to compile and run the program?

- a. Prints: v w x y z
- b. Prints: v w x x y z z

- c. Prints: v w w x y z
  - d. Run-time error.
  - e. **Compile-time error.**
  - f. None of the above.
- 

### Question 3

```
class JMM107 {
 public static void main(String[] args) {
 boolean b = true;
 if (b = false) {System.out.print("A");}
 } else if (b) {System.out.print("B");}
 } else {System.out.print("C");}
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
  - b. Prints: B
  - c. **Prints: C**
  - d. Run-time error
  - e. Compile-time error
  - f. None of the above
- 

### Question 4

```
class JMM108 {
 static boolean b;
 public static void main(String[] args) {
 if (b) {System.out.print("A");}
 } else if (b = false) {System.out.print("B");}
 } else if (b) {System.out.print("C");}
 } else if (!b) {System.out.print("D");}
```

```
 } else {System.out.print("E");}
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
  - b. Prints: B
  - c. Prints: C
  - d. Prints: D
  - e. Prints: E
  - f. Run-time error
  - g. Compile-time error
  - h. None of the above
- 

### Question 5

```
class JMM109 {
 public static void main(String[] args) {
 boolean b;
 if (b = false) {System.out.print("A");}
 } else if (b) {System.out.print("B");}
 } else if (!b) {System.out.print("C");}
 } else {System.out.print("D");}
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
- b. Prints: B
- c. Prints: C
- d. Prints: D
- e. Run-time error
- f. Compile-time error

- g. None of the above
- 

### Question 6

```
class JMM122 {
 public static void main (String[] args) {
 for (int i = 0; i < 4; i++) {
 switch (i) {
 case 0: System.out.print("A");
 case 1: System.out.print("B");
 case 2: System.out.print("C");
 }}}}
```

What is the result of attempting to compile and run the program?

- a. Prints: ABC
  - b. Prints: ABCC
  - c. Prints: CBA
  - d. Prints: ABCBCC
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- 

### Question 7

```
class JMM123 {
 public static void main (String args[]) {
 int i = 0, j = 0, k = 0;
 label1:
 for (int h = 0; h < 6; h++) {
 label2:
 do { i++; k = h + i + j;
 switch (k) {
```

```
 default: break label1;
 case 1: continue label2;
 case 2: break;
 case 3: break label2;
 case 4: continue label2;
 case 5: continue label1;
 }
} while (++j<5);
}
System.out.println(h + "," + i + "," + j);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0
  - b. Prints: 0,2,1
  - c. Prints: 1,3,1
  - d. Prints: 2,4,1
  - e. Run-time error
  - f. **Compile-time error**
  - g. None of the above
- 

### Question 8

```
class JMM103 {
 public static void main(String args[]) {
 byte b = -1;
 switch(b) {
 case 0: System.out.print("zero "); break;
 case 100: System.out.print("100 "); break;
 case 1000: System.out.print("1000 "); break;
 default: System.out.print("Default ");
 }
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: zero
  - b. Prints: 100
  - c. Prints: 1000
  - d. Prints: Default
  - e. Run-time error
  - f. Compile-time error**
  - g. None of the above
- 

### Question 9

```
class JMM104 {
 public static void main (String args[]) {
 char c = 'b';
 switch(c) {
 case 'a': System.out.print("1");
 case 'b': System.out.print("2");
 case 'c': System.out.print("3");
 default: System.out.print("4");
 }}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 3
  - b. Prints: 34
  - c. Prints: 234**
  - d. Prints: 1234
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
-

### Question 10

```
class JMM105 {
 public static void main(String args[]) {
 int x = 6; int success = 0;
 do {
 switch(x) {
 case 0: System.out.print("0"); x += 5; break;
 case 1: System.out.print("1"); x += 3; break;
 case 2: System.out.print("2"); x += 1; break;
 case 3: System.out.print("3"); success++; break;
 case 4: System.out.print("4"); x -= 1; break;
 case 5: System.out.print("5"); x -= 4; break;
 case 6: System.out.print("6"); x -= 5; break;
 }
 } while ((x != 3) || (success < 2));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 60514233
  - b. Prints: 6152433
  - c. Prints: 61433
  - d. Prints: 6143
  - e. Run-time error
  - f. Compile-time error
- 

### Question 11

```
class JMM106 {
 public static void main(String args[]) {
 int x = -5; int success = 0;
 do {
 switch(x) {
 case 0: System.out.print("0"); x += 5; break;
 }
 } while ((x != 3) || (success < 2));
 }
}
```

```

 case 1: System.out.print("1"); x += 3; break;
 case 2: System.out.print("2"); x += 1; break;
 case 3: System.out.print("3"); success++; break;
 case 4: System.out.print("4"); x -= 1; break;
 case 5: System.out.print("5"); x -= 4; break;
 case 6: System.out.print("6"); x -= 5; break;
 default: x += x < 0 ? 2 : -2;
 }
} while ((x != 3) || (success < 2));
})

```

What is the result of attempting to compile and run the program?

- a. Prints: 60514233
  - b. Prints: 1433
  - c. Prints: 61433
  - d. Prints: 051433
  - e. Run-time error
  - f. Compile-time error
- 

## Question 12

```

class JMM124 {
 public static void main(String args[]) {
 int k;
 for (int i=0, j=0; i<2; i++,j++) {System.out.print(i); } // 1
 for (int i=0, k=0; i<2; i++,k++) {System.out.print(i); } // 2
 for (int i=0, int j=0; i<2; i++,j++) {System.out.print(i); } // 3
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 012345
- b. Prints: 010101

- c. Compile-time error at line 1
  - d. **Compile-time error at line 2**
  - e. Compile-time error at line 3
  - f. Run-time error
- 

### Question 13

```
class JMM125 {
 static int i;
 public static void main(String args[]) {
 for (i=1; i<3; i++) {System.out.print(i);} // 1
 for (int i=1; i<3; i++) {System.out.print(i);} // 2
 int i; // 3
 for (i=0; i<2; i++) {System.out.print(i);} // 4
 System.out.print(JMM125.i);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 1212010
  - b. **Prints: 1212013**
  - c. Compile-time error at line 1
  - d. Compile-time error at line 2
  - e. Compile-time error at line 4
  - f. Run-time error
  - g. None of the above
- 

### Question 14

```
class JMM126 {
 static int i;
 public static void main(String args[]) {
 for (i=1; i<3; i++) {System.out.print(i);} // 1
```

```
for (int i=1; i<3; i++) {System.out.print(i);} // 2
int i; // 3
for (int i=0; i<2; i++) {System.out.print(i);} // 4
System.out.print(JMM126.i),
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1212010
  - b. Prints: 1212013
  - c. Compile-time error at line 1
  - d. Compile-time error at line 2
  - e. **Compile-time error at line 4**
  - f. Run-time error
  - g. None of the above
- 

### Question 15

```
class JMM101 {
 public static void main(String[] args) {
 int i = 0;
 while (i++ < args.length) {
 System.out.print(args[i]);
 }
 }
}
java JMM101 A B C D E F
```

What is the result of attempting to compile and run the program using the specified command line?

- a. Prints: JMM101ABCDEF
- b. Prints: ABCDEF
- c. Compile-time error
- d. **Run-time error**

- e. None of the above
- 

### Question 16

```
class JMM110 {
 public static void main (String[] args) {
 int j = 0;
 do for (int i = 0; i++ < 2;) {
 System.out.print(i);
 } while (j++ < 2);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0001
  - b. Prints: 012
  - c. Prints: 012012
  - d. Prints: 012345
  - e. Prints: 001122
  - f. Prints: 1112
  - g. Prints: 111222
  - h. Prints: 121212
  - i. Run-time error
  - j. Compile-time error
  - k. None of the above
- 

### Question 17

```
class JMM111 {
 public static void main (String[] args) {
 int j = 0;
 for (int i = 0; i < 2; i++) do
 }
```

```
System.out.print(i);
while (j++ < 2);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0001
  - b. Prints: 012
  - c. Prints: 012012
  - d. Prints: 012345
  - e. Prints: 001122
  - f. Prints: 1112
  - g. Prints: 111222
  - h. Prints: 121212
  - i. Run-time error
  - j. Compile-time error
  - k. None of the above
- 

### Question 18

```
class JMM112 {
 public static void main (String[] args) {
 int j = 0;
 for (int i = 0; i++ < 2; do
 System.out.print(i);
 while (j++ < 2);
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0001
- b. Prints: 012
- c. Prints: 012012

- d. Prints: 012345
  - e. Prints: 001122
  - f. Prints: 1112
  - g. Prints: 111222
  - h. Prints: 121212
  - i. Run-time error
  - j. Compile-time error
  - k. None of the above
- 

### Question 19

```
class JMM113 {
 public static void main (String[] args) {
 int i = 0, j = 0, k = 0;
 do while (i++ < 3)
 System.out.print(k++);
 while (j++ < 3);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0001
- b. Prints: 012
- c. Prints: 012012
- d. Prints: 012345
- e. Prints: 001122
- f. Prints: 1112
- g. Prints: 111222
- h. Prints: 121212
- i. Run-time error
- j. Compile-time error
- k. None of the above

---

### Question 20

```
class JMM114 {
 public static void main (String[] args) {
 int i = 0, j = 0;
 while (i++ < 3) do
 System.out.print(j);
 while (j++ < 3);
 }}
What is the result of attempting to compile and run the program?
a. Prints: 0001
b. Prints: 001122
c. Prints: 012012
d. Prints: 012345
e. Prints: 1112
f. Prints: 111222
g. Prints: 121212
h. Run-time error
i. Compile-time error
j. None of the above
```

---

### Question 21

```
class JMM115 {
 static int m1(String s, int i) {
 System.out.print(s + i);
 return i;
 }
 public static void main (String[] args) {
 int i = 0, j = 0, k = 0;
```

```
do while (m1("i", ++i) < 2)
 System.out.print("k" + ++k);
while (m1("j", ++j) < 2);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: i1k1i2k2j1i3j2
  - b. Prints: i1k1i2k2j1i1k1i2k2j2
  - c. Prints: i1k1i2j1i3j2
  - d. Run-time error
  - e. Compile-time error
  - f. None of the above
- 

## Question 22

```
class JMM116 {
 static int m1(String s, int i) {
 System.out.print(s + i);
 return i;
 }
 public static void main (String[] args) {
 int j = 0;
 for (int i = m1("A",0); m1("B",i) < 2; m1("C",++i)) {
 m1("J",++j);
 }
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A0B0C1J1B1C2J2B2
- b. Prints: A0B0J1C1B1J2C2B2
- c. Prints: A0B0J1C1A1B1J2C2A2B2
- d. Run-time error

- e. Compile-time error
  - f. None of the above
- 

### Question 23

```
class JMM117 {
 public static void main (String[] args) {
 int i = 0, j = 9;
 do {
 i++;
 if (j-- < i++) {break;}
 } while (i < 5);
 System.out.print(i + "," + j);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 5,4
  - b. Prints: 6,3
  - c. Prints: 6,6
  - d. Prints: 7,2
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- 

### Question 24

```
class JMM118 {
 public static void main (String[] args) {
 int i = 0, j = 9;
 while (i++ <= j--) {i++; if (j < 5) break;}
 System.out.print(i + "," + j);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 4,7
  - b. Prints: 6,6
  - c. Prints: 7,2
  - d. Prints: 8,5
  - e. Prints: 9,4
  - f. Run-time error
  - g. Compile-time error
  - h. None of the above
- 

### Question 25

```
class JMM119 {
 public static void main (String[] args) {
 int i = 0, j = 9;
 do {
 if (j < 4) {break;} else if (j-- < 7) {continue;}
 i++;
 } while (i++ < 7);
 System.out.print(i + "," + j);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 4,7
- b. Prints: 6,6
- c. Prints: 6,5
- d. Prints: 6,4
- e. Prints: 7,5
- f. Prints: 8,4
- g. Run-time error

- h. Compile-time error
  - i. None of the above
- 

### Question 26

```
class JMM120 {
 public static void main (String args[]) {
 int i = 0, j = 0, k = 0;
 label1:
 for (;;) { i++;
 label2:
 do {
 k = i + j;
 switch (k) {
 case 0: continue label2;
 case 1: continue label1;
 case 2: break;
 case 3: break label2;
 case 4: break label1;
 default: break label1;
 }
 } while (++j<5);
 }
 System.out.println(i + "," + j);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 2,1
- b. Prints: 2,2
- c. Prints: 3,1
- d. Prints: 3,2
- e. Prints: 3,3
- f. Run-time error

- g. Compile-time error
  - h. None of the above
- 

### Question 27

```
class JMM121 {
 public static void main (String args[]) {
 int h = 0, i = 0, j = 0, k = 0;
 label1:
 for (;;) { h++;
 label2:
 do { i++; k = h + i + j;
 switch (k) {
 default: break label1;
 case 1: continue label1;
 case 2: break;
 case 3: break label2;
 case 4: continue label2;
 case 5: continue label1;
 }
 } while (++j < 5);
 }
 System.out.println(h + "," + i + "," + j);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,2,3
- b. Prints: 1,3,2
- c. Prints: 2,2,2
- d. Prints: 2,4,1
- e. Prints: 2,4,2
- f. Run-time error
- g. Compile-time error

h. None of the above

1.c .Prints: v w x x y z z .Cases one and three have no break statement, so the next case is also executed and x and z are printed twice.

2.e .Compile-time error. .The keyword, case, is missing from the case labels.

3.c .Prints: C .The boolean type variable, b, is initialized to true. The boolean expression of the first if statement, b = false, is a simple assignment expression that sets b to false. Always look carefully at the boolean expression of an if statement to verify that the expected equality operator (==) has not been replaced by the simple assignment operator (=).

4.d .Prints: D .The expression, b = false, appears to be testing the value of b, but it is really setting the value of b. Always look carefully at the boolean expression of an if statement to verify that the expected equality operator (==) has not been replaced by the simple assignment operator (=).

5.c .Prints: C .The first if statement initializes the value of b. The expression, b = false, appears to be testing the value of b, but it is really setting the value of b. Always look carefully at the boolean expression of an if statement to verify that the expected equality operator (==) has not been replaced by the simple assignment operator (=).

6.d .Prints: ABCBCC .The switch statement does not contain any break statements. The first case falls through to the second. The second case falls through to the third. There is no default switch label, so nothing is printed when variable i is 3.

7.f .Compile-time error .The loop variable, h, is local to the for statement. The print statement causes a compile-time error, because it refers to the out-of-scope local variable, h. Always check for variables that are out-of-scope!

8.f .Compile-time error .The case constant expression, 1000, produces a compile-time error, because it exceeds the range of the switch expression type, byte. The type of a switch expression can be byte, short, int or char. A constant expression is associated with each case label. Each case constant expression must be assignable to the type of the switch expression. In this case, the switch expression, b, is of type byte; so the maximum positive value of each case constant expression is limited to the maximum byte value, 127.

9.c .Prints: 234 .No break statements appear inside the switch statement, so more than one case is processed.

10.c .Prints: 61433 .On the first pass through the loop, the value of x is 6, so 5 is subtracted from x. On the second pass, the value of x is 1, so 3 is added to x. On the third pass, the value of x is 4, so 1 is subtracted from x. On the fourth pass, the value of x is 3, so the variable, success, is incremented from zero to one. On the final pass, the value of x is 3 and the variable, success, is incremented to the value, 2. The boolean expression of the do loop is now false, so control passes out of the loop.

11.b .Prints: 1433 .On the first pass through the loop, the switch expression, x, has the value, -5. None of the case constants are matched, so the statement following the default label is executed causing the value of x to be set to -3. On the second pass, the default case of the switch statement is executed again, and two is again added to the value of x. The new value is -1. On the third pass, the value of x is again incremented, and the new value is +1. After that, the value of x is printed on each pass through the loop. For the last two passes, the value of x is 3.

12.d e .Compile-time error at line 2 Compile-time error at line 3 .A compile-time error occurs at line 2 as a result of the attempt to shadow the method local variable, k, within the for-loop. A variable declared within the for-loop can not shadow a local variable of the enclosing method. At line 3, the declaration of variable j causes a compile-time error. The initialization part of a for statement may use a single local variable declaration statement to declare more than one local variable. Each of the new variables is declared in a comma separated list of variable declarators. In this program, the local variables should have been declared as follows: "int i=0, j=0;". Instead, the type of variable j has been incorrectly added to the declarator: "int i=0, int j=0;". The result is a compile-time error.

13.b .Prints: 1212013 .A method local variable, i, is declared at line 3. At line 4, the initialization part of the for statement initializes the method local variable to the value zero. The for statement does not declare a new variable, i, so the method local variable is not shadowed within the loop. Suppose a local variable, v, is declared within a statement or block. Variable, v, can shadow a class variable or an instance variable of the same name, but a compile-time error is generated if v

shadows a method local variable. Please note that a compile-time error is not generated if a local variable is shadowed within the declaration of a class that is nested within the scope of the local variable.

14.e .Compile-time error at line 4 .A method local variable, i, is declared at line 3. At line 4, the initialization part of the for statement attempts to shadow the method local variable with a new variable that is intended to be local to the for statement. The result is a compile-time error. At line 2, a new variable, i, is declared within the for statement. That variable shadows the class variable, but it does not shadow the method local variable declared at line 3. The scope of a method local variable begins with its own initializer--if present--and extends to the end of the method body. At line 2, the method local variable is not in scope, so shadowing does not occur. Suppose a local variable, v, is declared within a statement or block. Variable, v, can shadow a class variable or an instance variable of the same name, but a compile-time error is generated if v shadows a method local variable. Please note that a compile-time error is not generated if a local variable is shadowed within the declaration of a class that is nested within the scope of the local variable.

15.d .Run-time error .The index, i, is incremented before the array access expression is evaluated, so the first element accessed is at index one instead of zero. The arguments, BCDEF, are printed before an ArrayIndexOutOfBoundsException is thrown when the attempt is made to access an element beyond the end of the array.

16.h .Prints: 121212 .This trick question has a for loop nested inside of a do loop. For each iteration, the values of i and j are as follows: (1,0)(2,0)(1,1)(2,1)(1,2)(2,2).

17.a .Prints: 0001 .This trick question has a do-loop nested inside of a for-loop. For each iteration, the values of i and j are as follows: (0,0)(0,1)(0,2)(1,3).

18.f .Prints: 1112 .This trick question has a do-loop nested inside of a for-loop. For each iteration, the values of i and j are as follows: (1,0)(1,1)(1,2)(2,3).

19.b .Prints: 012 .This trick question has a while-loop nested inside of a do-loop. For each iteration, the values of i, j and k are as follows: (1,0,0)(2,0,1)(3,0,2).

20.d .Prints: 012345 .This trick question has a do-loop nested inside of a while-loop. For each iteration, the values of i and j are as follows: (1,0)(1,1)(1,2)(1,3)(2,4)(3,5).

21.c .Prints: i1k1i2j1i3j2 .A while loop is nested inside of a do loop. The while loop iterates once during the first iteration of the do loop. The body of the while loop does not execute during the final iteration of the do loop.

22.b .Prints: A0B0J1C1B1J2C2B2 .The initialization statement, labeled A, is processed first. The boolean expression, labeled B, is processed before each iteration of the loop. The body of the loop is processed next followed by the update expression, labeled C.

23.c .Prints: 6,6 .Suppose the print statement, System.out.print("(" + i + "," + j + ")");, is inserted at the top of the loop. The output of the program would be as follows: (0,9)(2,8)(4,7)6,6. The variable, i, is incremented twice with each pass through the loop. The variable, j, is decremented once with each pass. The loop terminates when i reaches six.

24.e .Prints: 9,4 .Suppose the print statement, System.out.print("(" + i + "," + j + ")");, is inserted at the top of the loop. The output of the program would be as follows: (1,8)(3,7)(5,6)(7,5)9,4. The variable, i, is incremented twice with each pass through the loop. The variable, j, is decremented once with each pass. The boolean expression of the while loop, i++ <= j--, is false when i reaches 8 and j reaches 5. The value of i is subsequently incremented and j is decremented yielding 9 and 4 respectively. Those values are printed.

25.f .Prints: 8,4 .Suppose the print statement, System.out.print("(" + i + "," + j + ")");, is inserted at the top of the loop. The output of the program would be as follows: (0,9)(2,8)(4,7)(6,6)(7,5)8,4. The initial value of j is 9. With each pass through the loop, the expression, j-- < 7, decrements j. The initial value of variable i is 0. With each pass through the loop, the expression, i++ < 7, increments variable i. Inside the body of the loop, i is also incremented as long as j is greater than or equal to 7. When j is less than seven, variable i is no longer incremented inside the body of the loop, but it is still incremented by the expression, i++ < 7.

26.c .Prints: 3,1 .Suppose the print statement, System.out.print("(+i+","+j+","+k+")");, is inserted before the switch statement. The output of the program would be as follows: (1,0,1)(2,0,2)(2,1,3)(3,1,4)3,1. On the first iteration, case 1 is processed. The "continue label1;" statement causes control to go directly to the top of the for-loop following the label, label1. The boolean expression of the do-loop is not processed. On the second iteration, case 2 is processed. The break statement

causes the switch statement to complete, and control passes to the boolean expression of the do-loop. On the third iteration, case 3 is processed. The break with label statement, "break label2;", completes abruptly; and the do-loop completes abruptly without processing the loop expression,  $++j < 5$ , so j is not incremented. On the fourth iteration, case 4 is processed. The break with label statement, "break label1;", completes abruptly, and the for-loop completes abruptly.

27.b .Prints: 1,3,2 .Suppose the print statement, `System.out.print("(+h+","+i+","+j+","+k+)");`, is inserted before the switch statement. The output of the program would be as follows: (1,1,0,2)(1,2,1,4)(1,3,2,6)1,3,2. Three iterations of the loop are executed. On the first iteration, the switch expression, k, matches the case constant, 2. The subsequent break statement is executed and the switch statement completes normally. On the second iteration, the switch expression matches the case constant, 4; and the subsequent continue statement causes control to pass to the loop-continuation point of the do statement where the expression,  $++j < 5$ , is evaluated and found to be true. On the final iteration, the switch expression does not match any case constant; so the break statement following the default label is processed.

## Exception Handling

### Question 1

```
class A {
 public static void main (String[] args) {
 Error error = new Error();
 Exception exception = new Exception();
 System.out.print(exception instanceof Throwable) + ",";
 System.out.print(error instanceof Throwable);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compile-time error
- f. Run-time error
- g. None of the above

---

### Question 2

```
class A {A() throws Exception {}} // 1
class B extends A {B() throws Exception {}} // 2
class C extends A {C() {}} // 3
```

Which of the following statements are true?

- a. class A extends Object.
  - b. Compile-time error at 1.
  - c. Compile-time error at 2.
  - d. **Compile-time error at 3.**
- 

### Question 3

```
class A {
 public static void main (String[] args) {
 Object error = new Error();
 Object runtimeException = new RuntimeException();
 System.out.print((error instanceof Exception) + ",");
 System.out.print(runtimeException instanceof Exception);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
  - b. **Prints: false,true**
  - c. Prints: true,false
  - d. Prints: true,true
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 4

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
 public static void main(String args[]) {
 int a,b,c,d,f,g,x;
 a = b = c = d = f = g = 0;
 x = 1;
 try {
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 } a++; }
 catch (Level2Exception e) {b++;}
 finally {c++;}
 }
 catch (Level1Exception e) { d++;}
 catch (Exception e) {f++;}
 finally {g++;}
 System.out.print(a+","+b+","+c+","+d+","+f+","+g);
 }}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,0,0
  - b. Prints: 0,0,1,1,0,1
  - c. Prints: 0,1,1,1,0,1
  - d. Prints: 1,0,1,1,0,1
  - e. Prints: 1,1,1,1,0,1
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
-

## Question 5

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
 public static void main(String args[]) {
 int a,b,c,d,f,g,x;
 a = b = c = d = f = g = 0;
 x = 2;
 try {
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 } a++; }
 catch (Level2Exception e) {b++;}
 finally {c++;}
 }
 catch (Level1Exception e) { d++;}
 catch (Exception e) {f++;}
 finally {g++;}
 System.out.print(a+","+b+","+c+","+d+","+f+","+g);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,1,0,0,1
- b. Prints: 0,1,0,0,0,0
- c. Prints: 0,1,1,0,0,1
- d. Prints: 0,1,0,0,0,1
- e. Prints: 1,1,1,0,0,1
- f. Compile-time error
- g. Run-time error

- h. None of the above
- 

### Question 6

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
 public static void main(String args[]) {
 int a,b,c,d,f,g,x;
 a = b = c = d = f = g = 0;
 x = 3;
 try {
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 } a++; }
 catch (Level2Exception e) {b++;}
 finally {c++;}
 }
 catch (Level1Exception e) { d++;}
 catch (Exception e) {f++;}
 finally {g++;}
 System.out.print(a+","+b+","+c+","+d+","+f+","+g);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1,1,0,0,1
- b. Prints: 0,1,1,0,0,1
- c. Prints: 0,1,0,0,0,0
- d. Prints: 0,1,0,0,0,1

- e. Prints: 0,0,1,0,0,1
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### Question 7

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
 public static void main(String args[]) {
 int a,b,c,d,f,g,x;
 a = b = c = d = f = g = 0;
 x = 4;
 try {
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 case 4: throw new Exception();
 } a++; }
 catch (Level2Exception e) {b++;}
 finally{c++;}
 }
 catch (Level1Exception e) { d++;}
 catch (Exception e) {f++;}
 finally {g++;}
 System.out.print(a+","+b+","+c+","+d+","+f+","+g);
 }
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,0,0,1

- b. Prints: 0,0,0,0,1,0
  - c. Prints: 0,0,1,0,0,1
  - d. Prints: 0,0,1,0,1,1
  - e. Prints: 0,1,1,1,1,1
  - f. Prints: 1,1,1,1,1,1
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

### Question 8

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
 public static void main(String args[]) {
 int a,b,c,d,f,g,x;
 a = b = c = d = f = g = 0;
 x = 5;
 try {
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 case 4: throw new Exception();
 } a++; }
 catch (Level2Exception e) {b++;}
 finally {c++;}
 }
 catch (Level1Exception e) { d++;}
 catch (Exception e) {f++;}
 finally {g++;}
 System.out.print(a+","+b+","+c+","+d+","+f+","+g);
 }
}
```

}}

What is the result of attempting to compile and run the program?

- a. Prints: 1,0,0,0,0,0
  - b. Prints: 1,0,1,0,0,1
  - c. Prints: 0,0,1,0,0,1
  - d. Prints: 1,1,1,1,1,1
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 9

```
class ColorException extends Exception {}
class WhiteException extends ColorException {}
class White {
 void m1() throws ColorException {throw new WhiteException();}
 void m2() throws WhiteException {}
 public static void main (String[] args) {
 White white = new White();
 int a,b,d,f; a = b = d = f = 0;
 try {white.m1(); a++;} catch (ColorException e) {b++;}
 try {white.m2(); d++;} catch (WhiteException e) {f++;}
 System.out.print(a+","+b+","+d+","+f);
 }}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
- b. Prints: 1,1,0,0
- c. Prints: 0,1,1,0
- d. Prints: 1,1,1,0

- e. Prints: 1,1,1,1
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### Question 10

```
class ColorException extends Exception {}
class WhiteException extends ColorException {}
class White {
 void m1() throws ColorException {throw new ColorException();}
 void m2() throws WhiteException {throw new ColorException();}
 public static void main (String[] args) {
 White white = new White();
 int a,b,d,f; a = b = d = f = 0;
 try {white.m1(); a++;} catch (ColorException e) {b++;}
 try {white.m2(); d++;} catch (WhiteException e) {f++;}
 System.out.print(a+","+b+","+d+","+f);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
  - b. Prints: 1,1,0,1
  - c. Prints: 0,1,0,1
  - d. Prints: 0,1,1,1
  - e. Prints: 1,1,1,1
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### Question 11

```

class ColorException extends Exception {}
class WhiteException extends ColorException {}
class White {
 void m1() throws ColorException {throw new ColorException();}
 void m2() throws WhiteException {throw new WhiteException();}
 public static void main (String[] args) {
 White white = new White();
 int a,b,d,f; a = b = d = f = 0;
 try {white.m1(); a++;} catch (WhiteException e) {b++;}
 try {white.m2(); d++;} catch (WhiteException e) {f++;}
 System.out.print(a+","+b+","+d+","+f);
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
  - b. Prints: 1,1,0,1
  - c. Prints: 0,1,0,1
  - d. Prints: 0,1,1,1
  - e. Prints: 1,1,1,1
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### **Question 12**

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Brown {
 public static void main(String args[]) {
 int a, b, c, d, f; a = b = c = d = f = 0;
 int x = 1;
 try {

```

```

switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
} a++;
catch (Level3Exception e) {b++;}
catch (Level2Exception e) {c++;}
catch (Level1Exception e) {d++;}
finally {f++;}
System.out.print(a+","+b+","+c+","+d+","+f);
}}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,1
  - b. Prints: 0,0,1,1,1
  - c. Prints: 0,1,1,1,1
  - d. Prints: 1,1,1,1,1
  - e. Prints: 0,0,1,0,1
  - f. Prints: 0,1,0,0,1
  - g. Prints: 1,0,0,0,1
  - h. Compile-time error
  - i. Run-time error
  - j. None of the above
- 

### Question 13

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Brown {
 public static void main(String args[]) {
 int a, b, c, d, f; a = b = c = d = f = 0;
 int x = 2;
}

```

```

try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 } a++;
}
catch (Level3Exception e) {b++;}
catch (Level2Exception e) {c++;}
catch (Level1Exception e) {d++;}
finally {f++;}
System.out.print(a+","+b+","+c+","+d+","+f);
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,1
  - b. Prints: 0,0,1,1,1
  - c. Prints: 0,1,1,1,1
  - d. Prints: 1,1,1,1,1
  - e. Prints: 0,0,1,0,1
  - f. Prints: 0,1,0,0,1
  - g. Prints: 1,0,0,0,1
  - h. Compile-time error
  - i. Run-time error
  - j. None of the above
- 

#### **Question 14**

```

class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Brown {
 public static void main(String args[]) {
 int a, b, c, d, f; a = b = c = d = f = 0;
 }
}

```

```

int x = 4;
try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 case 3: throw new Level3Exception();
 } a++;
}
catch (Level3Exception e) {b++;}
catch (Level2Exception e) {c++;}
catch (Level1Exception e) {d++;}
finally {f++;}
System.out.print(a+","+b+","+c+","+d+","+f);
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,1,1
  - b. Prints: 0,0,1,1,1
  - c. Prints: 0,1,1,1,1
  - d. Prints: 1,1,1,1,1
  - e. Prints: 0,0,1,0,1
  - f. Prints: 0,1,0,0,1
  - g. Prints: 1,0,0,0,1
  - h. Compile-time error
  - i. Run-time error
  - j. None of the above
- 

### Question 15

```

class ColorException extends Exception {}
class WhiteException extends ColorException {}
abstract class Color {
 abstract void m1() throws ColorException;
}

```

```

class White extends Color {
 void m1() throws WhiteException {throw new WhiteException();}
 public static void main (String[] args) {
 White white = new White();
 int a,b,c; a = b = c = 0;
 try {white.m1(); a++;}
 catch (WhiteException e) {b++;}
 finally {c++;}
 System.out.print(a+","+b+","+c);
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0
  - b. Prints: 0,0,1
  - c. Prints: 0,1,0
  - d. Prints: 0,1,1
  - e. Prints: 1,0,0
  - f. Prints: 1,0,1
  - g. Prints: 1,1,0
  - h. Prints: 1,1,1
  - i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

### Question 16

```

class RedException extends Exception {}
class BlueException extends Exception {}
class White {
 void m1() throws RedException {throw new RedException();}
 public static void main (String[] args) {
 White white = new White();
 int a,b,c,d; a = b = c = d = 0;
 }
}

```

```
try {white.m1(); a++;}
 catch (RedException e) {b++;}
 catch (BlueException e) {c++;}
 finally {d++;}
 System.out.print(a+","+b+","+c+","+d);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1,0,0
  - b. Prints: 1,1,0,1
  - c. Prints: 0,1,0,1
  - d. Prints: 0,1,1,1
  - e. Prints: 1,1,1,1
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### Question 17

```
class Level1Exception extends Exception {}
class Level2Exception extends Level1Exception {}
class Level3Exception extends Level2Exception {}
class Purple {
 public static void main(String args[]) {
 int a,b,c,d,f,g,x;
 a = b = c = d = f = g = 0;
 x = 1;
 try {
 throw new Level1Exception();
 try {
 switch (x) {
 case 1: throw new Level1Exception();
 case 2: throw new Level2Exception();
 }
 }
 }
}
```

```

 case 3: throw new Level3Exception();
 } a++;
}
catch (Level2Exception e) {b++;}
finally {c++;}
}
catch (Level1Exception e) { d++;}
catch (Exception e) {f++;}
finally {g++;}
System.out.print(a+","+b+","+c+","+d+","+f+","+g);
}}

```

What is the result of attempting to compile and run the program?

- a. Prints: 1,1,1,0,0,1
- b. Prints: 0,1,1,0,0,1
- c. Prints: 0,1,0,0,0,0
- d. Prints: 0,1,0,0,0,1
- e. Prints: 0,0,1,0,0,1
- f. Compile-time error
- g. Run-time error
- h. None of the above

1.d .Prints: true,true .Both Error and Exception are subclasses of Throwable.

2.a d .class A extends Object. Compile-time error at 3. .The constructors for class B and class C both invoke the constructor for A. The constructor for class A declares Exception in the throws clause. Since the constructors for B and C invoke the constructor for A, it is necessary to declare Exception in the throws clauses of B and C. A compile-time error is generated at marker 3, because the constructor does not declare Exception in the throws clause.

3.b .Prints: false,true .Error is a direct subclass of Throwable. RuntimeException is a direct subclass of Exception.

4.b .Prints: 0,0,1,1,0,1 .The nested catch clause is able to catch a Level2Exception or any subclass of it. The switch statement throws a Level1Exception that can not be caught by the nested catch clause; so the nested finally block is executed as control passes to the first of the two outer catch clauses. The outer finally block is executed as control passes out of the try statement.

5.c .Prints: 0,1,1,0,0,1 .The nested catch block is able to catch a Level2Exception or any subclass of it causing b to be incremented. Both of the finally blocks are then executed.

6.b .Prints: 0,1,1,0,0,1 .The nested catch block is able to catch a Level2Exception or any subclass of it causing b to be incremented. Both of the finally blocks are then executed.

7.d .Prints: 0,0,1,0,1,1 .The nested catch clause is able to catch a Level2Exception or any subclass of it. The switch statement throws an Exception that can not be caught by the nested catch clause; so the nested finally block is executed as control passes to the second of the two outer catch clauses. The outer finally block is executed as control passes out of the try statement.

8.b .Prints: 1,0,1,0,0,1 .The switch statement does not throw an exception; so the switch completes normally. The subsequent statement increments the variable, a; and the try block completes normally. Both of the finally blocks are then executed.

9.c .Prints: 0,1,1,0 .The first try block contains two statements. The first invokes method m1, and the subsequent statement contains a post increment expression with the variable, a, as the operand. Method m1 throws a WhiteException exception, so variable a is not incremented as control passes to the catch block where b is incremented. The throws clause of m1 declares a ColorException, so the body may throw a ColorException or any subclass of ColorException. The second try block also contains two statements. The first invokes method m2, and the subsequent statement contains a post increment expression with the variable, d, as the operand. Method m2 does not throw an exception, so d is incremented, and the try block completes normally. Although the throws clause of m2 declares a WhiteException, there is no requirement to throw any exception.

10.f .Compile-time error .The throws clause of White.m2 declares a WhiteException, so the body of m2 may throw a WhiteException or any subclass of WhiteException. Instead, the body of m2 throws a superclass of WhiteException. The result is a compile-time error.

11.f .Compile-time error .The throws clause of White.m1 declares a ColorException, but the catch clause in the main method catches only a subclass of ColorException. The result is a compile-time error.

12.a .Prints: 0,0,0,1,1 .The first catch clause has a parameter e of type Level3Exception, so the first catch clause is able to catch any exception type that is assignable to type Level3Exception. Since Level2Exception is the superclass of Level3Exception, an instance of Level2Exception is not assignable to a catch clause parameter of type Level3Exception. Similarly, Level1Exception is also a superclass of Level3Exception, so an instance of Level1Exception is not assignable to a catch clause parameter of type Level3Exception. The only exception type that can be caught by the first catch clause is a Level3Exception. The second catch clause has a parameter e of type Level2Exception, so the second catch clause is able to catch a Level2Exception. The Level1Exception is the superclass of Level2Exception. An instance of Level1Exception is not assignable to a catch clause parameter of type Level2Exception, so the second catch clause can not catch a Level1Exception. Since a Level3Exception is a subclass of Level2Exception an exception of type Level3Exception is assignable to a catch clause parameter type Level2Exception. All exceptions of type Level3Exception will be caught by the first catch clause, so the second catch clause in this program will not have an opportunity to catch a Level3Exception. The third catch clause has a parameter e of type Level1Exception, so the third catch clause is able to catch a Level1Exception. The exceptions of type Level2Exception and Level3Exception are assignable to the catch clause parameter of the third catch clause, but the exceptions of those subclass types will be caught by the first two catch clauses. The switch statement throws a Level1Exception. The try block completes abruptly as control passes to the third catch block where d is incremented. The finally block is also executed, so f is incremented.

13.e .Prints: 0,0,1,0,1 .The first catch block is able to catch a Level3Exception or any subclass of Level3Exception. The second catch block is able to catch a Level2Exception or any subclass of Level2Exception. The switch statement throws a Level2Exception. The try block completes abruptly as control passes to the second catch block where c is incremented. The finally block is also executed, so f is incremented.

14.g .Prints: 1,0,0,0,1 .The switch statement does not throw an exception; so the switch completes normally. The subsequent statement increments the variable, a; and the try block completes normally. The finally block is also executed, so f is incremented.

15.d .Prints: 0,1,1 .The try block contains two statements. The first invokes method m1, and the subsequent statement contains a post increment expression with the variable, a, as the operand. Method m1 throws a WhiteException exception, so variable a is not incremented as control passes to the catch block where b is incremented. Although Color.m1 declares a ColorException in the throws clause, a subclass of Color is free to declare only a subclass of ColorException in the throws clause of the overriding method.

16.f .Compile-time error .A compile-time error is generated, because the second catch clause attempts to catch an exception that is never thrown in the try block.

17.f .Compile-time error .A throw statement is the first statement in the outer try block. A throw statement appearing in a try block causes control to pass out of the block. Consequently, statements can not be reached if they appear in the block after the throw statement. The switch statement that appears after the throw statement is unreachable and results in a compile-time error.

## Assertions

### Question 1

Which of the following are command-line switches used to enable assertions in non-system classes?

- a. -ea
  - b. -ae
  - c. -aon
  - d. -aoff
  - e. -enableassertions
  - f. -assertionsenable
  - g. -assertionson
  - h. -assertionsoff
- 

### Question 2

Which of the following are command-line switches used to disable assertions in non-system classes?

- a. -da
- b. -ad
- c. -aon
- d. -aoff
- e. -disableassertions
- f. -assertionsdisable
- g. -assertionson
- h. -assertionsoff

---

### **Question 3**

Which of the following are command-line switches used to disable assertions in non-system classes?

- a. **-disableassertions**
  - b. -disableAssertions
  - c. -assertionsDisable
  - d. -assertionsOn
  - e. -assertionsOff
  - f. None of the above
- 

### **Question 4**

Which of the following are command-line switches used to enable assertions in system classes?

- a. -saon
  - b. -saoff
  - c. **-esa**
  - d. -sae
  - e. -systemassertionson
  - f. -systemassertionsoff
  - g. **-enablesystemassertions**
  - h. -systemassertionsenable
- 

### **Question 5**

Which of the following statements are true?

- a. By default assertions are disabled at run-time.

- b. Assertions can be selectively enabled for any named class.
  - c. If assertions are selectively enabled for a named class then assertions are automatically enabled for any subclass of the named class.
  - d. Assertions can be selectively enabled for any named package.
  - e. If assertions are selectively enabled for any named package then assertions are automatically enabled for the subpackages.
  - f. Assertions can be selectively enable for the unnamed package in the current working directory.
  - g. Assertions can be selectively enable for any unnamed package in any named directory.
- 

### Question 6

Which of the following is thrown to indicate that an assertion has failed?

- a. AssertionError
  - b. AssertException
  - c. Assertion**Error**
  - d. AssertionException
  - e. None of the above
- 

### Question 7

```
class A {
 void m1(int i) {
 int j = i % 3;
 switch (j) {
 case 0: System.out.print("0"); break;
 case 1: System.out.print("1"); break;
 default:
 assert j == 2;
 System.out.print(j);
 }
 }
 public static void main (String[] args) {
 A a = new A();
 for (int i=5; i >= -1; i--) {a.m1(i);}
 }
}
```

Which statements are true?

