# Java SE 8 Programmer I

**Java Basics**
- Define the scope of variables
- Define the structure of a Java class
- Create executable Java applications with a main method; run a Java program from the command line; including console output.
- Import other Java packages to make them accessible in your code
- Compare and contrast the features and components of Java such as: platform independence, object orientation, encapsulation, etc.

**Working With Java Data Types**
- Declare and initialize variables (including casting of primitive data types)
- Differentiate between object reference variables and primitive variables
- Know how to read or write to object fields
- Explain an Object's Lifecycle (creation, "dereference by reassignment" and garbage collection)
- Develop code that uses wrapper classes such as Boolean, Double, and Integer.

**Using Operators and Decision Constructs**
- Use Java operators; including parentheses to override operator precedence
- Test equality between Strings and other objects using == and equals ()
- Create if and if/else and ternary constructs
- Use a switch statement

**Creating and Using Arrays**
- Declare, instantiate, initialize and use a one-dimensional array
- Declare, instantiate, initialize and use multi-dimensional array

**Using Loop Constructs**
- Create and use while loops
- Create and use for loops including the enhanced for loop
- Create and use do/while loops
- Compare loop constructs
- Use break and continue

**Working with Methods and Encapsulation**
- Create methods with arguments and return values; including overloaded methods
- Apply the static keyword to methods and fields
- Create and overload constructors; including impact on default constructors
- Apply access modifiers
- Apply encapsulation principles to a class
- Determine the effect upon object references and primitive values when they are passed into methods that change the values

**Working with Inheritance**
- Describe inheritance and its benefits
- Develop code that demonstrates the use of polymorphism; including overriding and object type versus reference type
- Determine when casting is necessary
- Use super and this to access objects and constructors
- Use abstract classes and interfaces

**Handling Exceptions**
- Differentiate among checked exceptions, unchecked exceptions, and Errors
- Create a try-catch block and determine how exceptions alter normal program flow
- Describe the advantages of Exception handling
- Create and invoke a method that throws an exception
- Recognize common exception classes (such as NullPointerException, ArithmeticExcpetion, ArrayIndexOutOfBoundsException, ClassCastException)

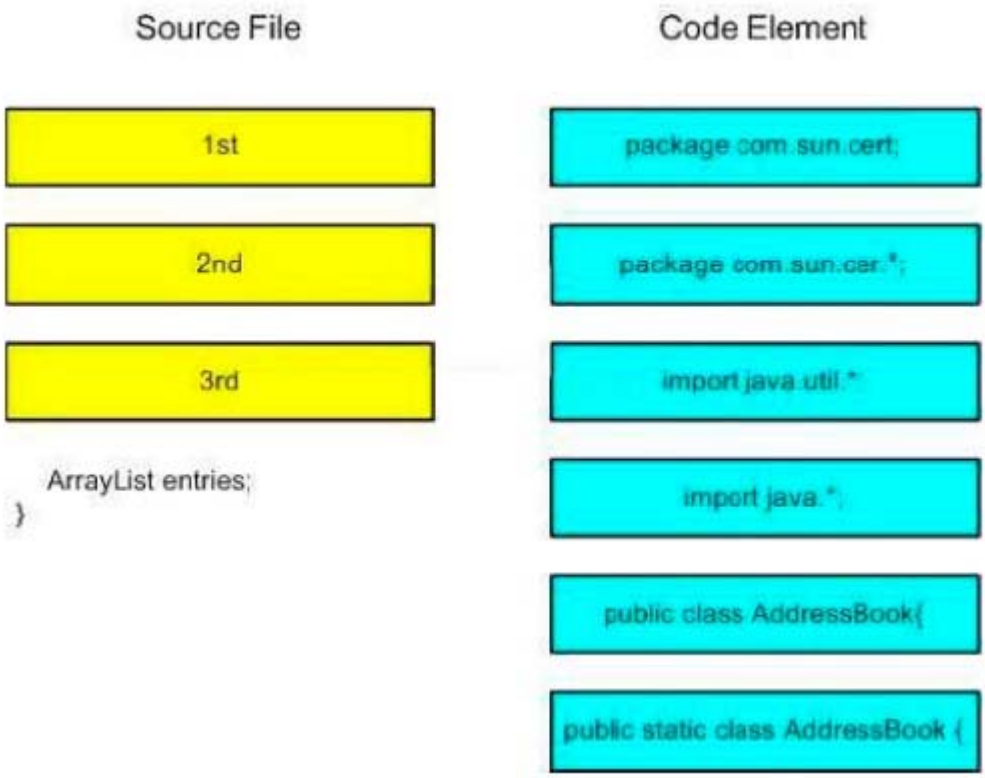
**Working with Selected classes from the Java API**
- Manipulate data using the StringBuilder class and its methods

- Creating and manipulating Strings
- Create and manipulate calendar data using classes from java.time.LocalDateTime, java.time.LocalDate, java.time.LocalTime, java.time.format.DateTimeFormatter, java.time.Period
- Declare and use an ArrayList of a given type
- Write a simple Lambda expression that consumes a Lambda Predicate expression

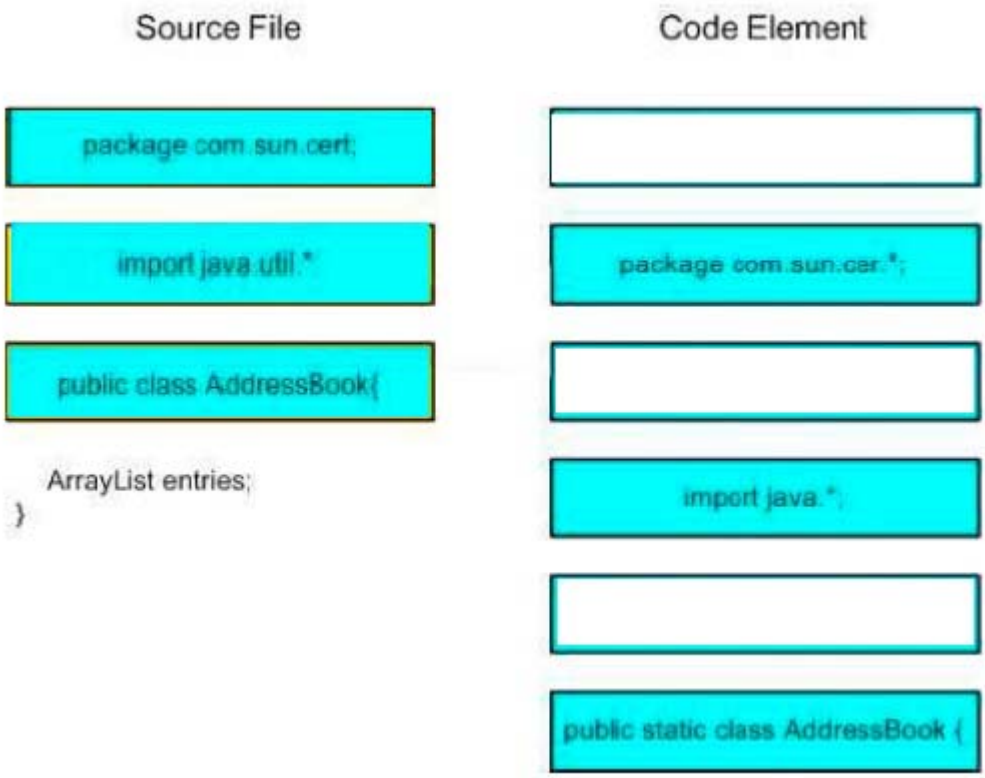
**Exam**

**QUESTION 1**
**Place the code elements in order so that the resulting Java source file will compile correctly, resulting in a class called com.sun.cert.AddressBook.**

**Select and Place:**




**Correct Answer:**




**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 2**
**Which of the following keywords may occur multiple times in a Java source file?**

A. `import`
B. `class`
C. `private`
D. `package`
E. `public`

**Correct Answer:** ABCE
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
There can be at most one package statement in a Java source file and it must be the first statement in the file.