- a. With assertions enabled it prints 210210-1 followed by an AssertionError message.
  - b. With assertions enabled it prints 210210 followed by an AssertionError message.
  - c. With assertions enabled it prints only 210210.
  - d. With assertions enabled it prints nothing.
  - e. With assertions disabled it prints 210210-1
  - f. With assertions disabled it prints only 210210
  - g. Assertions should not be used within the default case of a switch statement.
- 

### Question 8

```
class B {
 private static int x1;
 public void setX(int x) {x1 = x;}
 public int getX() {return x1;}
 public static void main(String[] args) {
 int i1 = 0, j1 = 0;
 do {
 System.out.print(j1++);
 assert (i1 = j1 + x1) < 6;
 } while (i1 < 3);
 } }
```

Assuming that the setX method is never invoked, which statements are true?

- a. With assertions enabled it prints 012345 followed by an AssertionError message.
- b. With assertions enabled it prints 012.
- c. With assertions disabled it prints: 012345
- d. With assertions disabled it prints: 012
- e. With assertions disabled it attempts to print an infinite sequence of numbers.
- f. With assertions disabled the variable i1 is incremented with each pass through the loop.
- g. As a rule, the boolean expression of an assert statement should not be used to perform actions that are required for normal operation of the program.

---

### **Question 9**

Which statements are true?

- a. Assertions should not be used to validate arguments passed to public methods.
  - b. Assertions should not be used to validate arguments passed to nonpublic methods.
  - c. Assertions should not be used within the default case of a switch statement.
  - d. Assertions should not be used to perform processing that is required for the normal operation of the application.
- 

### **Question 10**

Which statements are true?

- a. Assertions should never be placed at any point in the code that should not be reached under normal circumstances.
  - b. The compiler will generate an error if an assert statement is "unreachable" as defined by the Java Language Specification.
  - c. A catch clause should not be used to catch an AssertionError.
  - d. AssertionError is a checked exception.
- 

### **Question 11**

```
class C {
 String m1(int i) {
 switch (i) {
 case 0: return "A";
 case 1: return "B";
 case 2: return "C";
 default: throw new AssertionError();
 }
 }
 public static void main(String[] args) {
 C c = new C();
 for (int i = 0; i < 4; i++) {
```

```
 System.out.print(c.m1(i));
 }}}
```

Which statements are true?

- a. With assertions enabled it prints ABC followed by an AssertionError message.
  - b. With assertions disabled it prints ABC followed by an AssertionError message.
  - c. Assertions should not be used within the default case of a switch statement.
  - d. In this code example an assert statement could not be used in place of the "throw" statement.
- 

### Question 12

```
class C {
 String m1(int i) {
 switch (i) {
 case 0: return "A";
 case 1: return "B";
 case 2: return "C";
 default:
 assert false;
 }
 return "E";
 }
 public static void main(String[] args) {
 C c = new C();
 for (int i = 0; i < 4; i++) {
 System.out.print(c.m1(i));
 }
 }
}
```

Which statements are true?

- a. With assertions enabled it prints ABC followed by an AssertionError message.
- b. With assertions enabled it prints ABCE followed by an AssertionError message.
- c. With assertions disabled it prints ABC

- d. With assertions disabled it prints ABCE
  - e. Assertions should not be used within the default case of a switch statement.
- 

### Question 13

```
class C {
 String m1(int i) {
 switch (i) {
 case 0: return "A";
 case 1: return "B";
 case 2: return "C";
 default:
 assert false;
 }
 }
 public static void main(String[] args) {
 C c = new C();
 for (int i = 0; i < 4; i++) {
 System.out.print(c.m1(i));
 }
 }
}
```

Which statements are true?

- a. With assertions enabled it prints ABC followed by an AssertionError message.
  - b. With assertions disabled it prints ABC followed by an AssertionError message.
  - c. Assertions should not be used within the default case of a switch statement.
  - d. In this code example a throw statement must be used in place of the assert statement.
  - e. Compile-time error
- 

### Question 14

```
class D {
```

```

private boolean b1, b2;
public void setB1(boolean b) {b1 = b;}
public void setB2(boolean b) {b2 = b;}
public void m1 () {
 if (!b2 & !b1) {System.out.print("A");}
 } else if (!b2 & b1) {System.out.print("B");}
 } else if (b2 & !b1) {System.out.print("C");}
 } else {assert false;}
}
public static void main (String[] args) {
 D d = new D();
 d.setB1(true); d.setB2(true);
 d.m1();
}
}

```

Which statements are true?

- a. With assertions enabled it prints an AssertionError message.
  - b. With assertions enabled it prints nothing.
  - c. With assertions disabled it prints an AssertionError message.
  - d. With assertions disabled it prints nothing.
  - e. An assertion should not be placed at any location that the programmer believes will never be reached under normal operating conditions.
  - f. The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program.
- 

### Question 15

```

class A {
 private void m1 (int i) {
 assert i < 10 : i; System.out.print(i);
 }
 public void m2 (int i) {
 assert i < 10 : i; System.out.print(i);
 }
 public static void main (String[] args) {
 A a = new A(); a.m1(11); a.m2(12);
 }
}

```

}}

Which statements are true?

- a. If assertions are enabled at run time it prints an error message.
  - b. With assertions enabled it prints nothing.
  - c. With assertions disabled it prints an error message.
  - d. With assertions disabled it prints 1112.
  - e. With assertions disabled it prints nothing.
  - f. The assert statements are being used to check a precondition--something that must be true when the method is invoked.
  - g. Method m1 is an example of an improper use of an assert statement: an assert statement should not be used for argument checking in a non-public method.
  - h. Method m2 is an example of an improper use of an assert statement: an assert statement should not be used for argument checking in a public method.
- 

### Question 16

```
class B {
 int a, b, c;
 private void setA(int i) {a = i;}
 private void setB(int i) {b = i;}
 private int m1 (int i) {
 c = a + b + i; assert c < 200 : c; return c;
 }
 public static void main (String[] args) {
 B b = new B(); b.setA(50); b.setB(100); b.m1(50);
 }}
```

Which statements are true?

- a. If assertions are not enabled at run time it prints an error message.
- b. If assertions are not enabled at run time it prints nothing.
- c. With assertions enabled it prints an error message.
- d. With assertions enabled it prints nothing.
- e. The assert statement is being used to check a postcondition--something that must be true when the method completes successfully.

---

### Question 17

```
class C {
 int a, b, c;
 public void setA(int i) {a = i; assert validateC() : c;}
 public void setB(int i) {b = i; assert validateC() : c;}
 private boolean validateC() {
 return c > a + 2 * b;
 }
 public int m1(int i) {
 c = a + b + i;
 assert validateC() : c;
 return c;
 }
 public C(int i) {
 c = i; assert validateC() : c;
 }
 public static void main(String[] args) {
 C c = new C(251); c.setA(50); c.setB(100);
 }}}
```

Which statements are true?

- a. If assertions are not enabled at run time it prints an error message.
  - b. If assertions are not enabled at run time it prints nothing.
  - c. With assertions enabled it prints an error message.
  - d. With assertions enabled it prints nothing.
  - e. The assert statement is being used to check a class invariant--something that must be true about each instance of the class.
  - f. The assert statements are being used to check a precondition--something that must be true when the method is invoked.
- 

### Question 18

```
class E {
```

```

private boolean b1, b2, b3;
public void setB1(boolean b) {b1 = b;}
public void setB2(boolean b) {b2 = b;}
public void setB3(boolean b) {b3 = b;}
public void m1 (int i) {
 b2 = i % 2 == 0;
 if (!b3 & !b2 & !b1) {System.out.print("A");}
 } else if (!b3 & !b2 & b1) {System.out.print("B");}
 } else if (!b3 & b2 & !b1) {System.out.print("C");}
 } else { // Only b3 is true.
 assert b3 & !b2 & !b1;
 }
 System.out.print(b1 + "," + b2 + "," + b3);
 b1 = b2 = b3 = false;
}
public static void main (String[] args) {
 E e = new E(); e.setB1(true); e.m1(2);
}

```

Which statements are true?

- With assertions enabled it prints an error message.
  - With assertions enabled it prints: true,true,false
  - With assertions disabled it prints an error message.
  - With assertions disabled it prints: true,true,false
  - With assertions disabled it prints nothing.
  - The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2 or b3, to be true.
  - The assert statement is being used to check a precondition--something that must be true when the method begins.
  - The assert statement is being used to check an internal invariant--something that the programmer assumes to be true at a particular point in the program.
- 

### Question 19

```

class F {
 private boolean b1, b2, b3;

```

```

public void setB1(boolean b) {b1 = b;}
public void setB2(boolean b) {b2 = b;}
public void setB3(boolean b) {b3 = b;}
public String m1 (int i) {
 b2 = i % 2 == 0;
 if (!b3 & !b2 & !b1) {return "None are true.";}
 } else if (!b3 & !b2 & b1) {return "Only b1 is true.";}
 } else if (!b3 & b2 & !b1) {return "Only b2 is true.";}
 } else if (b3 & !b2 & !b1) {return "Only b3 is true.";}
 } else {throw new AssertionError();}
}
public static void main (String[] args) {
 F f = new F();
 f.setB1(true);
 System.out.println(f.m1(5));
 System.out.println(f.m1(6));
}

```

Which statements are true?

- Prints "Only b1 is true" followed by an error message.
  - An assertion should not be placed at any location that the programmer believes will never be reached under normal operating conditions.
  - The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2, or b3, to be true.
  - The assert statement is being used to check a precondition--something that must be true when the method begins.
  - The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program.
  - The throw statement could be replaced by an assert statement.
- 

## Question 20

An assert statement can be used to check a control-flow invariant to verify which of the following?

- A particular assumption is true when the flow of control enters a method.
- The flow of control does not reach a particular point in the program.
- The normal flow of control has reached a particular point in the program.

- d. The normal flow of control has reached the end of a method.
- e. The default case of a switch statement is not reached.
- f. The else block of an if/else statement is not reached.

1.a e .-ea -enableassertions .Two command-line switches used to enable assertions in non-system classes are -ea and -enableassertions.

2.a e .-da -disableassertions .Two command-line switches used to disable assertions are -da and -disableassertions. Remember that all of the letters are lower case.

3.a .-disableassertions .Two command-line switches used to disable assertions in non-system classes are -da and -disableassertions. Remember that all of the letters are lower case.

4.c g .-esa -enablesystemassertions .Two command-line switches used to enable assertions in system classes are -esa and -enablesystemassertions. Remember that all of the letters are lower case.

5.a b d e f .By default assertions are disabled at run-time. Assertions can be selectively enabled for any named class. Assertions can be selectively enabled for any named package. If assertions are selectively enabled for any named package then assertions are automatically enabled for the subpackages. Assertions can be selectively enable for the unnamed package in the current working directory. .

6.c .AssertionError .An AssertionError is thrown to indicate that an assertion has failed. Don't be fooled by the name AssertionException.

7.b e .With assertions enabled it prints 210210 followed by an AssertionError message. With assertions disabled it prints 210210-1 .If, under normal operating circumstances, the default label of a switch statement should not be reached, then an assert statement can be placed after the default label to verify that an unexpected condition has not occurred.

8.b e g .With assertions enabled it prints 012. With assertions disabled it attempts to print an infinite sequence of numbers. As a rule, the boolean expression of an assert statement should not be used to perform actions that are required for normal operation of the program. .An assert statement should not be used as demonstrated in the program. The boolean expression of the do-loop depends on the value of the local variable i1. The value of i1 is set within the boolean expression of the assert statement. If assertions are disabled, then the boolean expression of the assert statement is not processed and the value of i1 is not updated with each iteration of the loop; so the loop runs indefinitely.

9.a d .Assertions should not be used to validate arguments passed to public methods. Assertions should not be used to perform processing that is required for the normal operation of the application. .Assertions may be enabled or disabled at run time. Since assertions are not always enabled, they should not be used to validate the parameters of public methods. Parameter checking is typically published in the API specification of a method and must be enforced even when assertions are not enabled. Rather than use an assertion, an appropriate runtime exception should be thrown such as IllegalArgumentException, IndexOutOfBoundsException, or NullPointerException. However, an assertion may be used to validate the parameters of a nonpublic method. Since assertions are not always enabled, an assertion should not be used to perform operations that are required for the normal operation of the program. For example, the boolean expression of an assertion should not be used to produce the side effect of incrementing a variable that controls a loop statement. If assertions are disabled then the loop is unlikely to function as intended. Section 14.20 of the Java Language Specification defines "unreachable" statements. If an assert statement is "unreachable" as defined by the JLS, then a compile-time error is generated. In contrast, a programmer may believe that some points in the code will not be reached as a result of design assumptions. For example, a programmer may believe that the default case of a switch statement will never be reached. An assertion can be placed in the default case to verify the behavior of the switch statement.

10.b c .The compiler will generate an error if an assert statement is "unreachable" as defined by the Java Language Specification. A catch clause should not be used to catch an AssertionError. .Section 14.20 of the Java Language Specification defines "unreachable" statements. If an assert statement is "unreachable" as defined by the JLS, then a compile-time error is generated. In contrast, a programmer may believe that some points in the code will not be reached as a result of

design assumptions. For example, a programmer may believe that the default case of a switch statement will never be reached. An assertion can be placed in the default case to verify the behavior of the switch statement. While the exception handling mechanisms of Java have been designed to allow for recovery from Exceptions, the assertion mechanisms have been designed to discourage recovery attempts. An assertion is used to verify that the program has not strayed beyond the bounds of expected behavior. For example, suppose that you go to bed one night, and your pet dog is sleeping on the floor next to your bed. Before going to sleep, you make the assertion that your dog will still be there in the morning. When you wake up, you find that a different dog is sleeping in place of your pet. How do you recover from the failure of your assertion? Since you probably did not expect your dog to be mysteriously replaced during the night, it is unlikely that you have already developed an effective recovery routine. However, if you had planned for a dog swapping exception, then the recovery should be handled by the exception handling mechanism rather than the assertion mechanism.

11.a b d .With assertions enabled it prints ABC followed by an AssertionError message. With assertions disabled it prints ABC followed by an AssertionError message. In this code example an assert statement could not be used in place of the "throw" statement. If the default label of a switch statement should not be reached under normal operating circumstances, then the default label might be a good location for an assert statement. If a method is declared with a non-void return type and if no return statement appears after the switch statement, then each case of the switch must have a return statement or a throw statement. The throw statement is used rather than an assert, because the compiler knows that the assert statement is not functional when assertions are disabled.

12.a d .With assertions enabled it prints ABC followed by an AssertionError message. With assertions disabled it prints ABCE .If the default label of a switch statement should not be reached under normal operating circumstances, then the default label might be a good candidate for the use of an assert statement.

13.d e .In this code example a throw statement must be used in place of the assert statement. Compile-time error .If the default label of a switch statement should not be reached under normal operating circumstances, then the default case becomes a good candidate for the use of an assert statement. If a method is declared with a non-void return type and if no return statement appears after the switch statement, then each case of the switch must have a return statement or a throw statement. The throw statement is used rather than an assert, because the compiler knows that the assert statement is not functional when assertions are disabled.

14.a d f .With assertions enabled it prints an AssertionError message. With assertions disabled it prints nothing. The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program. The assert statement indicates that the programmer believes that b1 and b2 will never be true simultaneously, and the assert statement should not be reached under normal operating conditions.

15.a d f h .If assertions are enabled at run time it prints an error message. With assertions disabled it prints 1112. The assert statements are being used to check a precondition--something that must be true when the method is invoked. Method m2 is an example of an improper use of an assert statement: an assert statement should not be used for argument checking in a public method. Assertions may be enabled or disabled at run time. Since assertions are not always enabled, they should not be used to validate the parameters of public methods. Parameter checking is typically published in the API specification of a method and must be enforced even when assertions are not enabled. Rather than use an assertion, an appropriate runtime exception should be thrown such as IllegalArgumentException, IndexOutOfBoundsException, or NullPointerException. However, an assertion may be used to validate the parameters of a nonpublic method.

16.b c e .If assertions are not enabled at run time it prints nothing. With assertions enabled it prints an error message. The assert statement is being used to check a postcondition--something that must be true when the method completes successfully. Variable c equals 200 when the assertion is checked.

17.b d e .If assertions are not enabled at run time it prints nothing. With assertions enabled it prints nothing. The assert statement is being used to check a class invariant--something that must be true about each instance of the class. This question is an example of using assertions to check a class invariant--something that must be true about each instance of the class. Although a class invariant must be true before and after the execution of each public method, the invariant is typically only checked at the end of each method and constructor.

18.a d f h .With assertions enabled it prints an error message. With assertions disabled it prints: true,true,false The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2 or b3, to be true. The assert statement is being used to check an internal

invariant--something that the programmer assumes to be true at a particular point in the program. .Method m1 has a series of if/else statements. The first if statement is processed if none of the booleans are true. The second is processed if only b1 is true. The third is processed if only b2 is true. A set of three booleans can exist is eight states. The three if statements account for three of those states; so five more states remain. The assert statement indicates that the programmer assumes that only one of those five remaining states is valid--that is the state where only b3 is true. The combination of the three if statements and the assert statement indicate that the programmer believes that no more than one of the booleans will be true at that point in the program. That assumption is called an internal invariant.

19.a c e .Prints "Only b1 is true" followed by an error message. The combination of the if/else statements and the assert statement indicate that the programmer expects no more than one boolean, b1, b2, or b3, to be true. The assert statement is being used to check a control-flow invariant to verify that the control flow never reaches a point in the program. .Method m1 has a series of if/else statements. The first if statement is processed if none of the booleans are true. The second is processed if only b1 is true. The third is processed if only b2 is true. The fourth is processed if only b3 is true. A set of three booleans can exist in one of eight states. The first four if statements account for four of those states; so four more states remain. The combination of the three if statements and the fact that an AssertionError is thrown from the last else block indicates that the programmer believes that no more than one of the booleans will be true when method m1 is being processed. An assumption concerning the state of a set of variables is called an internal invariant. In this case, however, the assertion was tested by verifying that control never reached a particular point in the program. Based on the testing technique, we would say that the assertion tests a control-flow invariant. A throw statement is used in place of an assert statement, because the throw statement can not be disabled. As a result, the method is certain to generate an error once control passes beyond all of the return statements. The declared return type of method m1 is String. No return statement appears after the sequence of if statements; therefore, every if statement must either return a String or throw an exception. Assertions can be disabled at run time, so an assert statement in the final if block is no guarantee that an exception will be thrown. For that reason, an assert can not replace the throw statement.

20.b e f .The flow of control does not reach a particular point in the program. The default case of a switch statement is not reached. The else block of an if/else statement is not reached. .A control-flow invariant is placed at a point in the program that the programmer assumes will never be reached. Two examples are the default case of a switch statement or the else block of an if/else statement. It makes no sense to use an assert statement to verify that the flow of control does reach a particular point in the program, because it is unlikely that an assertion error is helpful when the program is found to be functioning correctly. An assert statement placed at the beginning of a method is generally used to check a precondition. An assert statement that is placed at the end of a method to check the state of some variables is generally said to be checking a post condition. However, it is also possible that an assert statement placed at the end of a method might also be checking a control-flow invariant. The correct term depends on the usage.

## Garbage Collection

### Question 1

```
class B {
 private String name;
 public B(String s) {name = s;}
 protected void finalize() {System.out.print(name);} }
class E {
 public static void m() {
 B x1 = new B("X"), y1 = new B("Y");
 } }
```

```
}

public static void main(String[] args) {
 m(); System.gc();
}

}
```

Which of the following could not be a result of attempting to compile and run the program?

- a. Prints: XY
  - b. Prints: YX
  - c. Prints: XXYY
  - d. Nothing is printed.
  - e. None of the above
- 

## Question 2

```
void m1() {
 Q q1 = null;
 for (int i = 0; i < 10; i++) {
 q1 = new Q(); // 1
 m2(q1); // 2
 }
 System.out.print("All done"); // 3
}
```

When the processing of line 3 begins, how many objects of type Q that were created at line 1 have become eligible for garbage collection?

- a. 0
- b. 1
- c. 9
- d. 10
- e. Indeterminate.
- f. Compile-time error
- g. Run-time error

- h. None of the above
- 

### Question 3

```
class Q {
 private int id;
 protected void finalize() {System.out.print(id);}
 public Q(int i) {id = i;}
}
class R {
 public static void main(String[] args) {
 Q q1 = null;
 for (int i = 0; i < 10; i++) {q1 = new Q(i);} // 1
 System.gc(); // 2
 }
}
```

When the processing of line 2 begins, how many objects of type Q that were created at line 1 have become eligible for garbage collection?

- a. 0
  - b. 1
  - c. 9
  - d. 10
  - e. Indeterminate.
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### Question 4

```
class I {
 private I other;
 public void other(I i) {other = i;}
}
```

```
class J {
 private void m1() {
 I i1 = new I(), i2 = new I();
 I i3 = new I(), i4 = new I();
 i1.other(i3); i2.other(i1);
 i3.other(i2); i4.other(i4);
 }
 public static void main (String[] args) {
 new J().m1();
 }
}
```

Which object is not eligible for garbage collection after method m1 returns?

- a. i1
  - b. i2
  - c. i3
  - d. i4
  - e. Compile-time error
  - f. Run-time error
  - g. **None of the above**
- 

### Question 5

```
class I {
 private String name;
 public I(String s) {name = s;}
 private I other;
 public void other(I i) {other = i;}
}
class J {
 private I i1 = new I("A"), i2 = new I("B"), i3 = new I("C");
 private void m1() {
 i1.other(i2); i2.other(i1); i3.other(i3);
 i1 = i3; i2 = i3;
 }
}
```

```
m2();
}
private void m2() /* Do amazing things. */
public static void main (String[] args) {
 new J().m1();
}
```

Which of the three objects, A, B or C, is not eligible for garbage collection when method m2 begins to execute?

- a. A
  - b. B
  - c. **C**
  - d. None of the above
- 

### Question 6

```
class I {
 private String name;
 protected void finalize() {System.out.print(name);}
 public I(String s) {name = s;}
}
class J {
 private static void m1(I[] a1) {
 a1[0] = a1[1] = a1[2] = null;
 }
 public static void main (String[] args) {
 I[] a1 = new I[3]; // 1
 a1[0] = new I("A"); // 2
 a1[1] = new I("B"); // 3
 a1[2] = new I("C"); // 4
 m1(a1);
 System.gc();
 }
}
```

After method m1 returns, the object created on which line is not eligible for garbage collection?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
  - f. Compile-time error
  - g. Run-time error
- 

### Question 7

```
class I {
 private String name;
 public String toString() {return name;}
 public I(String s) {name = s;}
}
class J {
 private static void m1(I[] a1) {a1 = null;}
 public static void main (String[] args) {
 I[] a1 = new I[3]; // 1
 a1[0] = new I("A"); // 2
 a1[1] = new I("B"); // 3
 a1[2] = new I("C"); // 4
 m1(a1);
 for (int i = 0; i < a1.length; i++) {
 System.out.print(a1[i]);
 }}}
```

After method m1 returns, the object created on which line is eligible for garbage collection?

- a. 1
- b. 2
- c. 3
- d. 4

- e. Compile-time error
  - f. Run-time error
  - g. **None of the above**
- 

### Question 8

```
class A {
 private String name;
 private A otherA;
 public A(String name) {this.name = name;}
 public void other(A otherA) {this.otherA = otherA;}
 public A other() {return otherA;}
 public String toString() {return name;}
 protected void finalize() {System.out.print(name);} }

class B {
 public static void m1() {
 A a1 = new A("A1"), a2 = new A("A2"), a3 = new A("A3"), a0 = a3;
 a1.other(a2); a2.other(a3); a3.other(a1);
 for(int i = 0; i<4; i++){System.out.print(a0 = a0.other());}
 }
 public static void main(String[] args) {m1(); System.gc();}
}
```

Which of the following could be a result of attempting to compile and run the program?

- a. A1A2A3A1
  - b. A0A0A0A0A1A2A3
  - c. A1A2A3A1A2A3
  - d. **A1A2A3A1A1A2A3**
  - e. A1A2A3A1A3A2A1
  - f. A0A1A2A3A1A2A3
-

### Question 9

```
class B {
 private String name;
 public B(String name) {this.name = name;}
 public String toString() {return name;}
 protected void finalize() {System.out.print(name);} }

class H {
 static B ba = new B("Ba");
 static int i = 1;
 static B m1(B b) {return b = new B("B" + i++);}
 public static void main (String[] args) {
 B x = m1(ba); m1(x);
 System.out.println(", " + ba + ", " + x);
 } }
```

Which of the following could be a result of attempting to compile and run the program?

- a. Ba, B1, B2
  - b. B1, Ba, B2
  - c. , Ba, B1
  - d. B2, Ba, B1
  - e. BaB1b2, null, null
  - f. B1B2, ba, null
- 

### Question 10

```
class B {
 private String name;
 public B(String name) {this.name = name;}
 public String toString() {return name;}
 protected void finalize() {System.out.print(name);} }

class J {
```

```
static B bc;
static int i = 1;
static B m1(B b) {bc = b; return new B("B" + i++);}
public static void main (String[] args) {
 B x = m1(new B("Ba")), y = m1(new B("Bb"));
 System.out.println(", " + x + ", " + y + ", " + bc);
}}
```

Which of the following could be a result of attempting to compile and run the program?

- a. BaBb, B1, B2, B2
  - b. B1B2, null, null, Bb
  - c. , Ba, Bb, Bb
  - d. BaBbB1B2, null, null, null
  - e. **Ba, B1, B2, Bb**
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### Question 11

```
class I {
 private String name;
 public String name() {return name;}
 public I(String s) {name = s;}
}
class J {
 public static void m1(I i) {i = null;}
 public static void main (String[] args) {
 I i = new I("X"); // 1
 m1(i); // 2
 System.out.print(i.name()); // 3
 }
}
```

Which of the following is a true statement?

- a. The object created at line 1 is eligible for garbage collection after line 2.
- b. A NullPointerException is generated at line 3.
- c. The program compiles, runs and prints X.
- d. The program fails to compile.
- e. None of the above

1 c Prints: XXYY The program will not print XXYY. Please note that the question asks which could NOT be a result of attempting to compile and run the program. The finalize method of each instance can only run once; so X or Y can never be printed more than once. The instances referenced by x1 and y1 become eligible for garbage collection when method m returns; so both could be finalized at that time, but there is no guarantee that they will be. Even though System.gc is invoked in the main method, there is no guarantee that the garbage collector will run at that time. If the garbage collector does run before the program terminates, then the name of each object could be printed at most one time. The order in which the names are printed depends on the order in which the objects are finalized. If the garbage collector does not run, then nothing will be printed.

2 e Indeterminate. Since we don't know what method m2 might be doing, we can not know if the objects are eligible for garbage collection. Suppose that method m2 is declared inside of a class that also contains 10 instance variables (instance variables are non-static member fields) that are references to instances of class A. The argument that appears in the method invocation expression m2(q1) is a reference to an instance of class Q. Suppose that m2 saves each argument value in one of the ten instance variables or in an element of an array of type Q[]. When the loop in method m1 runs to completion, each instance of class Q would still be referenced by one of the ten instance variables. Since the instance variables would continue to reference each instance of class Q when line 3 is executed, none of the instances would be eligible for garbage collection at that point. A second possibility is that method m2 does not save the reference values. In that case, all of the instances that were created inside the loop would be eligible for garbage collection when line 3 is executed.

3 c 9 With each pass through the loop, q1 references a new object, and the old object becomes eligible for garbage collection. When the processing of line 2 begins, the last object referenced by q1 is not eligible for garbage collection.

4 g None of the above Please note that this question asks which object is NOT eligible for garbage collection after method m1 returns. The objects referenced by i1, i2 and i3 form a ring such that each object is referenced by another. Even so, nothing outside of method J.m1 references any of those objects. When method J.m1 returns, the ring becomes an island of isolated objects that are not reachable by any part of the user program. A key point to remember is that an object that is referenced by another object can be eligible for garbage collection if the two objects form an island of isolated objects.

5 c C Please note that this question asks which objects are NOT eligible for garbage collection when method m2 begins to execute? All three references, i1, i2 and i3, refer to object named C; so C is not eligible for garbage collection when method m2 begins to execute. The objects named A and B have references to each other, but no other objects refer to A and B. The objects A and B form an island of isolated objects and are eligible for garbage collection.

6 a 1 Please note that this question asks which objects are NOT eligible for garbage collection after method m1 returns. After method m1 returns, the array a1 created on line 1 is not eligible for garbage collection. Method m1 sets all elements of the array to null; so the objects created on lines 2, 3 and 4 are eligible for garbage collection when method m1 returns.

7 g None of the above After method m1 returns, none of the objects are eligible for garbage collection. Method m1 sets the parameter variable a1 to null, but that does not change the reference a1 in the J.main method. Since array a1 continues to reference all three objects, none of the three are eligible for garbage collection.

8 a c d e A1A2A3A1 A1A2A3A1A2A3 A1A2A3A1A1A2A3 A1A2A3A1A3A2A1 The three instances of class A form an isolated ring where each instance references the next instance and the third references the first instance. Four iterations of the for loop are processed. Inside the body of the for statement, the invocation of the print method contains the argument expression `a0 = a0.other()`. On the first iteration, the reference variable `a0` references the instance named A3. The value returned by the method named other is a reference to the instance named A1. The reference is assigned to the reference variable `a0` and is also the value produced by the expression `a0 = a0.other()`. That reference value is passed as an argument to the print method, and the print method invokes the `A.toString` method. With each iteration of the loop, the reference moves to the next object in the loop and the name of the object is printed. After four iterations, the loop ends and the method `m1` returns. The invocation of the `System.gc` method serves as a suggestion that the garbage collector should be allowed to run. The system could ignore the suggestion, so there is no guarantee that the eligible arguments will be garbage collected. If they are collected, there is no guarantee which will be collected first. The only guarantee is that the finalize method will be invoked on each particular instance before the resources that had been allocated to that instance are reclaimed.

9 c d , Ba, B1 B2, Ba, B1 Class H declares two static member variables named `ba` and `i`. The type of `i` is `int`, and the value is initialized to 1. The type of `ba` is `B`. The declaration of `ba` contains the class instance creation expression `new B("Ba")`. The constructor of class `B` assigns the argument value to the instance variable called `name`. Inside the main method of class `H`, the method invocation expression `m1(ba)` invokes method `m1`. The argument is the static member variable `ba`. The body of method `m1` contains a return statement with the expression `b = new B("B" + i++)`. The assignment expression contains the class instance creation expression `new B("B" + i++)` which creates a new instance of the class `B`. For this first invocation of method `m1`, the argument appearing in the class instance creation expression is the String value `B1`. The reference to the new String is assigned to the parameter variable `b`, but that assignment does not change the value of the member variable `ba`. The value of the assignment expression is the reference to the new instance of class `B` with the name `B1`, and that reference value is returned by the method `m1`. The returned value is assigned to the local variable `x`. The next statement inside the main method is another invocation of method `m1`. The argument appearing in the method invocation expression `m1(x)` is the local reference variable `x`. The method invocation does not change the value of `x`. The value returned by this second invocation of `m1` is a reference to a new instance of class `B` that has the name `B2`. The returned reference value is not assigned to a variable, so the instance named `B2` is eligible for garbage collection. There is no guarantee that the garbage collector will run before the print statement is invoked. If it does run, then the instance named `B2` could be finalized causing the name to be printed.

10 e Ba, B1, B2, Bb Class J declares two static member variables named `bc` and `i`. The type of `i` is `int`, and the value is initialized to 1. The type of `bc` is `B`. Inside the main method of class `J`, the method invocation expression `m1(new B("Ba"))` invokes method `m1`. The argument is the class instance creation expression `new B("Ba")`. The constructor of class `B` assigns the argument value to the instance variable called `name`, so a new instance of class `B` named `Ba` is created. The reference to the new instance of class `B` is the argument that is passed to method `m1`. The body of method `m1` contains two statements. The first contains the assignment expression `bc = b` that assigns the value of the method parameter `b` to the static member variable `bc`, so `bc` now references the instance of class `B` named `Ba`. The second statement in the body of `m1` is a return statement with the class instance creation expression `new B("B" + i++)`. For this first invocation of method `m1`, the argument appearing in the class instance creation expression is the String value `B1`. The reference to the new String is returned by method `m1`. The returned value is assigned to the local variable `x`. Inside the main method, the declaration of the local variable `y` contains another invocation of method `m1`. The argument appearing in the method invocation expression `m1(new B("Bb"))` is the class instance creation expression `new B("Bb")`, so the argument value for the invocation of method `m1` is a reference to a new instance of class `B` named `Bb`. Inside the body of method `m1`, the reference to the new instance of class `B` named `Bb` is assigned to the static member variable `Bc`. At that point, the instance of class `B` named `Ba` becomes eligible for garbage collection. Method `m1` returns a reference to a new instance of class `B` named `B2`. There is no guarantee that the garbage collector will run before the print statement is invoked. If it does run, then the instance named `Ba` could be finalized causing the name to be printed.

11 c The program compiles, runs and prints X. The parameter i of method m1 is a copy of the local variable i of method J.main. Setting the parameter variable i of method m1 to null does not change the local variable i of method J.main.

## Packages

### Question 1

```
package com.dan.chisholm;
public class A {
 public void m1() {System.out.print("A.m1, ");}
 protected void m2() {System.out.print("A.m2, ");}
 private void m3() {System.out.print("A.m3, ");}
 void m4() {System.out.print("A.m4, ");}
}
class B {
 public static void main(String[] args) {
 A a = new A();
 a.m1(); // 1
 a.m2(); // 2
 a.m3(); // 3
 a.m4(); // 4
 }
}
```

Assume that the code appears in a single file named A.java. What is the result of attempting to compile and run the program?

- a. Prints: A.m1, A.m2, A.m3, A.m4,
  - b. Compile-time error at 1.
  - c. Compile-time error at 2.
  - d. Compile-time error at 3.**
  - e. Compile-time error at 4.
  - f. None of the above
- 

### Question 2

```
// Class A is declared in a file named A.java.
```

```
package com.dan.chisholm;
public class A {
 public void m1() {System.out.print("A.m1, ");}
 protected void m2() {System.out.print("A.m2, ");}
 private void m3() {System.out.print("A.m3, ");}
 void m4() {System.out.print("A.m4, ");}
}
// Class D is declared in a file named D.java.
package com.dan.chisholm.other;
import com.dan.chisholm.A;
public class D {
 public static void main(String[] args) {
 A a = new A();
 a.m1(); // 1
 a.m2(); // 2
 a.m3(); // 3
 a.m4(); // 4
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A.m1, A.m2, A.m3, A.m4,
  - b. Compile-time error at 1.
  - c. **Compile-time error at 2.**
  - d. **Compile-time error at 3.**
  - e. **Compile-time error at 4.**
- 

### Question 3

```
// Class A is declared in a file named A.java.
package com.dan.chisholm;
public class A {
 public void m1() {System.out.print("A.m1, ");}
 protected void m2() {System.out.print("A.m2, ");}
 private void m3() {System.out.print("A.m3, ")}
```

```

void m4() {System.out.print("A.m4, ");}
}
// Class C is declared in a file named C.java.
package com.dan.chisholm.other;
import com.dan.chisholm.A;
public class C extends A {
 public static void main(String[] args) {
 C c = new C();
 c.m1(); // 1
 c.m2(); // 2
 c.m3(); // 3
 c.m4(); // 4
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: A.m1, A.m2, A.m3, A.m4,
- b. Compile-time error at 1.
- c. Compile-time error at 2.
- d. Compile-time error at 3.**
- e. Compile-time error at 4.**

1.d .Compile-time error at 3. .Both class A and B are declared in the same package, so class B has access to the public, protected, and package access methods of class A.

2.c d e .Compile-time error at 2. Compile-time error at 3. Compile-time error at 4. .Classes A and D are not declared in the same package, so class D does not have access to package access method, m4. Since class D does not extend class A, class D does not have access to the protected method, m2, of class A.

3.d e .Compile-time error at 3. Compile-time error at 4. .Class A and C are not declared in the same package; therefore, class C does not have access to package access method, m4. Since class C extends class A, class C does have access to the protected method, m2, of class A.

## Interfaces

### Question 1

```

interface A {
 void m1(); // 1
 public void m2(); // 2
 protected void m3(); // 3
}

```

```
private void m4(); // 4
}
```

Compile-time errors are generated at which lines?

- a. 1
  - b. 2
  - c. 3
  - d. 4
- 

## Question 2

Which of the following statements is not true?

- a. An interface that is declared within the body of a class or interface is known as a nested interface.
  - b. A constant can be a member of an interface.
  - c. A class declaration can be a member of an interface.
  - d. If an interface is named in the implements clause of a class, then the class must implement all of the methods declared within the interface.
  - e. None of the above.
- 

## Question 3

Which of the following statements are true?

- a. An interface declaration can be a member of an interface.
  - b. A method declared within an interface must have a body represented by empty curly braces.
  - c. An interface can implement another interface.
  - d. An abstract class that implements an interface must implement all abstract methods declared within the interface.
  - e. An abstract method declaration can be a member of an interface.
- 

## Question 4

Which of the following are modifiers that can be applied to an interface that is a member of a directly enclosing interface?

- a. **Abstract**
  - b. implements
  - c. Final
  - d. Private
  - e. protected
  - f. **public**
- 

### Question 5

Which of the following are modifiers that can be applied to an interface that is a member of a directly enclosing class?

- a. **abstract**
  - b. extends
  - c. final
  - d. **private**
  - e. **protected**
  - f. **Public**
- 

### Question 6

Which of the following is a modifier that can be applied to an interface that is a member of a directly enclosing class or interface?

- a. **Static**
  - b. synchronized
  - c. Transient
  - d. Volatile
  - e. implements
  - f. None of the above.
-

## Question 7

Suppose that an interface, I1, is not a member of an enclosing class or interface. Which of the following modifiers can be applied to interface I1?

- a. Abstract
  - b. Final
  - c. Private
  - d. Protected
  - e. Public
- 

## Question 8

Suppose that an interface, I1, is not a member of an enclosing class or interface. Which of the following modifiers can be applied to interface I1?

- a. Abstract
  - b. Public
  - c. Static
  - d. synchronized
  - e. Transient
  - f. Volatile
- 

## Question 9

Which of the following are modifiers that can be applied to a field declaration within an interface?

- a. abstract
- b. Const
- c. Final
- d. private
- e. protected
- f. Public

---

### Question 10

Which of the following is a modifier that can be applied to a field declaration within an interface?

- a. static
  - b. Synchronized
  - c. Transient
  - d. Volatile
  - e. None of the above.
- 

### Question 11

Which of the following modifiers can be applied to a class that is declared within an enclosing interface?

- a. Public
  - b. Protected
  - c. Private
  - d. Abstract
  - e. Static
  - f. Final
- 

### Question 12

```
interface A {
 int a = 1; // 1
 public int b = 2; // 2
 public static int c = 3; // 3
 public static final int d = 4; // 4
}
```

Which field declaration results in a compile-time error?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
- 

### Question 13

```
interface A {
 protected int e = 5; // 1
 private int f = 6; // 2
 volatile int g = 7; // 3
 transient int h = 8; // 4
 public static final int d = 9; // 5
}
```

Which of the field declarations does not result in a compile-time error?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. None of the above
- 

### Question 14

Which of the following are modifiers that can be applied to a method declaration within an interface?

- a. abstract
- b. Final
- c. private

- d. protected
  - e. Public
- 

### Question 15

Which of the following is a modifier that can be applied to a method declaration within an interface?

- a. Static
  - b. synchronized
  - c. transient
  - d. volatile
  - e. Native
  - f. None of the above
- 

### Question 16

```
interface A {void m1();} // 1
class B implements A {public void m1() {} } // 2
class C implements A {protected void m1() {} } // 3
class D implements A {private void m1() {} } // 4
class E implements A {void m1() {} } // 5
```

Compile-time errors are generated at which lines?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
- 

### Question 17

```
interface A {
 void m1(); // 1
 public void m2(); // 2
 protected void m3(); // 3
 private void m4(); // 4
 abstract void m5(); // 5
}
```

Compile-time errors are generated at which lines?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
- 

### Question 18

```
interface A {
 final void m1(); // 1
 synchronized void m2(); // 2
 native void m3(); // 3
 abstract void m4(); // 4
 public void m5(); // 5
}
```

Compile-time errors are generated at which lines?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
-

### Question 19

```
interface A { void main(String[] args); } // 1
interface B { public void main(String[] args); } // 2
interface C { public static void main(String[] args); } // 3
interface D { protected void main(String[] args); } // 4
interface E { private void main(String[] args); } // 5
```

Which interface declarations generate a Compile-time error?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
- 

### Question 20

```
interface F { abstract void main(String[] args); } // 1
interface G { synchronized void main(String[] args); } // 2
interface H { final void main(String[] args); } // 3
interface I { native void main(String[] args); } // 4
```

Which interface declaration does not generate a compile-time error?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
- 

### Question 21

```
interface A {String s1 = "A"; String m1();}
interface B implements A {String s1 = "B"; String m1();}
class C implements B {
 public String m1() {return s1;}
 public static void main(String[] args) {
 A a = new C(); System.out.print(a.m1());
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: A
  - b. Prints: B
  - c. **Compile-time error**
  - d. Run-time error
  - e. None of the above
- 

## Question 22

```
interface A {int i = 1; int m1();}
interface B extends A {int i = 10; int m1();}
class C implements B {
 public int m1() {return ++i;}
 public static void main(String[] args) {
 System.out.print(new C().m1());
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 2
- b. Prints: 11
- c. **Compile-time error**
- d. Run-time error
- e. None of the above

---

### Question 23

```
interface Z {void m1();} // 1
class A implements Z {void m1() {} } // 2
class B implements Z {public void m1() {} } // 3
abstract class C implements Z {public abstract void m1();} // 4
```

A Compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
- 

### Question 24

```
interface Z {void m1();} // 1
class D implements Z {public final void m1() {} } // 2
class E implements Z {public synchronized void m1() {} } // 3
class G implements Z {public native void m1();} // 4
```

A Compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
- 

### Question 25

```
interface I10 {String name = "I10"; String s10 = "I10.s10";}
interface I20 {String name = "I20"; String s20 = "I20.s20";}
class C10 implements I10, I20 { // 1
 public static void main(String[] args) {
 System.out.print(s10+","); // 2
 System.out.print(s20+","); // 3
 System.out.print(name); // 4
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: I10.s10,I20.s20,I10
  - b. Prints: I10.s10,I20.s20,I20
  - c. Prints: I10.s10,I20.s20,
  - d. Prints: I10.s10,I20.s20,null
  - e. Compile-time error at line 1
  - f. Compile-time error at line 2
  - g. Compile-time error at line 3
  - h. **Compile-time error at line 4**
  - i. Run-time error
  - j. None of the above
- 

## Question 26

```
interface I10 {String name = "I10"; String s10 = "I10.s10";}
interface I20 {String name = "I20"; String s20 = "I20.s20";}
class C20 implements I10, I20 { // 1
 public static void main(String[] args) {
 System.out.print(I10.s10+","); // 2
 System.out.print(I20.s20+","); // 3
 System.out.print(I20.name); // 4
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: I10.s10,I20.s20,I10
- b. Prints: I10.s10,I20.s20,I20
- c. Prints: I10.s10,I20.s20,
- d. Prints: I10.s10,I20.s20,null
- e. Compile-time error at line 1
- f. Compile-time error at line 2
- g. Compile-time error at line 3
- h. Compile-time error at line 4
- i. Run-time error
- j. None of the above

1 c d 3 4 Methods declared within an interface are implicitly public. If no access modifier is included in the method declaration; then, the declaration is implicitly public. An attempt to declare the method using a weaker access privilege, private or protected, results in a compile-time error.

2 d If an interface is named in the implements clause of a class, then the class must implement all of the methods declared within the interface. This question asks which answer option is not true. Some true statements are as follows. An interface can be declared within an enclosing class or interface. The members of an interface can be constants, abstract method declarations, class declarations or interface declarations. If an interface is named in the implements clause of a class, then the class must implement all of the methods declared within the interface or the class must be declared abstract. The untrue answer option did not mention that an abstract class is not required to implement any of the methods declared in an interface that is named in the implements clause of the class declaration.

3 a e An interface declaration can be a member of an interface. An abstract method declaration can be a member of an interface. An interface can be declared within an enclosing class or interface. The members of an interface can be constants, abstract method declarations, class declarations, or interface declarations. The body of a method declared within an interface is a semicolon. An interface can extend another interface, but can not implement an interface. An abstract class that has an interface, I1, in its implements clause is not required to implement any of the methods declared within I1.

4 a f abstract public All interfaces are implicitly abstract. The explicit application of the abstract modifier to an interface declaration is redundant and is strongly discouraged. The declaration of an interface within the body of an enclosing class or interface is called a member type declaration. Every member type declaration appearing within the body of a directly enclosing interface is implicitly static and public. Use of the access modifiers, private or protected, is contradictory and results in a compile-time error. In contrast, the modifiers, private and protected, are applicable to a member type declaration appearing within the body of a directly enclosing class. The modifier, final, is never applicable to an interface. The keyword, implements, is not a modifier.

5 a d e f abstract private protected public All interfaces are implicitly abstract. The explicit application of the modifier, abstract, to an interface is redundant and is strongly discouraged. The declaration of an interface within the body of an enclosing class or interface is called a member type declaration. The private, protected and static modifiers are applicable to a member type declaration that appears in the body of a directly enclosing class. In contrast, the modifiers, private and protected, are not applicable to a member type declaration appearing within the body of a directly enclosing interface. The modifier, final, is never applicable to an interface. The keyword, extends, is not a modifier.

6 a static A member interface is always implicitly static. The modifier, static, can not be applied to an interface that is not a member interface. The modifier, synchronized, is applicable to a concrete implementation of a method, but is not applicable to any interface. The modifiers, volatile and transient, are only applicable to variables that are members of a class. The keyword, implements, is not a modifier.

7 a e abstract public The modifier, abstract, is applicable to an interface declaration, but its use is strongly discouraged; because every interface is implicitly abstract. An interface can not be final. The modifiers, private and protected, are applicable only to an interface declaration that is a member of a directly enclosing class declaration. If an interface is not a member of a directly enclosing class, or if the interface is a member of a directly enclosing interface; then, the modifiers, private and protected, are not applicable. If an interface is declare public, then the compiler will generate an error if the class is not stored in a file that has the same name as the interface plus the extension .java.

8 a b abstract public The modifier, abstract, is applicable to an interface declaration, but its use is strongly discouraged; because every interface is implicitly abstract. If an interface is declare public, then the compiler will generate an error if the class is not stored in a file that has the same name as the interface plus the extension .java. The modifier, static, is applicable to a member interface, but not to an interface that is not nested. The modifier, synchronized, is applicable only to concrete implementations of methods. The modifiers, transient and volatile, are applicable only to variables.

9 c f final public The modifier, abstract, is not applicable to a variable. All field declarations within an interface are implicitly public, static and final. Use of those modifiers is redundant but legal. Although const is a Java keyword, it is not currently used by the Java programming language. An interface member can never be private or protected.

10 a static All field declarations within an interface are implicitly public, static and final. Use of these modifiers is redundant but legal. A field that is declared final can not also be declared volatile; so a field of an interface can not be declared volatile. The modifier, synchronized, is never applicable to a field.

11 a d e f public abstract static final A class that is declared within an enclosing interface is implicitly public and static; so the access modifiers, protected and private, are not applicable.

12 e None of the above All field declarations within an interface are implicitly public, static and final. Use of these modifiers is redundant but legal. No other modifiers can be applied to a field declaration within an interface.

13 e 5 All field declarations within an interface are implicitly public, static and final. Use of these modifiers is redundant but legal. No other modifiers can be applied to a field declaration within an interface.

14 a e abstract public All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. An abstract method can not also be declared private, static, final, native or synchronized; so the same restriction applies to methods declared within an interface.

15 f None of the above All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. An abstract method can not also be declared private, static, final, native or synchronized; so the same restriction applies to methods declared within an interface. Transient and volatile are not method modifiers.

16 c d e 3 4 5 Methods declared within an interface are implicitly public even if the modifier, public, is omitted from the declaration. Within the body of a class declaration, an attempt to implement the method using a weaker access privilege, private, protected or package access, results in a compile-time error.

17 c d 3 4 All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. Since all methods declared within an interface are implicitly public, a weaker access level can not be declared.

18 a b c 1 2 3 All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. The final, synchronized and native modifiers can not appear in the declaration of an abstract method, and can not be applied to an abstract method declared within an interface.

19 c d e 3 4 5 All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. Since all methods declared within an interface are implicitly public, a weaker access level can not be declared.

20 a 1 All methods declared within an interface are implicitly abstract. The final, synchronized and native modifiers can not appear in the declaration of an abstract method, and can not be applied to an abstract method declared within an interface.

21 c Compile-time error In the declaration of interface B, the keyword, extends, has been replaced by the keyword, implements.

22 c Compile-time error Fields declared within an interface are implicitly public, final, and static. A compile-time error is generated in response to the attempt to increment the value of i.

23 b 2 All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. Methods declared within an interface are implicitly public even if the modifier, public, is omitted from the declaration. Within the body of a class declaration, an attempt to implement the method using a weaker access privilege, private, protected or package access, results in a compile-time error. An abstract class that implements an interface is free to override any of the inherited method declarations with another abstract method declaration.

24 e None of the above All methods declared within an interface are implicitly abstract and public. Although the abstract and public modifiers can legally be applied to a method declaration within an interface, the usage is redundant and is discouraged. The modifiers, final, synchronized and native, can not appear in the declaration of an abstract method, but they can be added to an implementation of an abstract method.

25 h Compile-time error at line 4 Class C10 inherits ambiguous declarations of the name field. As long as the field is not referenced as a member of class C10; then, no compile-time error occurs. Line 4 generates the compile-time error, because it is the first to access the name field as a member of class C10.

26 b Prints: I10.s10,I20.s20,I20 Class C20 inherits ambiguous declarations of the name field. As long as the field is not referenced as a member of class C20; then, no compile-time error occurs. Although line 4 may appear to generate the compile-time error it does not, because name is accessed directly as a member of interface I20. Therefore, the compiler does not encounter an ambiguity.

## Main Method

### Question 1

```
class GRC10 {
 public static void main (String[] s) {
 System.out.print(s[1] + s[2] + s[3]);
 }
}
java GRC10 A B C D E F
```

What is the result of attempting to compile and run the program using the specified command line?

- a. Prints: ABC
- b. Prints: BCD

- c. Prints: CDE
  - d. Prints: A B C
  - e. Prints: B C D
  - f. Prints: C D E
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

## Question 2

```
class GRC4 {public static void main(String[] args) {}} // 1
class GRC5 {public static void main(String []args) {}} // 2
class GRC6 {public static void main(String args[]) {}} // 3
```

What is the result of attempting to compile and run the above programs?

- a. Compile-time error at line 1.
  - b. Compile-time error at line 2.
  - c. Compile-time error at line 3.
  - d. An attempt to run GRC4 from the command line fails.
  - e. An attempt to run GRC5 from the command line fails.
  - f. An attempt to run GRC6 from the command line fails.
  - g. **None of the above**
- 

## Question 3

```
class GRC7 {public void main(String[] args) {}} // 1
class GRC8 {public void main(String []args) {}} // 2
class GRC9 {public void main(String args[]) {}} // 3
```

What is the result of attempting to compile and run the above programs?

- a. Compile-time error at line 1.
  - b. Compile-time error at line 2.
  - c. Compile-time error at line 3.
  - d. An attempt to run GRC7 from the command line fails.
  - e. An attempt to run GRC8 from the command line fails.
  - f. An attempt to run GRC9 from the command line fails.
- 

#### Question 4

```
class GRC1 {public static void main(String[] args) {} } // 1
class GRC2 {protected static void main(String[] args) {} } // 2
class GRC3 {private static void main(String[] args) {} } // 3
```

What is the result of attempting to compile each of the three class declarations and invoke each main method from the command line?

- a. Compile-time error at line 1.
- b. Compile-time error at line 2.
- c. Compile-time error at line 3.
- d. An attempt to run GRC1 from the command line fails.
- e. An attempt to run GRC2 from the command line fails.
- f. An attempt to run GRC3 from the command line fails.

1 b Prints: BCD The index for the first element of an array is zero so the first argument printed by this program is the second argument on the command line following the name of the class.

2 g None of the above Section 12.1.4 of the Java Language Specification requires that the main method must accept a single argument that is an array of components of type String. In each of the three class declarations, the single argument is indeed an array of components of type String. Please note that the square brackets within an array declaration may appear as part of the type or part of the declarator (i.e. array name).

3 d e f An attempt to run GRC7 from the command line fails. An attempt to run GRC8 from the command line fails. An attempt to run GRC9 from the command line fails. Section 12.1.4 of the JLS requires the main method to be declared static. In this example, each of the three main methods are not declared static. The result is an error at run-time.

4 e f An attempt to run GRC2 from the command line fails. An attempt to run GRC3 from the command line fails. Section 12.1.4 of the JLS requires the main method to be declared public. The main methods of GRC2 and GRC3 are not declared public and can not be invoked from the command line using a JVM that is compliant with section 12.1.4. Not every JVM enforces the rule. Even so, for the purposes of the SCJP exam, the main method should be declared as required by the JLS.

## **Keywords**

### **Question 1**

Which of these words belongs to the set of Java keywords?

- a. qualified
  - b. record
  - c. repeat
  - d. restricted
  - e. label
  - f. to
  - g. type
  - h. until
  - i. value
  - j. virtual
  - k. xor
  - l. **None of the above**
- 

### **Question 2**

Which of these words belong to the set of Java keywords?

- a. Exit
- b. **Strictfp**
- c. Enum
- d. **Super**
- e. Abort
- f. Event
- g. **Goto**
- h. **Native**

i. Exception

---

**Question 3**

Which of these words belong to the set of Java keywords?

- a. Double
  - b. Goto
  - c. Min
  - d. Extern
  - e. New
  - f. Signed
  - g. Finally
  - h. Const
  - i. And
  - j. Array
  - k. Do
- 

**Question 4**

Which of these words belong to the set of Java keywords?

- a. Declare
- b. Global
- c. Const
- d. preserve
- e. Continue
- f. Float
- g. Extends
- h. Select

- i. Break
  - j. Dim
  - k. instanceof
- 

### Question 5

Which of these words belong to the set of Java keywords?

- a. next
  - b. catch
  - c. function
  - d. instanceof
  - e. Mod
  - f. Const
  - g. Or
  - h. Boolean
  - i. Goto
  - j. Import
  - k. transient
- 

### Question 6

Which of these words belong to the set of Java keywords?

- a. Byte
- b. Short
- c. Int
- d. Long
- e. Decimal
- f. int64

- g. **Float**
  - h. Single
  - i. **Double**
  - j. **Boolean**
  - k. **Char**
  - l. Unsigned
  - m. Array
  - n. String
- 

### Question 7

Which of these words belongs to the set of Java keywords?

- a. clause
  - b. loop
  - c. expression
  - d. overrides
  - e. statement
  - f. assertion
  - g. validate
  - h. exception
  - i. cast
  - j. **None of the above**
- 

### Question 8

Which of these words belong to the set of Java keywords?

- a. **transient**
- b. Serializable

- c. Runnable
  - d. Run
  - e. volatile
  - f. Externalizable
  - g. Cloneable
- 

### Question 9

Which of these words belong to the set of Java keywords?

- a. virtual
  - b. goto
  - c. ifdef
  - d. typedef
  - e. friend
  - f. struct
  - g. Implements
  - h. Union
  - i. const
- 

### Question 10

Which of these lists contains at least one word that is not a Java keyword?

- a. abstract, default, if, private, this
  - b. do, implements, protected, boolean, throw
  - c. case, extends, int, short, try
  - d. import, break, double, exception, throws
  - e. byte, else, instanceof, return, transient
  - f. None of the above
-

## Question 11

Which of these lists contains at least one word that is not a Java keyword?

- a. interface, static, void, catch, final
- b. char, strictfp, finally, long, volatile
- c. native, super, class, float, while
- d. const, for, new, switch, import
- e. continue, finalize, goto, package, synchronized
- f. None of the above

1 1 None of the above All of these are keywords of the Pascal programming language, but none are Java keywords.

2 b d g h strictfp super goto native

3 b e g h k goto new finally const do

4 c e g i const continue extends break All of the letters of all Java keywords are lower case. The word instanceof is a Java keyword, but instanceof is not.

5 b d f i j k catch instanceof const goto import transient

6 a b c d g i j k byte short int long float double boolean char

7 j None of the above

8 a e transient volatile Serializable, Runnable, Externalizable, and Cloneable are all interfaces. Thread.run is a method. The keywords transient and volatile are field modifiers.

9 b g i goto implements const The words virtual, ifdef, typedef, friend, struct and union are all words related to the C programming language. Although the words const and goto are also related to the C programming language, they are also Java keywords.

10 d import, break, double, exception, throws The word exception is not a Java keyword. The words import, break, double and throws are Java keywords.

11 e continue, finalize, goto, package, synchronized The word finalize is the name of a method of the Object class: It is not a keyword. The words continue, goto, package and synchronized are all Java keywords.

## Initialization

### Question 1

```
class GFM11{
 public static void main (String[] args) {
 int x,y,z;
 System.out.println(x+y+z);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints nothing.
  - b. Prints an undefined value.
  - c. Prints: null
  - d. Prints: 0
  - e. Run-time error
  - f. **Compile-time error**
  - g. None of the above
- 

## Question 2

```
class GFM12 {
 static int x; // 1
 public static void main(String[] args) { // 2
 int y; // 3
 System.out.println("x="+x); // 4
 System.out.println("y="+y); // 5
 }
}
```

What is the result of attempting to compile and run the program?

- a. Compile-time error at line 1.
  - b. Compile-time error at line 2.
  - c. Compile-time error at line 3.
  - d. Compile-time error at line 4.
  - e. **Compile-time error at line 5.**
  - f. Run-time error
  - g. None of the above
- 

## Question 3

```
class GFM13 {
 static byte a; static short b; static char c;
```

```
static int d; static long e; static String s;
public static void main(String[] args) {
 System.out.println(a+b+c+d+e+s);
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 00000null
  - b. Prints: 00000
  - c. Prints: Onull
  - d. Prints: 0
  - e. Prints: null
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

#### Question 4

```
class GFM14 {
 static byte a; static short b; static char c;
 static int d; static long e; static String s;
 public static void main(String[] args) {
 System.out.println(a+","+b+","++(int)c+","+d+","+e+","+s);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,0,null
- b. Prints: 0,0,0,0,
- c. Prints: 0,0, ,0,0,
- d. Compile-time error
- e. Run-time error
- f. None of the above

---

### Question 5

```
class GFM15 {
 static int a; static float b; static double c;
 static boolean d; static String s;
 public static void main(String[] args) {
 System.out.println(a+","+b+","+c+","+d+","+s);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0,false,null
  - b. Prints: 0,0,0,false,
  - c. Prints: 0,0,0,0,false,null
  - d. Prints: 0,0,0,0,0,false,
  - e. Prints: 0,0,0,0,0,true,null
  - f. Prints: 0,0,0,0,0,true,
  - g. Prints: 0,0,0,true,null
  - h. Prints: 0,0,0,true,
  - i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

### Question 6

```
class GFM16 {
 static int m1 (int i1, int i2) {
 int i3;
 if (i1 > 0) {i3 = i1 + i2;}
 return i3;
 }
 public static void main(String[] args) {
```

```
 System.out.println(m1(1,2));
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0
  - b. Prints: 1
  - c. **Compile-time error**
  - d. Run-time error
  - e. None of the above
- 

### Question 7

```
class GFM17 {
 int x; // 1
 public static void main(String[] args) { // 2
 int y = 0; // 3
 System.out.print(x+","); // 4
 System.out.print(y); // 5
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0
- b. Compile-time error at line 1.
- c. Compile-time error at line 1.
- d. Compile-time error at line 2.
- e. Compile-time error at line 3.
- f. **Compile-time error at line 4.**
- g. Compile-time error at line 5.
- h. Run-time error
- i. None of the above

- 1 f Compile-time error Variables declared inside of a block or method are called local variables; they are not automatically initialized. The compiler will generate an error as a result of the attempt to access the local variables before a value has been assigned.
- 2 e Compile-time error at line 5. The local variable y has not been initialized so the attempt to access the variable results in a compile-time error.
- 3 c Prints: 0null The numeric sum of variables a, b, c, d and e is zero. The zero is converted to a String and concatenated with s.
- 4 a Prints: 0,0,0,0,null The default value of type char is the null character. When it is cast to an int the value is interpreted as zero.
- 5 c Prints: 0,0,0,0,false,null Generally speaking, numeric type variables are initialized to zero. Primitive boolean variables are initialized to false. Reference type variables are initialized to null.
- 6 c Compile-time error Local variables are not initialized automatically, and must be initialized explicitly before attempting to access the value. The local variable i3 will not be initialized if i1 is less than or equal to zero; so the result is a compile-time error.
- 7 f Compile-time error at line 4. Anytime a field is accessed from within a static context it is very important to verify that the field is also static. If the field is instead an instance variable then the result is a Compile-time error.

## Range

### Question 1

```
class JJF1 {
 public static void main (String args[]) {
 System.out.print(Byte.MIN_VALUE+",");
 System.out.print(Byte.MAX_VALUE);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,255
- b. Prints: 0,256
- c. Prints: -127,128
- d. Prints: -128,127
- e. Compile-time error
- f. Run-time error
- g. None of the above

---

### Question 2

```
class JJF2 {
 public static void main (String args[]) {
 System.out.print(Short.MIN_VALUE+",";
 System.out.print(Short.MAX_VALUE);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: -32767,32768
  - b. Prints: -32768,32767
  - c. Prints: 0,65535
  - d. Prints: 0,65536
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 3

```
class JJF3 {
 public static void main(String args[]) {
 System.out.print(Integer.toBinaryString(Byte.MAX_VALUE)+",";
 System.out.print(Integer.toOctalString(Byte.MAX_VALUE)+",";
 System.out.print(Integer.toString(Byte.MAX_VALUE)+",";
 System.out.print(Integer.toHexString(Byte.MAX_VALUE));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 1111111,177,127,7f
- b. Prints: 11111111,377,256,ff
- c. Compile-time error
- d. Run-time error
- e. None of the above

---

#### Question 4

```
class JJF4 {
 public static void main(String args[]) {
 System.out.print(Long.toHexString(Byte.MAX_VALUE)+",");
 System.out.print(Long.toHexString(Character.MAX_VALUE)+",");
 System.out.print(Long.toHexString(Short.MAX_VALUE));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: f,ff,7f
  - b. Prints: f,ff,ff
  - c. Prints: 7f,ffff,7fff
  - d. Prints: ff,ffff,ffff
  - e. Prints: 7fff,fffffff,7fffffff
  - f. Prints: ffff,fffffff,fffffff
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

#### Question 5

```
class JJF5 {
 public static void main(String args[]) {
 System.out.print(Integer.toHexString(Integer.MIN_VALUE)+",");
 System.out.print(Integer.toHexString(Integer.MAX_VALUE));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0000,ffff

- b. Prints: 00000000,ffffffffff
  - c. Prints: 7fff,8000
  - d. Prints: 8000,7fff
  - e. Prints: 7fffffff,80000000
  - f. Prints: 80000000,7fffffff
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

### Question 6

Which of the following represent the full range of type char?

- a. '\u0000' to '\u7fff'
  - b. '\u0000' to '\uffff'
  - c. 0 to 32767
  - d. 0 to 65535
  - e. -32768 to 32767
  - f. -65536 to 65535
- 

### Question 7

Which of the following represent the full range of type char?

- a. -0x8000 to 0x7fff
- b. 0x0000 to 0xffff
- c. -0100000 to 077777
- d. 0 to 0177777
- e. 0 to 32767
- f. 0 to 65535

g. -32768 to 32767

---

### Question 8

```
class JJF6 {
 char c1 = 0xffff; // 1
 char c2 = 0xfffff; // 2
 byte b1 = 0xffff; // 3
 byte b2 = 0x7f; // 4
 byte b3 = 0xff; // 5
 byte b4 = -0x80; // 6
}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5
- f. 6

1 d Prints: -128,127 A byte is an 8 bit signed value; so the minimum byte value is -(2<sup>7</sup>) and the maximum value is (2<sup>7</sup> - 1).

2 b Prints: -32768,32767 A short is a 16 bit signed value; so the minimum short value is -(2<sup>15</sup>) and the maximum value is (2<sup>15</sup> - 1).

3 a Prints: 1111111,177,127,7f A byte is an 8 bit signed value. The left most bit is the sign bit. The sign bit is set to zero for positive numbers and is set to one for negative numbers. The most positive byte value is represented as a sign bit that is set to zero and all of the other bits set to one. The Integer.toBinaryString method does not print leading zeros; so only seven bits are printed. An eight bit binary value is represented as three octal digits. The first of the three digits represents the left most two bits of the binary value. In this case, the left most two bits are zero and one. The second octal digit represents the next three bits of the binary value. The last of the octal digits represents the right most three bits of binary value. Note that the Integer.toOctalString method does not print a leading zero as is required for an octal literal value. An eight bit binary value is represented as two hexadecimal digits. The left hex digit represents the left most four bits of the binary value. The right hex digit represents the right most four bits of the binary value.

4 c Prints: 7f,ffff,7fff A byte is an 8 bit signed value. A char is a 16 bit unsigned value. A short is a 16 bit signed value. The left most bit of a signed value is the sign bit. The sign bit is zero for positive numbers and one for negative numbers. The maximum byte value in hexadecimal format is 7f and in decimal format is 127. The minimum byte value in hexadecimal format is 80 and in decimal format is -128. The byte value of decimal -1 is ff in hexadecimal.

5 f Prints: 80000000,7fffffff An int is a 32 bit signed value. The left most bit is the sign bit. The sign bit is zero for positive numbers and one for negative numbers.

6 b d '\u0000' to '\uffff' 0 to 65535 A char is a 16 bit unsigned value; so none of the char values are negative and the minimum value is zero. The maximum value is 216 - 1.

7 b d f 0x0000 to 0xffff 0 to 0177777 0 to 65535 A char is a 16 bit unsigned value; so none of the char values are negative and the minimum value is zero. The maximum value is 216 - 1.

8 b c e 2 3 5 The maximum char value is 0xffff. The maximum byte value is 127 = 0x7f. The hex value 0xff is of type int, and the decimal value is 255. The maximum positive value of type byte is 127. The value 255 is beyond the range of type byte; so 255 can not be assigned to type byte without an explicit cast. The assignment expression b3 = (byte)0xff would assign the value -1 to variable b3.

## Literals

### Question 1

```
class MCZ11 {
 public static void main (String[] args) {
 char a = 'c'; // 1
 char b = 'r'; // 2
 char c = '\"'; // 3
 char d = 'b'; // 4
 char e = '\"'; // 5
 }
}
```

A compile-time error is generated at which line?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5
- f. None of the above

---

### Question 2

```
class MCZ27 {
 public static void main (String[] args) {
 char a = '\f'; // 1
 char b = '\n'; // 2
 char c = '\r'; // 3
 char d = '\l'; // 4
 char e = '\\\'; // 5
 } }
```

A compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. None of the above
- 

### Question 3

```
class MCZ28 {
 public static void main (String[] args) {
 char a = '\b'; // 1
 char b = '\d'; // 2
 char c = '\f'; // 3
 char d = '\t'; // 4
 char e = '\"'; // 5
 } }
```

A compile-time error is generated at which line?

- a. 1
- b. 2
- c. 3

- d. 4
  - e. 5
  - f. None of the above
- 

#### Question 4

```
class MCZ29 {
 public static void main (String[] args) {
 char a = '\a'; // 1
 char b = '\b'; // 2
 char c = '\f'; // 3
 char d = '\n'; // 4
 char e = '\r'; // 5
 } }
```

A compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. None of the above
- 

#### Question 5

```
class MCZ30 {
 public static void main (String[] args) {
 char a = '\t'; // 1
 char b = '\W'; // 2
 char c = '\?'; // 3
 char d = '\"'; // 4
 char e = '\"'; // 5
 } }
```

}}

A compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. None of the above
- 

### Question 6

```
class MCZ31 {
 public static void main (String[] args) {
 char a = '\t'; // 1
 char b = '\W'; // 2
 char c = '\"'; // 3
 char d = '\"'; // 4
 char e = '\?'; // 5
 } }
```

A compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. None of the above
- 

### Question 7

```
class MCZ13 {
 public static void main (String[] args) {
 String s = null;
 System.out.print(s);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints nothing.
  - b. Prints: null
  - c. Compile-time error
  - d. Run-time error
  - e. None of the above
- 

### Question 8

```
class MCZ15 {
 public static void main (String[] args) {
 float a = 1.1e1f; // 1
 float b = 1e-1F; // 2
 float c = .1e1f; // 3
 double d = .1d; // 4
 double e = 1D; // 5
 }
}
```

A compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. None of the above
-

### Question 9

```
class MCZ16 {
public static void main (String[] args) {
 float a = 1; // 1
 float b = 1L; // 2
 float c = 1F; // 3
 float d = 1.0; // 4
}}
```

A compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
- 

### Question 10

```
class MCZ17 {
public static void main (String[] args) {
 String a = "\n"; // 1
 String b = "\r"; // 2
 String c = "\u000a"; // 3 \u000a = new line
 String d = "\u000d"; // 4 \u000d = return
}}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4

---

### Question 11

```
class MCZ18 {
 public static void main (String[] args) {
 String a = "abcd"; // 1
 String b = "\u0041"; // 2
 String c = "\u0041"; // 3
 String d = "\uD7AF"; // 4
 System.out.print(a+b+c+d); // 5
 }
}
```

A compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. None of the above
- 

### Question 12

```
class MCZ20 {
 public static void main (String[] args) {
 // Insert code here.
 }
}
```

Which of the following lines can be inserted at the specified location without generating a compile-time error?

- a. String a = 'a';
- b. String b = 'abc';
- c. String c = '\u0041';

- d. String d = "\uabcd";
  - e. None of the above
- 

### Question 13

```
class MCZ21 {
 public static void main (String[] args) {
 // Insert code here.
 }
}
```

Which of the following lines can be inserted at the specified location without generating a compile-time error?

- a. char a = a;
  - b. char b = abc;
  - c. char c = \u0041;
  - d. char d = \abcd;
  - e. None of the above
- 

### Question 14

```
class MCZ24 {
 public static void main (String[] args) {
 char a = 061; // 1
 char b = '\61'; // 2
 char c = '\061'; // 3
 char d = 0x0031; // 4
 char e = '\u0031'; // 5
 System.out.print(""+a+b+c+d+e);
 }
}
```

A compile-time error is generated at which line?

- a. 1

- b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. None of the above
- 

### Question 15

```
class MCZ25 {
 public static void main (String[] args) {
 char a = '\7'; // 1
 char b = '\61'; // 2
 char c = '\062'; // 3
 char d = '\x7'; // 4
 char e = '\x41'; // 5
 System.out.print(""+a+b+c+d+e);
 } }
```

Compile-time errors are generated at which lines?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
- 

### Question 16

```
class MCZ19 {
 public static void main (String[] args) {
 String a1 = null; // 1
 String b1 = 'null'; // 2
 String c1 = "null"; // 3
```

```
String d1 = "null"; // 4
}}
```

A compile-time error is generated at which line?

- a. 1
  - b. **2**
  - c. 3
  - d. 4
  - e. None of the above
- 

### Question 17

```
class MCZ22 {
 public static void main (String[] args) {
 // Insert code here.
 }}
```

Which of the following lines will generate a compile-time error if inserted at the specified location?

- a. char a = 0x0041;
  - b. char b = '\u0041';
  - c. char c = 0101;
  - d. **char d = -1;**
  - e. char e = (char)-1;
  - f. None of the above
- 

### Question 18

```
class MCZ23 {
 public static void main (String[] args) {
 // Insert code here.
 }}
```

Which of the following lines can be inserted at the specified location without generating a compile-time error?

- a. boolean b1 = true;
- b. boolean b2 = TRUE;
- c. boolean b3 = 'true';
- d. boolean b4 = "TRUE";
- e. boolean b5 = 0;
- f. None of the above

1 a 1 The escape sequences are as follows: '\b' (backspace), '\f' (formfeed), '\n' (newline), '\r' (carriage return), '\t' (horizontal tab), '\\' (backslash), '\"' (double quote), '\'' (single quote). Yes, you must memorize the escape sequences! Just remember "big farms need red tractors".

2 d 4 The escape sequences are as follows: '\b' (backspace), '\f' (formfeed), '\n' (newline), '\r' (carriage return), '\t' (horizontal tab), '\\' (backslash), '\"' (double quote), '\'' (single quote). Yes, you must memorize the escape sequences! Just remember "big farms need red tractors".

3 b 2 The escape sequences are as follows: '\b' (backspace), '\f' (formfeed), '\n' (newline), '\r' (carriage return), '\t' (horizontal tab), '\\' (backslash), '\"' (double quote), '\'' (single quote). Yes, you must memorize the escape sequences! Just remember "big farms need red tractors".

4 a 1 The escape sequences are as follows: '\b' (backspace), '\f' (formfeed), '\n' (newline), '\r' (carriage return), '\t' (horizontal tab), '\\' (backslash), '\"' (double quote), '\'' (single quote). Yes, you must memorize the escape sequences! Just remember "big farms need red tractors".

5 c 3 The escape sequences are as follows: '\b' (backspace), '\f' (formfeed), '\n' (newline), '\r' (carriage return), '\t' (horizontal tab), '\\' (backslash), '\"' (double quote), '\'' (single quote). Yes, you must memorize the escape sequences! Just remember "big farms need red tractors".

6 e 5 The escape sequences are as follows: '\b' (backspace), '\f' (formfeed), '\n' (newline), '\r' (carriage return), '\t' (horizontal tab), '\\' (backslash), '\"' (double quote), '\'' (single quote). Yes, you must memorize the escape sequences! Just remember "big farms need red tractors".

7 b Prints: null The System.out.print method prints the word null if the argument is a String reference that is null.

8 f None of the above Floating-point literals are covered in section 3.10.2 of the JLS. A floating-point literal can begin with either a digit or a decimal point. Optionally, it can have a fractional part, an exponent part and a floating point suffix--f, F, d, or D.

9 d 4 The literal 1.0 is a double and can not be used to initialize a float without an explicit cast.

10 c d 3 4 The compiler interprets \u000a as a line terminator. The escape sequence \n should be used instead. Similarly, \u000d is interpreted as a line terminator. The escape sequence \r should be used instead.

11 f None of the above All of the declarations are legal. String b is a single quote followed by the letter A followed by another single quote. String c is the letter A. String d is the Unicode character that is represented by the hexadecimal value D7AF. String literals are covered in section 3.10.5 of the JLS.

12 e None of the above String literals are declared using double quotes, but all of the declarations here use single quotes.

13 e None of the above Unicode char literals are declared using single quotes, but none of the declarations here use single quotes. The declaration of char b, is also problematic, because it contains more than one char.

14 f None of the above All of the declarations are legal. The first three ( 061, '\61', '\061' ) are declared in octal format. The fourth (0x0031) is declared as a hexadecimal literal. The fifth ('\u0031') is a Unicode escape sequence.

15 d e 4 5 All of the escape sequences used in this question are defined for the C programming language. Those that are not also Java escape sequences result in a compile-time error. Java does not accept the hexadecimal escape sequences of the C programming language. However, Java does accept Unicode escapes (JLS 3.3).

16 b 2 The reference a1 is set to null. String b1 generates a compile-time error, because String literals must be enclosed by double quotes. String c1 is the word null. String d1 is a single quote followed by the word null followed by another single quote. String literals are covered in section 3.10.5 of the JLS.

17 d char d = -1; The assignment of -1 to char d generates a compile-time error, because the primitive char type is unsigned. A negative int can not be assigned to a char without an explicit cast. If the literal value -1 were cast to type char then the result would be \uffff.

18 a boolean b1 = true; There are two primitive boolean values: true and false. Both must be written with lower case letters. Although the C programming language accepts zero as a boolean value, the Java programming language does not.

## java.lang.Math

### Question 1

```
class SRC102 {
 public static void main (String[] args) {
 int i1 = Math.round(0.5f);
 int i2 = Math.round(1.5d);
 System.out.print(i1 + "," + i2);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1
- b. Prints: 0,2
- c. Prints: 1,1
- d. Prints: 1,2
- e. **Compile-time error**
- f. Run-time error
- g. None of the above

---

### Question 2

```
class SRC103 {
```

```
public static void main (String[] args) {
 int i1 = Math.round(0.5f);
 int i2 = Math.round(1.5f);
 System.out.print(i1 + "," + i2);
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,1
  - b. Prints: 0,2
  - c. Prints: 1,1
  - d. Prints: 1,2
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 3

```
class SRC104 {
 public static void main (String[] args) {
 System.out.print(Math.round(Float.NaN));
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: NaN
  - b. Prints: 0.0
  - c. Prints: 0
  - d. Compile-time error
  - e. Run-time error
  - f. None of the above
-

#### **Question 4**

```
class SRC105 {
 public static void main(String[] args) {
 double d1 = Math.ceil(0.5);
 double d2 = Math.ceil(1.5);
 System.out.print(d1 + "," + d2);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0.0,1.0
  - b. Prints: 0.0,2.0
  - c. Prints: 1.0,1.0
  - d. Prints: 1.0,2.0
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

#### **Question 5**

```
class SRC106 {
 public static void main(String[] args) {
 double d1 = Math.floor(0.5);
 double d2 = Math.floor(1.5);
 System.out.print(d1 + "," + d2);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0.0,1.0
- b. Prints: 0.0,2.0
- c. Prints: 1.0,1.0

- d. Prints: 1.0,2.0
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 6

```
class SRC107 {
 public static void main (String[] args) {
 int a = -1; long b = -2;
 float c = -3.0f; double d = -4.0;
 a = Math.abs(a); b = Math.abs(b);
 c = Math.abs(c); d = Math.abs(d);
 System.out.print(a+b+c+d);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 10.0
  - b. Compile-time error
  - c. Run-time error
  - d. None of the above
- 

### Question 7

```
class SRC109 {
 public static void main (String[] args) {
 short x3 = 0; short x4 = 1;
 short b1 = Math.min(x3,x4); // 1
 int c1 = Math.min(0,1); // 2
 long d1 = Math.min(0L,+1L); // 3
 System.out.print(b1+","+c1+","+d1);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0
  - b. **Compile-time error at 1**
  - c. Compile-time error at 2
  - d. Compile-time error at 3
  - e. Run-time error
  - f. None of the above
- 

### Question 8

```
class SRC110 {
 public static void main (String[] args) {
 byte x1 = 0; byte x2 = 1;
 byte a1 = Math.min(x1,x2); // 1
 int c1 = Math.min(0,1); // 2
 long d1 = Math.min(0L,+1L); // 3
 System.out.print(a1+"," +c1+"," +d1);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,0
  - b. **Compile-time error at 1**
  - c. Compile-time error at 2
  - d. Compile-time error at 3
  - e. Run-time error
  - f. None of the above
- 

### Question 9

```
class SRC111 {
```

```
static String m(byte i) {return "byte";}
static String m(short i) {return "short";}
static String m(char i) {return "char";}
static String m(int i) {return "int";}
static String m(double i) {return "double";}
public static void main (String[] args) {
 byte b = 0; short s = 0; char c = 0;
 System.out.print(m(Math.min(b,b))+",");
 System.out.print(m(Math.min(s,s))+",");
 System.out.print(m(Math.min(c,c)));
}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: byte,byte,byte
  - b. Prints: short,short,short
  - c. Prints: int,int,int
  - d. Prints: byte,short,int
  - e. Prints: short,short,int
  - f. Prints: byte,short,char
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

### Question 10

```
class SRC112 {
 static String m(byte i) {return "byte";}
 static String m(int i) {return "int";}
 static String m(long i) {return "long";}
 static String m(double i) {return "double";}
 public static void main (String[] args) {
 byte b = 0;
 System.out.print(m(Math.min(b,b))+",");
 }
}
```

```
System.out.print(m(Math.min(b,1))+",");
System.out.print(m(Math.min(b,1L)));
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: byte,byte,byte
  - b. Prints: byte,int,long
  - c. Prints: int,int,int
  - d. Prints: int,int,long
  - e. Prints: long,long,long
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### Question 11

```
class SRC113 {
 static String m(byte i) {return "byte";}
 static String m(short i) {return "short";}
 static String m(char i) {return "char";}
 static String m(int i) {return "int";}
 static String m(long i) {return "long";}
 static String m(float i) {return "float";}
 static String m(double i) {return "double";}
 public static void main (String[] args) {
 byte b = 0; short s = 0; char c = 0;
 System.out.print(m(Math.min(b,1L))+",");
 System.out.print(m(Math.min(s,1.0f))+",");
 System.out.print(m(Math.min(c,1.0)));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: int,int,int
  - b. Prints: byte,short,char
  - c. Prints: long,float,double
  - d. Prints: double,double,double
  - e. Prints: long,long,long
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### Question 12

```
class SRC114 {
 static String m(int i) {return "int";}
 static String m(long i) {return "long";}
 static String m(float i) {return "float";}
 static String m(double i) {return "double";}
 public static void main (String[] args) {
 System.out.print(m(Math.random()));
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: int
  - b. Prints: long
  - c. Prints: float
  - d. Prints: double
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 13

```
class SRC115 {
 static String m(int i) {return "int";}
 static String m(long i) {return "long";}
 static String m(float i) {return "float";}
 static String m(double i) {return "double";}
 public static void main (String[] args) {
 long seed = 1L;
 System.out.print(m(Math.random(seed)));
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: int
  - b. Prints: long
  - c. Prints: float
  - d. Prints: double
  - e. **Compile-time error**
  - f. Run-time error
  - g. None of the above
- 

#### Question 14

```
class SRC116 {
 static String m(int i) {return "int";}
 static String m(long i) {return "long";}
 static String m(float i) {return "float";}
 static String m(double i) {return "double";}
 public static void main (String[] args) {
 System.out.print(m(Math.sin(0.0f))+",";
 System.out.print(m(Math.cos(0.0f))+",";
 System.out.print(m(Math.tan(0.0f)));
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: int,int,int
  - b. Prints: long,long,long
  - c. Prints: float,float,float
  - d. Prints: double,double,double
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 15

```
class SRC117 {
 public static void main (String[] args) {
 double d1 = Math.random();
 boolean b1 = (d1 < 0.0), b2 = (d1 <= 0.0), b3 = (d1 == 0.0);
 boolean b4 = (d1 >= 0.0), b5 = (d1 < 1.0), b6 = (d1 <= 1.0);
 boolean b7 = (d1 == 1.0), b8 = (d1 >= 1.0), b9 = (d1 > 1.0);
 }
}
```

Which of the boolean variables will never be initialized to the value true?