**QUESTION 3**
**Given:**

```
public class Test {
}
```

**Which packages are automatically imported into the java source file by the java compiler?**

A. java.lang
B. java.awt

C. javax.net
D. java.*
E. The package with no name
F. java.util
G. System

**Correct Answer:** AE
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
If there is no package statement in the source file, the class is assumed to be created in a default package that has no name. In this case, all the types created in this default package will be available to this class without any import statement.

However, note that this default package cannot be imported in classes that belong to any other package at all, not even with any sort of import statement. So for example, if you have a class named SomeClass in package test, you cannot access TestClass defined in the problem statement (as it is defined in the default package) at all because there is no way to import it.

As per JLS Section 7.5: A type in an unnamed package has no canonical name, so the requirement for a canonical name in every kind of import declaration implies that (a) types in an unnamed package cannot be imported, and (b) static members of types in an unnamed package cannot be imported.

**QUESTION 4**
Which package would you import to use the Random class?

A. `java.util`
B. `java.lang`
C. `java.io`
D. `java.math`

**Correct Answer:** A
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 5**
**Which of the given options can be successfully inserted at line 1....**

```
//line 1
public class A {

}
```

A. `import java.lang.*;`
B. `package p.util;`
C. `public class MyClassA { }`
D. `class MyClassB { }`
E. `private class MyClassC { }`
F. `static class MyClassD { }`

**Correct Answer:** ABD
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
import java.lang.*; ---> Although this package is automatically imported, it is not an error to import it

explicitly.
package p.util; ---> It is a perfectly valid package statement.

You can have more than one classes in a file but at most one of them can be public.
There can be only 1 "public" class within package scope in a file. You can have additional inner classes that are public though.

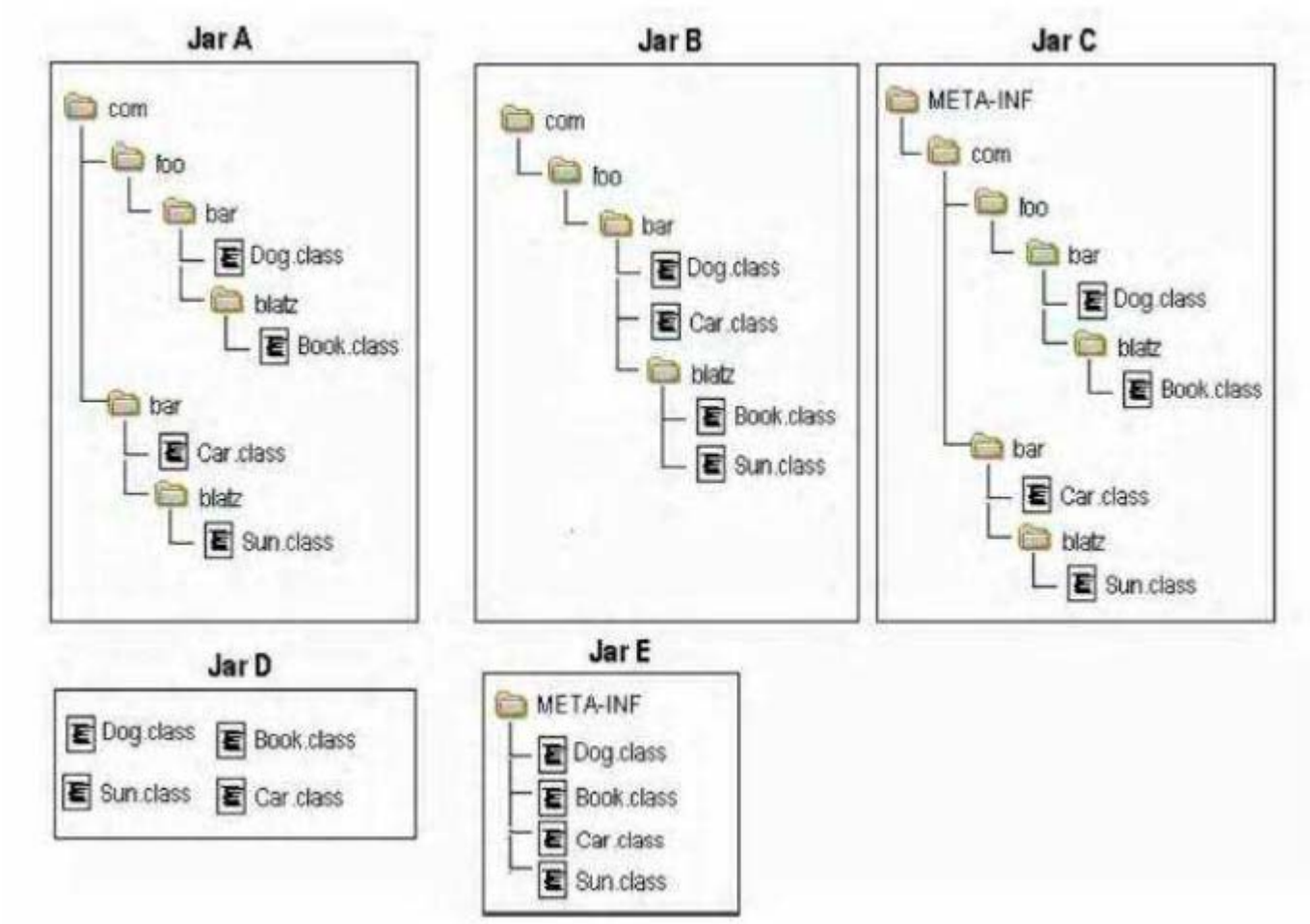
**QUESTION 6**
**Which method identifier is correct according to Java naming conventions?**

A. `Calculator`
B. `calculateBill`
C. `calculatebill`
D. `BillCalculator`

**Correct Answer:** B
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 7**
**Given the following JARs:**




**and given the fully-qualified class names:**
`com.foo.bar.Dog`
`com.foo.bar.blatz.Book`
`com.bar.Car`

```
com.bar.blatz.Sun
```

**Which graph represents the correct directory structure for a JAR file from which those classes can be used by the compiler and JVM?**

A. Jar A
B. Jar B
C. Jar C
D. Jar D
E. Jar E

**Correct Answer:** A
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 8**
**Given the following directory structure:**

```
org
   | -- Robot.class
   |
   | -- ex
        |-- Pet.class
        |
        |-- why
            |-- Dog.class
```

**And the following source file:**

```
class MyClass {
  Robot r;
  Pet p;
  Dog d;
}
```

**Which statement(s) must be added for the source file to compile? (Choose all that apply.)**

A. package org;
B. import org.*;
C. package org.*;
D. package org.ex;
E. import org.ex.*;
F. package org.ex.why;
G. package org.ex.why.Dog;

**Correct Answer:** BEF
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 9**
**Given the classes:**

Apple.java

```
package fruits;
```

```
class Apple {
   public void getApple() { }
}
```

Salad.java

```
package food;

//line n1

class Salad {
   Apple apple = new Apple(); // line n2
   public void prepareSalad() {
      apple.getApple();
   }
}
```

**Which modifications, independently, enable the Salad.java file to compile?**

A. `Replace line n1 with import fruits.Apple.getApple();`
B. `Replace line n1 with import fruits.Apple;`
C. `Replace line n1 with import fruits;`
D. `Replace line n2 with fruits.Apple apple = new Apple();`
E. `Replace line n2 with fruits.Apple apple = new fruits.Apple();`
F. `The code will not compile`

**Correct Answer:** F
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 10**
**Given the code:**

```
package handy.dandy;

public class Keystroke {
   public void typeExclamation() {
      System.out.println("!");
   }
}
```

**And**

```
1. package handy;
2. public class Greet {
3.    public static void main(String[] args) {
4.       String greeting = "Hello";
5.       System.out.print(greeting);
6.       Keystroke stroke = new Keystroke();
7.       stroke.typeExclamation();
8.    }
9. }
```

**What three modifications, made independently, enable the code to compile and run?**

A. `line 6 replaced with handy.dandy.Keystroke stroke = new Keystroke();`
B. `line 6 replaced with handy.*.Keystroke stroke = new Keystroke();`
C. `line 6 replaced with handy.dandy.Keystroke stroke = new`

```
handy.dandy.Keystroke();
```

D. `import handy.*; added before line 1`
E. `import handy.dandy.*; added after line 1`
F. `import handy.dandy.Keystroke; added after line 1`
G. `import handy.dandy.Keystroke.typeExclamation(); added before line 1`

**Correct Answer:** CEF
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 11**
**Given:**

```
1.  class Test {
2.     /* comment text 1 */
3.     // comment text 2 //
3.     // comment text 3
5.     and comment text 4 //
6.     /* comment text 5
7.     and comment text 6 */
8   }
```

**At which line does a compilation error occur?**

A. `line 5`
B. `line 2`
C. `line 3`
D. `line 7`

**Correct Answer:** A
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 12**
**What is the correct parameter specification for the standard main method?**

A. `void`
B. `String[ ] args`
C. `Strings args[ ]`
D. `String args`
E. `String args[ ]`
F. `String... args`
G. `String.. args`

**Correct Answer:** BEF
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
There is a no difference for args whether it is defined as String[] args or String args[].

However, there is an important difference in the way it is defined as illustrated by the following:

1. String[] sa1, sa2;
Here, both - sa1 and sa2 are String arrays.

2. String sa1[], sa2;
Here, only sa1 is a String array. sa2 is just a String.

**QUESTION 13**
**Given:**

```
public class App {
   public static void main(String[] args) {
       System.out.println("Hello Java!");
   }
}
```

**Which statement is true about the `main` method of a Java application?**

A. Its parameter can be of type Integer[].
B. It can be a non-static method.
C. It can be a non-public method.
D. It can be defined in a non-public class.

**Correct Answer:** D
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 14**
**Which method declarations will enable a class to be run as a standalone program?**

A. static void main(String args[ ])
B. public void static main(String args[ ])
C. public static main(String[ ] argv)
D. final public static void main(String [ ] array)
E. static public void main(String args[ ])
F. public static void main(String [ ] array)

**Correct Answer:** DEF
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
A valid declaration of "the" main() method must be public and static, should return void, and should take a single array of Strings as a parameter.
The order of the static and public keywords is irrelevant. But the return type should always come just before the method name.
Applying final to the method does not change the method signature.

final only means that subclasses cannot shadow (in case of static methods) or override (in case of instance methods) it.

However, for the purpose of Java Certification exam, it should be assumed that for the JVM to execute a class using the standard main method, the accessibility of the main method must be public.

```
package test;

public class TestClass{
        private static void main(String args[]) {
               System.out.println("hello");
```

```
        }
}
```

In some versions of JDK, even a private or protected main() method works from command line. However, for the purpose of Java Certification exam, it should be assumed that for the JVM to execute a class using the standard main method, the accessibility of the main method must be public.

**QUESTION 15**
**What will the following code print when run?**

```
public class Pass2 {
   public static void main(String [] args) {
     int x;
     int y;
     System.out.print(" main x = " + x++);
     System.out.print(" main y = " + ++y);
   }
}
```

A. Compilation fails.
B. An exception is thrown at runtime.
C. main x = 0 main y = 1
D. main x = 1 main y = 0
E. main x = 0 main y = 0
F. main x = 1 main y = 1

**Correct Answer:** A
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 16**
**What will the following code print when run?**

```
public class Pass2 {
   public void main(String [] args) {
     int x = 0;
     int y = 0;
     System.out.print(" main x = " + x++);
     System.out.print(" main y = " + ++y);
   }
}
```

**What is the result?**

A. Compilation fails.
B. An error is thrown at runtime.
C. main x = 0 main y = 1
D. main x = 1 main y = 0
E. main x = 0 main y = 0
F. main x = 1 main y = 1

**Correct Answer:** B
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 17**
**What will the following code print when run?**

```
public class TestClass {
    public static long main(String[] args){
         System.out.println("Hello");
         return 10L;
    }
}
```

A. Hello
B. It will not print anything.
C. It will not compile
D. It will throw an Error at runtime.
E. None of the above.

**Correct Answer:** D
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
When the program is run, the JVM looks for a method named main() which takes an array of Strings as input and returns nothing (i.e. the return type is void).
But in this case, it doesn't find such a method (the given main() method is returning long!) so it throws a **java.lang.NoSuchMethodError**.

Note that java.lang.Error does not extend Exception class. It extends java.lang.Throwable and so it can be "thrown".

**QUESTION 18**
**What will the following code print when run?**

```
public class Pass2 {
   public static void main(String [] args) {
     int x = 6;
     doStuff(x);
     System.out.print(" main x = " + x);
   }
   void doStuff(int x) {
     System.out.print(" doStuff x = " + ++x);
   }
}
```

A. Compilation fails.
B. An exception is thrown at runtime.
C. doStuff x = 6 main x = 6
D. doStuff x = 6 main x = 7
E. doStuff x = 7 main x = 6
F. doStuff x = 7 main x = 7

**Correct Answer:** A
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 19**
**Which statement is valid?**

A. `int total score = 0;`
B. `int totalScore2 = 0;`
C. `int 2totalScore = 0;`
D. `int total-score = 0;`

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 20**
**Which are valid declarations? (Choose all that apply)**

A. int $x;
B. int 123;
C. int _123;
D. int #dim;
E. int %percent;
F. int *divide;
G. int central_sales_region_Summer_2005_gross_sales;

**Correct Answer:** ACG
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 21**
**Given:**

```
public static void main (String [] args) {
   int a, b, c = 0;
   int a, b, c;
   int g, int h, int i = 0;
   int d, e, F;
   int k, l, m; = 0;
}
```

**Which two declarations will compile?**

A. `int a, b, c = 0;`
B. `int a, b, c;`
C. `int g, int h, int i = 0;`
D. `int d, e, F;`
E. `int k, l, m; = 0;`

**Correct Answer:** AD
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 22**
**Which of the given code fragments will not compile? (Select all that apply)**

A.
```
byte a = 128;
```
B.
```
byte b = 100;
b *= 1;
```
C.
```
byte b = 1;
b = b+1;
```
D.
```
int i= 128;
```

**Correct Answer:** AC
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 23**
**Which of the given code snippets will compile? (Select all that apply)**