- a. b1
  - b. b2
  - c. b3
  - d. b4
  - e. b5
  - f. b6
  - g. b7
  - h. b8
  - i. b9
- 

### Question 16

Which method of the `java.lang.Math` class is not static?

- a. `abs`
  - b. `ceil`
  - c. `floor`
  - d. `max`
  - e. `min`
  - f. `random`
  - g. `round`
  - h. `sin`
  - i. `cos`
  - j. `tan`
  - k. `sqrt`
  - l. **None of the above**
- 

### Question 17

Some of the following method names are overloaded and some are not. Which of the following method names are overloaded such that for each of the following four primitive types `int`, `long`, `float` and `double` there is at least one corresponding method that returns that type?

- a. **Abs**
- b. `ceil`
- c. `floor`
- d. **max**
- e. **Min**
- f. `random`
- g. `round`
- h. `sin`
- i. `cos`
- j. `tan`
- k. `sqrt`

---

### Question 18

What is the result of attempting to compile and run the program?

- a. Prints: DDDDDD
  - b. Prints: LLLLLL
  - c. Prints: DLLLDD
  - d. Prints: DIDLDD
  - e. Prints: DLDLDD
  - f. Prints: DDDLDD
  - g. Prints: DIDIDD
  - h. None of the above
- 

### Question 19

```
class SRC119 {
 static String m1(int i) {return "I";}
 static String m1(long i) {return "L";}
 static String m1(float i) {return "F";}
 static String m1(double i) {return "D";}
 public static void main (String[] args) {
 System.out.print(m1(Math.floor(1.0f)) + m1(Math.floor(1.0d)));
 System.out.print(m1(Math.ceil(1.0f)) + m1(Math.ceil(1.0d)));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: IIII
- b. Prints: ILIL

- c. Prints: LLLL
- d. Prints: FDFD
- e. Prints: DDDD
- f. None of the above

1 e Compile-time error There are two versions of the Math.round method. One accepts an argument of type float; the return type is int. The other accepts an argument of type double; the return type is long. A compile-time error is generated here due to the attempt to assign the long return value to the int variable, i2.

2 d Prints: 1,2 The Math.round method returns the floor of the argument value plus 0.5. If the argument is of type float, then the return type is int. If the argument is of type double, then the return type is long.

3 c Prints: 0 If NaN is the argument passed to Math.round, then the return value is zero. If the argument type is float, then the return type is int. If the argument type is double, then the return type is long.

4 d Prints: 1.0,2.0 The Math.ceil method name is not overloaded. The Math.ceil method accepts an argument of type double and returns a double. The returned value is the smallest whole number that is greater than or equal to the argument.

5 a Prints: 0.0,1.0 The Math.floor method name is not overloaded. The Math.floor method accepts an argument of type double and returns a double. The returned value is the largest whole number that is smaller than or equal to the argument.

6 a Prints: 10.0 The Math class declares four versions of the abs method; each declares one parameter. The parameter can be of type int, long, float or double. The return type is the same as the argument type. At run-time, the argument might not match the parameter type; so the argument might require an implicit conversion to an acceptable type. If the argument is of type byte, short or char, then it will be promoted to type int.

7 b Compile-time error at 1 At line 1, the invocation of the Math.min method produces a return value of type int. The variable b1 is of type short; so the assignment expression results in a compile-time error. The Math class declares four versions of the min method; each declares a pair of parameters of the same type. The parameter pair can be of type int, long, float or double. The return type is the same as the argument types. At run-time, the arguments might not match the declared parameter types; so one argument or both might require an implicit conversion to an acceptable type. If both arguments are of type byte, short or char, then both will be promoted to type int. If only one argument is of type byte, short or char, then it will be promoted to the type of the other argument. If both arguments are of type int, long, float or double but the types differ, then a primitive widening conversion will be applied to one of the two arguments.

8 b Compile-time error at 1 At line 1, the invocation of the Math.min method produces a return value of type int. The variable a1 is of type byte; so the assignment expression results in a compile-time error. The Math class declares four versions of the min method; each declares a pair of parameters of the same type. The parameter pair can be of type int, long, float or double. The return type is the same as the argument types. At run-time, the arguments might not match the declared parameter types; so one argument or both might require an implicit conversion to an acceptable type. If both arguments are of type byte, short or char, then both will be promoted to type int. If only one argument is of type byte, short or char, then it will be promoted to the type of the other argument. If both arguments are of type int, long, float or double but the types differ, then a primitive widening conversion will be applied to one of the two arguments.

9 c Prints: int,int,int The Math class declares four versions of the min method; each declares a pair of parameters of the same type. The parameter pair can be of type int, long, float or double. The return type is the same as the argument types. At run-time, the arguments might not match the declared parameter types; so one argument or both might require an implicit conversion to an acceptable type. If both arguments are of type byte, short or char, then both will be promoted to type int. If only one argument is of type byte, short or char, then it will be promoted to the type of the other argument. If both arguments are of type int, long, float or double but the types differ, then a primitive widening conversion will be applied to one of the two arguments.

10 d Prints: int,int,long The Math class declares four versions of the min method; each declares a pair of parameters of the same type. The parameter pair can be of type int, long, float or double. The return type is the same as the argument types. At run-time, the arguments might not match the declared parameter types; so

one argument or both might require an implicit conversion to an acceptable type. If both arguments are of type byte, short or char, then both will be promoted to type int. If only one argument is of type byte, short or char, then it will be promoted to the type of the other argument. If both arguments are of type int, long, float or double but the types differ, then a primitive widening conversion will be applied to one of the two arguments.

11 c Prints: long,float,double The Math class declares four versions of the min method; each declares a pair of parameters of the same type. The parameter pair can be of type int, long, float or double. The return type is the same as the argument types. At run-time, the arguments might not match the declared parameter types; so one argument or both might require an implicit conversion to an acceptable type. If both arguments are of type byte, short or char, then both will be promoted to type int. If only one argument is of type byte, short or char, then it will be promoted to the type of the other argument. If both arguments are of type int, long, float or double but the types differ, then a primitive widening conversion will be applied to one of the two arguments.

12 d Prints: double The Math.random method returns a value of type double. The range of values is greater than or equal to zero and less than one.

13 e Compile-time error The Math.random method does not accept a seed value. If your application requires a seed, then use java.util.Random.

14 d Prints: double,double,double The Math.sin, Math.cos and Math.tan methods return a value of type double.

15 a g h i b1 b7 b8 b9 The Math.random method returns a value that is equal to or greater than zero and less than 1.0.

16 l None of the above All methods of the java.lang.Math class are static.

17 a d e abs max min

18 f Prints: DDDLD The round method does not return either of the two floating point primitive types, float or double. The ceil, sin and sqrt methods return only a value of type double. The abs and max methods are able to return an int, long, float or double depending on the argument type.

19 e Prints: DDDD The floor and ceil methods are not overloaded. There is only one version of each method. The parameter type is double, and the return type is double

## Question 1

```
class SRC120 {
 static String m1(int i) {return "I";}
 static String m1(long i) {return "L";}
 static String m1(float i) {return "F";}
 static String m1(double i) {return "D";}
 public static void main (String[] args) {
 System.out.print(m1(Math.abs(1.0f)) + m1(Math.abs(1.0d)));
 System.out.print(m1(Math.sqrt(1.0f)) + m1(Math.sqrt(1.0d)));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: IIII
- b. Prints: ILIL
- c. Prints: LLLL
- d. Prints: FDFD
- e. Prints: FDDD

- f. Prints: DDFD
  - g. Prints: DDDD
  - h. None of the above
- 

## Question 2

```
class SRC121 {
 static String m1(byte i) {return "B";}
 static String m1(char i) {return "C";}
 static String m1(int i) {return "I";}
 static String m1(long i) {return "L";}
 static String m1(float i) {return "F";}
 static String m1(double i) {return "D";}
 public static void main (String[] args) {
 System.out.print(m1(Math.min((byte)1,(byte)2)));
 System.out.print(m1(Math.min('A','B')));
 System.out.print(m1(Math.min(1,2)));
 System.out.print(m1(Math.min((long)1,2)));
 System.out.print(m1(Math.min(1.0f,1)));
 System.out.print(m1(Math.min(1.0,1)));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: DDDDDDD
  - b. Prints: FFFFFFD
  - c. Prints: FFFDFD
  - d. Prints: IIILFD
  - e. Prints: IILLDD
  - f. Prints: IIIIDD
  - g. Prints: BCILFD
  - h. Prints: CCILFD
  - i. None of the above
-

### Question 3

```
class SRC122 {
 static String m1(int i) {return "T";}
 static String m1(long i) {return "L";}
 static String m1(float i) {return "F";}
 static String m1(double i) {return "D";}
 public static void main (String[] args) {
 System.out.print(m1(Math.abs(1.0f)) + m1(Math.abs(1.0)));
 System.out.print(m1(Math.round(1.0f)) + m1(Math.round(1.0)));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: IIII
  - b. Prints: LLLL
  - c. Prints: FFFF
  - d. Prints: DDDD
  - e. Prints: FDFD
  - f. Prints: ILIL
  - g. Prints: FDIL
  - h. Prints: ILFD
  - i. None of the above
- 

### Question 4

Which of the following methods of the java.lang.Math class accepts an argument of type float or double, but has a return type of type int or long respectively.

- a. abs
- b. ceil
- c. floor
- d. max
- e. min
- f. random
- g. round
- h. sin
- i. cos

- j. tan
  - k. sqrt
  - l. None of the above
- 

#### Question 5

Which of the following methods of the java.lang.Math class declares a non-primitive argument type?

- a. abs
  - b. ceil
  - c. floor
  - d. max
  - e. min
  - f. random
  - g. round
  - h. sin
  - i. cos
  - j. tan
  - k. sqrt
  - l. None of the above
- 

#### Question 6

Which of the following methods of the java.lang.Math class declares a non-primitive return type?

- a. abs
- b. ceil
- c. floor
- d. max
- e. min
- f. random
- g. round
- h. sin
- i. cos
- j. tan

- k. sqrt
  - l. None of the above
- 

#### Question 7

Which of the following statements is true in terms of the java.lang.Math.round method?

- a. The returned value is of a floating-point primitive type.
  - b. The returned value is always of type int.
  - c. The returned value is always of type long.
  - d. The returned value is of type long only if the argument is of type long.
  - e. The returned value is of type long only if the argument is of type double.
  - f. None of the above
- 

#### Question 8

Which of the following statements is true in terms of the value returned by the java.lang.Math.round method?

- a. Rounds the argument to the nearest whole number. If the result is equally close to two whole numbers, then the even one is returned.
  - b. Rounds the argument to the nearest whole number. If the result is equally close to two whole numbers, then the odd one is returned.
  - c. Adds 0.5 to the argument and takes the floor of the result.
  - d. Adds 0.5 to the argument and takes the ceil of the result.
  - e. None of the above
- 

#### Question 9

Suppose that the java.lang.Math.round method is invoked with an argument of type float, and the result exceeds the range of type int. Which of the following occurs?

- a. The result is implicitly widened to type long.
- b. The returned value is Float.NaN.
- c. A run-time exception is thrown.
- d. The bits that overflow are ignored and no exception is thrown.
- e. The returned value is zero but no exception is thrown.

f. None of the above

---

#### Question 10

Which of the following statements is true in terms of the java.lang.Math.sqrt method?

- a. It is not overloaded.
  - b. Returns a float if the argument type is float.
  - c. If the argument is negative, then the returned value is the square root of the absolute value of the argument.
  - d. Throws an ArithmeticException if the argument is negative.
  - e. None of the above
- 

#### Question 11

Which of the following statements are true in terms of the java.lang.Math.sin method?

- a. It is not overloaded.
  - b. Returns a float if the argument type is float.
  - c. The argument is an angle measured in radians.
  - d. The argument is an angle measured in degrees.
  - e. Throws an ArithmeticException if the argument is greater than 360.
- 

#### Question 12

Which of the following statements is true in terms of the java.lang.Math.random method?

- a. The random method name is overloaded.
  - b. The argument type is a double that represents a seed value.
  - c. Throws an ArithmeticException if the seed value is negative.
  - d. The returned value is always greater than zero and less than or equal to one.
  - e. None of the above
-

### Question 13

Which of the following statements is true in terms of the java.lang.Math.floor method?

- a. Four overloaded versions of floor exist.
  - b. An ArithmeticException is declared in the throws clause.
  - c. The type of the return value depends on the argument type.
  - d. The returned value is always of an integral primitive type.
  - e. Returns the largest whole number that is less than or equal to the argument value.
  - f. Returns the smallest whole number that is greater than or equal to the argument value.
  - g. None of the above
- 

### Question 14

Which of the following statements is true in terms of the java.lang.Math.ceil method?

- a. Four overloaded versions of ceil exist.
  - b. An ArithmeticException is declared in the throws clause.
  - c. The type of the return value depends on the argument type.
  - d. The returned value is always of an integral primitive type.
  - e. Returns the largest whole number that is less than or equal to the argument value.
  - f. Returns the smallest whole number that is greater than or equal to the argument value.
  - g. None of the above
- 

### Question 15

Which of the following statements are true in terms of the java.lang.Math.abs method?

- a. Four overloaded versions of abs exist.
  - b. An ArithmeticException is declared in the throws clause.
  - c. The type of the return value depends on the argument type.
  - d. The returned value is always of a floating-point primitive type.
  - e. If the argument is a negative integral value, then the returned value is always positive.
  - f. If the argument is positive, then the argument is returned.
-

### Question 16

The java.lang.Math.abs method can return which of the following?

- a. Negative infinity
  - b. Positive infinity
  - c. NaN
  - d. Short.MIN\_VALUE
  - e. Integer.MIN\_VALUE
  - f. Long.MIN\_VALUE
  - g. Negative zero
  - h. Positive zero
- 

### Question 17

```
class SRC123 {
 public static void main (String[] args) {
 System.out.print(Math.floor(-3.6) + "," + Math.ceil(-3.6)+ ",");
 System.out.print(Math.floor(3.6) + "," + Math.ceil(3.6));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: -3.0,-4.0,3.0,4.0
  - b. Prints: -3.0,-4.0,4.0,3.0
  - c. Prints: 4.0,-3.0,3.0,4.0
  - d. Prints: -4.0,-3.0,4.0,3.0
  - e. Prints: -4.0,-4.0,3.0,3.0
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### Question 18

```
class SRC124 {
```

```
public static void main (String[] args) {
 System.out.print(Math.round(-3.6) + "," + Math.round(-3.4)+ ",");
 System.out.print(Math.round(3.4) + "," + Math.round(3.6));
}
```

What is the result of attempting to compile and run the program?

- a. Prints: -3.0,-4.0,3.0,4.0
  - b. Prints: -3.0,-4.0,4.0,3.0
  - c. Prints: -4.0,-3.0,3.0,4.0
  - d. Prints: -4.0,-3.0,4.0,3.0
  - e. Prints: -4.0,-4.0,3.0,3.0
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

Question 19

```
class SRC125 {
 public static void main (String[] args) {
 System.out.print(Math.round(-3.6) + Math.round(3.6) + ",");
 System.out.print(Math.round(-3.4) + Math.round(3.4));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0
- b. Prints: 0,-1
- c. Prints: -1,0
- d. Prints: -1,-1
- e. Prints: 0,1
- f. Prints: 1,0
- g. Prints: 1,1
- h. Compile-time error
- i. Run-time error
- j. None of the above

- 1 e Prints: FDDD The method name abs is overloaded. There are versions that accept one argument of type int, long, float or double. The type of the return value is the same as the argument type. The sqrt method is not overloaded. The parameter type is double and the return type is double.
- 2 d Prints: IIILFD There are four overloaded versions of the Math.min method with versions that declare one parameter of type int, long, float or double. The return type is the same as the parameter type. Arguments of type byte, char and short are promoted to type int.
- 3 g Prints: FDIL The method name abs is overloaded. There are versions that accept one argument of type int, long, float or double. The type of the return value is the same as the argument type. There are two versions of the Math.round method. One accepts an argument of type float; the return type is int. The other accepts an argument of type double; the return type is long.
- 4 g round
- 5 l None of the above
- 6 l None of the above
- 7 e The returned value is of type long only if the argument is of type double.
- 8 c Adds 0.5 to the argument and takes the floor of the result.
- 9 f None of the above If the java.lang.Math.round method is invoked with an argument of the float type and the result exceeds the range of type int, then the result is Integer.MAX\_VALUE.
- 10 a It is not overloaded. The argument and returned value are both of type double. If the argument is negative, then the returned value is NaN.
- 11 a c It is not overloaded. The argument is an angle measured in radians. The argument is of type double and represents an angle measured in radians. The returned value is of type double. The method does not throw any exceptions.
- 12 e None of the above The random method is not overloaded, does not declare any parameters and does not throw any exceptions. The returned value is greater than or equal to zero and is less than but not equal to one.
- 13 e Returns the largest whole number that is less than or equal to the argument value. The floor method is not overloaded, and declares only one parameter of type double. The return value is always of type double. The floor method does not declare any exceptions.
- 14 f Returns the smallest whole number that is greater than or equal to the argument value. The ceil method is not overloaded, and declares only one parameter of type double. The return value is always of type double. The ceil method does not declare any exceptions.
- 15 a c f Four overloaded versions of abs exist. The type of the return value depends on the argument type. If the argument is positive, then the argument is returned. The method name abs is overloaded. There are versions that accept one argument of type int, long, float or double. The type of the return value is the same as the argument type. No exceptions are declared. If the argument is a negative integral value, then the returned value is the two's complement of the argument. The magnitude of Integer.MIN\_VALUE is one greater than the magnitude of Integer.MAX\_VALUE; therefore, the absolute value of Integer.MIN\_VALUE exceeds the range of Integer.MAX\_VALUE. Due to the limited range of type int, the two's complement of Integer.MIN\_VALUE is Integer.MIN\_VALUE. For that reason, the Math.abs method returns Integer.MIN\_VALUE when the argument is Integer.MIN\_VALUE.
- 16 b c e f h Positive infinity NaN Integer.MIN\_VALUE Long.MIN\_VALUE Positive zero The method name abs is overloaded. There are versions that accept one argument of type int, long, float or double. The type of the return value is the same as the argument type. The result is positive infinity if the argument is negative infinity. The result is positive zero if the argument is negative zero. The result is NaN if the argument is NaN. If the argument is a negative integral value, then the returned value is the two's complement of the argument. The magnitude of Integer.MIN\_VALUE is one greater than the magnitude of Integer.MAX\_VALUE; therefore, the absolute value of Integer.MIN\_VALUE exceeds the range of Integer.MAX\_VALUE. Due to the limited range of type int, the two's complement of Integer.MIN\_VALUE is Integer.MIN\_VALUE. For that reason, the Math.abs method returns Integer.MIN\_VALUE when the argument is Integer.MIN\_VALUE. The negation of Short.MIN\_VALUE is well within the range of type int; so Short.MIN\_VALUE is never returned by the Math.abs method.

17 c Prints: -4.0,-3.0,3.0,4.0 The Math.floor method returns a primitive of type double that is equal to the largest integral value that is smaller than or equal to the argument. For example, -4.0 is the floor of -3.6, because -4.0 is the largest floating-point value that is equal to an integral value that is closer to negative infinity than -3.6. Similarly, the Math.ceil method returns a primitive of type double that is equal to an integral value, and is smaller than any other such value that is larger than the argument. For example, -3.0 is the ceil of -3.6. The term "larger" refers to values that are closer to positive infinity. For example, -3 is larger than -4, because -3 is closer to positive infinity.

18 h None of the above The math.round method name is overloaded. If the argument is of type double, then the return value is of type long. If the argument is of type float, then the return value is of type int. Both return types are integral primitive types; so a decimal point should not be included in the output. The actual output is -4,-3,3,4.

19 a Prints: 0,0 The math.round method adds 0.5 to the argument, and then takes the floor of the sum. The result of Math.round with an argument of -3.6 is -4. The result of Math.round with an argument of 3.6 is 4.0. The sum of the two results is zero.

## Strings

### Question 1

Which of the following are not methods of the java.lang.String class?

- a. Append
  - b. Concat
  - c. Delete
  - d. Insert
  - e. Replace
  - f. Substring
  - g. ValueOf
- 

### Question 2

Which of these methods modify the contents of the String instance on which it is invoked?

- a. Append
- b. Concat
- c. Delete
- d. Insert

- e. Replace
  - f. substring
  - g. valueOf
  - h. None of the above.
- 

### Question 3

Which of these methods is a static member of the java.lang.String class?

- a. append
  - b. concat
  - c. delete
  - d. insert
  - e. replace
  - f. substring
  - g. ValueOf
  - h. None of the above.
- 

### Question 4

```
class MWC106 {
 static void m1(String s) {
 s = s.trim(); s = s.concat("D");
 }
 public static void main(String[] s) {
 String s1 = "A", s2 = " B ", s3 = "C";
 m1(s2);
 System.out.print(s1 + s2 + s3);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: ABC
  - b. Prints: A B C
  - c. Prints: ABCD
  - d. Prints: ABDC
  - e. Prints: A B CD
  - f. Prints: A B DC
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

### Question 5

```
class MWC107 {
 public static void main(String[] s) {
 String s1 = "A", s2 = " B ", s3 = "C";
 s2.trim(); s3.concat("D");
 System.out.print(s1 + s2 + s3);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: ABC
  - b. Prints: A B C
  - c. Prints: ABCD
  - d. Prints: ABDC
  - e. Prints: A B CD
  - f. Prints: A B DC
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
-

## Question 6

```
class MWC108 {
 public static void main(String[] s) {
 String s1 = "A", s2 = " B ", s3 = "C";
 s2.trim(); s3.append("D");
 System.out.print(s1 + s2 + s3);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: ABC
  - b. Prints: A B C
  - c. Prints: ABCD
  - d. Prints: ABDC
  - e. Prints: A B CD
  - f. Prints: A B DC
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

## Question 7

```
class MWC110 {
 static void m1(String s1, String s2) {
 s1.insert(1,"D"); s1 = s1 + s2;
 }
 public static void main(String[] s) {
 String s1 = "A", s2 = "B";
 m1(s1,s2);
 System.out.print(s1 + s2);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AB
  - b. Prints: ABB
  - c. Prints: ADB
  - d. Prints: ADBB
  - e. **Compile-time error**
  - f. Run-time error
  - g. None of the above
- 

### Question 8

```
class MWC111 {
 static void m1(String s1) {
 s1.replace('A','Y'); System.out.print(s1);
 }
 static void m2(String s1) {
 s1 = s1.replace('A','Z'); System.out.print(s1);
 }
 public static void main(String[] s) {
 String s1 = "A"; m1(s1); m2(s1);
 System.out.print(s1);
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
  - b. Prints: YZA
  - c. Prints: YAA
  - d. **Prints: AZA**
  - e. Prints: AZZ
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
-

### Question 9

```
class MWC112 {
 static void m1(String s1) {
 s1 = s1.toUpperCase(); System.out.print(s1);
 }
 static void m2(String s1) {
 s1.toLowerCase(); System.out.print(s1);
 }
 public static void main(String[] s) {
 String s1 = "Ab";
 m1(s1); m2(s1);
 System.out.print(s1);
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: AbAbAb
  - b. Prints: ABabab
  - c. Prints: ABabAb
  - d. Prints: ABAbAb**
  - e. Prints: Ababab
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### Question 10

```
class MWC113 {
 public static void main(String[] s) {
 String s1 = "1", s2 = "2", s3 = s1 + s2;
 s1 += s2; s3 += s1;
 System.out.println(s3);
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: 3
  - b. Prints: 6
  - c. Prints: 33
  - d. Prints: 1212
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 11

```
class MWC114 {
 public static void main(String[] s) {
 String s1 = new String("ABCDEFG"), s2 = new String("EFGHIJ");
 String s3 = s1.substring(4,7), s4 = s2.substring(0,3);
 System.out.println((s3 == s4) + "," + (s3 + s4).equals(s4 + s3));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
  - b. Prints: false,true
  - c. Prints: true,false
  - d. Prints: true,true
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 12

```
class MWC115 {
```

```
public static void main(String[] s) {
 String s1 = new String("ABCDEFG"), s2 = s1.substring(0,3);
 String s3 = s1.substring(4,6); char c1 = s1.charAt(3);
 System.out.println(s2.concat(String.valueOf(c1)).concat(s3));
}
```

What is the result of attempting to compile and run the program?

- a. Prints: "ABCDEFG"
  - b. Prints: "ABCEFG"
  - c. Prints: "ABCDEF
  - d. Prints: "ABCDEF"
  - e. Prints: "ABCEF"
  - f. Prints: "ABCDEF"
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

### Question 13

```
class MWC118 {
 public static void main(String[] args) {
 byte[] b = {'a', 'b', 'c'};
 char[] c = {'a', 'b', 'c'};
 String s = "abc";
 StringBuffer sb = new StringBuffer("abc");
 byte b2 = 'a';
 char c2 = 'a';
 String s1 = new String(b); // 1
 String s2 = new String(c); // 2
 String s3 = new String(s); // 3
 String s4 = new String(sb); // 4
 String s5 = new String(b2); // 5
 String s6 = new String(c2); // 6
```

}}

Compile-time errors are generated at which lines?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. 6
- 

#### Question 14

```
class MWC101 {
 public static void main(String[] args) {
 String s1 = "A", s2 = "a", s3 = "b";
 s1.toLowerCase(); s3.replace('b','a');
 System.out.print((s1.equals(s2)) + "," + (s2.equals(s3)));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
  - b. Prints: false,true
  - c. Prints: true,false
  - d. Prints: true,true
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

#### Question 15

```
class MWC102 {
 public static void main (String[] args) {
 String s1 = "ABCDE";
 System.out.print(s1.substring(1,2)+s1.substring(3));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AABC
  - b. Prints: ACDE
  - c. Prints: ABABC
  - d. Prints: ABCDE
  - e. Prints: BABCD
  - f. Prints: **BDE**
  - g. Prints: BCABCD
  - h. Prints: BCDE
  - i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

### Question 16

```
class MWC104 {
 public static void main (String[] args) {
 char[] c = {'a','b','c','d'};
 String s1 = new String(c);
 boolean b = true;
 for (int i = 0; i < s1.length; i++) {
 b &= (c[i] == s1.charAt(i));
 }
 System.out.print(b);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false
  - b. Prints: true
  - c. **Compile-time error**
  - d. Run-time error
  - e. None of the above
- 

### Question 17

```
class MWC109 {
 public static void main(String args[]) {
 String a = "A", b = "B", c = a+b, d = a+b;
 System.out.print(((a+b)==(a+b)) + ",");
 System.out.print((c==d) + ",");
 System.out.print(c.equals(d));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
  - b. **Prints: false,false,true**
  - c. Prints: false,true,false
  - d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
- 

### Question 18

```
class MWC116 {
 public static void main(String[] args) {
 String s1 = " B C D E ";
 }
}
```

```
System.out.print('A' + s1.trim() + 'F');
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: ABCDEF
  - b. Prints: A B C D E F
  - c. Prints: A BCDEF
  - d. Prints: ABCDE F
  - e. Prints: AB C D EF
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### Question 19

```
class MWC117 {
 public static void main (String[] args) {
 System.out.print(String.valueOf(1) + String.valueOf(2));
 String s1 = "S1";
 String s2 = s1.toString();
 System.out.print(", " + (s1==s2));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 3,false
- b. Prints: 3,true
- c. Prints: 12,false
- d. Prints: 12,true
- e. Compile-time error
- f. Run-time error
- g. None of the above

1.a c d .append delete insert .The StringBuffer class has methods named append, delete and insert, but the String class does not. A typical trick question will attempt to invoke StringBuffer methods on a String instance.

2.h .None of the above. .Strings are immutable. No method of the of a String class will change the contents of a String instance. Some String methods such as concat, replace, substring and trim will return a new String with the desired modifications. The StringBuffer methods append, delete and insert are not members of the java.lang.String class. A typical trick question will attempt to invoke StringBuffer methods on a String instance.

3.g .valueOf .The valueOf method is static. It returns a String representation of the argument. The StringBuffer methods append, delete and insert are not members of the java.lang.String class. A typical trick question will attempt to invoke StringBuffer methods on a String instance.

4.b .Prints: A B C .The String instance referenced by s2 is passed to the m1 method by passing the value of the reference. The reference value used in method m1 is a local copy of the reference. If the local copy used in method m1 is changed, then the original reference variable in the main method remains unchanged.

5.b .Prints: A B C .Strings are immutable. No method will change the contents of a String instance. Some String methods such as concat, replace, substring and trim will return a new String instance with the desired modifications. In this case, the String instances returned by the methods trim and concat are ignored.

6.g .Compile-time error .The StringBuffer class has an append method, but the String class does not. A compile-time error is generated due to the attempt to invoke the append method on an instance of type String.

7.e .Compile-time error .The StringBuffer class has an insert method, but the String class does not. A compile-time error is generated due to the attempt to invoke the insert method on an instance of type String.

8.d .Prints: AZA .Instances of type String are immutable. In method m1, the replace method returns a new instance of type String that contains the value Y, but the String instance referenced by s1 remains unchanged. The original value, A, is printed in method m1. In method m2, the replace method returns a new instance of type String that contains the value Z, and a reference to the new instance is assigned to reference variable s1. The new value, Z, is printed in method m2. In the main method, a copy of the reference value contained by the reference variable s1 is passed as an argument to methods m1 and m2. Since String instances are immutable, methods m1 and m2 can not change the original String instance that is declared in the main method. Since references are passed by value, methods m1 and m2 can not change the reference variable declared in the main method. Regardless of anything that happens in methods m1 and m2, the reference variable s1 that is declared in the main method will continue to reference the original String instance that contains the value A.

9.d .Prints: ABAbAb .Instances of type String are immutable. In method m1, the toUpperCase method returns a new instance of type String that contains the value AB, and a reference to the new instance is assigned to reference variable s1. The new value, AB, is printed in method m1. In method m2, the toLowerCase method returns a new instance of type String that contains the value ab, but the String instance referenced by s1 remains unchanged. The original value, Ab, is printed in method m2. In the main method, a copy of the reference value contained by the reference variable s1 is passed as an argument to methods m1 and m2. Since String instances are immutable, methods m1 and m2 can not change the original String instance that is declared in the main method. Since references are passed by value, methods m1 and m2 can not change the reference variable declared in the main method. Regardless of anything that happens in methods m1 and m2, the reference variable s1 that is declared in the main method will continue to reference the original String instance that contains the value Ab.

10.d .Prints: 1212 .The reference variable s3 is initialized with a reference to an instance of type String containing the value "12". The expression s1 += s2 is equivalent to the expression s1 = s1 + s2. Further simplification produces s1 = "1" + "2" = "12". The expression s3 += s1 is equivalent to the expression s3 = "12" + "12" = "1212".

11.b .Prints: false,true .The reference variable s1 is initialized with a reference to an instance of type String containing the value "ABCDEFG". The reference variable s2 is initialized with a reference to an instance of type String containing the value "EFGHIJ". The expression s3 = s1.substring(4,7) initializes the

reference variable s3 with a reference to a unique instance of type String containing the value "EFG". The expression s4 = s2.substring(0,3) initializes the reference variable s4 with a reference to a unique instance of type String containing the value "EFG". The expression s3 == s4 compares two unique reference values and produces the value false even though s3 and s4 reference two String instances that contain the same value, "EFG". The expression s3 + s4 produces a unique instance of type String containing the value "EFGEFG". Similarly, the expression s4 + s3 produces a unique instance of type String containing the value "EFGEFG". The expression (s3 + s4).equals(s4 + s3) compares the contents of two unique instances of type String that contain the value "EFGEFG". The result of the comparison is true.

12.d .Prints: "ABCDEF" .The reference variable s1 is initialized with a reference to an instance of type String containing the value "ABCDEFG". The expression s2 = s1.substring(0,3) initializes the reference variable s2 with a reference to a unique instance of type String containing the value "ABC". The expression s3 = s1.substring(4,6) creates a unique instance of type String containing the value "EF". The expression c1 = s1.charAt(3) initializes the primitive variable c1 with the value 'D'. The expression String.valueOf(c1) invokes the static valueOf method with an argument of type primitive char and value 'D'. The valueOf method creates a new instance of type String. The value contained by the new instance is "D". The expression s2.concat(String.valueOf(c1)) invokes the concat method on the instance of type String referenced by the variable s2. The instance referenced by s2 contains the value "ABC". The value contained by the argument is "D". The result of the concatenation operation is a new instance of type String containing the value "ABCD". The expression s2.concat(String.valueOf(c1)).concat(s3) invokes the concat method on the previously created instance containing the value "ABCD". The instance referenced by the argument s3 contains the value "EF". The result of the concatenation operation is a new instance of type String containing the value "ABCDEF".

13.e f .5 6 .The overloaded constructors for the String class accept a parameter of type String or StringBuffer or an array of byte or char. There is no constructor that will accept a primitive byte or char. Please note that if you want to convert a primitive to a String then you can use the static String.valueOf method.

14.a .Prints: false,false .String instances are immutable. String methods such as String.toLowerCase and String.replace create and return a new String instance. The instance on which the method has been invoked remains unchanged. In the program, the equals method is invoked on the original instances of s1 and s2--not the new instances.

15.f .Prints: BDE .The substring method returns a new String instance that is a substring of the original String instance. The single parameter form of the substring method creates a new String that begins at the index specified by the argument value. The two parameter form creates a substring that starts at the index of the first parameter and ends at the index of the second parameter. The character at the start index is included in the substring, but the character at the end index is not.

16.c .Compile-time error .A compile-time error is generated due to the attempt to access the length method of the String class as though it were a variable.

17.b .Prints: false,false,true .At run-time, the expression a+b is evaluated four times. Each evaluation produces a new String instance containing the value "AB". Each of the four instances has a unique reference. If any two of the four instances appear as the operands of the equality operator, then the result is always false. The left and right operands of the equality expression (a+b)==(a+b) reference unique String instances. Since the two references are not the same, the equality expression produces the value false. Similarly, the expression c==d produces the value false. Since the contents of the String instances referenced by c and d are the same, the method invocation expression c.equals(d) produces the value true.

18.e .Prints: AB C D EF .The trim method creates a new String instance with the leading and trailing white space removed.

19.d .Prints: 12,true .The valueOf method returns a String representation of the input parameter. The input parameter may be of type Object, char[], or any primitive type. The toString method returns a reference to the existing String instance. It does not create a new instance of the String.

## Stringbuffer

### Question 1

```
class MWC204 {
 static void m1(StringBuffer s1) {
 s1 = s1.append("B"); System.out.print(s1);
 }
 static void m2(StringBuffer s1) {
 s1.append("C"); System.out.print(s1);
 }
 public static void main(String[] s) {
 StringBuffer s1 = new StringBuffer("A");
 m1(s1); m2(s1);
 System.out.print(s1);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABAA
- c. Prints: ABACA
- d. Prints: ABABAB
- e. Prints: ABABCAB
- f. Prints: ABABCABC
- g. Prints: ABCABCABC
- h. Compile-time error
- i. Run-time error

- j. None of the above
- 

### Question 2

```
class MWC205 {
 static void m1(StringBuffer s1) {
 s1.append("B"); System.out.print(s1);
 }
 static void m2(StringBuffer s1) {
 s1 = new StringBuffer("C"); System.out.print(s1);
 }
 public static void main(String[] s) {
 StringBuffer s1 = new StringBuffer("A");
 m1(s1); m2(s1);
 System.out.print(s1);
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
  - b. Prints: ABCA
  - c. Prints: ABCAB
  - d. Prints: ABCABC
  - e. Prints: ABCAC
  - f. Prints: ABABCABC
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
- 

### Question 3

```
class MWC200 {
 public static void main (String[] args) {
```

```
String s1 = "ABC";
StringBuffer s2 = new StringBuffer(s1);
System.out.print(s2.equals(s1) + "," + s1.equals(s2));
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
  - b. Prints: false,true
  - c. Prints: true,false
  - d. Prints: true,true
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

#### Question 4

```
class MWC201 {
 public static void main (String[] args) {
 String s1 = new String("ABC"), s2 = new String("ABC");
 StringBuffer sb1 = new StringBuffer(s1);
 StringBuffer sb2 = new StringBuffer(s2);
 System.out.print(s1.equals(s2) + "," + sb1.equals(sb2));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. Compile-time error
- f. Run-time error

- g. None of the above
- 

### Question 5

What is the result of attempting to compile and run the program?

- a. Prints: false,false
  - b. Prints: false,true
  - c. Prints: true,false
  - d. Prints: true,true
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 6

```
class MWC203 {
 public static void main (String[] args) {
 String s1 = new String("ABC"), s2 = new String("ABC");
 StringBuffer sb1 = new StringBuffer(s1);
 StringBuffer sb2 = new StringBuffer(s2);
 boolean b1 = s1.hashCode() == s2.hashCode();
 boolean b2 = sb1.hashCode() == sb2.hashCode();
 System.out.print(b1 + "," + b2);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true

c. Prints: true,false

d. Prints: true,true

e. Compile-time error

f. Run-time error

g. None of the above

1 f Prints: ABABCABC Instances of type StringBuffer are not immutable. In method m1, the method invocation expression s1.append("B") appends the String literal "B" to the StringBuffer instance referenced by variable s1. The append method returns a reference to the same StringBuffer instance on which it is invoked; so the assignment expression s1 = s1.append("B") does not assign a different reference value to variable s1. The new value, AB, is printed in method m1. In method m2, the method invocation expression s1.append("C") appends the String literal "C" to the StringBuffer instance referenced by variable s1. The new value, ABC, is printed in method m2. In the main method, a copy of the reference value contained by the reference variable s1 is passed as an argument to methods m1 and m2. Since StringBuffer instances are not immutable, methods m1 and m2 are able to change the original StringBuffer instance that is declared in the main method. The new value, ABC, is printed in the main method.

2 c Prints: ABCAB Instances of type StringBuffer are not immutable. In method m1, the method invocation expression s1.append("B") appends the String literal "B" to the StringBuffer instance referenced by the parameter variable s1. The new value, AB, is printed in method m1. The reference variable s1 declared in the main method refers to the same modified StringBuffer instance. In method m2, the class instance creation expression new StringBuffer("C") creates a new instance of type StringBuffer containing the value C. The assignment expression s1 = new StringBuffer("C") assigns a reference to the new StringBuffer instance to the method parameter variable s1. The value C is printed in method m1. The method local reference variable s1 in the main method remains unchanged by the assignment expression contained in method m2. In the main method, a copy of the reference value contained by the reference variable s1 is passed as an argument to methods m1 and m2. Since StringBuffer instances are not immutable, method m1 is able to change the original instance of the StringBuffer declared in the main method. Since references are passed by value, method m2 can not change the reference variable declared in the main method. Regardless of anything that happens in method m2, the reference variable s1 that is declared in the main method will continue to reference the original StringBuffer instance. Since the content of the original instance was modified by method m1, the new value, AB, is printed in the main method.

3 a Prints: false,false StringBuffer.equals does not override the Object.equals method; so StringBuffer.equals just compares reference values. Since the reference variables s1 and s2 refer to different objects, the equals method of the StringBuffer instance s2 returns the value false. The String.equals method does override the Object.equals method. The String.equals method returns false anytime the argument is not an instance of type String. The method invocation expression s1.equals(s2) produces the value false, because the argument is an instance of type StringBuffer.

4 c Prints: true,false String.equals overrides Object.equals. The String.equals method compares the contents of the String instances--not the references. Since the contents of s1 and s2 are the same, the method invocation expression s1.equals(s2) produces the value true. The StringBuffer.equals method does not override Object.equals. The StringBuffer.equals method compares the reference values--not the contents of the StringBuffer instances. Since the reference values sb1 and sb2 are different, the method invocation expression sb1.equals(sb2) produces the value false.

5 a Prints: false,false StringBuffer.equals does not override Object.equals. The StringBuffer.equals method compares the reference values--not the contents of the StringBuffer instances. The expressions sb1==sb2 and sb1.equals(sb2) produce the same results.

6 c Prints: true,false The StringBuffer class does not override the equals and hashCode methods of the Object class. The Object.equals method does not return the value true unless the argument is a reference to the same object on which the method is invoked. For example, the method invocation expression obj1.equals(obj2) only produces the value true when obj1 == obj2 is also true. The Object.hashCode method tends to return distinct hashCode values for distinct objects regardless of the internal contents of the object. Suppose that the reference variables sb1 and sb2 are of type StringBuffer. The expression sb1.hashCode()

`==` `sb2.hashCode()` will not produce the value true unless the expression `sb1 == sb2` is also true. The `String` class does override the `equals` and `hashCode` methods of the `Object` class. The `String.hashCode` method returns a hashcode value that is computed based on the contents of the `String` object. Suppose that the reference variables `s1` and `s2` are of type `String`. The expression `s1.hashCode() == s2.hashCode()` must produce the value true anytime the method invocation expression `s1.equals(s2)` produces the value true.

## Wrapper

### Question 1

```
class A {
 public static void main (String args[]) {
 byte primitiveByte = 1; // 1
 Byte b1 = new Byte(primitiveByte); // 2
 Byte b2 = new Byte(1); // 3
 System.out.print(b1.byteValue() + b2.byteValue());
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 2
  - b. Prints: 11
  - c. Compile-time error at 1
  - d. Compile-time error at 2
  - e. Compile-time error at 3
  - f. Run-time error
  - g. None of the above
- 

### Question 2

```
class A {
 public static void main (String args[]) {
 byte primitiveByte = 1;
 char primitiveChar = 'b'-'a';
```

```
int primitiveInt = 1;
short primitiveShort = 1;
String s = "1";
Integer i1 = new Integer(primitiveByte);
Integer i2 = new Integer(primitiveChar);
Integer i3 = new Integer(primitiveShort);
Integer i4 = new Integer(primitiveInt);
Integer i5 = new Integer(s);
int p1 = i1.intValue() + i2.intValue() +
 i3.intValue() + i4.intValue() +
 i5.intValue();
System.out.print(p1);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 5
  - b. Prints: 5.0
  - c. Compile-time error
  - d. Run-time error
  - e. None of the above
- 

### Question 3

```
class A {
 public static void main (String args[]) {
 byte primitiveByte = 1;
 char primitiveChar = 1;
 double primitiveDouble = 1;
 String s = "1";
 Double d1 = new Double(primitiveByte);
 Double d2 = new Double(primitiveChar);
 Double d3 = new Double(primitiveDouble);
 Double d4 = new Double(s);
 double d5 = d1.doubleValue() + d2.doubleValue() +
```

```
d3.doubleValue() + d4.doubleValue();
System.out.print(d5);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 4
  - b. Prints: 4.0
  - c. Compile-time error
  - d. Run-time error
  - e. None of the above
- 

#### Question 4

```
class A {
 public static void main (String args[]) {
 byte primitiveByte = 1;
 int primitiveInt = 1;
 long primitiveLong = 1L;
 float primitiveFloat = 1f;
 String s = "1";
 Long i1 = new Long(primitiveByte);
 Long i2 = new Long(primitiveInt);
 Long i3 = new Long(primitiveLong);
 Long i4 = new Long(primitiveFloat);
 Long i5 = new Long(s);
 int i6 = i1.intValue() + i2.intValue() +
 i3.intValue() + i4.intValue() +
 i5.intValue();
 System.out.print(i6);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 5
  - b. Prints: 5.0
  - c. Compile-time error
  - d. Run-time error
  - e. None of the above
- 

### Question 5

```
class C {
 public static void main (String[] args) {
 Float f1 = new Float(1.0); // 1
 Float f2 = new Float("1.0"); // 2
 Float f3 = new Float("1.0f"); // 3
 Float f4 = new Float("1e1f"); // 4
 Float f5 = new Float(".1e1d"); // 5
 }
}
```

What is the result of attempting to compile and run the program?

- a. Compile-time error at 1
  - b. Compile-time error at 2
  - c. Compile-time error at 3
  - d. Compile-time error at 4
  - e. Compile-time error at 5
  - f. Run-time error at 1
  - g. Run-time error at 2
  - h. Run-time error at 3
  - i. Run-time error at 4
  - j. Run-time error at 5
  - k. None of the above
-

## Question 6

```
class A {
 public static void main(String[] args) {
 Boolean b1 = new Boolean(true); // 1
 Boolean b2 = new Boolean(false); // 2
 Boolean b3 = new Boolean(TRUE); // 3
 Boolean b4 = new Boolean(FALSE); // 4
 Boolean b5 = new Boolean("TrUe"); // 5
 Boolean b6 = new Boolean("fAlSe"); // 6
 } }
```

Compile-time errors are generated at which lines?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. 6
- 

## Question 7

Which of the following class instance creation expressions would generate a compile-time error?

- a. new Short(1)
  - b. new Short('1')
  - c. new Short('b' - 'a')
  - d. new Short((short)1 - (short)2)
  - e. new Short((byte)1)
  - f. new Short((short)1)
-

## Question 8

```
class B {
 public static void main (String args[]) {
 Byte b1 = new Byte(1); // 1
 Byte b2 = new Byte('2'); // 2
 Byte b3 = new Byte("3"); // 3
 byte i1 = b1.byteValue()+b2.byteValue()+b3.byteValue(); // 4
 System.out.print(i1);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 6
  - b. Compile-time error at 1
  - c. Compile-time error at 2
  - d. Compile-time error at 3
  - e. Compile-time error at 4
  - f. Run-time error
- 

## Question 9

```
class A {
 public static void main (String args[]) {
 int primitiveInt = 1;
 long primitiveLong = 1L;
 float primitiveFloat = 1.0f;
 String s = "1";
 Integer i1 = new Integer(primitiveInt);
 Integer i2 = new Integer(primitiveLong);
 Integer i3 = new Integer(primitiveFloat);
 Integer i4 = new Integer(s);
 int i5 = i1.intValue() + i2.intValue() +
 i3.intValue() + i4.intValue();
 System.out.print(i5);
 }
}
```

}}

What is the result of attempting to compile and run the program?

- a. Prints: 4
  - b. Prints: 4.0
  - c. Compile-time error
  - d. Run-time error
  - e. None of the above
- 

### Question 10

```
class A {
 public static void main (String args[]) {
 Double d1 = new Double(1.0);
 Double d2 = new Double(d1);
 System.out.print(d1.equals(d2));
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false
  - b. Prints: true
  - c. Compile-time error
  - d. Run-time error
  - e. None of the above
- 

### Question 11

```
class B {
 public static void main (String args[]) {
 Long i1 = new Long(1);
 Long i2 = new Long(i1);
```

```
System.out.print((i1==i2) + "," + i1.equals(i2));
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
  - b. Prints: false,true
  - c. Prints: true,false
  - d. Prints: true,true
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

## Question 12

```
class F {
 public static void main (String[] args) {
 Float f1 = new Float('B' - 'A'); // 1
 Float f2 = new Float(010); // 2
 Float f3 = new Float(0x10); // 3
 Float f4 = new Float(.1e1); // 4
 Float f5 = new Float(1.0); // 5
 Float f6 = new Float(1.0d); // 6
 System.out.print(f1+"," +f2+"," +f3+"," +f4+"," +f5+"," +f6);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Compile-time error at 1
- b. Compile-time error at 2
- c. Compile-time error at 3
- d. Compile-time error at 4
- e. Compile-time error at 5

- f. Compile-time error at 6
  - g. Run-time error at 1
  - h. Run-time error at 2
  - i. Run-time error at 3
  - j. Run-time error at 4
  - k. Run-time error at 5
  - l. Run-time error at 6
  - m. Prints: 1.0,10.0,10.0,1.0,1.0,1.0
  - n. Prints: 1.0,8.0,16.0,1.0,1.0,1.0
  - o. None of the above
- 

### Question 13

```
class B {
 public static void main(String[] args) {
 Boolean b1 = new Boolean(true);
 Boolean b2 = new Boolean(true);
 Boolean b3 = new Boolean("TrUe");
 Boolean b4 = new Boolean("tRuE");
 System.out.print((b1==b2) + ",");
 System.out.print((b1.booleanValue()==b2.booleanValue()) + ",");
 System.out.println(b3.equals(b4));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true

- g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. None of the above
- 

#### Question 14

Which of the following class instance creation expressions would generate a compile-time error?

- a. new Short("1")
  - b. new Short("-1")
  - c. new Short("1.0")
  - d. new Short("0x1")
  - e. new Short("011")
  - f. None of the above
- 

#### Question 15

```
class D {
 public static void main (String args[]) {
 Byte a = new Byte("1");
 byte b = a.byteValue();
 short c = a.shortValue();
 char d = a.charValue();
 int e = a.intValue();
 long f = a.longValue();
 float g = a.floatValue();
 double h = a.doubleValue();
 System.out.print(b+c+d+e+f+g+h);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 7

- b. Prints: 7.0
  - c. Compile-time error
  - d. Run-time error
  - e. None of the above
- 

### Question 16

```
class A {
 public static void main (String args[]) {
 Integer i1 = new Integer(1);
 Integer i2 = new Integer(i1);
 System.out.print(i1.equals(i2));
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false
  - b. Prints: true
  - c. Compile-time error
  - d. Run-time error
  - e. None of the above
- 

### Question 17

```
class B {
 public static void main (String args[]) {
 Double a = new Double(0xFFFF);
 byte b = a.byteValue();
 short c = a.shortValue();
 int e = a.intValue();
 long f = a.longValue();
 float g = a.floatValue();
 double h = a.doubleValue();
 } }
```

```
System.out.print(b+"," +c+" "+ (e+f+g+h == 4 * 0xFFFF));
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0xFFFF,0xFFFF,false
  - b. Prints: 0xFFFF,0xFFFF,true
  - c. Prints: -1,-1,false
  - d. Prints: -1,-1,true
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 18

```
class C {
 public static void main (String args[]) {
 Long a = new Long(1);
 byte b = a.byteValue();
 short s = a.shortValue();
 char c = a.charValue();
 int d = a.intValue();
 long e = a.longValue();
 float f = a.floatValue();
 double g = a.doubleValue();
 System.out.print(b+s+c+d+e+f+g);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: 7
- b. Prints: 7L
- c. Prints: 7.0

d. Compile-time error

e. Run-time error

f. None of the above

e Compile-time error at 3 The Byte class has only two constructors: one accepts a primitive byte; the other accepts a String. The argument that appears in the class instance creation expression new Byte(1) is of type int, but the compiler will not implicitly narrow the int to a byte. At line 1, the assignment expression primitiveByte = 1 causes the compiler to implicitly narrow a literal of type int to type byte. The compiler will implicitly do a narrowing conversion for an assignment expression if the right hand operand is a compile-time constant of type byte, short, char or int and the value falls within the range of the variable on the left and the variable is of type byte, short, or char. For that reason, a constant int expression that is equal to 1 may be assigned to a byte without an explicit cast. The same is not true for the parameters of a method or constructor. The designers of the Java programming language felt that implicit narrowing conversions of method and constructor arguments would add unnecessary complexities to the process of resolving overloaded method calls.

2 a Prints: 5 The Integer class has only two constructors: one accepts a primitive int, and the other accepts a String. If the argument is of type byte, short or char, then it is implicitly promoted to type int.

3 b Prints: 4.0 The Double constructor is overloaded: one accepts an argument of type primitive double; the other accepts an argument of type String.

4 c Compile-time error Long has two constructors: one accepts an argument of type primitive long; the other accepts an argument of type String. The class instance creation expression new Long(primitiveFloat) generates a compile-time error, because the compiler will not implicitly apply a primitive narrowing conversion to the argument.

5 k None of the above The Float constructor is overloaded: one version accepts a primitive of type float; one accepts a primitive of type double; one accepts a String representation of a floating-point literal.

6 c d 3 4 The boolean literals true and false are written with lower case letters; upper case letters produce a compile-time error. A String representation of a boolean literal is acceptable in both upper and lower case letters.

7 a b c d new Short(1) new Short('1') new Short('b' - 'a') new Short((short)1 - (short)2) The Short class has only two constructors: one accepts a primitive short; the other accepts a String. The argument that appears in the class instance creation expression new Short(1) is of type int, but the compiler will not implicitly narrow the int to a short. The argument that appears in the class instance creation expression new Short('1') is of type char, but the compiler will not implicitly convert the char to a short. The argument that appears in the class instance creation expression new Short('b' - 'a') is of type int, but the compiler will not implicitly narrow the int to a short. The argument that appears in the class instance creation expression new Short((short)1 - (short)2) is of type int, but the compiler will not implicitly narrow the int to a short. If both operands of a binary arithmetic expression are of type byte, char or short; then both are implicitly widened to type int, and the result of the expression is of type int.

8 b c e Compile-time error at 1 Compile-time error at 2 Compile-time error at 4 The Byte class has only two constructors: one accepts a primitive byte; the other accepts a String. The argument that appears in the class instance creation expression new Byte(1) is of type int, but the compiler will not implicitly narrow it to type byte. The argument that appears in the class instance creation expression new Byte('2') is of type char, but the compiler will not implicitly narrow it to type byte. At line 4, the result of the additive expression is of type int, but the variable is of type byte. The assignment expression generates a compile-time error.

9 c Compile-time error The Integer class has only two constructors: one accepts a primitive int, and the other accepts a String. The class instance creation expression new Integer(primitiveLong) generates a compile-time error, because the compiler will not apply an implicit narrowing conversion to the argument. For the same reason, the class instance creation expression new Integer(primitiveFloat) generates a compile-time error.

10 c Compile-time error The Double constructor is overloaded: one accepts an argument of type primitive double; the other accepts an argument of type String. There is no constructor that accepts a reference to an instance of type Double.

11 e Compile-time error Long has two constructors: one accepts an argument of type primitive long; the other accepts an argument of type String. The class instance creation expression new Long(i1) generates a compile-time error, because there is no constructor that accepts a reference to an instance of type Long.

12 n Prints: 1.0,8.0,16.0,1.0,1.0,1.0 The Float constructor is overloaded: one version accepts a primitive of type float; one accepts a primitive of type double; one accepts a String representation of a floating-point literal. All numeric values can be promoted to type double; so all numeric values are accepted by the float constructor.

13 d Prints: false,true,true Four instances of type Boolean containing the value true are created. The equality expression b1==b2 evaluates to false, because the unique instances of type Boolean have different reference values. The instance method Boolean.booleanValue returns a copy of the primitive boolean value that is contained by the instance. Since the boolean values of b1 and b2 are the same, the result of the equality expression b1.booleanValue()==b2.booleanValue() is true. The equals method compares the values of the primitives that are wrapped by the Boolean instances. Since the wrapped primitive values are the same, the result of the method invocation expression b3.equals(b4) is true.

14 f None of the above None of the String arguments would generate a compile-time error, but two would generate a run-time error. The argument 1.0 would generate a run-time error, because a floating-point value is not acceptable. The argument 0x1 would generate a run-time error, because a hexadecimal integer literal is not acceptable. The argument must be a decimal integer literal. The leading 0 of the argument 011 is ignored and the argument is interpreted as a decimal integer literal.

15 c Compile-time error A compile-time error is generated, because the Byte class does not have a charValue method. The Byte class extends the Number class and implements all six of the methods declared in Number.

16 c Compile-time error The Integer class has only two constructors: one accepts a primitive int, and the other accepts a String. There is no constructor that accepts an argument that is an instance of type Integer.

17 d Prints: -1,-1,true Double is a subclass of the abstract class Number, and implements all of the methods of Number such as byteValue, shortValue, floatValue, etc. In this case, the Double instance contains the value 0xFFFF. When that value is converted to type byte the result is 0xFF which is also the two's complement representation of the byte value -1. Similarly, 0xFFFF is the two's complement representation of the short value -1. Please note there is no Double.charValue method.

18 d Compile-time error Long is a subclass of the abstract class Number, and Long implements all of the methods of Number: byteValue, shortValue, intValue, longValue, floatValue and doubleValue. The attempt to invoke the charValue method on an instance of Long generates a compile-time error, because there is no charValue method.

## Question 1

Which of the instance creation expressions produce a run-time error?

- a. new Float('A')
- b. new Float("A")
- c. new Float(1L)
- d. new Float("1L")

- e. new Float(0x10)
  - f. new Float("0x10")
  - g. new Float("010")
- 

## Question 2

```
class C {
 public static void main(String[] args) {
 Boolean b1 = Boolean.valueOf(true);
 Boolean b2 = Boolean.valueOf(true);
 Boolean b3 = Boolean.valueOf("TrUe");
 Boolean b4 = Boolean.valueOf("tRuE");
 System.out.print((b1==b2) + ",");
 System.out.print((b1.booleanValue()==b2.booleanValue()) + ",");
 System.out.println(b3.equals(b4));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
  - b. Prints: false,false,true
  - c. Prints: false,true,false
  - d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

## Question 3

Which of the following class instance creation expressions would generate a run-time error?

- a. new Short("1")
  - b. new Short("-1")
  - c. new Short("+1")
  - d. new Short("1.0")
  - e. new Short("0x1")
  - f. new Short("011")
- 

#### Question 4

```
class E {
 public static void main (String[] args) {
 Byte b1 = new Byte("1"), b2 = new Byte("1");
 System.out.print((b1==b2)+","+(b1.equals(b2)));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
  - b. Prints: false,true
  - c. Prints: true,false
  - d. Prints: true,true
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

#### Question 5

```
class B {
 public static void main (String args[]) {
 Integer a = new Integer(256);
 }
}
```

```
byte b = a.byteValue();
short c = a.shortValue();
int e = a.intValue();
long f = a.longValue();
float g = a.floatValue();
double h = a.doubleValue();
System.out.print(b+","+c+","+ (e+f+g+h == 4 * 256));
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: 0,0,false
  - b. Prints: 0,0,true
  - c. Prints: 0,-1,false
  - d. Prints: 0,-1,true
  - e. Prints: -1,0,false
  - f. Prints: -1,0,true
  - g. Prints: -1,-1,false
  - h. Prints: -1,-1,true
  - i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

## Question 6

```
class D {
 static boolean m(double v) {
 return(v != v == Double.isNaN(v));
 }
 public static void main (String args[]) {
 double d1 = Double.NaN;
 double d2 = Double.POSITIVE_INFINITY;
 double d3 = Double.MAX_VALUE;
 System.out.print(m(d1) + "," + m(d2) + "," + m(d3));
 }
}
```

}}

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
  - b. Prints: false,false,true
  - c. Prints: false,true,false
  - d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

### Question 7

```
class F {
 static String m(long i) {return "long";}
 static String m(Long i) {return "Long";}
 static String m(double i) {return "double";}
 static String m(Double i) {return "Double";}
 static String m(String i) {return "String";}
 public static void main (String[] args) {
 System.out.print(m(Long.parseLong("1")));
 }}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: long
- b. Prints: Long
- c. Prints: double

- d. Prints: Double
  - e. Prints: String
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

### Question 8

```
class E {
 public static void main (String args[]) {
 String s1 = Float.toString(1.0); // 1
 String s2 = Float.toString(1.0f); // 2
 String s3 = Float.toString(0xf); // 3
 String s4 = Float.toString(010); // 4
 String s5 = Float.toString('A'); // 5
 } }
```

What is the result of attempting to compile and run the program?

- a. Compile-time error at 1
  - b. Compile-time error at 2
  - c. Compile-time error at 3
  - d. Compile-time error at 4
  - e. Compile-time error at 5
  - f. Run-time error at 1
  - g. Run-time error at 2
  - h. Run-time error at 3
  - i. Run-time error at 4
  - j. Run-time error at 5
  - k. None of the above
-

### Question 9

```
class D {
 public static void main (String[] args) {
 Boolean b1 = new Boolean("trUE"); // 1
 Boolean b2 = new Boolean("What's This?"); // 2
 Boolean b3 = new Boolean(null); // 3
 System.out.print(b1 + "," + b2 + "," + b3);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
  - b. Prints: false,false,true
  - c. Prints: false,true,false
  - d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

### Question 10

```
class F {
 public static void main (String[] args) {
 Short s1 = new Short("1"), s2 = new Short("1");
 int a1 = s1.hashCode(), b1 = s2.hashCode();
 System.out.print((s1==s2)+","+(s1.equals(s2))+","+(a1==b1));
 } }
```

What is the result of attempting to compile and run the program?

- a. false,false,false
  - b. false,false,true
  - c. false,true,false
  - d. false,true,true
  - e. true,false,false
  - f. true,false,true
  - g. true,true,false
  - h. true,true,true
  - i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

### Question 11

```
class E {
 public static void main (String[] args) {
 Byte b1 = new Byte("1"), b2 = new Byte("1");
 int a = b1.hashCode(), b = b2.hashCode();
 System.out.print((b1.equals(b2))+","+(a==b));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
  - b. Prints: false,true
  - c. Prints: true,false
  - d. Prints: true,true
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
-

## Question 12

```
class E {
 static String m(int i) {return "int primitive";}
 static String m(Integer i) {return "Integer instance";}
 static String m(double i) {return "double primitive";}
 static String m(Double i) {return "Double instance";}
 static String m(String i) {return "String instance";}
 public static void main (String[] args) {
 System.out.print(m(Integer.parseInt("1")));
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: int primitive
  - b. Prints: double primitive
  - c. Prints: Double instance
  - d. Prints: String instance
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

## Question 13

Which of the following methods are static members of the java.lang.Double class?

- a. DoubleValue
- b. FloatValue
- c. IntValue
- d. LongValue
- e. ParseDouble
- f. toString(double)
- g. ValueOf

---

#### Question 14

```
class F {
 static String m(long i) {return "long";}
 static String m(Long i) {return "Long";}
 static String m(double i) {return "double";}
 static String m(Double i) {return "Double";}
 static String m(String i) {return "String";}
 public static void main (String[] args) {
 System.out.print(m(Long.parseLong("1L")));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: long
  - b. Prints: Long
  - c. Prints: double
  - d. Prints: Double
  - e. Prints: String
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

#### Question 15

```
class E {
 static String m(float f) {return "f";}
 static String m(Float f) {return "F";}
 public static void main (String[] args) {
 System.out.print(m(Float.parseFloat("1")));
 System.out.print(m(Float.floatValue()));
 System.out.print(m(Float.valueOf("1")));
 }
}
```

}}

What is the result of attempting to compile and run the program?

- a. Prints: fff
  - b. Prints: ffF
  - c. Prints: fFf
  - d. Prints: Fff
  - e. Prints: Fff
  - f. Prints: FfF
  - g. Prints: FFf
  - h. Prints: FFF
  - i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

### Question 16

```
class D {
 public static void main (String[] args) {
 Boolean b1 = Boolean.valueOf("trUE"); // 1
 Boolean b2 = Boolean.valueOf("Even more true"); // 2
 Boolean b3 = Boolean.valueOf(null); // 3
 System.out.print((b1==b2) + ",");
 System.out.print((b2==b3) + ",");
 System.out.println(b3==b1);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false

- d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

### Question 17

```
class E {
 static String m(short s) {return "p";}
 static String m(Short s) {return "S";}
 public static void main (String[] args) {
 Short s1 = new Short("1");
 System.out.print(m(s1.shortValue())+",");
 System.out.print(m(Short.parseShort("1"))+",");
 System.out.print(m(Short.valueOf("1")));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: p,p,p
- b. Prints: p,p,S
- c. Prints: p,S,p
- d. Prints: p,S,S
- e. Prints: S,p,p
- f. Prints: S,p,S
- g. Prints: S,S,p
- h. Prints: S,S,S
- i. Compile-time error

- j. Run-time error
  - k. None of the above
- 

### Question 18

```
class F {
 public static void main (String[] args) {
 System.out.print(Byte.MIN_VALUE+","+Byte.MAX_VALUE);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: -128,127
- b. Prints: -127,128
- c. Prints: 0,255
- d. Compile-time error
- e. Run-time error
- f. None of the above

1 b d f new Float("A") new Float("1L") new Float("0x10") The Float constructor is overloaded: one version accepts a primitive of type float; one accepts a primitive of type double; one accepts a String representation of a floating-point literal. The primitive char literal 'A' is converted to a float, and is accepted by the constructor that declares a parameter of type float. The String literals "NaN" and "Infinity" are accepted by the Float constructor. A sign (+ or -) is optional. The API specification states that any other String must represent a floating-point value; however, a little experimentation proves that a String is acceptable if it can be parsed as a decimal integer value. The leading 0 of an octal value is ignored, and the String is parsed as a decimal value. A String representation of a hexadecimal value is not acceptable. The String "A" does not represent a floating-point literal value; therefore, a NumberFormatException is thrown. Arguments of type String can not contain an integer type suffix, L or l. A floating-point suffix, F, f, D or d, is acceptable, but the suffix has no impact on the result.

2 h Prints: true,true,true The Boolean class contains two public static final Boolean instances: Boolean.FALSE wraps the primitive boolean value false; Boolean.TRUE wraps the primitive boolean value true. Depending on the value of the argument, the Boolean.valueOf method returns a reference to either Boolean.FALSE or Boolean.TRUE. Reference variables b1 and b2 are both initialized with a reference value returned by the method invocation expression Boolean.valueOf(true); so the equality expression b1==b2 is true. Please note that the valueOf method that accepts an argument of type primitive boolean was introduced in the 1.4 version of Java.

3 c d e new Short("+1") new Short("1.0") new Short("0x1") The Short class has only two constructors: one accepts a primitive short; the other accepts a String. A String argument must represent an integral primitive type. A leading minus sign can be added to indicate a negative value. A leading plus sign generates a run-time error. The constructor is not able to determine the radix of the String value by examining a prefix such as 0 or 0x. The 0 prefix used to identify octal

values is accepted, but the String is parsed as a decimal value. The prefix 0x generates a run-time error. A run-time error is generated if the String argument is not formatted as a decimal integer. A floating-point format results in a run-time error.

4 b Prints: false,true,true The expression `b1==b2` compares the references of two instances of Byte. The result is false, because the instances are distinct. The expression `b1.equals(b2)` compares the contents of two instances of Byte. The result is true, because the two instances contain the same value.

5 k None of the above The binary representation of 256 is one bit that is set to one followed by eight bits that are set to zero. When 256 is converted to an eight bit byte value, the bit that is set to one is lost and only the bits that are set to zero remain. When 256 is converted to a short, no information is lost; so the value remains 256.

6 h Prints: true,true,true NaN is the only value that is not equal to itself. The Double.isNaN method returns the result of the expression (`v != v`).

7 a Prints: long The Long.parseLong method returns a primitive long.

8 a Compile-time error at 1 The Float.toString method is overloaded: one declares no parameters and returns the value wrapped by the Float instance; the other accepts a primitive of type float. The literal, 1.0, is of type double and can not be implicitly narrowed to type float.

9 e Prints: true,false,false The Boolean constructor is overloaded: one version accepts a primitive boolean argument; the other accepts a String. If the String value is the word true, then the new Boolean instance will contain the value true. Both upper and lower case letters are acceptable. If the String contains any word other than true or if the reference is null, then the new instance will contain the value false.

10 d false,true,true The equality expression `s1==s2` compares the reference values of two distinct instances of type Short. Since the instance are distinct, the equality expression is false. The expression `s1.equals(s2)` compares the values of two instances of type Short. Since both instances contain the value 1, the returned value is true. The expression `a1==b1` compares the hash codes of two instances of Short. The result is true, because the two instances contain the same value.

11 d Prints: true,true The expression `b1.equals(b2)` compares the values of two instances of type Byte. Since both instances contain the value 1, the return value is true. The expression `a==b` compares the hash codes of two instances of Byte. The result is true, because the two instances contain the same value.

12 a Prints: int primitive The Integer.parseInt method returns a primitive of type int.

13 e f g parseDouble toString(double) valueOf

14 g Run-time error The Long.parseLong method accepts a String parameter that represents a numeric value. Long.parseLong assumes that the input String represents a decimal value unless a second parameter is provided to specify the radix. All of the characters in the String must be digits of the specified radix. The parseLong method does not determine the type of the numeric value based on a suffix such as l, L, f, F, d, or D. Use of any such suffix generates a NumberFormatException at run-time.

15 i Compile-time error The Float.floatValue method is not static; it must be invoked on an instance of type Float.

16 c Prints: false,true,false The Boolean.valueOf method is overloaded: one version accepts a primitive boolean argument; the other accepts a String. If the String value is the word true, then the new Boolean instance will contain the value true. Both upper and lower case letters are acceptable. If the String contains any word other than true or if the String reference is null, then the new instance will contain the value false.

17 b Prints: p,p,S The Short.shortValue method and the Short.parseShort method both return primitives of type short. The Short.valueOf method returns an instance of type Short.

18 a Prints: -128,127

## Question 1

```
class D {
 public static void main (String args[]) {
```

```
boolean b1 = Integer.MIN_VALUE == 0x80000000;
boolean b2 = Integer.MAX_VALUE == 0x7FFFFFFF;
System.out.print(b1 + "," + b2);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
  - b. Prints: false,true
  - c. Prints: true,false
  - d. Prints: true,true
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 2

Which methods of the java.lang.Double class return a primitive value?

- a. doubleValue
  - b. floatValue
  - c. intValue
  - d. longValue
  - e. parseDouble
  - f. toString(double)
  - g. valueOf
- 

### Question 3

```
class G {
 public static void main (String[] args) {
 Long l1 = new Long("1"), l2 = new Long("1");
```

```
int h1 = l1.hashCode(), h2 = l2.hashCode();
StringBuffer s1 = new StringBuffer("1");
StringBuffer s2 = new StringBuffer("1");
int h3 = s1.hashCode(), h4 = s2.hashCode();
System.out.print((l1.equals(l2) & (h1==h2)) + ",");
System.out.print(s1.equals(s2) | (h3==h4));
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
  - b. Prints: false,true
  - c. Prints: true,false
  - d. Prints: true,true
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

#### Question 4

```
class F {
 static String m(float f) {return "f";}
 static String m(Float f) {return "F";}
 public static void main (String[] args) {
 Float f1 = new Float(1);
 System.out.print(m(f1.parseFloat("1")));
 System.out.print(m(f1.floatValue()));
 System.out.print(m(f1.valueOf("1")));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: fff
- b. Prints: ffF

- c. Prints: fFf
  - d. Prints: fFF
  - e. Prints: Fff
  - f. Prints: FfF
  - g. Prints: FFf
  - h. Prints: FFF
  - i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

### Question 5

```
class E {
 static String m1(boolean b) {return "b";}
 static String m1(Boolean b) {return "B";}
 public static void main(String[] args) {
 Boolean b1 = new Boolean(true);
 System.out.print(m1(Boolean.valueOf(null)));
 System.out.print(m1(b1.booleanValue()));
 System.out.println(m1(Boolean.TRUE));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: bbb
- b. Prints: bbB
- c. Prints: bBb
- d. Prints: bBB
- e. Prints: Bbb
- f. Prints: BbB
- g. Prints: BBb
- h. Prints: BBB

- i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

### Question 6

```
class C {
 public static void main (String args[]) {
 String s1 = "1", s2 = "2";
 Byte b1 = Byte.parseByte(s1);
 Byte b2 = Byte.parseByte(s2);
 System.out.print(b1.byteValue() + b2.byteValue());
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 3
  - b. Prints: 12
  - c. Compile-time error
  - d. Run-time error
  - e. None of the above
- 

### Question 7

```
class A {
 public static void main (String[] args) {
 String s = "11";
 int i1 = Integer.parseInt(s);
 System.out.print(new Integer(i1).equals(new Integer(i1)) + ",");
 System.out.print(new Integer(i1).equals(new Integer(s)) + ",");
 System.out.print(new Integer(i1) == new Integer(i1));
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
  - b. Prints: false,false,true
  - c. Prints: false,true,false
  - d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

### Question 8

Which method of the java.lang.Double class returns an instance of type Double?

- a. doubleValue
  - b. floatValue
  - c. intValue
  - d. longValue
  - e. parseDouble
  - f. toString(double)
  - g. valueOf
  - h. None of the above
- 

### Question 9

Which of the method invocation expressions would produce a run-time error?

- a. Long.parseLong("1")
  - b. Long.parseLong("1L")
  - c. Long.parseLong("010")
  - d. Long.parseLong("0x10")
  - e. Long.parseLong("1.0")
- 

### Question 10

```
class B {
 public static void main (String[] args) {
 byte b1 = 11;
 System.out.print(new Integer(b1).equals(new Integer(b1)) + ",");
 System.out.print(new Integer(b1).equals(new Short(b1)) + ",");
 System.out.print(new Integer(b1).equals(new Byte(b1)));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
  - b. Prints: false,false,true
  - c. Prints: false,true,false
  - d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. Compile-time error
  - j. Run-time error
  - k. None of the above
- 

### Question 11

Which methods of the `java.lang.Double` class declare a parameter of type `String`?

- a. `doubleValue`
  - b. `floatValue`
  - c. `intValue`
  - d. `longValue`
  - e. `parseDouble`
  - f. `valueOf`
- 

### Question 12

Which of the method invocation expressions would produce a run-time error?

- a. `Long.parseLong("-1")`
  - b. `Long.parseLong("+1")`
  - c. `Long.parseLong("01")`
  - d. `Long.parseLong("1L")`
  - e. `Long.parseLong("1.0")`
- 

### Question 13

Which of the following methods of the `java.lang.Integer` class returns a primitive `int` type?

- a. `intValue`
  - b. `parseInt`
  - c. `valueOf`
- 

### Question 14

Which methods of the `java.lang.Double` class declare a `NumberFormatException` in the `throws` clause?