A.
```
long id = 6;
int i = id;
```
B.
```
float f = 6.32f;
int i = f;
```
C.
```
char c = 'A';
int i = c;
```
D.
```
int i = 0;
byte b = (byte)i;
```

**Correct Answer:** CD
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 24**
**Given the code fragment:**

```
5. float fValue = 120;
6. int iValue = fValue;
7. double dValue = fValue;
8. long lValue = fValue;
```

**At which line does a compilation error occur?**

A. lines 6 and 8
B. line 5
C. line 7
D. lines 5 and 7

**Correct Answer:** A
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 25**
**Given:**

```
public class Course {
   public static void main(String[] args) {
       double courseFee = 1000.0;
```

```
        float percentage = 5.0f;
        // line n1
        newFee = courseFee * percentage;
        System.out.println(newFee);
    }
}
```

**Which statement, when inserted at line n1, enables the `Course` class to compile?**

A. `int newFee;`
B. `long newFee;`
C. `float newFee;`
D. `double newFee;`

**Correct Answer:** D
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 26**
**Which code fragment causes a compilation error?**

A. `float flt = 100F;`
B. `float flt = (float) 1_11.00;`
C. `float flt = 100;`
D.
```
double y1 = 203.22;
float flt = y1;
```
E.
```
int y2 = 100;
float flt = (float)y2;
```

**Correct Answer:** D
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 27**
**Which of the following declarations are valid?**

A. `float f1 = 1.0;`
B. `float f = 43e1;`
C. `float f = -1;`
D. `float f = 0x0123;`
E. `float f = 4;`

**Correct Answer:** CDE
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
Although the values in the option 1 and 2 are compile time constants and the values i.e. 1.0 and 43e1 can fit into a float, implicit narrowing does not happen because implicit narrowing is permitted only among byte, char, short, and int.

**QUESTION 28**
**Which of the following is illegal ?**

A. `char c = 320;`
B. `float f = 320;`
C. `double d = 320;`
D. `byte b = 320;`
E. `float f = 22.0f/7.0f;`
F. None of the above is illegal.

**Correct Answer:** D
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
`char c = 320;`
This is valid because 320 is below the maximum value that a char can take, which is 2^16 -1. Remember that char can take only positive values.

`float f = 320;`
320 is an int and any valid int can be assigned to a float or a double variable without a cast. Note that f = 320.0 is not valid as 320.0 would be a double and a double can only be assigned to a double without a cast.

`double d = 320;`
This is valid because any valid int can be assigned to a double or even a float without any cast.

`byte b = 320;`
320 cannot fit into a byte so you must cast it.: byte b = (byte) 320;

`float f = 22.0f/7.0f;`
Since both the operands of / are floats, it will result in a float, which can be assigned to f. If you have, 22.0f/7.0, then it would not compile because 7.0 is a double and so 22.0f/7.0 will return a double, which cannot be assigned to a float.

**QUESTION 29**
**What will be the result of attempting to compile and run the following class?**

```
public class TestClass {
    public static void main(String args[ ] ){
        int i, j, k;
        i = j = k = 9;
        System.out.println(i);
    }
}
```

**Please select 2 options**

A. The code will not compile because unlike in c++, operator '=' cannot be chained i.e. a = b = c = d is invalid.
B. The code will not compile as 'j' is being used before getting initialized.
C. The code will compile correctly and will display '9' when run.
D. The code will not compile as 'j' and 'i' are being used before getting initialized.
E. All the variables will get a value of 9.

**Correct Answer:** CE
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
= can be chained. For example, assuming all the variables are declared appropriately before hand, a = b = c = d; is valid

However, chaining to use a value of a variable at the time of declaration is not allowed. For example, int a

= b = c = 100; is invalid if b and c are not already declared. Had b and c been already declared, int a = b = c = 100; would have been valid.

Every expression has a value, in this case the value of the expression is the value that is assigned to the right hand side of the equation.
k has a value of 9 which is assigned to j and then to i.

Another implication of this is :
```
boolean b = false;
if( b = true) { System.out.println("TRUE");}
```

The above code is valid and will print TRUE. Because b = true has a boolean value, which is what an if statement expects.

Note that if( i = 5) { ... } is not valid because the value of the expression i = 5 is an int (5) and not a boolean.

**QUESTION 30**
**Which of these assignments are valid?**

A. `short s = 12 ;`
B. `long g = 012 ;`
C. `int i = (int) false;`
D. `float f = -123;`
E. `float d = 0 * 1.5;`

**Correct Answer:** ABD
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
Note that
`float d = 0 * 1.5f;` and `float d = 0 * (float)1.5 ;` are OK

`short s = 12 ;`
This is valid since 12 can fit into a short and an implicit narrowing conversion can occur.

`long g = 012 ;`
012 is a valid octal number.

`int i = (int) false;`
Values of type boolean cannot be converted to any other types.

`float f = -123;`
Implicit widening conversion will occur in this case.

A narrowing primitive conversion may be used if all of the following conditions are satisfied:
1. The expression is a constant expression of type int.
2. The type of the variable is byte, short, or char.
3. The value of the expression (which is known at compile time, because it is a constant expression) is representable in the type of the variable.

Note that narrowing conversion does not apply to long or double. So, char ch = 30L; will fail even though 30 is representable in char.

**QUESTION 31**
**What gets written on the screen when the following program is compiled and run. Select the one right answer.**

```
public class test {
   public static void main(String args[]) {
      int i;
      float f = 2.3f;
      double d = 2.7;
```

```
      i = ((int)Math.ceil(f)) * ((int)Math.round(d));
      System.out.println(i);
  }
}
```

A. 4
B. 5
C. 6
D. 6.1
E. 9
F. Compilation fails
G. An exception is thrown

**Correct Answer:** E
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 32**
**Given the following code fragment:**

```
public static void main(String[] args) {
   double num = -25.67;
   System.out.println(Math.abs(num));
}
```

**What is the output?**

A. 26
B. 25.7
C. 25.00
D. 25.67

**Correct Answer:** D
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 33**
**Given the following code fragment:**

```
public static void main(String[] args) {
    int a = 10, b = 15;
    boolean result = false;
    // line n1
    System.out.println(result);
}
```

**Which two statements, when inserted at line n1 independently, enable the code to print true?**

A. `result = a == b;`
B. `result = a != b;`
C. `result = a > b;`
D. `result = !( a > b);`
E. `result = (a !> b);`

**Correct Answer:** BD
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 34**
**What is the output of the following code?**

```
public class Foo {
   public static void main(String[] args) {
     int a = 10;
     long b = 20;
     short c = 30;
     System.out.println(++a + b++ * c);
   }
}
```

A. 611
B. 641
C. 930
D. 960

**Correct Answer:** A
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 35**
**Given the following code fragment:**

```
int a = 3;
a = ++a + a++;
a = --a - a--;
System.out.println(a);
```

**What is the output?**

A. A compilation error occurs
B. `0`
C. `8`
D. `3`

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 36**
**Given the following code fragment:**

```
int a = 3;
a = ++a + ++a;
a = --a - --a;
System.out.println(a);
```

**What is the output?**

A. A compilation error occurs
B. `0`
C. `3`
D. `2`
E. `1`

**Correct Answer:** E
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 37**
**What will the following code print?**

```
public class TestClass {
    public static void main(String[] args) {
        int x = 1____3;   //1
        long y = 1_3;     //2
        float z = 3.234_567f; //3
        System.out.println(x+" "+y+" "+z);
    }
}
```

A. `Compilation error at //1`
B. `Compilation error at //2`
C. `Compilation error at //3`
D. `Compilation error at //1 and //3`
E. `10003 103 3.234567`
F. `13 13 3.234567`
G. `Compilation error at //1, //2 and //3`

**Correct Answer:** F
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
The number at //1 and //2 are actually the same. Although confusing, it is legal to have multiple underscores between two digits.

You may use underscore for all kinds of numbers including long, double, float, binary, as well as hex. For example, the following are all valid numbers

```
int hex = 0xCAFE_BABE;
float f = 9898_7878.333_333f;
int bin = 0b1111_0000_1100_1100;
```

**QUESTION 38**
**Which of the following is/are valid double values for 10 million? (A million has 6 zeros)**

A. `double d = 10,000,000.0;`
B. `double d = 10-000-000;`
C. `double d = 10_000_000;`
D. `double d = 10 000 000;`

**Correct Answer:** C
**Section: Working With Java Data Types**

**Explanation**

**Explanation/Reference:**
Beginning with Java 7, you can include underscores in between the digits. This helps in writing long numbers. For example, if you want to write 1 million, you can write: 1_000_000, which is easier than 1000000 for humans to interpret.

Note that you cannot start or end a value with an underscore though. Thus, 100_ or _100 are invalid values. _100 is actually a valid variable name!

You may use underscore for all kinds of numbers including long, double, float, binary, as well as hex.

For example, the following are all valid numbers
```
int hex = 0xCAFE_BABE;
float f = 9898_7878.333_333f;
int bin = 0b1111_0000_1100_1100;
```

**QUESTION 39**
**Given that the bit pattern for the amount $20,000 is 100111000100000, which of the following is/are valid declarations of an int variable that contains this value?**

A. `int b = (binary)100111000100000;`
B. `int b = 01001110_00100000;`
C. `int b = 0b01001110_00100000;`
D. `int b = b1001110_00100000;`

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
(binary) is invalid because binary is not a valid keyword.
In fact, since it starts with 0, it will be interpreted as an octal number.
If you want to write a value in binary, it must be prefixed with 0b or 0B.

Beginning with Java 7, you can include underscores in between the digits. This helps in writing long numbers. For example, if you want to write 1 million, you can write: 1_000_000, which is easier to interpret for humans than 1000000.

Note that you cannot start or end a value with an underscore though. Thus, 100_ or _100 would be invalid values. _100 would actually be a valid variable name

You may use underscore for all kinds of numbers including long, double, float, binary, as well as hex. For example, the following are all valid numbers
```
int hex = 0xCAFE_BABE;
float f = 9898_7878.333_333f;
int bin = 0b1111_0000_1100_1100;
```

**QUESTION 40**
**Identify the valid code fragments when occurring by themselves within a method.**

A. `long y = 123_456_L;`
B. `long z = _123_456L;`
C. `float f1 = 123_.345_667F;`
D. `float f2 = 123_345_667F;`
E. None of the above declarations are valid.

**Correct Answer:** D
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
An underscore can only occur in between two digits. So the _ before L is invalid.

An underscore can only occur in between two digits. So the _ before 1 is invalid. _123_456L is a valid variable name though.

So the following code is valid:
```
int _123_456L = 10;
long z = _123_456L;
```

An exception to this rule is that multiple continuous underscores can appear between two digits. For example, 2____3 is as good as 2_3 and is same as 23.

An underscore can only occur in between two digits. So the _ before . is invalid.

You may use underscore for all kinds of numbers including long, double, float, binary, as well as hex. For example, the following are all valid numbers
```
int hex = 0xCAFE_BABE;
float f = 9898_7878.333_333f;
int bin = 0b1111_0000_1100_1100;
```

**QUESTION 41**
**Which class definition prints `Hello Java` when it is compiled an run?**

A.
```
class Test {
        public String msg = "Hello Java";
        public static void main(String[] args) {
             System.out.println(msg);
        }
   }
```
B.
```
class Test {
        public void main() {
             System.out.print("Hello Java");
        }
        public static void main(String[] args) {
             main();
        }
   }
```
C.
```
class Test {
        {
             System.out.print("Hello Java");
        }
        public static void main(String[] args) {
            Test t = new Test();
        }
   }
```
D.
```
class Test {
        public static void main(String[] args) {
             private String msg = "Hello Java";
             System.out.print(msg);
        }
   }
```

**Correct Answer:** C
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 42**
**Under what situations does a class get a default constructor?**

A. All classes in Java get a default constructor.

B. You have to define at least one constructor to get the default constructor.
C. If the class does not define any constructors explicitly.
D. All classes get default constructor from Object class.
E. None of the above.

**Correct Answer:** C
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
If a class defines a constructor explicitly, it will not get the default constructor.

**QUESTION 43**
**Which of the following classes have a default constructor?**

```
class A {  }

class B {
    B(){ }
}

class C {
   C(String s){ }
}
```

A. A
B. A and B
C. B
D. C
E. B and C

**Correct Answer:** A
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
There is only one rule regarding the "default" constructor:
The Java compiler automatically adds a constructor that takes no argument and has the same access as the class, if and only if the programmer does not define ANY constructor in the class.

In this case, the programmer has not defined any constructor for class A, hence it will have the default constructor.

For class B, the programmer has defined a constructor that is exactly same as the default constructor that would have been provided automatically. It is a matter of interpretation whether it can be called a default constructor or not.

Based on Java Language Specification section 8.8.9, quoted below, our interpretation is that class B will not get a default constructor:

(http://docs.oracle.com/javase/specs/jls/se7/html/jls-8.html )

8.8.9 Default Constructor
If a class contains no constructor declarations, then a default constructor with no formal parameters and no throws clause is implicitly declared.

If the class being declared is the primordial class Object, then the default constructor has an empty body. Otherwise, the default constructor simply invokes the superclass constructor with no arguments.

It is a compile-time error if a default constructor is implicitly declared but the superclass does not have an accessible constructor (6.6) that takes no arguments and has no throws clause.

It follows that if the nullary constructor of the superclass has a throws clause, then a compile-time error will occur.

**QUESTION 44**
**You want to use the static member MYCONST belonging to class A in abc.org.project package.**

**Which one of the following statements shows the correct use of static import feature?**

A. `static import abc.org.project.A;`
B. `static import abc.org.project.A.MYCONST;`
C. `import static abc.org.project.A;`
D. `import static abc.org.project.A.MYCONST;`

**Correct Answer:** D
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 45**
**Which one of the following programs compiles without any errors and prints "hello world" in console when executed?**

A.
```
import static java.lang.System.out.println;

class StaticImport {
   public static void main(String []args) {
     println("hello world");
   }
}
```
B.
```
import static java.lang.System.out;

class StaticImport {
   public static void main(String []args) {
      out.println("hello world");
   }
}
```
C.
```
import static java.lang.System.out.*;

class StaticImport {
   public static void main(String []args) {
      out.println("hello world");
   }
}
```
D.
```
import static java.lang.System.out.*;

class StaticImport {
   public static void main(String []args) {
      println("hello world");
   }
}
```

**Correct Answer:** B
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 46**

**Which of the given options could be inserted at line 1 so that the following code can compile without any errors?**

```
package objective1;

// 1

public class StaticImports {
    public StaticImports() {
         out.println(MAX_VALUE);
    }
}
```

A. import static java.lang.Integer.*;
B. static import java.lang.Integer.*;
C. import java.lang.System;
D. import static java.lang.*;
E. static import java.lang.System.out;
F. import static java.lang.System.*;
G. static import java.lang.System.*;
H. import static java.lang.System.out;

**Correct Answer:** AFH
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
- The order of keywords for a static import must be "import static ... ".
- You can either import all the static members using import static java.lang.Integer.* or one specific member using import static java.lang.Integer.MAX_VALUE
- You must specify the full package name of the class that you are importing (just like the regular import statement). So, import static Integer.*; is wrong.