- a. doubleValue
  - b. floatValue
  - c. intValue
  - d. longValue
  - e. parseDouble
  - f. toString(double)
  - g. valueOf
- 

### Question 15

```
class F {
 public static void main (String[] args) {
 Integer i1 = new Integer("1");
 Integer i2 = new Integer("1");
 int h1 = i1.hashCode(), h2 = i2.hashCode();
 StringBuffer sb1 = new StringBuffer("1");
 StringBuffer sb2 = new StringBuffer("1");
 int h3 = sb1.hashCode(), h4 = sb2.hashCode();
 System.out.print((h1==h2)+","+(h3==h4));
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
  - b. Prints: false,true
  - c. Prints: true,false
  - d. Prints: true,true
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 16

```
class F {
 public static void main (String args[]) {
 Double d1 = new Double("0x10"); // 1
 double d2 = Double.parseDouble("0x10"); // 2
 Double d3 = Double.valueOf("0x10"); // 3
 System.out.print(d1+","+d2+","+d3);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: 10,10,10
  - b. Prints: 16,16,16
  - c. Compile-time error at line 1
  - d. Compile-time error at line 2
  - e. Compile-time error at line 3
  - f. Run-time error
  - g. None of the above
- 

### Question 17

```
class G {
 public static void main (String[] args) {
 Integer i1 = new Integer("1"), i2 = new Integer("1");
 StringBuffer sb1 = new StringBuffer("1");
 StringBuffer sb2 = new StringBuffer("1");
 System.out.print(sb1.equals(sb2)+","+i1.equals(i2));
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false

- d. Prints: true,true
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

### Question 18

```
class H {
 public static void main (String[] args) {
 int i1, i2; Integer i3, i4;
 i1=new Integer(Integer.parseInt("1")).intValue(); // 1
 i3=new Integer(Integer.parseInt("1")).intValue(); // 2
 i2=Integer.parseInt(Integer.valueOf("1").toString()); // 3
 i4=Integer.parseInt(Integer.valueOf("1").intValue()); // 4
 }
}
```

What is the result of attempting to compile and run the program?

- a. Compile-time error at 1.
- b. Compile-time error at 2.
- c. Compile-time error at 3.
- d. Compile-time error at 4.
- e. Compile-time error
- f. Run-time error

1 d Prints: true,true An int is a 32-bit signed integral value that is stored in two's complement format. The left most bit is the sign bit. The sign bit is set to one for negative numbers and is set to zero for positive numbers.

2 a b c d e doubleValue floatValue intValue longValue parseDouble

3 c Prints: true,false Long.equals overrides Object.equals. The Long.equals method compares the data values contained in the Long instances. The StringBuffer.equals method does not override Object.equals. The StringBuffer.equals method compares the reference values of the instances. Two distinct instances of StringBuffer will not have the same reference values; so the equals method returns false. The StringBuffer.hashCode method typically returns a value that is based on the internal address of the StringBuffer instance. Two instances of StringBuffer will not have the same hash code.

4 b Prints: ffF

5 f Prints: BbB The Boolean.valueOf method returns a Boolean instance. The Boolean.booleanValue method returns a primitive. The Boolean.TRUE field is a reference to an instance of type Boolean that wraps the primitive value true.

6 c Compile-time error The Byte.parseByte method returns a primitive byte. A compile-time error is generated as a result of the attempt to assign a primitive byte to a Byte reference variable.

7 g Prints: true,true,false Integer.equals overrides Object.equals. The Integer.equals method compares the data values contained in the Integer instances. If the argument is of type Integer and if the value contained in the argument is the same as the value contained in the instance on which the method is invoked, then the result is true. The equality operator, ==, does not compare data values. Instead, the equality operator compares references. Distinct instances of any two objects can not have the same reference value; so the expression new Integer(i1) == new Integer(i1) is false.

8 g valueOf

9 b d e Long.parseLong("1L") Long.parseLong("0x10") Long.parseLong("1.0") Long.parseLong is overloaded: one version accepts a String argument that represents an integral value; the other accepts both a String argument and an argument of type int. The int argument represents the radix (i.e. base) of the String argument. The Long.parseLong method is not able to determine the type of the String value by examining a suffix such as L. Any such suffix results in a run-time error. The Long.parseLong method is not able to determine the radix of the String value by examining a prefix such as 0 or 0x. The 0 prefix used to identify octal values is accepted, but the String is parsed as a decimal value. The prefix 0x generates a run-time error. The Long.parseLong method generates a run-time error if the String argument is not formatted as a decimal integer. A floating-point format results in a run-time error.

10 e Prints: true,false,false Integer.equals overrides Object.equals. The Integer.equals method compares the data values contained in the Integer instances. If the argument is of type Integer and if the value contained in the argument is the same as the value contained in the instance on which the method is invoked, then the result is true. If the argument is not of type Integer then the result is false.

11 e f parseDouble valueOf

12 b d e Long.parseLong("+1") Long.parseLong("1L") Long.parseLong("1.0") Long.parseLong is overloaded: one version accepts a String argument that represents an integral value; the other accepts both a String argument and an argument of type int. The int argument represents the radix (i.e. base) of the decimal integral value represented by the String argument. The Long.parseLong method is not able to determine the type of the String value by examining a suffix such as L. Any such suffix results in a run-time error. The Long.parseLong method is not able to determine the radix of the String value by examining a prefix such as 0 or 0x. The 0 prefix used to identify octal values is accepted, but the String is parsed as a decimal value. The prefix 0x generates a run-time error. A leading minus sign (-) can be added to indicate a negative value. A leading plus sign (+) results in a run-time error. The Long.parseLong method generates a run-time error if the String argument is not formatted as a decimal integer. A floating-point format results in a run-time error.

13 a b intValue parseInt Integer.valueOf returns an instance of the Integer wrapper class.

14 e g parseDouble valueOf

15 c Prints: true,false Integer.hashCode overrides Object.hashCode. The Integer.hashCode method calculates the hash code based on the value contained in the Integer instance. The StringBuffer.hashCode method does not override Object.hashCode. The StringBuffer.hashCode method typically returns a value that is based on the internal address of the StringBuffer instance. Two instances of StringBuffer will not have the same hash code.

16 f Run-time error An argument of type String must represent a floating-point value. The argument "0x10" represents a hexadecimal integer literal; so a NumberFormatException would be thrown at run-time.

17 b Prints: false,true Integer.equals overrides Object.equals. The Integer.equals method compares the data values contained in the Integer instances. The StringBuffer.equals method does not override Object.equals. The StringBuffer.equals method compares the reference values of the instances. Two distinct instances of StringBuffer will not have the same reference values; so the equals method will return false.

18 b d Compile-time error at 2. Compile-time error at 4. The Integer.parseInt method is overloaded: one version accepts one argument of type String; the other accepts one String and an int. Both versions of Integer.parseInt return a primitive int. The Integer.intValue method returns the value of the Integer instance as a primitive int. The Integer.valueOf method is overloaded: one version accepts one argument of type String; the other accepts one String and an int. Both versions of Integer.valueOf return an instance of Integer.

## Section 6: Overloading, Overriding, Runtime Type and Object Orientation

### 6.1 Encapsulation:

#### Question 1

Which of the following are true statements?

- a. Encapsulation is a form of data hiding.
  - b. A tightly encapsulated class is always immutable.
  - c. Encapsulation is always used to make programs run faster.
  - d. Encapsulation helps to protect data from corruption.
  - e. Encapsulation allows for changes to the internal design of a class while the public interface remains unchanged.
- 

#### Question 2

Which of the following are true statements?

- a. A top-level class can not be called "tightly encapsulated" unless it is declared private.

- b. Encapsulation enhances the maintainability of the code.
  - c. A tightly encapsulated class allows fast public access to member fields.
  - d. A tightly encapsulated class allows access to data only through accessor and mutator methods.
  - e. Encapsulation usually reduces the size of the code.
  - f. A tightly encapsulated class might have mutator methods that validate data before it is loaded into the internal data model.
- 

## Question 3

A class can not be called "tightly encapsulated" unless which of the following is true?

- a. The class is declared final.
  - b. All local variables are declared private.
  - c. All method parameters are declared final.
  - d. No method returns a reference to any object that is referenced by an internal data member.
  - e. None of the above
- 

## Question 4

A class can not be called "tightly encapsulated" unless which of the following is true?

- a. All of the methods are declared private.
  - b. All of the methods are synchronized.
  - c. All local variables are declared final.
  - d. The class is a direct subclass of Object.
  - e. Accessor methods are used to prevent fields from being set with invalid data.
  - f. None of the above
- 

## Question 5

A class can not be called "tightly encapsulated" unless which of the following are true?

- 
- a. The data members can not be directly manipulated by external code.
  - b. The class is declared final.
  - c. It has no public mutator methods.
  - d. The superclass is tightly encapsulated.**
- 

## Question 6

A class can not be called "tightly encapsulated" unless which of the following is true?

- a. The class is a nested class.
  - b. The constructors are declared private.
  - c. The mutator methods are declared private.
  - d. The class implements the Encapsulated interface.
  - e. None of the above**
- 

## Question 7

A class can not be called "tightly encapsulated" unless which of the following is true?

- a. All member fields are declared final.
  - b. The class is not anonymous.
  - c. The internal data model can be read and modified only through accessor and mutator methods.**
  - d. The class is an inner class.
  - e. None of the above
- 

## Question 8

```
class GFC500 {private String name;}\nclass GFC501 {\n private String name;\n private void setName(String name) {this.name = name;}}
```

```
private String getName() {return name;}
}
class GFC502 {
 private String name;
 public void setName(String name) {this.name = name;}
 public String getName() {return name;}
}
```

Which class is not tightly encapsulated?

- a. GFC501
  - b. GFC502
  - c. GFC503
  - d. None of the above
- 

## Question 9

```
class GFC505 extends GFC504 {
 public void setName(String name) {this.name = name;}
 public String getName() {return name;}
}
class GFC504 extends GFC503 {
 private void setName(String name) {this.name = name;}
 private String getName() {return name;}
}
class GFC503 {String name;}
```

Which class is tightly encapsulated?

- a. GFC503
  - b. GFC504
  - c. GFC505
  - d. None of the above
- 

## Question 10

```

class GFC506 {private String name;}
class GFC507 extends GFC506 {
 String name;
 public void setName(String name) {this.name = name;}
 public String getName() {return name;}
}
class GFC508 extends GFC506 {
 private String name;
 public GFC508(String name) {setName(name);}
 public void setName(String name) {this.name = name;}
 public String getName() {return name;}
}

```

Which class is not tightly encapsulated?

- a. GFC506
  - b. **GFC507**
  - c. GFC508
  - d. None of the above
- 

| No. | Answer | Remark                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | a d e  | Encapsulation is a form of data hiding. Encapsulation helps to protect data from corruption. Encapsulation allows for changes to the internal design of a class while the public interface remains unchanged. A tightly encapsulated class does not allow direct public access to the internal data model. Instead, access is permitted only through accessor (i.e. get) and mutator (i.e. set) methods. The additional time required to work through the accessor and mutator methods typically slows execution speed. Encapsulation is a form of data hiding. A tightly encapsulated class does not allow public access to any data member that can be changed in any way; so encapsulation helps to protect internal data from the possibility of corruption from external influences. The mutator methods can impose constraints on the argument values. If an argument falls outside of the acceptable range, then a mutator method could throw an <code>IllegalArgumentException</code> . The internal design of a tightly encapsulated class can change while the public interface remains unchanged. An immutable class is always tightly encapsulated, but not every tightly encapsulation class is immutable. |
| 2   | b d f  | Encapsulation enhances the maintainability of the code. A tightly encapsulated class allows access to data only through accessor and mutator methods. A tightly encapsulated class might have mutator methods that validate data before it is loaded into the internal data model. The data members of a tightly encapsulated class are declared private; so changes to the data model are less likely to impact external code. Access to internal data can be provided by public accessor (i.e. get) and mutator (i.e. set) methods. The mutator methods can be used to validate the data before it is loaded into the internal data model. The use of accessor and mutator methods is likely to increase the size of the code and slow execution speed.                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 3   | e      | None of the above If a class A has a method that returns a reference to an internal, mutable object; then external code can use the reference to modify the internal state of class A. Therefore, class A can not be considered tightly encapsulated. However, the methods of a tightly encapsulated class may return a reference to an immutable object or a reference to a copy or clone of an internal object.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

4 f None of the above One answer option reads as follows: "Accessor methods are used to prevent fields from being set with invalid data." The answer would be correct if the word "Accessor" were replaced by the word "Mutator". Accessor methods are used to read data members; mutator methods are used to set data members. The mutator methods can validate the parameter values before the values are used to change the state of the internal data model.

5 a d The data members can not be directly manipulated by external code. The superclass is tightly encapsulated. If a class A is not tightly encapsulated, then no subclass of A is tightly encapsulated.

6 e None of the above A tightly encapsulated class may have public mutator methods.

7 c The internal data model can be read and modified only through accessor and mutator methods. A class is not tightly encapsulated if the internal data model can be read and/or modified without working through accessor (i.e. get) and mutator (i.e. set) methods.

8 d None of the above All three classes are tightly encapsulated, because the data members are private. A tightly encapsulated class can have public accessor and mutator methods, but it is not required to have those methods.

9 d None of the above Class GFC503 is not tightly encapsulated; so no subclass of GFC503 is tightly encapsulated.

10 b GFC507 Class GFC507 has a public field; so it is not tightly encapsulated.

## 6.2 Inheritance:

### Question 1

Which of the following statements are true?

- a. A constructor can invoke another constructor of the same class using the alternate constructor invocation, "this(argumentListopt);".
- b. A constructor can invoke itself using the alternate constructor invocation, "this(argumentListopt);".
- c. The alternate constructor invocation, "this(argumentListopt);", can legally appear anywhere in the constructor body.
- d. A constructor can invoke the constructor of the direct superclass using the superclass constructor invocation, "super(argumentListopt);".
- e. The number of constructor invocations that may appear in any constructor body can equal but not exceed the number of alternate constructors declared in the same class.
- f. A constructor is not permitted to throw an exception.

---

### Question 2

Suppose that the superclass constructor invocation, "super(argumentListOpt);", appears explicitly in a subclass constructor. If a compile-time error is to be avoided then the arguments for the superclass constructor invocation, "super(argumentListOpt);", can not refer to which of the following?

- a. Static variables declared in this class or any superclass.
  - b. Instance variables declared in this class or any superclass.
  - c. Static methods declared in this class or any superclass.
  - d. Instance methods declared in this class or any superclass.
  - e. The keyword this.
  - f. The keyword super.
- 

## Question 3

```
class A {void m1(String s1) {}}
class B extends A {
 void m1(String s1) {} // 1
 void m1(boolean b) {} // 2
 void m1(byte b) throws Exception {} // 3
 String m1(short s) {return new String();} //4
 private void m1(char c) {} // 5
 protected void m1(int i) {} // 6
}
```

What is the result of attempting to compile the program?

- a. Compile-time error at line 1
  - b. Compile-time error at line 2
  - c. Compile-time error at line 3
  - d. Compile-time error at line 4
  - e. Compile-time error at line 5
  - f. Compile-time error at line 6
  - g. None of the above
- 

## Question 4

```
class A {void m1() {System.out.print("A.m1");}}
class B extends A {
 void m1() {System.out.print("B.m1");}
 static void m1(String s) {System.out.print(s+",");}
}
class C {
 public static void main (String[] args) {B.m1("main"); new B().m1();}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: main,B.m1
  - b. Compile-time error
  - c. Run-time error
  - d. None of the above
- 

## Question 5

Which of the following are true statements?

- a. The relationship between a class and its superclass is an example of a "has-a" relationship.
  - b. The relationship between a class and its superclass is an example of an "is-a" relationship.
  - c. The relationship between a class and an object referenced by a field within the class is an example of a "has-a" relationship.
  - d. The relationship between a class and an object referenced by a field within the class is an example of an "is-a" relationship.
- 

## Question 6

```
class A {String s1 = "A.s1"; String s2 = "A.s2";}
class B extends A {
 String s1 = "B.s1";
 public static void main(String args[]) {
 B x = new B(); A y = (A)x;
 System.out.println(x.s1+" "+x.s2+" "+y.s1+" "+y.s2);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: B.s1 A.s2 B.s1 A.s2
  - b. Prints: B.s1 A.s2 A.s1 A.s2
  - c. Prints: A.s1 A.s2 B.s1 A.s2
  - d. Prints: A.s1 A.s2 A.s1 A.s2
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- 

## Question 7

```
class C {
 void printS1() {System.out.print("C.printS1 ");}
 static void printS2() {System.out.print("C.printS2 ");}
}
class D extends C {
 void printS1(){System.out.print("D.printS1 ");}
 void printS2() {System.out.print("D.printS2 ");}
 public static void main (String args[]) {
 C c = new D(); c.printS1(); c.printS2();
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: C.printS1 C.printS2
  - b. Prints: C.printS1 D.printS2
  - c. Prints: D.printS1 C.printS2
  - d. Prints: D.printS1 D.printS2
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- 

## Question 8

```

class E {
 void printS1(){System.out.print("E.printS1");}
 static void printS2() {System.out.print("E.printS2");}
}
class F extends E {
 void printS1(){System.out.print("F.printS1");}
 static void printS2() {System.out.print("F.printS2");}
 public static void main (String args[]) {
 E x = new F(); x.printS1(); x.printS2();
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: E.printS1 E.printS2
  - b. Prints: E.printS1 F.printS2
  - c. Prints: F.printS1 E.printS2
  - d. Prints: F.printS1 F.printS2
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- 

## Question 9

```

class P {
 static void printS1(){System.out.print("P.printS1");}
 void printS2() {System.out.print("P.printS2");}
 void printS1S2(){printS1();printS2();}
}
class Q extends P {
 static void printS1(){System.out.print("Q.printS1");}
 void printS2(){System.out.print("Q.printS2");}
 public static void main(String[] args) {
 new Q().printS1S2();
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: P.printS1 P.printS2

- b. Prints: P.printS1 Q.printS2
  - c. Prints: Q.printS1 P.printS2
  - d. Prints: Q.printS1 Q.printS2
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- 

## Question 10

```
class R {
 private void printS1(){System.out.print("R.printS1 ");}
 protected void printS2() {System.out.print("R.printS2 ");}
 protected void printS1S2(){printS1();printS2();}
}
class S extends R {
 private void printS1(){System.out.print("S.printS1 ");}
 protected void printS2(){System.out.print("S.printS2 ");}
 public static void main(String[] args) {
 new S().printS1S2();
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: R.printS1 R.printS2
  - b. Prints: R.printS1 S.printS2
  - c. Prints: S.printS1 R.printS2
  - d. Prints: S.printS1 S.printS2
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- 

## Question 11

```
class T {
 private int i1, i2;
```

```

void printI1I2() {System.out.print("T, i1="+i1+", i2="+i2);}
T(int i1, int i2) {this.i1=i1; this.i2=i2;}
}
class U extends T {
private int i1, i2;
void printI1I2() {System.out.print("U, i1="+i1+", i2="+i2);}
U(int i1, int i2) {this.i1=i1; this.i2=i2;}
public static void main(String[] args) {
 T t = new U(1,2); t.printI1I2();
}
}

```

What is the result of attempting to compile and run the program?

- a. Prints: U, i1=1, i2=2
  - b. Prints: T, i1=1, i2=2
  - c. Prints: U, i1=null, i2=null
  - d. Prints: T, i1=null, i2=null
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- 

## Question 12

```

interface I {String s1 = "I";}
class A implements I {String s1 = "A";}
class B extends A {String s1 = "B";}
class C extends B {
 String s1 = "C";
 void printIt() {
 System.out.print(((A)this).s1 + ((B)this).s1 +
 ((C)this).s1 + ((I)this).s1);
 }
 public static void main (String[] args) {new C().printIt();}
}

```

What is the result of attempting to compile and run the program?

- a. Prints: ABCI

- b. Run-time error
  - c. Compile-time error
  - d. None of the above
- 

## Question 13

```
abstract class D {String s1 = "D"; String getS1() {return s1;}}
class E extends D {String s1 = "E"; String getS1() {return s1;}}
class F {
 public static void main (String[] s) {
 D x = new E(); System.out.print(x.s1 + x.getS1());
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: DD
  - b. Prints: DE
  - c. Prints: ED
  - d. Prints: EE
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- 

## Question 14

```
class A {static void m() {System.out.print("A");}}
class B extends A {static void m() {System.out.print("B");}}
class C extends B {static void m() {System.out.print("C");}}
class D {
 public static void main(String[] args) {
 C c = new C(); c.m(); B b = c; b.m(); A a = b; a.m();
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
  - b. Prints: ABC
  - c. Prints: CBA
  - d. Prints: CCC
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

## Question 15

```
class A {String s1 = "A";}
class B extends A {String s1 = "B";}
class C extends B {String s1 = "C";}
class D {
 static void m1(A x) {System.out.print(x.s1);}
 static void m1(B x) {System.out.print(x.s1);}
 static void m1(C x) {System.out.print(x.s1);}
 public static void main(String[] args) {
 A a; B b; C c; a = b = c = new C(); m1(a); m1(b); m1(c);
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
  - b. Prints: ABC
  - c. Prints: CBA
  - d. Prints: CCC
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

## Question 16

```

class Leg{}
class Fur{}
abstract class Pet {
 public abstract void eat();
 public abstract void sleep();
}
class Dog extends Pet {
 Leg leftFront = new Leg(), rightFront = new Leg();
 Leg leftRear = new Leg(), rightRear = new Leg();
 Fur fur = new Fur();
 public Fur shed() {return fur;}
 public void eat() {}
 public void sleep() {}
}
class Cat extends Dog {
 public void ignoreOwner() {}
 public void climbTree() {}
}

```

Which of the following statements is not a true statement?

- A Cat object inherits an instance of Fur and four instances of Leg from the Dog superclass.
  - A Cat object is able to sleep and eat.
  - A Cat object is able to climb a tree.
  - The relationship between Dog and Pet is an example of an appropriate use of inheritance.
  - The relationship between Cat and Dog is an example of an appropriate use of inheritance.
  - None of the above.
- 

## Question 17

```

class A {
 A() {System.out.print("CA ");}
 static {System.out.print("SA");}
}
class B extends A {
 B() {System.out.print("CB ");}
 static {System.out.print("SB");}
 public static void main (String[] args) {B b = new B();}
}

```

What is the result of attempting to compile and run the above program?

- a. Prints: SA CA SB CB
  - b. Prints: SA SB CA CB
  - c. Prints: SB SA CA CB
  - d. Prints: SB CB SA CA
  - e. Runtime Exception
  - f. Compiler Error
  - g. None of the above
- 

## Question 18

```
class A {String s1="A";}
class B extends A {String s1="B";}
class C extends B {String s1="C";}
class D extends C {
 String s1="D";
 void m1() {
 System.out.print(this.s1 + ","); // 1
 System.out.print(((C)this).s1 + ","); // 2
 System.out.print(((B)this).s1 + ","); // 3
 System.out.print(((A)this).s1); // 4
 }
 public static void main (String[] args) {
 new D().m1(); // 5
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: D,D,D,D
- b. Prints: D,C,B,A
- c. Compile-time error at 1.
- d. Compile-time error at 2.
- e. Compile-time error at 3.
- f. Compile-time error at 4.

- g. Compile-time error at 5.
  - h. Run-time error
  - i. None of the above
- 

## Question 19

```
class SuperA {String s1="SuperA";}
class SuperB {String s1="SuperB";}
class A extends SuperA {
 String s1="A";
 class B extends SuperB { // 1
 String s1="B";
 void m1() {
 System.out.print(this.s1 + ","); // 2
 System.out.print(super.s1 + ","); // 3
 System.out.print(A.this.s1 + ","); // 4
 System.out.print(A.super.s1); // 5
 }
 }
 public static void main (String[] args) {
 new A().new B().m1(); // 6
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: B,SuperB,B,SuperB
  - b. Prints: B,SuperB,A,SuperA
  - c. Compile-time error at 1.
  - d. Compile-time error at 2.
  - e. Compile-time error at 3.
  - f. Compile-time error at 4.
  - g. Compile-time error at 5.
  - h. Compile-time error at 6.
  - i. Run-time error
  - j. None of the above
-

## Question 20

```
class A {void m1() {System.out.print("A");}}
```

```
class B extends A {void m1(){System.out.print("B");}}
```

```
class C extends B {void m1() {System.out.print("C");}}
```

```
class D extends C {
```

```
 void m1() {System.out.print("D");}
```

```
 void m2() {
```

```
 m1();
```

```
 ((C)this).m1(); // 1
```

```
 ((B)this).m1(); // 2
```

```
 ((A)this).m1(); // 3
```

```
}
```

```
public static void main (String[] args) {
```

```
 new D().m2(); // 4
```

```
}
```

What is the result of attempting to compile and run the program?

- a. Prints: DCBA
  - b. Prints: DDDD
  - c. Compile-time error at 1.
  - d. Compile-time error at 2.
  - e. Compile-time error at 3.
  - f. Compile-time error at 4.
  - g. Run-time error
  - h. None of the above
- 

## Question 21

```
class A {public void m1() {System.out.print("A1");}}
```

```
class B extends A {
```

```
 public void m1() {System.out.print("B1");}
```

```
 public void m2() {System.out.print("B2");}
```

```
}
```

```
class C {
```

```
 public static void main(String[] args) {
```

```
 A a1 = new B();
```

```

a1.m1(); // 1
a1.m2(); // 2
((B)a1).m1(); // 3
((B)a1).m2(); // 4
}

```

What is the result of attempting to compile and run the program?

- a. Prints: A1B2B1B2
  - b. Prints: B1B2B1B2
  - c. Compile-time error at 1
  - d. Compile-time error at 2
  - e. Compile-time error at 3
  - f. Compile-time error at 4
  - g. Run-time error
  - h. None of the above
- 
- 

| No. | Answer  | Remark                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-----|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | a d     | A constructor can invoke another constructor of the same class using the alternate constructor invocation, "this(argumentListopt);". A constructor can invoke the constructor of the direct superclass using the superclass constructor invocation, "super(argumentListopt);". If an alternate constructor invocation appears in the body of the constructor, then it must be the first statement. The same is true for a superclass constructor invocation. A compile-time error is generated if a constructor attempts to invoke itself either directly or indirectly.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 2   | b d e f | Instance variables declared in this class or any superclass. Instance methods declared in this class or any superclass. The keyword this. The keyword super. If the superclass constructor invocation, "super(argumentListopt);", appears explicitly or implicitly, then it must be the first statement in the body of the constructor. Until the superclass constructor invocation runs to completion, no other statements are processed within the body of the constructor. The same is true of the constructors of any superclass. (Note: The primordial class, Object, does not have a superclass, so the constructors do not include a superclass constructor invocation statement.) Suppose class B is a subclass of A. The process of creating and initializing an instance of B includes the creation and initialization of an instance of the superclass A and an instance of the superclass Object. The superclass constructor invocation statement appearing in a constructor of B is invoked before the completion of the constructors of the superclasses A and Object. A superclass constructor invocation statement appearing in B can not refer to the non-static members of the superclasses, because the process of initializing those non-static superclass members is not complete when the superclass constructor invocation occurs in B. The same is true of the non-static members of B. |
| 3   | g       | None of the above If a superclass method is not private and is accessible to code in a subclass, then any subclass method that has the same signature as the superclass method must also have the same return type. In other words, if a superclass method is overridden or hidden in a                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

subclass, then the overriding or hiding subclass method must have the same return type as the superclass method. No such restriction applies to method overloading. If two methods share an overloaded method name but not the same parameter list, then the two methods need not have the same return type. Class A declares one method: The name is m1 and the single parameter is of type String. Class B extends A and declares six methods that overload the name m1. The method, B.m1(String s1), overrides the superclass method, A.m1(String s1). The overriding subclass method must have the same return type as the superclass method, but the methods that overload the name m1 are free to have different return types. An overriding subclass method is not permitted to throw any checked exception that is not listed or is not a subclass of any of those listed in the throws clause of the superclass method. No such restriction applies to method overloading. If two methods share an overloaded method name but not the same parameter list, then the two methods are free to have differing throws clauses.

4 a Prints: main,B.m1 Suppose a superclass method is not private and is accessible to code in a subclass. If the superclass method is declared static, then any subclass method sharing the same signature must also be declared static. Similarly, if the superclass method is not declared static, then any subclass method sharing the same signature must not be declared static. The rules governing method overloading are different. If a superclass method is declared static, then any subclass method that overloads the superclass method is free to be declared static or non-static. Similarly, if a method is declared non-static, then any overloading method is free to be declared static or non-static. Method B.m1() shares the same signature as the non-static superclass method A.m1(), so B.m1() must also be non-static. The method B.m1(String s) overloads the method name m1, but does not share the same signature with any superclass method; therefore, B.m1(String s) can be declared static even though the other methods of the same name are non-static.

5 b c The relationship between a class and its superclass is an example of an "is-a" relationship. The relationship between a class and an object referenced by a field within the class is an example of a "has-a" relationship. Inheritance is an example of an "is-a" relationship, because the subclass "is-a" specialized type of the superclass. The relationship between a class and an object referenced by a field declared within the class is an example of a "has-a" relationship, because the class "has-a" object.

6 b Prints: B.s1 A.s2 A.s1 A.s2 The variables of a subclass can hide the variables of a superclass or interface. The variable that is accessed is determined at compile-time based on the type of the reference--not the run-time type of the object. The two references x and y refer to the same instance of type B. The name x.s1 uses a reference of type B; so it refers to the variable s1 declared in class B. The name y.s1 uses a reference of type A; so it refers to the variable s1 declared in class A.

7 f Compile-time error Suppose a superclass method is not private and is accessible to code in a subclass. If the superclass method is declared static, then any subclass method sharing the same signature must also be declared static. Similarly, if the superclass method is declared non-static, then any subclass method sharing the same signature must also be declared non-static. The attempted declaration of the non-static method D.printS2 generates a compile-time error; because the superclass method, C.printS2, is static.

8 c Prints: F.printS1 E.printS2 A static method is selected based on the compile-time type of the reference--not the run-time type of the object. A non-static method is selected based on the run-time type of the object--not the compile-time type of the reference. Both method invocation expressions, x.printS1() and x.printS2(), use a reference of the superclass type, E, but the object is of the subclass type, F. The first of the two expressions invokes an instance method on an object of the subclass type; so the overriding subclass method is selected. The second invokes a static method using a reference of the superclass type; so the superclass method is selected.

9 b Prints: P.printS1 Q.printS2 Suppose a method m1 is invoked using the method invocation expression m1(). If m1 is a static member of the class where the invocation expression occurs, then that is the implementation of the method that is invoked at run-time regardless of the run-time type of the object. If m1 is non-static, then the selected implementation is determined at run-time based on the run-time type of the object. The program invokes method printS1S2 on an instance of class Q. The body of method printS1S2 contains two method invocation expressions, printS1() and printS2(). Since method printS1 is static, the implementation declared in class P is invoked. Since printS2 is non-static and the run-time type of the object is Q, the invoked method is the one declared in class Q.

10 b Prints: R.printS1 S.printS2 A private method of a superclass is not inherited by a subclass. Even if a subclass method has the same signature as a superclass method, the subclass method does not override the superclass method. Suppose a non-static method m1 is invoked using the method invocation expression m1(). If m1 is a private member of the class T where the invocation expression occurs, then the implementation in class T is selected at run-time regardless of the run-time type of the object. If the non-static method m1 is not private, then the selected implementation is determined at run-time based on the run-time type of the object. The program invokes the non-static method printS1S2 on an instance of class S, so the run-time type is S. The body of method R.printS1S2 contains two method invocation expressions, printS1() and printS2(). Since class R contains a private implementation of the instance method printS1, it is the implementation that is selected regardless of the run-time type of the object. Since printS2 is not private and not static, the selected implementation of printS2 depends on the run-time type of the object. The method printS1S2 is invoked on an instance of class S; so the run-time type of the object is S, and the implementation of printS2 declared in class S is selected.

11 f Compile-time error The two-parameter constructor of U does not explicitly invoke the two-parameter constructor of T; therefore, the constructor of U will try to invoke a no-parameter constructor of T, but none exists.

12 a Prints: ABCI Suppose that a class extends a superclass, X, or implements an interface, X. The field access expression (X)this.hiddenField is used to access the hidden field, hiddenField, that is accessible within the superclass or interface, X.

13 b Prints: DE At run-time, the actual field that is accessed depends on the compile-time type of the reference--not the run-time type of the object. The compile-time type of the reference x in the name x.s1 is D; so the selected field is the one declared in class D. A non-static method is selected based on the run-time type of the object--not the compile-time type of the reference. The same reference variable x is used in the method invocation expression x.getS1(), and the compile-time type is still D. At run-time, the actual type of the object is E; so the selected method is the one declared in class E.

14 c Prints: CBA Class C extends B, and B extends A. The static method C.m hides method B.m, and B.m hides A.m. In the method invocation expression c.m(), the compile-time type of the reference c is C. A static method is invoked based on the compile-time type of the reference; so the method invocation expression c.m() invokes the method m declared in class C. The compile-time type of the reference b is B; so the method invocation expression b.m() invokes the method m declared in class B. The compile-time type of the reference a is A; so the method invocation expression a.m() invokes the method m declared in class A.

15 b Prints: ABC In all three cases, the object passed to method m1 is an instance of class C; however, the type of the reference is different for each method. Since fields are accessed based on the type of the reference, the value printed by each method is different even though the same instance is used for each method invocation.

16 e The relationship between Cat and Dog is an example of an appropriate use of inheritance. An appropriate inheritance relationship includes a subclass that "is-a" special kind of the superclass. The relationship between the Dog subclass and the Pet superclass is an example of an appropriate inheritance relationship, because a Dog "is-a" Pet. The relationship between the Cat subclass and the Dog superclass is not an example of an appropriate use of inheritance, because a Cat is not a special kind of a Dog. The goal of the OO paradigm is to develop software models that are accurate and reusable. If the software model is not accurate, then it probably is not reusable and the goals of the OO paradigm are not achieved. Code reuse and maintenance becomes increasingly difficult when inheritance is used to model inappropriate relationships. For example, suppose that somebody implements a herdSheep method in the Dog class. The Cat subclass would inherit the method and suddenly each instance of Cat would acquire the unwanted capability to make an attempt to herd sheep. It is difficult to imagine that a Cat would perform well in that role, so additional maintenance would be required to resolve the problem.

17 b Prints: SA SB CA CB The static initializer of the super class runs before the static initializer of the subclass. The body of the superclass constructor runs to completion before the body of the subclass constructor runs to completion.

18 b Prints: D,C,B,A A field of a superclass can be inherited by a subclass if the superclass field is not private and not hidden by a field declaration in the subclass and is accessible to code in the subclass. The field D.s1 hides C.s1, and C.s1 hides B.s1, and B.s1 hides A.s1. The keyword this serves as a reference to the object on which a method has been invoked. In the field access expression this.s1 appearing on line 1, the keyword this denotes a reference to the object of type D on which method m1 has been invoked. In the field access expression ((C)this).s1 appearing on line 2, the reference denoted by the keyword this is cast from type D to type C. The field that is accessed at run-time depends on the compile-time type of the reference; so the field access expression ((C)this).s1 refers to the variable s1 declared in class C.

19 b Prints: B,SuperB,A,SuperA The expression A.this.s1 is an example of a qualified this expression. It accesses the variable s1 declared in class A. The expression A.super.s1 is equivalent to ((SuperA)A.this).s1. It accesses the variable s1 declared in class SuperA.

20 b Prints: DDDD The instance method that is invoked depends on the run-time type of the object--not the compile-time type of the reference. In each case, the method m1 is invoked on an object of type D; so the implementation of m1 in type D is selected each time.

21 d Compile-time error at 2 The method invocation expression a1.m2() generates a compile-time error, because the named method, m2, is declared in class B, but the reference is of the superclass type, A. The reference a1 is of type A; so a1 is able to access only those methods that are declared in class A and subclass methods that override those of class A. Only one method, m1, is declared in A; so a reference of type A can be used to invoke A.m1 or an overriding implementation of m1 that is declared in a subclass of A. Class B extends A and overrides method m1. A reference of type A can be used to invoke method m1 on an instance of type B. Class B declares an additional method, m2, that does not override a method of class A; so a reference of type A can not invoke B.m2.

## 6.3 Nested Classes:

### Question 1

```
class A {
```

```
private static String s1 = "s1";
final String s2 = "s2";
A () { new Z("s5","s6");}
class Z {
 final String s3 = "s3";
 String s4 = "s4";
 Z (final String s5, String s6) {
 System.out.print(???);
 }
 public static void main(String args[]) {new A();}
}
```

Which variable can not be substituted for ??? without causing a compile-time error?

- a. s1
  - b. s2
  - c. s3
  - d. s4
  - e. s5
  - f. s6
  - g. None of the above
- 

## Question 2

```
class B {
 private static String s1 = "s1";
 final String s2 = "s2";
 B () {new Z("s5","s6");}
 static class Z {
 final String s3 = "s3";
 static String s4 = "s4";
 Z (final String s5, String s6) {
 System.out.print(???);
 }
 public static void main(String args[]) {new B();}
 }
}
```

Which variable can not be substituted for ??? without causing a compile-time error?

- a. s1
  - b. s2
  - c. s3
  - d. s4
  - e. s5
  - f. s6
  - g. None of the above
- 

## Question 3

```
class C {
 private static String s1 = "s1";
 String s2 = "s2";
 C() {m1("s5","s6");}
 void m1(final String s5, String s6) {
 final String s3 = "s3"; String s4 = "s4";
 class Z {Z() {System.out.print(???);}}
 new Z();
 }
 public static void main(String args[]) {new C();}
}
```

Which variable names can be substituted for ??? without causing a compile-time error?

- a. s1
  - b. s2
  - c. s3
  - d. s4
  - e. s5
  - f. s6
- 

## Question 4

```
class D {
 D() {System.out.print("D");}
```

```
class Z {Z(){System.out.print("Z");}}
public static void main(String args[]) {
 new D.Z();
}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: D
  - b. Prints: Z
  - c. Prints: DZ
  - d. Prints: ZD
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
- 

## Question 5

```
class E {
 E() {System.out.print("E");}
 static class Z {Z(){System.out.print("Z");}}
 public static void main(String args[]) {
 new E.Z();
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: E
  - b. Prints: Z
  - c. Prints: EZ
  - d. Prints: ZE
  - e. Run-time error
  - f. Compile-time error
  - g. None of the above
-

## Question 6

```
class F {
 public void m1() {Z.m1();} // 1
 private static class Y {
 private static void m1() {
 System.out.print("Y.m1 ");
 }
 }
 private static class Z {
 private static void m1(){
 System.out.print("Z.m1 ");
 Y.m1(); // 2
 }
 }
 public static void main(String[] args) {
 new F().m1();
 }
}
```

What is the result of attempting to compile and run the program?

- a. Compile-time error at line 1
  - b. Compile-time error at line 2
  - c. Run-time error at line 1
  - d. Run-time error at line 2
  - e. Prints: Z.m1 Y.m1
  - f. None of the above
- 

## Question 7

```
class G {
 final String s1 = "G.s1";
 class Z {
 String s1;
 void m1() {System.out.println(???);}
 }
 public static void main(String args[]) {
 G g = new G(); g.new Z().m1();
 }
}
```

Which name or expression could be used in place of ??? to cause the program to print "G.s1"?

- a. s1
  - b. G.s1
  - c. ((G)this).s1
  - d. G.this.s1
  - e. G.super.s1
  - f. None of the above
- 

## Question 8

```
class Outer {
 static class StaticNested {
 static final int a = 25; // 1
 static final int b; // 2
 static int c; // 3
 int d; // 4
 static {b = 42;} // 5
 }
 class NonStaticInner {
 static final int e = 25; // 6
 static final int f; // 7
 static int g; // 8
 int h; // 9
 static {f = 42;} // 10
 }
}
```

Compile-time errors are generated at which lines?

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5
- f. 6
- g. 7
- h. 8
- i. 9

## Question 9

```
class Red {
 private static final int a = 10; // 1
 protected static int b = 20; // 2
 int c = 30; // 3
 static class StaticNested {
 int d = a; // 4
 int e = b; // 5
 int f = c; // 6
 }
 class NonStaticInner {
 int g = a; // 7
 int h = b; // 8
 int i = c; // 9
 }
}
```

A compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. 5
  - f. 6
  - g. 7
  - h. 8
  - i. 9
- 

## Question 10

```
class Red {
 static class StaticNested {interface ABC {}} // 1
 class NonStaticInner {interface DEF {}} // 2
```

```
interface GHI {} // 3
}
```

A compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. None of the above
- 

## Question 11

```
class A {
 private static int counter;
 public static int getCounter(){return counter++;}
 private static int innerCounter;
 public static int getInnerCounter(){return innerCounter++;}
 private String name;
 A() {name = "A" + getCounter();}
 class B {
 private String name;
 B() {
 name = "B" + getInnerCounter();
 System.out.print(A.this.name + name); // 1
 }
 }
 public static void main(String[] args) {
 new A().new B(); // 2
 A a1 = new A();
 a1.new B(); // 3
 a1.new B(); // 4
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A0B0A1B1A1B2
- b. Prints: A0B0A1B1A2B2
- c. Compile-time error at line 1
- d. Compile-time error at line 2

- e. Compile-time error at line 3
  - f. Compile-time error at line 4
  - g. Other compile-time error.
  - h. Run-time error
  - i. None of the above
- 

## Question 12

```
class A {
 private static int counter;
 public static int getCounter(){return counter++;}
 private static int innerCounter;
 public static int getInnerCounter(){return innerCounter++;}
 private String name;
 A() {name = "A" + getCounter();}
 class B {
 private String name;
 B() {
 name = "B" + getInnerCounter();
 System.out.print(A.this.name + name); // 1
 }
 void m1() {new A().new B();} // 2
 void m2() {this.new B();} // 3
 void m3() {new B();} // 4
 }
 public static void main(String[] args) {
 A a1 = new A();
 a1.m1(); a1.m2(); a1.m3();
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A0B0A1B1A1B2
- b. Prints: A0B0A1B1A2B2
- c. Prints: A1B0A0B1A0B2
- d. Compile-time error at line 1
- e. Compile-time error at line 2
- f. Compile-time error at line 3
- g. Compile-time error at line 4

- h. Other compile-time error.
  - i. Run-time error
  - j. None of the above
- 

## Question 13

```
class A {
 private static int counter;
 public static int getCounter(){return counter++;}
 private static int innerCounter;
 public static int getInnerCounter(){return innerCounter++;}
 private String name;
 A() {name = "A" + getCounter();}
 class B {
 private String name;
 B() {
 name = "B" + getInnerCounter();
 System.out.print(A.this.name + name);
 }
 void m1() {new A().new B();} // 1
 void m2() {new A.B();} // 2
 void m3() {new B();} // 3
 }
 public static void main(String[] args) {
 A a1 = new A();
 a1.m1(); a1.m2(); a1.m3();
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A0B0A1B1A1B2
- b. Prints: A0B0A1B1A2B2
- c. Prints: A1B0A0B1A0B2
- d. Compile-time error at line 1
- e. Compile-time error at line 2
- f. Compile-time error at line 3
- g. Compile-time error at line 4
- h. Other compile-time error.
- i. Run-time error

- j. None of the above
- 

## Question 14