**QUESTION 47**
**Given the following code:**

```
1. // insert code here
2. class StatTest {
3.     public static void main(String[] args) {
4.        System.out.println(MAX_VALUE);
5.     }
6. }
```

**Which, inserted independently at line 1, allow the code compiles? (Choose all that apply.)**

A. `import static java.lang;`
B. `import static java.lang.Integer;`
C. `import static java.lang.Integer.*;`
D. `import static java.lang.Integer.*_VALUE;`
E. `import static java.lang.Integer.MAX_VALUE;`
F. None of the above statements are valid import syntax

**Correct Answer:** CE
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 48**

**Given the following code:**

```
1. // insert code here
2. public class TestStaticImport {
3.    public static void main(String[] args) {
4.      out.println(MAX_VALUE);
5.      out.println(toHexString(42));
6.    }
7. }
```

**Which two imports are required to compile?**

A. `import java.lang.*;`
B. `import static java.lang.Integer.*;`
C. `import java.lang.System;`
D. `import static java.lang.System.out;`
E. `static import java.lang.System.*;`
F. `import java.lang.System.*;`
G. `import java.lang.Integer.toHexString;`

**Correct Answer:** BD
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 49**
**Given a class Repetition:**

```
package utils;

public class Repetition {
     public static String twice(String s) {
       return s + s;
     }
}
```

**and given another class Demo:**

```
01. public class Demo {
02.    public static void main(String[] args) {
03.       System.out.println(twice("pizza"));
04.    }
05. }
```

**Which code should be inserted at line 1 of Demo.java to compile and run Demo to print "pizzapizza"?**

A. `import utils.*;`
B. `static import utils.*;`
C. `import utils.Repetition.*;`
D. `static import utils.Repetition.*;`
E. `import utils.Repetition.twice();`
F. `import static utils.Repetition.twice;`
G. `static import utils.Repetition.twice;`

**Correct Answer:** F
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 50**
**Given classes defined in two different files:**

```
package util;

public class BitUtils {
   public static void process(byte[] b) {
      /* more code here */
   }
}
```

```
1. package app;
2.
3. public class SomeApp {
4.     public static void main(String[] args) {
5.       byte[] bytes = new byte[256];
6.       // insert code here
7.    }
8. }
```

**What is required at line 6 in class SomeApp to use the process method of BitUtils?**

A. `process(bytes);`
B. `BitUtils.process(bytes);`
C. `util.BitUtils.process(bytes);`
D. `SomeApp cannot use methods in BitUtils.`
E. `Insert at line 2:`
   `import util.BitUtils.*;`

   `Then at line 6:`
   `process(bytes);`

**Correct Answer:** C
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 51**
**Given these classes in different files:**

```
package xcom;

public class Useful {
   int increment(int x) {
     return ++x;
   }
}

import xcom.*; // line 1

class Needy3 {
    public static void main(String[] args) {
        xcom.Useful u = new xcom.Useful(); // line 2
        System.out.println(u.increment(5));
    }
}
```

**Which statements are true? (Choose all that apply)**

A. `The output is 0.`
B. `The output is 5.`
C. `The output is 6.`
D. `Compilation fails.`
E. `The code compiles if line 1 is removed.`
F. `The code compiles if line 2 is changed to read`
   `Useful u = new Useful();`

**Correct Answer:** D
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 52**
**Given classes defined in two different files:**

```
package utils;

public class StringUtils {
   static void process(String s) {}
}
```

```
01. package app;
02.
03. public class SomeApp {
04.     public static void main(String[] args) {
05.       String text = "hello";
06.       // insert code here
07.    }
08. }
```

**What is required at line 6 in class SomeApp to use the process method of StringUtils?**

A. `process(text);`
B. `StringUtils.process(text);`
C. `app.StringUtils.process(text);`
D. `utils.StringUtils.process(text);`
E. `Insert first at line 2:`
   `import static utils.StringUtils.process;`

   `Then at line 6:`
   `process(text);`
F. SomeApp cannot use the process method in StringUtils.

**Correct Answer:** F
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 53**
**You have written some Java code in MyFirstClass.java file. Which of the following commands will you use to compile and run it.**
**(Assume that the code has no package declaration.)**

A. `javac MyFirstClass.java`

```
   javar MyFirstClass
```
B.
```
   javap MyFirstClass.java
   javar MyFirstClass.java
```
C.
```
   java MyFirstClass.java
   java MyFirstClass.class
```
D.
```
   javac MyFirstClass.java
   javar MyFirstClass.java
```
E.
```
   javac MyFirstClass.java
   java MyFirstClass
```

**Correct Answer:** E
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
Remember that java code must be written in a file with .java extension. If you have a public class in the code, the name of the file must be same as the name of that public class.

Compilation and execution of a Java program is two step process. You first need to compile a java file using a Java compiler. Oracle's JDK comes with a compiler. It is contained in the executable file named javac. You will find it in <jdk installation folder>/bin.

javac compiles the source code and generates bytecode in a new file with the same name as the source file but with extension .class. By default, the class file in generated in the same folder but javac is capable of placing it in a different folder if you use the -d flag. [This is just FYI and not required for the exam. -d is a very important and useful flag and we recommend that you read about it even if it is not required for the exam.]

In second step, the Java virtual machine (JVM), aka Java interpreter is invoked to execute the .class file. Oracle's JVM is contained in the executable file named java. It is also present in the same bin folder of JDK installation. It takes the fully qualified name (i.e. name including package) of the class file without extension as a argument.

**QUESTION 54**
**What will be the output of the following program when it is compiled and run with the command line:**

```
java TestClass 1 2 3
```

```
public class TestClass {
    public static void main(String[] args) {
        System.out.println("Values : "+args[0]+args[1]);
    }
}
```

A.
```
Values : java TestClass
```
B.
```
Values : TestClass 1
```
C.
```
Values : 12
```
D.
```
Values : 23
```
E.
```
Values : 3
```

**Correct Answer:** C
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
In Java, command line arguments are passed into the program using the String[] parameter to the main method. The String array contains actual parameters and does not include java and the name of the class.

Therefore, in this case, args will point to an array of Strings with 3 elements - "1", "2", and "3". The program

prints out only args[0] and args[1], which are 1 and 2.

**QUESTION 55**
**Given the code from the `Greeting.java` file:**

```
public class Greeting {
    public static void main(String[] args) {
        System.out.println("Hello " + args[0]);
    }
}
```

**Which set of commands prints Hello Duke in the console?**

A.
```
javac Greeting
java Greeting Duke
```
B.
```
javac Greeting.java Duke
java Greeting
```
C.
```
javac Greeting.java
java Greeting Duke
```
D.
```
javac Greeting.java
java Greeting.class Duke
```

**Correct Answer:** C
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 56**
**Given:**

```
public class Palindrome {
   static void main(String[] args) {
      System.out.print(args[1]);
   }
}
```

**And the commands:**

```
javac Palindrome.java
java Palindrome Wow Mom
```

**What is the result?**

A. `Compilation fails.`
B. `The code compiles, but does not execute.`
C. `Palindrome`
D. `Wow`
E. `Mom`

**Correct Answer:** B
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 57**
**Given**:

```
import static java.lang.System.*;
```

```
class _ {
   static public void main(String... __A_V_) {
     String $ = "";
     for(int x=0; ++x < __A_V_.length; )
         $ += __A_V_[x];
     out.println($);
   }
}
```

**And the command line:**
`java _ - A .`

**What is the result?**

A. -A
B. A.
C. -A.
D. _A.
E. _-A.
F. Compilation fails
G. An exception is thrown at runtime

**Correct Answer:** B
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 58**
**Given the following code:**

```
class Application {
   public static void main(String[] args) {
      int b = 3;
      if ( !(b > 3)) {
          System.out.println("square ");
      }
      {
          System.out.println("circle ");
      }
      System.out.println("...");
    }
}
```

**And the invocation:**
`java Application`

**What is the result?**

A.
```
square
...
```
B.
```
circle
...
```
C.
```
square
circle
...
```
D. `Compilation fails.`
E. `An exception is thrown`

**Correct Answer:** C

**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 59**
**Given the following code:**

```
07. class Application {
08.  public static void main(String[] args) {
09.      test(null);
10.  }
11.  public static void test(String str) {
12.    if(str == null | str.length() == 0) {
13.      System.out.println("String is empty");
14.    } else {
15.      System.out.println("String is not empty");
16.    }
17.  }
18. }
```

**And the invocation:**
`java Application`

**What is the result?**

A. `An exception is thrown at runtime`
B. `"String is empty" is printed to output`
C. `Compilation fails because of an error in line 12.`
D. `"String is not empty" is printed to output.`

**Correct Answer:** A
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
`Exception in thread "main" java.lang.NullPointerException`

**QUESTION 60**
**Given**:

```
class Application {

   public static void main(String[] args) {
     if (args.length !=0 & !args[0].isEmpty()) {
        System.out.println("Parameter one has content");
     } else {
        System.out.println("Invocation without parameters");
     }
   }

}
```

**And the command-line invocation:**
`java Application`

**What is the result?**

A. `Parameter one has content`
B. `Invocation without parameters`
C. `Nothing is printed`

D. Compilation fails
E. An exception is thrown at runtime

**Correct Answer:** E
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
    at Main.main(Main.java:3)
```

**QUESTION 61**
**Given the following code:**

```
public class Application {

     public static void main(String[] args) {
          System.out.println(true ^ false);
          System.out.println(false ^ false);
     }
}
```

**And the invocation:**
```
java Application
```

**What is the result?**

A.
```
true
true
```
B. Compilation fails
C. An exception is thrown at runtime
D.
```
false
false
```
E.
```
true
false
```
F.
```
false
true
```

**Correct Answer:** E
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
```
Main.java:4: error: not a statement
2 * 3;
```

**QUESTION 62**
**Given the following code**:

```
class Application {
   public static void main(String[] args) {
      for(int i= 0; i <= 10; i++) {
         if (i > 6)
            break;
      }
      System.out.println(i);
   }
}
```

**And the invocation:**
```
java Application
```

**What is the result?**

A. 6
B. 7
C. 10
D. 11
E. Compilation fails
F. An exception is thrown

**Correct Answer:** E
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
```
Application.java:7: error: cannot find symbol
      System.out.println(i);
```

**QUESTION 63**
**Given the following code:**

```
public class Application {

     public static void main(String[] args) {
          int xx[] = null;
          for (int i : xx) {
              System.out.println(i);
          }
     }
}
```

**And the invocation:**
```
java Application
```

**What is the result?**

A. null
B. Compilation fails
C. An exception is thrown at runtime
D. 0
E. Nothing is printed

**Correct Answer:** C
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
```
Exception in thread "main" java.lang.NullPointerException
        at Application.main(Application.java:4)
```

**QUESTION 64**
**Given the following code:**

```
public class Application {

     public static void main(String[] args) {
          2 * 3;
          for (String s : args) {
              System.out.println(s);
          }
     }
}
```

**And the invocation:**
`java Application`

**What is the result?**

A. `null`
B. `Compilation fails`
C. `An exception is thrown at runtime`
D. `0`
E. `Nothing is printed`

**Correct Answer:** B
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
```
Main.java:4: error: not a statement
2 * 3;
```

**QUESTION 65**
**Given:**

```
class Jump {
    static String args[] = {"lazy", "lion", "is", "always"};
    public static void main(String[] args) {
       System.out.println(args[1] + " " + args[2] + " " + args[3] + "
jumping");
    }
}
```

**And the commands:**

```
javac Jump.java
java Jump crazy elephant is always
```

**What is the result?**

A. `lazy lion is jumping`
B. `lion is always jumping`
C. `crazy elephant is jumping`
D. `elephant is always jumping`
E. `Compilation fails.`
F. `An exception is thrown at runtime.`

**Correct Answer:** D
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 66**
**Given this code in a file Traveler.java:**

```
class Tours {
    public static void main(String[] args) {
       System.out.print("Happy Journey! " + args[1]);
    }
}

public class Traveler {
```

```
    public static void main(String[] args) {
       Tours.main(args);
    }
}
```

**And the commands:**

```
javac Traveler.java
java Traveler Java Duke
```

**What is the result?**

A. `Happy Journey! Duke`
B. `Happy Journey! Java`
C. `An exception is thrown at runtime.`
D. `The program fails to execute due to a runtime error.`
E. `A compilation error occurs`

**Correct Answer:** A
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 67**
**Given:**

```
public class Yippee {
   public static void main(String [] args) {
     for(String arg : args) {
       System.out.print(arg + " ");
     }
   }
}
```

**and two separate command line invocations:**

```
java Yippee
java Yippee 1 2 3 4
```

**What is the result?**

A. `No output is produced.`
   `1 2 3`
B. `No output is produced.`
   `2 3 4`
C. `No output is produced.`
   `1 2 3 4`
D. `An exception is thrown at runtime.`
   `1 2 3`
E. `An exception is thrown at runtime.`
   `2 3 4`
F. `An exception is thrown at runtime.`
   `1 2 3 4`

**Correct Answer:** C
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 68**
**Given:**

```
class Course {

}

public class Student {

   public static void main(String[] args) {
      System.out.println(args[0] + " is studying " + args[1]);
   }
}
```

**Which statement is true?**

A. The commands:
   javac Course.java
   javac Student.java
   java Course
   java Student "Richard William" Java

   are used to print Richard William is studying Java

B. The commands:
   javac Student.java
   java Student Richard William Java

   throw an error about a missing Course.class file

C. The commands:
   javac Student.java
   java Student "Richard William" Java

   are used to print Richard William is studying Java

D. The commands:
   javac Student.java
   java Student Richard William Java

   are used to print Richard William is studying Java

E. None of the above is valid

**Correct Answer:** C
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 69**
**Given the following code:**

```
public class Counter {
    public static int getCount(String[] arr) {
         int count = 0;
         for(String var : arr) {
            if (var!=null) count++;
         }
         return count;
    }
    public static void main(String[] args) {
         String[] arr = new String[4];
         arr[1] = "C";
         arr[2] = "";
         arr[3] = "Java";
         System.out.print(getCount(arr));
```

```
    }
}
```

**And the commands:**

```
javac Counter.java
java Counter
```

**What is the result?**

A. 2
B. 3
C. Compilation fails
D. An exception is thrown

**Correct Answer:** B
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 70**
**Given the following code:**