```
class A {
 private static int counter;
 public static int getCounter(){return counter++;}
 private static int innerCounter;
 public static int getInnerCounter(){return innerCounter++;}
 private String name;
 A() {name = "A" + getCounter();}
 class B {
 private String name;
 B() {
 name = "B" + getInnerCounter();
 System.out.print(A.this.name + name); // 1
 }
 static void m1() {new A().new B();} // 2
 static void m2() {this.new B();} // 3
 static void m3() {new B();} // 4
 public static void main(String[] args) {
 m1(); m2(); m3();
 }
 }
}
```

What are the results of attempting to compile and run the program?

- a. Prints: A0B0A1B1A1B2
- b. Prints: A0B0A1B1A2B2
- c. Prints: A1B0A0B1A0B2
- d. Compile-time error at line 1
- e. Compile-time error at line 2
- f. Compile-time error at line 3
- g. Compile-time error at line 4

No.      Answer      Remark

1      g      None of the above      Please note that this question asks which variable can NOT be substituted. Class Z is a non-static member class of class A; therefore, all of the fields and methods of class A (static, non-static, and private) are available to class Z.

2 b s2 Please note that this question asks which variable can NOT be substituted. Class Z is a static member class of class B; therefore, all of the static fields and methods of the enclosing class B are available to Z. For example, from within the static member class Z, the static field s1 of the enclosing class B can be accessed using the simple name s1. The simple name s1 does not need to be qualified with a reference to an instance of the enclosing class B, because s1 is not associated with a particular instance of the enclosing class. In contrast, non-static fields and methods of the enclosing class B are associated with a particular instance of class B. From the static context of class Z, the instance fields and methods of the enclosing class B can be accessed only if the simple name of the instance field or method is qualified with a reference to a specific instance of class B. Suppose a reference variable r1 refers to an instance of the enclosing class B. Then the instance member s2 of the enclosing class instance referenced by r1 could be accessed using the expression r1.s2.

3 a b c e s1 s2 s3 s5 Class Z is a local class defined within the code block of method m1 of class C. All of the fields and methods of class C (static, non-static, and private) are available to Class Z. Additionally, the parameters of method m1 that are declared final are available to class Z. If a final local variable declaration appears before the declaration of class Z, then it is also available to class Z. Parameter s6 and variable s4 are not final and are therefore not available to class Z. There are at least two reasons why non-final variables are not available to a local inner class. The first reason is because the life cycle of a local variable might be shorter than that of a local class. A local variable lives on the stack and disappears as soon as the method is exited. A local class, however, might continue to exist even after the method has been exited. The second reason is due to multithreading issues. Suppose a local class extends the Thread class or implements the Runnable interface, and it is used to spawn a new Thread from within the method m1. If the local variables of method m1 were available to the local class, then a mechanism for synchronizing access to the variables would be required.

4 f Compile-time error An instance of class Z must be associated with an enclosing instance of class D. In a static context, an unqualified class instance creation expression of the form new ClassType(ArgumentListopt) ClassBodyopt can not be used to create an instance of an inner class. Instead, a qualified class instance creation expression of the form Reference.new Identifier(ArgumentListopt) ClassBodyopt is required to create an association between an instance of the enclosing class and the new instance of the inner class. The reference could be provided by a reference variable of the type of the enclosing class, or it could be provided by a class instance creation expression such as new D(). In a static context, the expression new D().new Z() can be used to create the new instance of the enclosing class D and the new instance of the inner class Z.

5 b Prints: Z Class Z is a static member class of class E. Static member classes are similar to ordinary top-level classes with the added advantage that all of the static fields and methods of the enclosing class (including those that are private) are available to the member class. The class instance creation expression new E.Z() creates an instance of the static nested class E.Z. The instance of the static nested class is not associated with any instance of the enclosing class, and no instance of the enclosing class is created. For that reason, only the letter "Z" is printed.

6 e Prints: Z.m1 Y.m1 The private fields and methods of the member classes are available to the enclosing class. The private fields and methods of the member classes are also available to other member classes.

7 d G.this.s1 The qualified this expression ClassName.this.name can be used to access shadowed variables declared within an enclosing class.

8 g h j 7 8 10 A non-static nested class is called an inner class. An inner class can not declare a static field unless it is a compile-time constant. Even though f is final, it does not have a value at compile time; so it causes a compilation error. Member variable g also causes a compile-time

error, because it is static. The static initializer of NonStaticInner causes a compile-time error, because inner classes (i.e. non-static nested classes) can not declare static initializers.

9 f 6 The non-static members of an enclosing class are not directly available to a static nested class. From within StaticNested, the non-static members of the enclosing class can not be referred to by a simple name. Instead, a qualified name is required. Suppose a reference variable r1 refers to an instance of the enclosing class Red. Then the instance member c of the enclosing class instance referenced by r1 could be accessed using the qualified name r1.c.

10 b 2 A member interface is implicitly static; therefore, it can not be declared as a member of a non-static nested class.

11 a Prints: A0B0A1B1A1B2 Class A is the enclosing class of the inner class B. An instance of class B must be associated with an enclosing instance of class A. In a static context such as the main method, instantiation of a named inner class requires the use of a qualified class instance creation expression of the form Reference.new Identifier(ArgumentListopt). The reference could be provided by a reference variable of the type of the enclosing class, or it could be provided by another class instance creation expression. At line 2, the qualified class instance creation expression new A().new B() first creates a new instance of the enclosing class A, then it creates an instance of B. The new instance of A is the first instance created; so the name is A0. The new instance of B is the first instance created; so the name is B0. At line 3, the qualified class instance creation expression a1.new B() creates an instance of B that is associated with a previously existing instance of class A that is referenced by variable a1. The instance of class A referenced by variable a1 is the second instance created so the name is A1. The new instance of B is the second instance created; so the name is B1. At line 4, a new instance of B is created and associated with the instance of A this is referenced by variable a1.

12 c Prints: A1B0A0B1A0B2 Class A is the enclosing class of the inner class B. An instance of class B must be associated with an enclosing instance of class A. In a static context, instantiation of a named inner class requires the use of a qualified class instance creation expression of the form Reference.new Identifier(ArgumentListopt). The reference could be provided by a reference variable of the type of the enclosing class, or it could be provided by another class instance creation expression. If the enclosing class is not an inner class, then the enclosing class could be instantiated with an unqualified class instance creation expression such as the following, new EnclosingClass(). The qualified class instance creation expression new A().new B() first creates a new instance of A, then it creates an instance of B. The new instance of A is the second instance created; so the name is A1. The new instance of B is the first instance created; so the name is B0. In the qualified class instance creation expression this.new B(), the keyword this, denotes a reference to the enclosing instance on which the method m2 has been invoked. A new instance of the enclosing class is not created; so the name of the enclosing instance is A0. The new instance of B is the second instance created; so the name is B1. Since method m3 is an instance method, the inner class B can be instantiated using the unqualified class instance creation expression new B(). The enclosing instance is the instance on which the method m3 has been invoked. It is the same instance that is referenced by the keyword this and the reference variable a1.

13 c Prints: A1B0A0B1A0B2 Class A is the enclosing class of the inner class B. An instance of class B must be associated with an enclosing instance of class A. In a static context, instantiation of a named inner class requires the use of a qualified class instance creation expression of the form Reference.new Identifier(ArgumentListopt). The reference could be provided by a reference variable of the type of the enclosing class, or it could be provided by another class instance creation expression. If the enclosing class is not an inner class, then the enclosing class could be instantiated with an unqualified class instance creation expression such as the following, new EnclosingClass(). At line 1, the qualified class instance creation expression new A().new B() first creates a new instance of A, then it creates an instance of B. The new instance of A is the second instance created; so the

name is A1. The new instance of B is the first instance created; so the name is B0. At line 2, a new instance of the named inner class B is created using the class instance creation expression new A.B(). The fully qualified name of class B is A.B. Since the class instance creation expression new A.B() appears within a method that is a member of class A, the use of the fully qualified name is unnecessary. Within method A.m2, the class instance creation expression new A.B() could be replaced by the expression new B() without changing the result. Using either expression, a new instance of class B is created without creating a new instance of class A. Instead, the new instance of class B is associated with the same instance of class A on which the method m2 has been invoked. It is the same instance of class A that is referenced by the keyword this and the reference variable a1. Since it was the first instance created, the name is A0. The new instance of B is the second instance created; so the name is B1. At line 3, a new instance of the named inner class B is created using the unqualified class instance creation expression new B(). The new instance of B is the third instance created; so the name is B2. The new instance of the inner class is associated with the same instance of the enclosing class on which the method m3 has been invoked. It is the same instance that is referenced by the keyword this and the reference variable a1. Since it was the first instance created, the name is A0.

14 f g    Compile-time error at line 3    Compile-time error at line 4    Class A is the enclosing class of the inner class B. An instance of class B must be associated with an enclosing instance of class A. In a static context, instantiation of a named inner class requires the use of a qualified class instance creation expression of the form Reference.new Identifier(ArgumentListopt). The reference could be provided by a reference variable of the type of the enclosing class, or it could be provided by another class instance creation expression. If the enclosing class is not an inner class, then the enclosing class could be instantiated with an unqualified class instance creation expression such as the following, new EnclosingClass(). The qualified class instance creation expression new A().new B() first creates a new instance of A, then it creates an instance of B. The qualified class instance creation expression this.new B() generates a compile-time error, because the keyword this can not be used within a static method. Since methods m1, m2 and m3 are static methods, an instance of the named inner class B can not be created inside any of the three methods using an unqualified class instance creation expression of the form new ClassType(ArgumentListopt).

## 6.4Overloading:

### Question 1

```
class A {void m1(A a) {System.out.print("A");}}
```

```
class B extends A {void m1(B b) {System.out.print("B");}}
```

```
class C extends B {void m1(C c) {System.out.print("C");}}
```

```
class D extends C {
```

```
 void m1(D d) {System.out.print("D");}
```

```
 public static void main(String[] args) {
```

```
 A a1 = new A(); B b1 = new B(); C c1 = new C(); D d1 = new D();
```

```
 d1.m1(a1); d1.m1(b1); d1.m1(c1);
```

```
 }}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
  - b. Prints: ABC
  - c. Prints: DDD
  - d. Prints: ABCD
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

## Question 2

```
class A {} class B extends A {} class C extends B {}
class D {
 void m1(A a) {System.out.print("A");}
 void m1(B b) {System.out.print("B");}
 void m1(C c) {System.out.print("C");}
 public static void main(String[] args) {
 A c1 = new C(); B c2 = new C(); C c3 = new C(); D d1 = new D();
 d1.m1(c1); d1.m1(c2); d1.m1(c3);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
  - b. Prints: ABC
  - c. Prints: CCC
  - d. Compile-time error
  - e. Run-time error
  - f. None of the above
- 

## Question 3

```
class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
```

```
public static void main(String[] args) {
 A c1 = new C(); B c2 = new C(); C c3 = new C(); C c4 = new C();
 c4.m1(c1); c4.m1(c2); c4.m1(c3);
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
  - b. Prints: ABC**
  - c. Prints: CCC
  - d. Compile-time error
  - e. Run-time error
  - f. None of the above
- 

## Question 4

```
class A { void m1(A a) {System.out.print("A");} }
class B extends A {void m1(B b) {System.out.print("B");} }
class C extends B {void m1(C c) {System.out.print("C");} }
class D {
 public static void main(String[] args) {
 A c1 = new C(); C c2 = new C(); c1.m1(c2);
 } }
```

What is the result of attempting to compile and run the program?

- a. Prints: A**
  - b. Prints: B
  - c. Prints: C
  - d. Compile-time error
  - e. Run-time error
  - f. None of the above
- 

## Question 5

```
class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
 public static void main(String[] args) {
 A a1 = new A(); A b1 = new B(); A c1 = new C(); C c4 = new C();
 a1.m1(c4); b1.m1(c4); c1.m1(c4);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
  - b. Prints: ABC
  - c. Prints: CCC
  - d. Compile-time error
  - e. Run-time error
  - f. None of the above
- 

## Question 6

```
class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
 public static void main(String[] args) {
 A a1 = new A(); B b1 = new B(); C c1 = new C(); A c2 = new C();
 c2.m1(a1); c2.m1(b1); c2.m1(c1);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC
- c. Prints: CCC
- d. Compile-time error
- e. Run-time error
- f. None of the above

---

## Question 7

```
class A { void m1(A a) {System.out.print("A");} }
class B extends A { void m1(B b) {System.out.print("B");} }
class C extends B { void m1(C c) {System.out.print("C");} }
class D {
 public static void main(String[] args) {
 A a1 = new A(); B b1 = new A(); C c1 = new A(); C c2 = new C();
 c2.m1(a1); c2.m1(b1); c2.m1(c1);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
  - b. Prints: ABC
  - c. Prints: CCC
  - d. Compile-time error
  - e. Run-time error
  - f. None of the above
- 

## Question 8

```
class A { void m1(A a) {System.out.print("A");} }
class B extends A { void m1(B b) {System.out.print("B");} }
class C extends B { void m1(C c) {System.out.print("C");} }
class D {
 public static void main(String[] args) {
 A a1 = new A(); B b1 = new B(); C c1 = new C(); C c2 = new A();
 c2.m1(a1); c2.m1(b1); c2.m1(c1);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: AAA
- b. Prints: ABC

- c. Prints: CCC
- d. Compile-time error
- e. Run-time error
- f. None of the above

**No.**    **Answer**        **Remark**

1       b       Prints: ABC     The method invocation expression `d1.m1(a1)` uses reference `d1` of type `D` to invoke method `m1`. Since the reference `d1` is of type `D`, the class `D` is searched for an applicable implementation of `m1`. The methods inherited from the superclasses, `C`, `B` and `A`, are included in the search. The argument, `a1`, is a variable declared with the type `A`; so method `A.m1(A a)` is invoked.

2       b       Prints: ABC     Three methods overload the method name `m1`. Each has a single parameter of type `A` or `B` or `C`. For any method invocation expression of the form `m1(referenceArgument)`, the method is selected based on the declared type of the variable `referenceArgument`--not the run-time type of the referenced object. The method invocation expression `d1.m1(c1)` uses reference `d1` of type `D` to invoke method `m1` on an instance of type `D`. The argument, `c1`, is a reference of type `A` and the run-time type of the referenced object is `C`. The argument type is determined by the declared type of the reference variable `c1`--not the run-time type of the object referenced by `c1`. The declared type of `c1` is type `A`; so the method `m1(A a)` is selected. The declared type of `c2` is type `B`; so the method invocation expression `d1.m1(c2)` invokes method `m1(B b)`. The declared type of `c3` is type `C`; so the method invocation expression `d1.m1(c3)` invokes method `m1(C c)`.

3       b       Prints: ABC     Three methods overload the method name `m1`. Each has a single parameter of type `A` or `B` or `C`. For any method invocation expression of the form `m1(referenceArgument)`, the method is selected based on the declared type of the variable `referenceArgument`--not the run-time type of the referenced object. The method invocation expression `c4.m1(c1)` uses reference `c4` of type `C` to invoke method `m1` on an instance of type `C`. The argument, `c1`, is a reference of type `A` and the run-time type of the referenced object is `C`. The argument type is determined by the declared type of the reference variable `c1`--not the run-time type of the object referenced by `c1`. The declared type of `c1` is type `A`; so the method `A.m1(A a)` is selected. The declared type of `c2` is type `B`; so the method invocation expression `c4.m1(c2)` invokes method `B.m1(B b)`. The declared type of `c3` is type `C`; so the method invocation expression `c4.m1(c3)` invokes method `C.m1(C c)`.

4       a       Prints: A       The reference `c1` is of the superclass type, `A`; so it can be used to invoke only the method `m1` declared in class `A`. The methods that overload the method name `m1` in the subclasses, `B` and `C`, can not be invoked using the reference `c1`. A method invocation conversion promotes the argument referenced by `c2` from type `C` to type `A`, and the method declared in class `A` is executed. Class `A` declares only one method, `m1`. The single parameter is of type `A`. Class `B` inherits the method declared in class `A` and overloads the method name with a new method that has a single parameter of type `B`. Both methods sharing the overloaded name, `m1`, can be invoked using a reference of type `B`; however, a reference of type `A` can be used to invoke only the method declared in class `A`. Class `C` inherits the methods declared in classes `A` and `B` and overloads the method name with a new method that has a single parameter of type `C`. All three methods sharing the overloaded name, `m1`, can be invoked using a reference of type `C`; however, a reference of type `B` can be used to invoke only the method declared in class `B` and the method declared in the superclass `A`. The method invocation expression `c1.m1(c2)` uses reference `c1` of type `A` to invoke method `m1`. Since the reference `c1` is of type `A`, the search for an applicable implementation of `m1` is limited to class `A`. The subclasses, `B` and `C`, will not be searched; so the overloading methods declared in the subclasses can not be invoked using a reference of the superclass type.

5        a        Prints: AAA     The declared type of the reference variables, a1, b1 and c1, is the superclass type, A; so the three reference variables can be used to invoke only the method m1(A a) that is declared in the superclass, A. The methods that overload the method name m1 in the subclasses, B and C, can not be invoked using a reference variable of the superclass type, A. A method invocation conversion promotes the argument referenced by c4 from type C to type A, and the method declared in class A is executed.

6        a        Prints: AAA     The reference c2 is of the superclass type, A; so it can be used to invoke only the method, m1, declared in class A. The methods that overload the method name m1 in the subclasses, B and C, can not be invoked using the reference c2.

7        d        Compile-time error     The declarations of b1 and c1 cause compile-time errors, because a reference of a subclass type can not refer to an instance of the superclass type.

8        d        Compile-time error     The declaration of c2 causes a compile-time error, because a reference of a subclass type can not refer to an instance of the superclass class.

## 7.Threads

### Exam1:

#### Question 1

Which of the following methods are members of the Object class?

- a. Join
  - b. **notify**
  - c. **notifyAll**
  - d. Run
  - e. sleep
  - f. Start
  - g. yield
  - h. **Wait**
- 

#### Question 2

Which of the following methods are static members of the Thread class?

- a. Join
  - b. Notify
  - c. NotifyAll
  - d. Run
  - e. sleep
  - f. start
  - g. Yield
  - h. Wait
- 

## Question 3

Which of the following methods are deprecated members of the Thread class?

- a. Join
  - b. Notify
  - c. NotifyAll
  - d. Resume
  - e. Run
  - f. Sleep
  - g. Start
  - h. Stop
  - i. Suspend
  - j. Yield
  - k. Wait
- 

## Question 4

Which of the following methods name the InterruptedException in its throws clause?

- a. Join
- b. Notify

- c. NotifyAll
  - d. Run
  - e. Sleep
  - f. Start
  - g. Yield
  - h. Wait
- 

## Question 5

A timeout argument can be passed to which of the following methods?

- a. Join
  - b. Notify
  - c. NotifyAll
  - d. Run
  - e. Sleep
  - f. Start
  - g. Yield
  - h. Wait
- 

## Question 6

Which of the following instance methods should only be called by a thread that holds the lock of the instance on which the method is invoked?

- a. join
  - b. Notify
  - c. notifyAll
  - d. run
  - e. start
  - f. Wait
-

## Question 7

Which of the following is a checked exception?

- a. IllegalMonitorStateException
  - b. IllegalThreadStateException
  - c. IllegalArgumentException
  - d. **InterruptedException**
  - e. None of the above
- 

## Question 8

Which kind of variable would you prefer to synchronize on?

- a. A member variable of a primitive type
  - b. **A member variable that is an object reference**
  - c. A method local variable that is a reference to an instance that is created within the method
  - d. None of the above
- 

## Question 9

synchronized (expression) block

The synchronized statement has the form shown above. Which of the following are true statements?

- a. A compile-time error occurs if the expression produces a value of any reference type
- b. **A compile-time error occurs if the expression produces a value of any primitive type**
- c. A compile-time error does not occur if the expression is of type boolean
- d. The synchronized block may be processed normally if the expression is null
- e. **If execution of the block completes normally, then the lock is released**
- f. **If execution of the block completes abruptly, then the lock is released**

- g. A thread can hold more than one lock at a time
  - h. Synchronized statements can be nested
  - i. Synchronized statements with identical expressions can be nested
- 

## Question 10

Which of the following is a true statement?

- a. The process of executing a synchronized method requires the thread to acquire a lock
  - b. Any overriding method of a synchronized method is implicitly synchronized
  - c. If any method in a class is synchronized, then the class itself must also be declared using the synchronized modifier
  - d. If a thread invokes a static synchronized method on an instance of class A, then the thread must acquire the lock of that instance of class A
  - e. None of the above
- 

## Question 11

Which of the following thread state transitions model the lifecycle of a thread?

- a. The Dead state to the Ready state
  - b. The Ready state to the Not-Runnable state
  - c. The Ready state to the Running state
  - d. The Running state to the Not-Runnable state
  - e. The Running state to the Ready state
  - f. The Not-Runnable state to the Ready state
  - g. The Not-Runnable state to the Running state
- 

## Question 12

Which of the following are true statements?

- a. The Thread.yield method might cause the thread to move to the Not-Runnable state
  - b. The Thread.yield method might cause the thread to move to the Ready state
  - c. The same thread might continue to run after calling the Thread.yield method
  - d. The Thread.yield method is a static method
  - e. The behavior of the Thread.yield method is consistent from one platform to the next
  - f. The Thread.sleep method causes the thread to move to the Not-Runnable state
  - g. The Thread.sleep method causes the thread to move to the Ready state
- 

## Question 13

Which of the following will not force a thread to move into the Not-Runnable state?

- a. Thread.yield method
  - b. Thread.sleep method
  - c. Thread.join method
  - d. Object.wait method
  - e. By blocking on I/O
  - f. Unsuccessfully attempting to acquire the lock of an object
  - g. None of the above
- 

## Question 14

Which of the following will cause a dead thread to restart?

- a. Thread.yield method
  - b. Thread.join method
  - c. Thread.start method
  - d. Thread.resume method
  - e. None of the above
-

## Question 15

When a thread is created and started, what is its initial state?

- a. New
  - b. Ready
  - c. Not-Runnable
  - d. Runnning
  - e. Dead
  - f. None of the above
- 

## Question 16

Which of the following are true statements?

- a. The Thread.run method is used to start a new thread running
  - b. The Thread.start method causes a new thread to get ready to run at the discretion of the thread scheduler
  - c. The Runnable interface declares the start method
  - d. The Runnable interface declares the run method
  - e. The Thread class implements the Runnable interface
  - f. If an Object.notify method call appears in a synchronized block, then it must be the last method call in the block
  - g. No restriction is placed on the number of threads that can enter a synchronized method
  - h. Some implementations of the Thread.yield method will not yield to a thread of lower priority
- 

## Question 17

Which of the following are true statements?

- a. Thread.MAX\_PRIORITY == 10
- b. Thread.MAX\_PRIORITY == 5
- c. Thread.NORM\_PRIORITY == 5

- d. Thread.NORM\_PRIORITY == 3
  - e. Thread.NORM\_PRIORITY == 0
  - f. Thread.MIN\_PRIORITY == 1
  - g. Thread.MIN\_PRIORITY == 0
  - h. Thread.MIN\_PRIORITY == -5
  - i. Thread.MIN\_PRIORITY == -10
- 

## Question 18

Which of the following are true statements?

- a. A program will terminate only when all daemon threads stop running
  - b. A program will terminate only when all user threads stop running
  - c. A daemon thread always runs at Thread.MIN\_PRIORITY
  - d. A thread inherits its daemon status from the thread that created it
  - e. The daemon status of a thread can be changed at any time using the Thread.setDaemon method
  - f. The Thread.setDaemon method accepts one of two argument values defined by the constants Thread.DAEMON and Thread.USER
- 

## Question 19

```
class A extends Thread {
 public A(Runnable r) {super(r);}
 public void run() {System.out.print("A");}
}
class B implements Runnable {
 public void run() {System.out.print("B");}
}
class C {
 public static void main(String[] args) {
 new A(new B()).start();
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
  - b. Prints: B
  - c. Prints: AB
  - d. Prints: BA
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

## Question 20

```
class A implements Runnable {
 public void run() {System.out.print(Thread.currentThread().getName());}
}
class B implements Runnable {
 public void run() {
 new A().run();
 new Thread(new A(),"T2").run();
 new Thread(new A(),"T3").start();
 }
}
class C {
 public static void main (String[] args) {
 new Thread(new B(),"T1").start();
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: T1T1T1
  - b. Prints: T1T1T2
  - c. Prints: T1T2T2
  - d. Prints: T1T2T3
  - e. Prints: T1T1T3
  - f. Prints: T1T3T3
  - g. Compile-time error
  - h. Run-time error
  - i. None of the above
-

## Question 21

```
class AnException extends Exception {}
class A extends Thread {
 public void run() throws AnException {
 System.out.print("A"); throw new AnException();
 }
}
class B {
 public static void main (String[] args) {
 A a = new A(); a.start(); System.out.print("B");
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: A
  - b. Prints: B
  - c. Prints: AB
  - d. Prints: BA
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

## Question 22

```
class A extends Thread {
 public void run() {System.out.print("A");}
}
class B {
 public static void main (String[] args) {
 A a = new A();
 a.start();
 a.start(); // 1
 }
}
```

What is the result of attempting to compile and run the program?

- a. The program compiles and runs without error

- b. The second attempt to start thread t1 is successful
- c. The second attempt to start thread t1 is ignored
- d. Compile-time error at marker 1
- e. An IllegalThreadStateException is thrown at run-time
- f. None of the above

No.      Answer      Remark

1      b c h    notify    notifyAll    wait

2      e g      sleep    yield

3      d h i    resume    stop    suspend      **For the purposes of the exam, you don't need to memorize the deprecated methods of the Thread class.**  
**Even though a question such as this will not be on the exam, every Java programmer should know that the deprecated methods should not be used in new programs.**

4      a e h    join    sleep    wait

5      a e h    join    sleep    wait

6      b c f    notify    notifyAll    wait

7      d      InterruptedException      **The methods Object.wait, Thread.join and Thread.sleep name InterruptedException in their throws clauses.**

8      b      A member variable that is an object reference      Primitives don't have locks; therefore, they can not be used to synchronize threads.  
**A method local variable that is a reference to an instance that is created within the method should not be used to synchronize threads, because each thread has its own instance of the object and lock. Synchronization on an instance that is created locally makes about as much sense as placing on your doorstep a box full of keys to the door. Each person that comes to your door would have their own copy of the key; so the lock would provide no security.**

9      b e f g h i    A compile-time error occurs if the expression produces a value of any primitive type If execution of the block completes normally, then the lock is released If execution of the block completes abruptly, then the lock is released A thread can hold more than one lock at a time  
**Synchronized statements can be nested Synchronized statements with identical expressions can be nested**

10     a      The process of executing a synchronized method requires the thread to acquire a lock      The synchronized modifier can not be applied to a class. A method that overrides a synchronized method does not have to be synchronized. If a thread invokes a synchronized instance method on an instance of class A, then the thread must acquire the lock of that instance of class A. The same is not true for synchronized static methods. A synchronized static method is synchronized on the lock for the Class object that represents the class for which the method is a member.

11 c d e f      The Ready state to the Running state   The Running state to the Not-Runnable state   The Running state to the Ready state   The Not-Runnable state to the Ready state      A dead thread can not be restarted.

12 b c d f      The Thread.yield method might cause the thread to move to the Ready state   The same thread might continue to run after calling the Thread.yield method   The Thread.yield method is a static method   The Thread.sleep method causes the thread to move to the Not-Runnable state

The Thread.yield method is intended to cause the currently executing thread to move from the Running state to the Ready state and offer the thread scheduler an opportunity to allow a different thread to execute based on the discretion of the thread scheduler. The thread scheduler may select the same thread to run immediately, or it may allow a different thread to run. The Thread.yield method is a native method; so the behavior is not guaranteed to be the same on every platform. However, at least some implementations of the yield method will not yield to a thread that has a lower priority.

13 a      Thread.yield method   The Thread.yield method may cause a thread to move into the Ready state, but that state transition is not guaranteed. The JLS states that the Thread.yield method provides a hint to the thread scheduler, but the scheduler is free to interpret--or ignore--the hint as it sees fit. Nothing in the JLS suggests that the thread might move to the Not-Runnable state.

14 e      None of the above      A dead thread can not be restarted.

15 b      Ready

16 b d e h      The Thread.start method causes a new thread to get ready to run at the discretion of the thread scheduler   The Runnable interface declares the run method   The Thread class implements the Runnable interface   Some implementations of the Thread.yield method will not yield to a thread of lower priority      The Object.notify method can only be called by the thread that holds the lock of the object on which the method is invoked. Suppose that thread T1 enters a block that is synchronized on an object, A. Within the block, thread T1 holds the lock of A. Even if thread T1 calls the notify method immediately after entering the synchronized block, no other thread can grab the lock of object A until T1 leaves the synchronized block. For that reason, the transfer of control from thread T1 to any waiting thread can not be accelerated by moving the notify method to an earlier point in the synchronized block. The behavior of Thread.yield is platform specific. However, at least some implementations of the yield method will not yield to a thread that has a lower priority. Invoking the Thread.yield method is like offering a suggestion to the JVM to allow another thread to run. The response to the suggestion is platform specific.

17 a c f   Thread.MAX\_PRIORITY == 10   Thread.NORM\_PRIORITY == 5   Thread.MIN\_PRIORITY == 1

18 b d      A program will terminate only when all user threads stop running   A thread inherits its daemon status from the thread that created it

19 a      Prints: A      If a Runnable target object is passed to the constructor of the Thread class, then the Thread.run method will invoke the run method of the Runnable target. In this case, the Thread.run method is overridden by A.run. The A.run method does nothing more than print the letter A. The invocation of the A.start method inside the main method results in the invocation of A.run, and the letter A is printed. The B.run method is never invoked.

20 e      Prints: T1T1T3      The Thread.currentThread method returns a reference to the currently executing thread. When the run method is invoked directly it does not start a new thread; so T1 is printed twice.

21 e      **Compile-time error**    The Runnable.run method does not have a throws clause; so any implementation of run can not throw a checked exception.

22 e      An IllegalThreadStateException is thrown at run-time      For the purposes of the exam, invoking the start method on a thread that has already been started will generate an IllegalThreadStateException. The actual behavior of the method might be different. If the start method is invoked on a thread that is already running, then an IllegalThreadStateException will probably be thrown. However, if the thread is already dead then the second attempt to start the thread will probably be ignored, and no exception will be thrown. For the purposes of the exam, the exception is always thrown in response to the second invocation of the start method. This is a case where the exam tests your knowledge of the specification and ignores the actual behavior of the 1.4 version of the JVM.

## Exam 2:

### 7.2

#### Question 1

```
class A extends Thread {
 private int i;
 public void run() {i = 1;}
 public static void main(String[] args) {
 A a = new A(); a.start(); System.out.print(a.i);
 }
}
```

What are the possible results of attempting to compile and run the program?

- a. Prints nothing
  - b. Prints: 0
  - c. Prints: 1
  - d. Prints: 01
  - e. Prints: 10
  - f. Compile-time error
  - g. Run-time error
-

## Question 2

```
class A extends Thread {
 private int i;
 public void run() {i = 1;}
 public static void main(String[] args) {
 A a = new A(); a.run(); System.out.print(a.i);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints nothing
  - b. Prints: 0
  - c. Prints: 1
  - d. Prints: 01
  - e. Prints: 10
  - f. Compile-time error
  - g. Run-time error
  - h. None of the above
- 

## Question 3

```
class A extends Thread {
 public void run() {
 try {sleep(10000);} catch (InterruptedException ie){}
 }
 public static void main(String[] args) {
 A a1 = new A();
 long startTime = System.currentTimeMillis();
 a1.start();
 System.out.print(System.currentTimeMillis() - startTime);
 }
}
```

What are the possible results of attempting to compile and run the program?

- a. Prints a number greater than or equal to 0
- b. The number printed must always be greater than 10000

- c. This program will run for at least ten seconds
  - d. Compile-time error
  - e. Run-time error
- 

## Question 4

Which of the following is used to force each thread to reconcile its working copy of a variable with the master copy in main memory?

- a. Final
  - b. Static
  - c. synchronized
  - d. transient
  - e. volatile
  - f. native
- 

## Question 5

```
class A extends Thread {
 public void run() {
 synchronized (this) {
 try {wait(5000);} catch (InterruptedException ie){}
 }
 }
 public static void main(String[] args) {
 A a1 = new A();
 long startTime = System.currentTimeMillis();
 a1.start();
 System.out.print(System.currentTimeMillis() - startTime + ",");
 try {a1.join(6000);} catch (InterruptedException ie) {}
 System.out.print(System.currentTimeMillis() - startTime);
 }
}
```

What are the possible results of attempting to compile and run the program?

- a. The first number printed is greater than or equal to 0
- b. The first number printed must always be greater than 5000

- c. The second number printed must always be greater than 5000
  - d. The second number printed must always be greater than 6000
  - e. The synchronized block inside the run method is not necessary
  - f. Compile-time error
  - g. Run-time error
- 

## Question 6

```
class A extends Thread {
 String[] sa;
 public A(String[] sa) {this.sa = sa;}
 public void run() {
 synchronized (sa) {System.out.print(sa[0] + sa[1] + sa[2]);}
 }
}
class B {
 private static String[] sa = new String[]{"X","Y","Z"};
 public static void main (String[] args) {
 synchronized (sa) {
 Thread t1 = new A(sa); t1.start();
 sa[0] = "A"; sa[1] = "B"; sa[2] = "C";
 }}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: XYZ
  - b. Prints: AYZ
  - c. Prints: ABZ
  - d. Prints: ABC
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

## Question 7

```
class A extends Thread {
```

```

String[] sa;
public A(String[] sa) {this.sa = sa;}
public void run() {
 synchronized (sa) {
 while (!sa[0].equals("Done")) {
 try {sa.wait();} catch (InterruptedException ie) {}
 }
 System.out.print(sa[1] + sa[2] + sa[3]);
 }
}
class B {
 private static String[] sa = new String[]{"Not Done","X","Y","Z"};
 public static void main (String[] args) {
 Thread t1 = new A(sa); t1.start();
 synchronized (sa) {
 sa[0] = "Done";
 sa[1] = "A"; sa[2] = "B"; sa[3] = "C";
 sa.notify();
 }
 }
}

```

What is the result of attempting to compile and run the program?

- a. Prints: XYZ
  - b. Prints: AYZ
  - c. Prints: ABZ
  - d. Prints: ABC
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

## Question 8

Which of the following are true statements?

- a. The Thread.join method is static
- b. The Thread.join method is always invoked on an instance of Thread
- c. The Thread.join method causes the current thread to wait for the referenced thread to die
- d. The Thread.join method declares an InterruptedException in the throws clause

- e. The Thread.join method accepts a timeout value as an argument
  - f. The timeout value sets the minimum time that the current thread will wait for the death of the referenced thread
  - g. Thread.join will return immediately if the timeout value is zero
  - h. A timeout of zero will allow Thread.join to wait forever if necessary
- 

## Question 9

Which of the following allows a thread t1 to become the holder of the lock of object obj1.

- a. By blocking on I/O
  - b. By entering a synchronized instance method of the obj1
  - c. By invoking the wait method on the object
  - d. By entering the body of a block that is synchronized on obj1
  - e. By entering a synchronized static method of the obj1
  - f. By invoking the notify method on obj1
- 

## Question 10

After invoking the wait method on an object, obj1, a thread, T1, will remain in the wait set of obj1 until which of the following occurs?

- a. Another thread invokes the notify method on the object, obj1, and T1 is selected to move out of the wait set
  - b. Another thread invokes the notifyAll method on the object
  - c. Another thread invokes the resume method on thread T1
  - d. Another thread interrupts thread T1
  - e. The priority of thread T1 is increased
  - f. A specified timeout period has elapsed
  - g. Another thread invokes the join method on thread T1
- 

## Question 11

```
class A implements Runnable{public void run() {}}
class B {
 public static void main(String[] args) {
 Thread t1 = new Thread(); // 1
 Thread t2 = new Thread(new A()); // 2
 Thread t3 = new Thread(new A(), "A"); // 3
 Thread t4 = new Thread("A"); // 4
 }
}
```

A compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
- 

## Question 12

```
class A implements Runnable{public void run() {}}
class B {
 public static void main(String[] args) {
 Thread t1 = new Thread(); // 1
 Thread t2 = new Thread(new A()); // 2
 Thread t3 = new Thread("A", new A()); // 3
 Thread t4 = new Thread("A"); // 4
 }
}
```

A compile-time error is generated at which line?

- a. 1
  - b. 2
  - c. 3
  - d. 4
  - e. None of the above
-

## Question 13

```
class A extends Thread {
 private boolean done;
 public void setDone(boolean done) {this.done = done;}
 public void run() {
 synchronized (this) {
 while (!done) {try {wait();} catch (InterruptedException ie){}}
 }
 }
 public static void main(String[] args) {
 A a1 = new A();
 long startTime = System.currentTimeMillis();
 a1.start();
 System.out.print(System.currentTimeMillis() - startTime);
 }
}
```

Which is a possible result of attempting to compile and run the program?

- a. The number printed is greater than or equal to 0
  - b. The synchronized block inside the run method is not necessary
  - c. This program runs to completion after the elapsed time is printed
  - d. Compile-time error
  - e. Run-time error
  - f. None of the above
- 

## Question 14

```
class A extends Thread {
 public void run() {
 synchronized (this) {
 try {wait();} catch (InterruptedException ie){}
 }
 }
 public static void main(String[] args) {
 A a1 = new A(); a1.setDaemon(true);
 long startTime = System.currentTimeMillis();
 a1.start();
 System.out.print(System.currentTimeMillis() - startTime + ",");
 }
}
```

}}

Which is a possible result of attempting to compile and run the program?

- a. The number printed is greater than or equal to 0
  - b. The synchronized block inside the run method is not necessary
  - c. Thread a1 waits forever and the program runs forever
  - d. Compile-time error
  - e. Run-time error
  - f. None of the above
- 

## Question 15

```
class A extends Thread {
 private Object obj;
 public A(Object obj) {this.obj = obj;}
 public void run() {
 try {
 synchronized (obj) {obj.wait();}
 } catch (InterruptedException ie) {}
 System.out.print(Thread.currentThread().getName());
 }
}
class B {
 private void m1() {
 for (int i = 0; i < 10; i++) {
 A t1 = new A(this);
 t1.setName(String.valueOf(i)); t1.setDaemon(true); t1.start();
 }
 synchronized (this) {notifyAll();}
 }
 public static void main(String[] args) {new B().m1();}
}
```

What are the possible results of attempting to compile and run the program?

- a. All of the numbers 0 through 9 must always be printed
- b. Some or all of the numbers 0 through 9 could be printed
- c. Nothing is printed

d. Run-time error

---

## Question 16

```
class C extends Thread {
 private static String[] sa = new String[]{"Not Done","X","Y","Z"};
 public void run() {
 synchronized (sa) {
 while (!sa[0].equals("Done")) {
 try {sa.wait();} catch (InterruptedException ie) {}
 }
 System.out.print(sa[1] + sa[2] + sa[3]);
 }
 }
 public static void main (String[] args) {
 start();
 synchronized (sa) {
 sa[0] = "Done";
 sa[1] = "A"; sa[2] = "B"; sa[3] = "C";
 sa.notify();
 }
 }}}
```

Which is a possible result of attempting to compile and run the program?

- a. Prints: XYZ
  - b. Prints: AYZ
  - c. Prints: ABZ
  - d. Prints: ABC
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

## Question 17

```
class C extends Thread {
 private static String[] sa = new String[]{"Not Done","X","Y","Z"};
 public void run() {
 synchronized (this) {
```

```

while (!sa[0].equals("Done")) {
 try { wait(); } catch (InterruptedException ie) {}
}
System.out.print(sa[1] + sa[2] + sa[3]);
}
void m1() {
 start();
 synchronized (this) {
 sa[0] = "Done";
 sa[1] = "A"; sa[2] = "B"; sa[3] = "C";
 }
}
public static void main (String[] args) {
 new C().m1(); notify();
}

```

Which is a possible result of attempting to compile and run the program?

- a. Prints: XYZ
  - b. Prints: AYZ
  - c. Prints: ABZ
  - d. Prints: ABC
  - e. Compile-time error
  - f. Run-time error
  - g. None of the above
- 

## Question 18

```

class A extends Thread {
 public void run() {
 long startTime = System.currentTimeMillis();
 long endTime = startTime + 10000;
 while (System.currentTimeMillis() < endTime) {
 yield();
 }
 }
}
public static void main(String[] args) {
 A a1 = new A();
 long startTime = System.currentTimeMillis();
 a1.start(); sleep(1000); a1.interrupt(); a1.join();
 System.out.print(System.currentTimeMillis() - startTime);
}

```

Which is a possible result of attempting to compile and run the program?

- a. Prints a number that is less than 1000
  - b. Prints a number between 1000 and 9999
  - c. Prints a number larger than 10000
  - d. Compile-time error
  - e. Run-time error
  - f. None of the above
- 

## Question 19

```
class A extends Thread {
 public void run() {System.out.print("A");}
}
class B {
 public static void main (String[] args) {
 A a = new A(); a.start();
 try {
 a.join();
 } catch (InterruptedException ie) {ie.printStackTrace();}
 a.start(); // 1
 }
}
```

What is the result of attempting to compile and run the program?

- a. The program compiles and runs without error
  - b. The second attempt to start thread t1 is successful
  - c. The second attempt to start thread t1 is ignored
  - d. Compile-time error at marker 1
  - e. An IllegalThreadStateException is thrown at run-time
  - f. None of the above
- 

## Question 20

```
class A extends Thread {
```

```

private static B b = new B();
private String s1;
public void run() {System.out.print(b.m1(s1));}
A(String threadName, String s1) {
 super(threadName); this.s1 = s1;
}
public static void main (String[] args) {
 A a = new A("T1","A"), b = new A("T2","B"); a.start(); b.start();
}
class B {
 private String s1;
 public synchronized String m1(String s) {
 s1 = s;
 try {Thread.sleep(1);} catch (InterruptedException ie) {}
 return "["+Thread.currentThread().getName()+","+s1+"]";
}

```

What are the possible results of attempting to compile and run the program?

- a. Prints nothing
  - b. Prints: [T1,A][T2,B]
  - c. Prints: [T1,B][T2,B]
  - d. Prints: [T2,B][T1,A]
  - e. Prints: [T2,A][T1,A]
  - f. Compile-time error
  - g. Run-time error
- 

## Question 21

```

class A extends Thread {
 static long startTime;
 public void run() {
 for (int i = 0; i < 99999; i++) {Math.sin(i);}
 String name = Thread.currentThread().getName();
 long time = System.currentTimeMillis();
 System.out.println(name + " done at " + (time - startTime));
 }
 public static void main(String[] args) {
 A t1 = new A(); A t2 = new A();
 t1.setName("T1"); t2.setName("T2");
 }
}

```

```

t1.setPriority(Thread.MIN_PRIORITY);
t2.setPriority(Thread.MAX_PRIORITY);
startTime = System.currentTimeMillis();
t1.start(); t2.start();
}

```

Which of the following is a true statement?

- a. The priority assigned to thread T2 is greater than the priority assigned to T1
- b. Java guarantees that thread T2 will get more CPU time than T1
- c. Java guarantees that thread T2 will run to completion before T1
- d. None of the above

| No. | Answer                                         | Remark                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----|------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | b c Prints: 0 Prints: 1                        | The new thread is started before the print statement, but there is no guarantee that the new thread will run before the print statement is processed. The guarantee could be provided by placing the method invocation expression a.join() before the print statement, but the invocation of the join method does not appear in the program. If the new thread runs before the print statement is processed, then 1 is printed. Otherwise, 0 is printed.                                                                                                                                                                                                                                                                       |
| 2   | c Prints: 1                                    | The a.run() method was called instead of a.start(); so the entire program runs as a single thread, and a.run() is guaranteed to complete before the print statement is called.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 3   | a c Prints a number greater than or equal to 0 | This program will run for at least ten seconds Thread a1 will run for at least ten seconds, but the main method is likely to run to completion very quickly. The start method will return without waiting for thread a1 to complete. Since thread a1 immediately goes to sleep the thread that is processing the main method has an opportunity to complete the main method quickly. The number printed in the main method can be as small as zero.                                                                                                                                                                                                                                                                            |
| 4   | e volatile                                     | A field might be shared between two or more threads. Each thread is allowed to maintain a working copy of the field. If the threads do not reconcile the working copies then each might be working with a different value. The volatile modifier is used to force each thread to reconcile its working copy of the field with the master copy in main memory.                                                                                                                                                                                                                                                                                                                                                                  |
| 5   | a c                                            | The first number printed is greater than or equal to 0 The second number printed must always be greater than 5000 The notify method is never invoked on thread a1; so it will sleep for at least five seconds. The invocation of the join method forces the main thread to wait for the completion of thread a1. The argument of 6000 will allow the main thread to wait for six seconds if necessary, but we know that thread a1 will complete in only five seconds. The first number printed will be greater than or equal to zero, and the second number will be greater than or equal to 5000. The synchronized block is necessary, because it is necessary to hold the lock of an object when the wait method is invoked. |

6 d Prints: ABC The block inside the main method is synchronized on the String array object sa. Inside the block, a new thread t1 is started and will run at the discretion of the thread scheduler. The A.run method also contains a block that is synchronized on the String array object sa. Even if the thread scheduler moves thread t1 into the Running state, it will block while attempting to acquire the lock of the String array object sa. Thread t1 will continue to block until the synchronized block in the B.main method runs to completion. At that time, the contents of the String array object have all been updated.

7 d Prints: ABC Inside the main method, thread t1 is started and will move into the Running state at the discretion of the thread scheduler. The A.run method invokes the wait method on the String array object sa causing the thread to block until another thread invokes the sa.notify method. Before the B.main method invokes sa.notify, all of the elements of the String array object sa have already been updated.

8 b c d e h The Thread.join method is always invoked on an instance of Thread. The Thread.join method causes the current thread to wait for the referenced thread to die. The Thread.join method declares an InterruptedException in the throws clause. The Thread.join method accepts a timeout value as an argument. A timeout of zero will allow Thread.join to wait forever if necessary. The Thread.join method is not static. Thread.join is always invoked on an instance of Thread. Thread.join causes the current thread to wait until the referenced thread has died. The maximum time limit to wait for the death of the referenced thread can be specified in milliseconds by an argument. Thread.join will throw an InterruptedException if the interrupt method is invoked on the current Thread.

9 b d By entering a synchronized instance method of the obj1 By entering the body of a block that is synchronized on obj1 Blocking on I/O or invoking the Thread.sleep or Object.wait method causes a thread to enter the Not-Runnable state. Invoking the notify method on an object wakes up a thread that is waiting on the object. The thread that invokes wait or notify on an object should already hold the lock of the object. Invoking the wait or notify method does not cause the thread to hold the lock. Static methods are synchronized on the lock of the Class object of the class. Instance methods are synchronized on the lock of the instance of the class.

10 a b d f Another thread invokes the notify method on the object, obj1, and T1 is selected to move out of the wait set. Another thread invokes the notifyAll method on the object. Another thread interrupts thread T1. A specified timeout period has elapsed.

11 e None of the above All of the class instance creation expressions are legal. The String instance A is the name of the Thread. Yes, the exam requires you to memorize the Thread constructor signatures.

12 c 3 The position of the arguments have been reversed in the constructor on line 3. The Runnable argument should appear before the thread name argument. Yes, the exam requires you to memorize the Thread constructor signatures.

13 a The number printed is greater than or equal to 0 The main thread invokes the start method on thread a1. There is no way to predict when the new thread will start to run. At some point in time, the main thread will proceed to the print statement where the value of the startTime variable is subtracted from the current time. The value printed will be greater than or equal to one. At some point in time, thread a1 will begin to run and it will invoke the wait method. Since no other thread invokes the notify method on a1, it will wait forever, and the program will never run to completion.

14 a The number printed is greater than or equal to 0 The a1 thread is a daemon thread; so the program can run to completion even if thread a1 is still running, waiting or sleeping. The notify method is never invoked on thread a1. If thread a1 were not a daemon thread, then the program would wait forever. However, the program will run to completion without waiting for a1.

15 b c Some or all of the numbers 0 through 9 could be printed Nothing is printed All of the threads started in method B.m1 are daemon threads; so the program can run to completion even if some or all of the daemon threads have not run.

16 e Compile-time error Remember that the Thread.start method is an instance method and can not be invoked from a static context.

17 e Compile-time error Remember that the Object.notify method is an instance method and can not be invoked from a static context. Also, the thread that invokes the notify method on an object must hold the lock of the object.

18 d Compile-time error Both the sleep and join methods declare an InterruptedException that must be caught or declared in the throws clause of A.main.

19 e An IllegalThreadStateException is thrown at run-time For the purposes of the exam, invoking the start method on a thread that has already been started will generate an IllegalThreadStateException. The actual behavior of Java might be different. If the start method is invoked on a thread that is already running, then an IllegalThreadStateException will probably be thrown. However, if the thread is already dead then the second attempt to start the thread will probably be ignored, and no exception will be thrown. However, for the purposes of the exam, the exception is always thrown in response to the second invocation of the start method. This is a case where the exam tests your knowledge of the specification of the Thread.start method and ignores the actual behavior of the 1.4 version of the JVM. The Thread.join method is included here to verify that the thread is already dead before the start method is invoked the second time. If this code is executed using the 1.4 version of the JVM, the exception will not be thrown. However, for the purposes of the exam, the exception is always thrown. The real exam question will probably not include the invocation of the join method.

20 b d Prints: [T1,A][T2,B] Prints: [T2,B][T1,A] Since method m1 is synchronized, it is guaranteed that no more than one thread will execute the method at any one time. Even though the start method is invoked on thread T1 first, there is no guarantee that it will actually begin to run first.

21 a The priority assigned to thread T2 is greater than the priority assigned to T1 The Java Language Specification suggests that higher priority threads should be given preference over lower priority threads, but explicitly states that the preference is not a guarantee. It is very important to remember that no guarantee exists.

## Section 9: The Collections Framework

Collections

Question 1

Which implementation of the List interface produces the slowest access to an element in the middle of the list by means of an index?

- a. Vector
- b. ArrayList
- c. LinkedList
- d. None of the above

Question 2

```
import java.util.*;

class GFC100 {

 public static void main (String args[]) {

 Object a1 = new LinkedList(), b1 = new TreeSet();

 Object c1 = new TreeMap();

 System.out.print((a1 instanceof Collection)+",");

 System.out.print((b1 instanceof Collection)+",");

 System.out.print(c1 instanceof Collection);

 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 3

- Each element must be unique.
- Contains no duplicate elements.
- Elements are not key/value pairs.
- Accessing an element can be almost as fast as performing a similar operation on an array.

Which of these classes provides the specified features?

- a. LinkedList
- b. TreeMap
- c. TreeSet
- d. HashMap

- e. HashSet
- f. LinkedHashMap
- g. Hashtable
- h. None of the above

#### Question 4

```
import java.util.*;

class GFC101 {

 public static void main (String args[]) {

 Object a1 = new HashMap(), b1 = new ArrayList();

 Object c1 = new HashSet();

 System.out.print((a1 instanceof Collection)+",");

 System.out.print((b1 instanceof Collection)+",");

 System.out.print(c1 instanceof Collection);

 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true

- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

#### Question 5

- Entries are organized as key/value pairs.
- Duplicate entries replace old entries.

Which interface of the java.util package offers the specified behavior?

- a. List
- b. Map
- c. Set
- d. None of the above

#### Question 6

```
import java.util.*;

class GFC102 {

 public static void main (String args[]) {

 Object a = new HashSet();

 System.out.print((a instanceof Set)+",");
```

```
System.out.print(a instanceof SortedSet);
}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. None of the above

Question 7

Which implementation of the List interface provides for the fastest insertion of a new element into the middle of the list?

- a. Vector
- b. ArrayList
- c. LinkedList
- d. None of the above

Question 8

```
import java.util.*;

class GFC103 {

 public static void main (String args[]) {
```

```
Object a1 = new TreeSet();
System.out.print((a1 instanceof Set)+",");
System.out.print(a1 instanceof SortedSet);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false
- b. Prints: false,true
- c. Prints: true,false
- d. Prints: true,true
- e. None of the above

Question 9

- Stores key/value pairs.
- Duplicate entries replace old entries.
- Entries are sorted using a Comparator or the Comparable interface.

Which of these classes provides the specified features?

- a. LinkedList
- b. TreeMap
- c. TreeSet
- d. HashMap

- e. HashSet
- f. Hashtable
- g. None of the above

Question 10

```
import java.util.*;

class GFC104 {

 public static void main (String args[]) {

 LinkedList a1 = new LinkedList();

 ArrayList b1 = new ArrayList();

 Vector c1 = new Vector();

 System.out.print((a1 instanceof List)+",");

 System.out.print((b1 instanceof List)+",");

 System.out.print(c1 instanceof List);

 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true

- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

#### Question 11

- Entries are not organized as key/value pairs.
- Duplicate entries are rejected.

Which interface of the java.util package offers the specified behavior?

- a. List
- b. Map
- c. Set
- d. None of the above

#### Question 12

```
import java.util.*;

class GFC105 {

 public static void main (String args[]) {

 Object a = new HashSet(), b = new HashMap();

 Object c = new Hashtable();
```

```
System.out.print((a instanceof Collection)+",");
System.out.print((b instanceof Collection)+",");
System.out.print(c instanceof Collection);
}}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 13

- Elements are not key/value pairs.
- Contains no duplicate elements.
- The entries can be sorted using the Comparable interface.

Which of these classes provides the specified features?

- a. LinkedList
- b. TreeMap
- c. TreeSet
- d. HashMap
- e. HashSet
- f. Hashtable
- g. None of the above

#### Question 14

```
import java.util.*;

class GFC106 {

 public static void main (String args[]) {
 Object a = new HashSet(), b = new HashMap();
 Object c = new Hashtable();
 System.out.print((a instanceof Map)+",");
 System.out.print((b instanceof Map)+",");
 System.out.print(c instanceof Map);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false

- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

#### Question 15

- Stores key/value pairs.
- Allows null elements, keys, and values.
- Duplicate entries replace old entries.
- Entries are not sorted.

Which of these classes provides the specified features?

- a. LinkedList
- b. TreeMap
- c. TreeSet
- d. HashMap
- e. HashSet
- f. Hashtable

- g. None of the above

Question 16

```
import java.util.*;

class GFC107 {

 public static void main (String args[]) {

 Object a = new HashSet(), b = new HashMap();

 Object c = new Hashtable();

 System.out.print((a instanceof Cloneable)+",");

 System.out.print((b instanceof Cloneable)+",");

 System.out.print(c instanceof Cloneable);

 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false

h. Prints: true,true,true

i. None of the above

#### Question 17

- Entries are not organized as key/value pairs.
- Generally accepts duplicate elements.
- Entries may be accessed by means of an index.

Which interface of the java.util package offers the specified behavior?

- a. List
- b. Map
- c. Set
- d. None of the above

#### Question 18

```
import java.util.*;

import java.io.Serializable;

class GFC108 {

 public static void main (String args[]) {

 HashMap a = new HashMap();

 boolean b1, b2, b3;

 b1 = (a instanceof Cloneable) & (a instanceof Serializable);
```

```
b2 = a instanceof Map;
b3 = a instanceof Collection;
System.out.print(b1 + "," + b2 + "," + b3);
}
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 19

Which of the following classes allow unsynchronized read operations by multiple threads?

- a. Vector
- b. Hashtable
- c. TreeMap

- d. TreeSet
- e. HashMap
- f. HashSet

Question 20

- Entries are organized as key/value pairs.
- Duplicate entries replace old entries.
- Entries are sorted using a Comparator or the Comparable interface.

Which interface of the java.util package offers the specified behavior?

- a. List
- b. Map
- c. Set
- d. SortedSet
- e. SortedMap
- f. None of the above

Question 21

```
import java.util.*;

class GFC110 {

 public static void main (String[] args) {

 Object m = new LinkedHashMap();
```

```
System.out.print((m instanceof Collection)+",");

System.out.print((m instanceof Map)+",");

System.out.print(m instanceof List);

}}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

Question 22

```
import java.util.*;

class GFC109 {

 public static void main (String[] args) {

 Object v = new Vector();
```

```

System.out.print((v instanceof Collections)+",");
System.out.print((v instanceof Arrays)+",");
System.out.print(v instanceof List);
}

```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true
- c. Prints: false,true,false
- d. Prints: false,true,true
- e. Prints: true,false,false
- f. Prints: true,false,true
- g. Prints: true,true,false
- h. Prints: true,true,true
- i. None of the above

| No. | Answer                       |  | Remark                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----|------------------------------|--|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | c<br>LinkedList              |  | ArrayList and Vector both use an array to store the elements of the list; so access to any element using an index is very fast. A LinkedList is implemented using a doubly linked list; so access to an element requires the list to be traversed using the links.                                                                                                                                                             |
| 2   | g<br>Prints: true,true,false |  | The List and Set interfaces extend the Collection interface; so both List and Set could be cast to type Collection without generating a ClassCastException. Therefore, the first two of the three relational expressions return true. The Map interface does not extend Collection. The reference variable c1 refers to an instance of TreeMap; so the relational expression c1 instanceof Collection returns the value false. |
| 3   | e<br>HashSet                 |  | The elements of a Map are key/value pairs; so a Map is not a good choice. A List generally accepts duplicate                                                                                                                                                                                                                                                                                                                   |

| No. | Answer |                          | Remark                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----|--------|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     |        |                          | elements. A Set stores a collection of unique elements. Any attempt to store a duplicate element in a Set is rejected. Adding and removing an element in a TreeSet involves walking the tree to determine the location of the element. A HashSet stores the elements in a hashtable; so elements in a HashSet can be accessed almost as quickly as elements in an array as long as the hash function disperses the elements properly. Although the LinkedHashSet is not among the answer options it could arguably satisfy the requirements. However, the put and remove methods of the LinkedHashSet are a little slower than the same methods of the HashSet due to the need to maintain the linked list through the elements of the LinkedHashSet. |
| 4   | d      | Prints: false,true,true  | The Map interface does not extend Collection. The reference variable a1 refers to an instance of HashMap; so the relational expression a1 instanceof Collection returns the value false. The List and Set interfaces extend the Collection interface; so both List and Set could be cast to type Collection without generating a ClassCastException. Therefore, the second and third relational expressions return true.                                                                                                                                                                                                                                                                                                                              |
| 5   | b      | Map                      | The List and Set interfaces do not support key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 6   | c      | Prints: true,false       | HashSet implements the Set interface, but not the SortedSet interface.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 7   | c      | LinkedList               | ArrayList and Vector both use an array to store the elements of the list. When an element is inserted into the middle of the list the elements that follow the insertion point must be shifted to make room for the new element. The LinkedList is implemented using a doubly linked list; an insertion requires only the updating of the links at the point of insertion. Therefore, the LinkedList allows for fast insertions and deletions.                                                                                                                                                                                                                                                                                                        |
| 8   | d      | Prints: true,true        | TreeSet implements the Set interface and the SortedSet interface.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 9   | b      | TreeMap                  | The requirement to store key/value pairs is directly satisfied by a concrete implementation of the Map interface. The List and Set interfaces recognize objects, but do not recognize keys and values. TreeMap and TreeSet store elements in a sorted order based on the key, but the TreeSet does not support key/value pairs.                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 10  | h      | Prints: true,true,true   | The 1.2 version of Java introduced the updated Vector class that implements the List interface.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 11  | c      | Set                      | The Map interface organizes entries as key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 12  | e      | Prints: true,false,false | HashSet is a subclass of AbstractSet and AbstractCollection; therefore, it implements the Collection interface. HashMap and Hashtable do not implement the Collection interface. Instead, HashMap extends AbstractMap and implements the Map interface. Hashtable extends Dictionary and implements the Map interface.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 13  | c      | TreeSet                  | The elements are not key/value pairs; so a Map is not a good choice. A List generally accepts duplicate elements. A Set stores a collection of unique objects; so any attempt to store a duplicate object is rejected. TreeSet stores elements in an order that is determined either by a Comparator or by the Comparable interface.                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 14  | d      | Prints: false,true,true  | HashSet implements the Set interface, but not the Map interface. HashMap extends AbstractMap and implements the Map interface. Hashtable extends Dictionary and implements the Map interface.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

| No. | Answer        |                                          | Remark                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----|---------------|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15  | d             | HashMap                                  | The requirement to store key/value pairs is directly satisfied by a concrete implementation of the Map interface. The List and Set interfaces recognize objects, but do not recognize keys and values. The requirement to allow null elements is not satisfied by a Hashtable. TreeMap and TreeSet store elements in a sorted order based on the key.                                                                                                                                                                                                        |
| 16  | h             | Prints: true,true,true                   | All three implement Cloneable.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 17  | a             | List                                     | The Map interface organizes entries as key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries. A List allows entries to be accessed using an index.                                                                                                                                                                                                                                                                                                                                                                     |
| 18  | g             | Prints: true,true,false                  | HashMap does not implement the Collection interface.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 19  | c<br>d<br>e f | TreeMap<br>TreeSet<br>HashMap<br>HashSet | The Vector and Hashtable methods are synchronized and do not allow for simultaneous access by multiple threads. The concrete subclasses of the AbstractList, AbstractMap, and AbstractSet classes allow for unsynchronized read operations by multiple threads. Additionally, the synchronized wrapper methods of the Collections class allow for the instantiation of a Collection, List, Map, Set, SortedMap, or SortedSet with synchronized methods. If simultaneous read and write operations are necessary then a synchronized instance should be used. |
| 20  | e             | SortedMap                                | The List and Set interfaces do not support key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries. A Map organizes the entries as key/value pairs. The SortedMap is similar to a Map except that the ordering of the elements is determined by a Comparator or the Comparable interface.                                                                                                                                                                                                                                |
| 21  | c             | Prints: false,true,false                 | LinkedHashMap does not implement the Collection interface or the List interface.                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 22  | b             | Prints: false,false,true                 | The Collections class is not the same as the Collection interface. The Collections class contains a variety of methods used to work with collections. For example, Collections.shuffle is used to randomly shuffle the elements of a Collection. Similarly, the Arrays class provides utility methods for working with arrays.                                                                                                                                                                                                                               |

## Question 1

- Entries are not organized as key/value pairs.
- Duplicate entries are rejected.
- Entries are sorted using a Comparator or the Comparable interface.

Which interface of the java.util package offers the specified behavior?

- a. List
  - b. Map
  - c. Set
  - d. SortedSet
  - e. SortedMap
  - f. None of the above
- 

### Question 2

```
import java.util.*;
class GFC111 {
 public static void main (String[] args) {
 Object m = new LinkedHashMap();
 System.out.print((m instanceof Collection)+",");
 System.out.print((m instanceof Map)+",");
 System.out.print(m instanceof HashMap);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
  - b. Prints: false,false,true
  - c. Prints: false,true,false
  - d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. None of the above
- 

### Question 3

```
import java.util.*;
```

```
class GFC112 {
 public static void main (String[] args) {
 Object m = new LinkedHashSet();
 System.out.print((m instanceof Collection)+",");
 System.out.print((m instanceof Set)+",");
 System.out.print(m instanceof List);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
  - b. Prints: false,false,true
  - c. Prints: false,true,false
  - d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. None of the above
- 

Question 4

```
import java.util.*;
class GFC113 {
 public static void main (String[] args) {
 Object m = new LinkedHashSet();
 System.out.print((m instanceof Collection)+",");
 System.out.print((m instanceof Set)+",");
 System.out.print(m instanceof HashSet);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
- b. Prints: false,false,true

- c. Prints: false,true,false
  - d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. None of the above
- 

#### Question 5

```
import java.util.*;
class GFC116 {
 public static void main (String[] args) {
 Object x = new Vector().elements();
 System.out.print((x instanceof Enumeration)+",");
 System.out.print((x instanceof Iterator)+",");
 System.out.print(x instanceof ListIterator);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
  - b. Prints: false,false,true
  - c. Prints: false,true,false
  - d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. None of the above
- 

#### Question 6

```
import java.util.*;
class GFC117 {
 public static void main (String[] args) {
 Object i1 = new HashMap(), i2 = new TreeMap();
 System.out.print((i1 instanceof SortedMap)+",");
 System.out.print((i2 instanceof SortedMap)+",");
 System.out.print(i1 instanceof Collection);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
  - b. Prints: false,false,true
  - c. Prints: false,true,false
  - d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. None of the above
- 

#### Question 7

```
import java.util.*;
class GFC118 {
 public static void main (String[] args) {
 Object i = new ArrayList().listIterator();
 System.out.print((i instanceof List)+",");
 System.out.print((i instanceof Iterator)+",");
 System.out.print(i instanceof ListIterator);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false

- b. Prints: false,false,true
  - c. Prints: false,true,false
  - d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. None of the above
- 

#### Question 8

Which of the following classes allow elements to be accessed in the order that they were added?

- a. HashMap
  - b. HashSet
  - c. Hashtable
  - d. TreeMap
  - e. TreeSet
  - f. LinkedHashMap
  - g. LinkedHashSet
- 

#### Question 9

Which of the following are true statements?

- a. The Enumeration interface was introduced with the collections framework with Java 1.2.
  - b. The Enumeration interface declares only two methods: hasMoreElements and nextElement.
  - c. The Iterator interface extends the Enumeration interface.
  - d. The Iterator interface declares a total of three methods.
- 

#### Question 10

Which of the following is a true statement?

- a. The Iterator interface declares only two methods: hasMoreElements and nextElement.
  - b. The ListIterator interface extends both the List and Iterator interfaces.
  - c. The ListIterator interface was introduced with Java 1.2 to replace the older Iterator interface that was released with Java 1.0.
  - d. The ListIterator interface declares only three methods: hasNext, next and remove.
  - e. None of the above.
- 

Question 11

- Stores key/value pairs.
- Allows null elements, keys, and values.
- Duplicate entries replace old entries.
- Entries are not sorted using a Comparator or the Comparable interface.
- The iteration order is unspecified.

Which of these classes provides the specified features?

- a. LinkedList
  - b. LinkedHashMap
  - c. LinkedHashSet
  - d. TreeMap
  - e. TreeSet
  - f. HashMap
  - g. HashSet
  - h. Hashtable
  - i. None of the above
- 

Question 12

- Stores key/value pairs.

- Does not allow null elements, keys, and values.

Which of these classes provides the specified features?

- a. LinkedList
  - b. LinkedHashMap
  - c. LinkedHashSet
  - d. TreeMap
  - e. TreeSet
  - f. HashMap
  - g. HashSet
  - h. Hashtable
  - i. None of the above
- 

Question 13

- Stores key/value pairs.
- Duplicate entries replace old entries.
- Provides constant-time performance for the add, contains and remove operations.

Which of these classes provides the specified features?

- a. LinkedHashMap
  - b. LinkedHashSet
  - c. LinkedList
  - d. TreeMap
  - e. TreeSet
  - f. HashMap
  - g. HashSet
  - h. Hashtable
-

#### Question 14

```
import java.util.*;
class GFC114 {
 public static void main (String[] args) {
 Object a = new ArrayList();
 System.out.print((a instanceof Collections)+",");
 System.out.print((a instanceof Arrays)+",");
 System.out.print(a instanceof List);
 }
}
```

What is the result of attempting to compile and run the program?

- a. Prints: false,false,false
  - b. Prints: false,false,true
  - c. Prints: false,true,false
  - d. Prints: false,true,true
  - e. Prints: true,false,false
  - f. Prints: true,false,true
  - g. Prints: true,true,false
  - h. Prints: true,true,true
  - i. None of the above
- 

#### Question 15

- Each element must be unique.
- Contains no duplicate elements.
- Elements are not key/value pairs.
- Entries are not sorted using a Comparator or the Comparable interface.
- The iteration order is determined by the insertion order.

Which of these classes provides the specified features?

- a. HashMap
  - b. HashSet
  - c. Hashtable
  - d. LinkedHashMap
  - e. LinkedHashSet
  - f. LinkedList
  - g. TreeMap
  - h. TreeSet
  - i. None of the above
- 

#### Question 16

Suppose that you would like to create a new instance of a class that implements the Set interface, and you would like the new instance to be initialized with the elements of an existing Set. If you would like the iteration order of the new Set to be the same as that of the existing Set, then which concrete implementation of the Set interface should be used for the new instance?

- a. Hashtable
  - b. HashMap
  - c. HashSet
  - d. LinkedHashSet
  - e. TreeMap
  - f. TreeSet
  - g. None of the above.
- 

#### Question 17

Suppose that you would like to create a new instance of a class that implements the Map interface, and you would like the new instance to be initialized with the elements of an existing Map. If you would like the iteration order of the new Map to be the same as that of the existing Map, then which concrete implementation of the Map interface should be used for the new instance?

- a. Hashtable
- b. HashSet

- c. HashMap
  - d. TreeMap
  - e. TreeSet
  - f. LinkedHashMap
  - g. None of the above.
- 

#### Question 18

- Stores key/value pairs.
- Allows null elements, keys, and values.
- Duplicate entries replace old entries.
- The least recently used element can be removed automatically when a new element is added.

Which of these classes provides the specified features?

- a. LinkedHashMap
  - b. LinkedHashSet
  - c. LinkedList
  - d. TreeMap
  - e. TreeSet
  - f. HashMap
  - g. HashSet
  - h. Hashtable
  - i. None of the above
- 

#### Question 19

Which of the following classes would provide the most efficient implementation of a First In First Out queue?

- a. ArrayList
- b. LinkedHashMap

- c. LinkedHashSet
  - d. LinkedList
  - e. HashMap
  - f. HashSet
  - g. Hashtable
  - h. TreeMap
  - i. TreeSet
  - j. Vector
  - k. None of the above
- 

#### Question 20

In addition to implementing the List interface, which of the following also provides methods to get, add and remove elements from the head and tail of the list without specifying an index?

- a. Collection
  - b. ArrayList
  - c. LinkedList
  - d. List
  - e. Vector
  - f. None of the above
- 

#### Question 21

Which of the following is a true statement?

- a. All implementations of the List interface provide fast random access.
- b. A LinkedList provides faster random access than an ArrayList.
- c. The LinkedList implements the RandomAccess interface.
- d. Each collection that implements the List interface must also implement the RandomAccess interface.
- e. The RandomAccess interface declares methods that use of an index argument to access elements of the List.
- f. The RandomAccess interface declares the next and hasNext methods.

g. None of the above.

---

## Question 22

Which of the following is not a true statement?

- a. The Iterator interface declares only three methods: hasNext, next and remove.
- b. The ListIterator interface extends both the List and Iterator interfaces.
- c. The ListIterator interface provides forward and backward iteration capabilities.
- d. The ListIterator interface provides the ability to modify the List during iteration.
- e. The ListIterator interface provides the ability to determine its position in the List.
- f. None of the above.

| No. | Answer | Remark                                                                                                                                                                                                                                                                                                |
|-----|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | d      | SortedSet      The Map interface organizes entries as key/value pairs. A list generally allows duplicate entries. A Set rejects duplicate entries. The SortedSet is similar to a Set except that the ordering of the elements is determined by a Comparator or the Comparable interface.              |
| 2   | d      | Prints: false,true,true    LinkedHashMap does not implement the Collection interface. LinkedHashMap extends HashMap and implements Map, Cloneable and Serializable.                                                                                                                                   |
| 3   | g      | Prints: true,true,false    LinkedHashSet does not implement the List interface. LinkedHashSet extends HashSet and implements Collection, Set, Cloneable and Serializable.                                                                                                                             |
| 4   | h      | Prints: true,true,true    LinkedHashMap is a subclass of HashSet; therefore, it is also a subclass of AbstractSet, and it implements the Collection interface.                                                                                                                                        |
| 5   | e      | Prints: true,false,false      The Vector.elements method returns an Enumeration over the elements of the Vector. The Vector class implements the List interface and extends AbstractList; so it is also possible to obtain an Iterator over a Vector by invoking the iterator or listIterator method. |
| 6   | c      | Prints: false,true,false      Both HashMap and TreeMap are subclasses of type AbstractMap; so both implement the Map interface. Neither implements the Collection interface. The TreeMap implements the SortedMap interface, but HashMap does not.                                                    |
| 7   | d      | Prints: false,true,true    ListIterator extends Iterator.                                                                                                                                                                                                                                             |

8 f g LinkedHashMap HashSet     The HashMap, HashSet and Hashtable classes are all implemented with an internal hashtable that organizes the elements in buckets according to the hashCode and not according to the order of insertion. The LinkedHashMap and HashSet classes also use an internal hashtable, and both also maintain a linked list through all of the elements. The order of the list is determined by the order of insertion. Optionally, the LinkedHashMap can maintain the order of the list based on the time of the most recent access of each element. The TreeMap and TreeSet classes are both implemented using a tree structure that is ordered based on a Comparator or the Comparable interface.

9 b d     The Enumeration interface declares only two methods: hasMoreElements and nextElement. The Iterator interface declares a total of three methods.     The Enumeration interface was introduced with Java 1.0 to provide an easy means of moving through the elements of a Vector or the keys or values of a Hashtable. The Iterator interface was introduced with the collections framework with Java 1.2. The Iterator interface declares three methods: hasNext, next and remove. The first two methods, hasNext and next, are similar to the two methods declared in the Enumeration interface, hasMoreElements and nextElement. The third method of the Iterator interface, remove, provides new functionality relative to the Enumeration interface.

10 e     None of the above.     The Iterator interface declares three methods: hasNext, next and remove. The ListIterator interface was introduced in Java 1.2 along with the Iterator interface. The ListIterator interface extends the Iterator interface and declares additional methods to provide forward and backward iteration capabilities, List modification capabilities and the ability to determine the position of the iterator in the List. The ListIterator interface does not extend the List interface.

11 f     HashMap     The requirement to store key/value pairs is directly satisfied by a concrete implementation of the Map interface. The List and Set interfaces recognize objects, but do not recognize keys and values. The requirement to allow null elements is not satisfied by a Hashtable. TreeMap and TreeSet store elements in a sorted order based on the key. The iteration order of LinkedHashMap and HashSet is not unspecified. By default, the iteration order of LinkedHashMap and HashSet is based on the order in which elements were inserted. Optionally, the iteration order of the LinkedHashMap can be set to the order in which the elements were last accessed.

12 h     Hashtable     The requirement to store key/value pairs is directly satisfied by a Hashtable or any concrete implementation of the Map interface. The List and Set interfaces recognize objects, but do not recognize keys and values. The requirement to NOT allow null elements is satisfied by Hashtable, but not by HashMap or any of the other Collection implementations that were introduced with Java 1.2 and later.

13 a f h     LinkedHashMap HashMap Hashtable     The requirement to store key/value pairs is directly satisfied by a concrete implementation of the Map interface. The List and Set interfaces recognize objects, but do not recognize keys and values. The Hashtable, HashMap and LinkedHashMap classes store elements in a hashtable. Elements are accessed using a hashCode that identifies the bucket that contains the element. Access time is therefore not dependent on the number of buckets. As long as the hashCode methods of the elements are properly implemented, the time required to access an element in a hashtable remains constant as the number of buckets in the hashtable grows. In contrast, the TreeMap and TreeSet classes store elements in a sorted order in a tree structure. Access to any element requires walking the tree; so access time depends on the size of the tree.

14 b Prints: false,false,true The Collections class is not the same as the Collection interface. The Collections class contains a variety of methods used to work with collections. For example, Collections.shuffle is used to randomly shuffle the elements of a Collection. Similarly, the Arrays class provides utility methods for working with arrays.

15 e LinkedHashSet The elements of a Map are key/value pairs; so a Map is not a good choice. A List generally accepts duplicate elements. A Set stores a collection of unique elements. Any attempt to store a duplicate element in a Set is rejected. TreeSet stores elements in a sorted order based on the key. HashSet does not sort the elements based on the key. The iteration order of LinkedHashMap and LinkedHashSet is clearly defined. By default, the iteration order of LinkedHashMap and LinkedHashSet is based on the order in which elements were inserted. While a LinkedHashSet rejects duplicate entries, the LinkedHashMap allows duplicate entries to replace old entries.

16 d LinkedHashSet The iteration order of a Set is the order in which an iterator moves through the elements of the Set. The iteration order of a LinkedHashSet is determined by the order in which elements are inserted. When a reference to an existing Set is passed as an argument to the constructor of LinkedHashSet, the Collection.addAll method will add the elements of the existing Set to the new instance. Since the iteration order of the LinkedHashSet is determined by the order of insertion, the iteration order of the new LinkedHashSet must be the same as the iteration order of the old Set.

17 f LinkedHashMap The iteration order of a Map is the order in which an iterator moves through the elements of the Map. The iteration order of a LinkedHashMap is determined by the order in which elements are inserted. When a reference to an existing Map is passed as an argument to the constructor of LinkedHashMap, the Collection.addAll method will add the elements of the existing Map to the new instance. Since the iteration order of the LinkedHashMap is determined by the order of insertion, the iteration order of the new LinkedHashMap must be the same as the iteration order of the old Map.

18 a LinkedHashMap The requirement to store key/value pairs is directly satisfied by a concrete implementation of the Map interface. The List and Set interfaces recognize objects, but do not recognize keys and values. The requirement to allow null elements is not satisfied by a Hashtable. The LinkedHashMap offers the option to remove the least recently used (LRU) element when a new element is added. The LinkedHashSet does not offer the LRU option.

19 d LinkedList A stack or queue must be implemented using a data structure that stores the elements based on the order of insertion. Any data structure that is implemented using a hashtable is not a good choice. The ArrayList and Vector are both implemented using an internal array. Although an array allows elements to be easily organized based on the order of insertion, an array does not allow the list of elements to be easily shifted in memory as elements are appended to the tail of the list and removed from the head of the list. The LinkedList is implemented using a doubly linked list that allows elements to be easily appended to the tail of the list, and removed from the head of the list.

20 c LinkedList The LinkedList class provides methods such as addFirst, addLast, getFirst, getLast, removeFirst and removeLast that facilitate the implementation of stacks and queues.

21 g None of the above. The RandomAccess interface is a marker interface; so it does not declare any methods. Its purpose is to provide information about the RandomAccess capabilities of a List implementation. Generic list algorithms can check to see if an instance of a List implements the RandomAccess marker interface. If not, then the algorithm can avoid operations that require fast random access. Both Vector and ArrayList implement the RandomAccess interface. LinkedList does not implement RandomAccess.

22 b The ListIterator interface extends both the List and Iterator interfaces. The ListIterator interface extends the Iterator interface and declares additional methods to provide forward and backward iteration capabilities, List modification capabilities and the ability to determine the position of the iterator in the List.

Hashcodes

Question 1

Suppose that an instance of class C has legal implementations of the hashCode and equals methods. Within any one execution of the Java application, the hash code contract requires that each invocation of the hashCode method on the same instance of class C must consistently return the same result as long as the fields used for the equals comparison remain unchanged.

- a. false
- b. true

Question 2

If two instances of a class type are equal according to the equals method, then the same integer value must be returned by the hashCode method of the two objects.

- a. false
- b. true

Question 3

If two instances of a class type are not equal according to the equals method, then the same integer value must not be returned by the hashCode method of the two objects.

- a. false
- b. true

#### Question 4

```
class A {

 static void m1 (B a, B b, B c, B d, B e, B f, B g, B h) {

 if (a.equals(b)) {System.out.print("A");}

 if (!c.equals(d)) {System.out.print("B");}

 if (e.hashCode() == f.hashCode()) {System.out.print("C");}

 if (g.hashCode() != h.hashCode()) {System.out.print("D");}

 }{}
```

Suppose that method m1 is invoked with eight instances of the same class and the output is ABCD. If the B.equals and B.hashCode methods are implemented according to the hash code contract, then which of the following statements must always be true?

- a. (a.hashCode() == b.hashCode())
- b. (c.hashCode() != d.hashCode())
- c. (e.equals(f))
- d. (!g.equals(h))

#### Question 5

```
class B {
 private int i1;

 public int hashCode() {return 1;}
}

class C {
 private int i1;

 public int hashCode() {return -1;}
}

class D {
 private int i1;

 public int hashCode() {return i1;}
}
```

Suppose that the equals method of classes B, C and D all make use of the value of the int variable, i1. Which class has a hashCode method that is not consistent with the hash code contract?

- a. B
- b. C
- c. D
- d. None of the above

Question 6

Which of the following classes override both the equals and hashCode methods?

- a. java.lang.Byte
- b. java.lang.Integer
- c. java.util.Vector
- d. java.lang.String
- e. java.lang.StringBuffer

Question 7

```
class A {
 int i1, i2;

 public void setI1(int i) {i1 = i;}

 public int getI1() {return i1;}

 public void setI2(int i) {i2 = i;}

 public int getI2() {return i2;}

 public A(int ii1, int ii2) {i1 = ii1; i2 = ii2;}

 public boolean equals(Object obj) {
 if (obj instanceof A) {
 return (i1 == ((A)obj).getI1());
 }
 return false;
 }
}
```

```
public int hashCode() {
 // Insert statement here.
}
```

Which of the following statements could be inserted at the specified location without violating the hash code contract?

- a. return 31;
- b. return getI1();
- c. return getI2();
- d. return 31 \* getI1() + getI2();

#### Question 8

```
class A {

 int i1, i2;

 public void setI1(int i) {i1 = i;}

 public int getI1() {return i1;}

 public void setI2(int i) {i2 = i;}

 public int getI2() {return i2;}

 public A(int ii1, int ii2) {i1 = ii1; i2 = ii2;}

 public boolean equals(Object obj) {

 if (obj instanceof A) {

 return (i1 == ((A)obj).getI1()) & (i2 == ((A)obj).getI2());
 }
 }
}
```

```
}

return false;

}

public int hashCode() {

 // Insert statement here.

}}
```

If inserted at the specified location, which of the following statements would produce the most efficient hashCode method?

- a. return 31;
- b. return getI1();
- c. return getI2();
- d. return getI1() + getI2();
- e. return 31 \* getI1() + getI2();
- f. None of the above

1 b true

2 b true If two objects are equal according to the equals method, then the hashcodes produced by the hashCode method must also be equal. If two objects are not equal according to the equals method, then the hashcodes may or may not be equal. It is preferable that unequal objects have different hashcodes, but that is not always possible. Since the hash code value is a 32 bit primitive int, it is not possible to produce a unique hash code for each value of a primitive long.

3 a false If two objects are equal according to the equals method, then the hashcodes must also be equal. If two objects are not equal according to the equals method, then the hashcodes may or may not be equal. It is preferable that unequal objects have different hashcodes, but that is

not always possible. Since the hash code value is a 32 bit primitive int, it is not possible to produce a unique hash code for each value of a primitive long.

4 a d (a.hashCode() == b.hashCode()) (!g.equals(h)) If two objects are equal according to the equals method, then the hashcodes must also be equal. If two objects are not equal according to the equals method, then the hashcodes may or may not be equal. If two objects have the same hash code, then the objects may or may not be equal. If two objects have different hashcodes, then the objects must not be equal.

5 d None of the above Suppose that the hashCode method is invoked on the same object more than once during the same execution of a Java application. If no information used in the equals comparison is modified, then each invocation of the hashCode method must produce the same hash code value. The hashCode methods of classes B and C will always return the same value during an execution of the Java application and are therefore consistent with the hash code contract. Even so, the hashCode methods of classes B and C are not efficient, because they will cause hashtables to place every instance of classes B and C in the same bucket. The hashCode method of class D is appropriate and will allow a hash table to operate efficiently.

6 a b c d java.lang.Byte java.lang.Integer java.util.Vector java.lang.String The wrapper classes and the collection classes override the equals and hashCode methods. The String class overrides the equals and hashCode methods, but StringBuffer does not.

7 a b return 31; return getI1(); A hashCode method that returns the constant value 31 is consistent with the hash code contract. Even so, a hashCode method that returns the same value regardless of the internal state of the object is not very good, because it will cause hashtables to place every instance of the class in the same bucket. If the equals method determines that two instances are equal, then the two instances must produce the same hash code. For that reason, the hash code must not be calculated using fields that are not used to determine equality. In this case, the equals method determines equality based on the value of i1. The value of i2 is not used to determine equality; therefore, i2 can not be used to calculate the hash code.

8 e return 31 \* getI1() + getI2(); All of the statements would produce a hashCode method that is consistent with the hash code contract. The expression 31 \* getI1() + getI2() produces the most efficient hashCode method, because it is most likely to produce unique hashcodes for various combinations of i1 and i2. The expression getI1() + getI2() is less efficient, because it produces the same hash code when the values of i1 and i2 are swapped.