```
public class Sequence {
  Sequence() {
    System.out.print("c ");
  }
  {
    System.out.print("y ");
  }
  public static void main(String[] args) {
    new Sequence().go();
  }
  void go() {
    System.out.print("g ");
  }
  static {
    System.out.print("x ");
  }
}
```

**What is the result when this code is executed?**

A. c x y g
B. c g x y
C. x c y g
D. x y c g
E. y x c g
F. y c g x
G. Compilations fails
H. An exception is thrown

**Correct Answer:** D
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 71**
**Given the following code:**

```
class Init {
  Init(int x) {
    System.out.println("1-arg const");
  }
  Init() {
    System.out.println("no-arg const");
  }
  static {
    System.out.println("1st static init");
  }
  {
    System.out.println("1st instance init");
  }
  {
    System.out.println("2nd instance init");
  }
  static {
    System.out.println("2nd static init");
  }
  public static void main(String [] args) {
    new Init();
    new Init(7);
  }
}
```

**What is the result when this code is executed?**

A. `Compilation fails`

B. `An exception is thrown`

C.
```
1st static init
2nd static init
1st instance init
2nd instance init
no-arg const
1st instance init
2nd instance init
1-arg const
```

D.
```
1st static init
2nd static init
no-arg const
1st instance init
2nd instance init
1-arg const
1st instance init
2nd instance init
```

E.
```
1st static init
2nd static init
no-arg const
1-arg const
```

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 72**
**What will be the result of attempting to compile and run the following code?**

```
public class InitClass{
   public static void main(String args[ ] ){
      InitClass obj = new InitClass(5);
   }
   int m;
   static int i1 = 5;
   static int i2 ;
   int  j = 100;
   int x;
   public InitClass(int m){
      System.out.println(i1 + "  " + i2 + "   " + x + "  " + j + "  " + m);
   }
   { j = 30; i2 = 40; }
   static { i1++; }
}
```

A. The code will fail to compile since the instance initializer tries to assign a value to a static member.
B. The code will fail to compile since the member variable x will be uninitialized when it is used.
C. The code will compile without error and will print 6 40 0 30 5 when run.
D. The code will compile without error and will print 5, 0, 0, 100, 5 when run.
E. The code will compile without error and will print 5, 40, 0, 30, 0 when run.

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
The value 5 is passed to the constructor to the local (automatic) variable m. So the instance variable m is shadowed. Before the body of the constructor is executed, the instance initializer is executed and assigns values 30 and 40 to variables j and i2, respectively. A class is loaded when it is first used.

For example,

```
class A1{
  static int i = 10;
  static { System.out.println("A1 Loaded "); }
}
public class A{
  static { System.out.println("A Loaded "); }
  public static void main(String[] args){
    System.out.println(" A should have been loaded");
    A1 a1 = null;
    System.out.println(" A1 should not have been loaded");
    System.out.println(a1.i);
  }
}
```

When you run it you get the output:

```
A Loaded
 A should have been loaded
 A1 should not have been loaded
A1 Loaded
10
```

Now, A should be loaded first as you are using its main method. Even though you are doing A1 a1 = null; A1 will not be loaded as it is not yet used (so the JVM figures out that it does not need to load it yet.) When you do a1.i, you are using A1, so before you use it, it must be loaded. That's when A1 is loaded. Finally 10 is printed.

**QUESTION 73**
**Given the following code:**

```
public class Foo {
```

```
    static int[] a;
    static {
       a[0]=2;
    }
    public static void main(String[] args ) {
       System.out.println(a[0]);
    }
}
```

**Which is the output?**

A. An java.lang.StackOverflowError is thrown
B. An java.lang.IllegalStateException is thrown
C. An java.lang.ExceptionInInitializerError is thrown
D. An java.lang.ArrayIndexOutOfBoundsException is thrown
E. An java.lang.NullPointerException is thrown
F. 2
G. Compilation fails

**Correct Answer:** C
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

```
java.lang.ExceptionInInitializerError
   Caused by: java.lang.NullPointerException
            at Main.<clinit>(Main.java:4)
```

**QUESTION 74**
**Given that TestClass is a class, how many objects and reference variables are created by the following code?**

```
TestClass t1, t2, t3, t4;
t1 = t2 = new TestClass();
t3 = new TestClass();
```

A. 2 objects, 3 references.
B. 2 objects, 4 references.
C. 3 objects, 2 references.
D. 2 objects, 2 references.
E. None of the above.

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
A declared reference variable exists regardless of whether a reference value (i.e. an object) has been assigned to it or not.

two news => two objects. t1, t2, t3, t4 => 4 references

**QUESTION 75**
**Given the code fragment:**

```
class Student {
   String name;
   int age;
}
```

**And,**

```
1. public class Test {
2.    public static void main (String[] args) {
3.       Student s1 = new Student();
4.       Student s2 = new Student();
5.       Student s3 = new Student();
6.       s1 = s3;
7.       s3 = s2;
8.       s2 = null;
9.    }
10. }
```

**Which statement is true?**

A. After line 8, three objects are eligible for garbage collection.
B. After line 8, two objects are eligible for garbage collection.
C. After line 8, one object is eligible for garbage collection.
D. After line 8, none of the objects are eligible for garbage collection.

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 76**
**In the following code, after which statement (earliest), the object originally held in s, may be garbage collected?**

```
1. public class TestClass {
2.   public static void main (String args[]){
3.      Student s = new Student("Vaishali", "930012");
4.      s.grade();
5.      System.out.println(s.getName());
6.      s = null;
7.      s = new Student("Vaishali", "930012");
8.      s.grade();
9.      System.out.println(s.getName());
10.     s = null;
11.   }
12. }


public class Student {
    private String name, rollNumber;
    public Student(String name, String rollNumber) {
        this.name = name;
        this.rollNumber = rollNumber;
    }
    //valid setter and getter for name and rollNumber follow
    public void grade() {    }
}
```

A. It will not be Garbage Collected till the end of the program.
B. Line 5
C. Line 6
D. Line 7.
E. Line 10.

**Correct Answer:** C
**Section: Working With Java Data Types**

**Explanation**

**Explanation/Reference:**
In this case, since there is only one reference to Student object, as soon as it is set to null, the object held by the reference is eligible for GC, here it is done at line 6.

Note that although an object is created at line 7 with same parameters, it is a different object and it will be eligible for GC after line 10.

The official exam objectives now explicitly mention Garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.
2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling System.gc();

**QUESTION 77**
**Given**

```
class CardBoard {
    Short story = 5;
    CardBoard go(CardBoard cb) {
        cb = null;
        return cb;
    }
    public static void main(String[] args) {
        CardBoard c1 = new CardBoard();
        CardBoard c2 = new CardBoard();
        CardBoard c3 = c1.go(c2);
        c1 = null;
        // do Stuff
}
```

**When //doStuff is executed, how many objects are candidates for GC?**

A. 0
B. 1
C. 2
D. Impossible to know
E. Compilation fails
F. An exception is thrown

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 78**
**Given**:

```
06. public class Icelandic {
07.     Long weight = 1200L;
08.     public void makeNoise() { System.out.println("vinny"); }
09.     public static void main(String[] args) {
10.         Icelandic i1 = new Icelandic();
11.         Icelandic i2 = new Icelandic();
12.         Icelandic i3 = new Icelandic();
13.         i3 = i1; i1 = i2; i2 = null; i3 = i1;
14.     }
15. }
```

**When line 14 is reached, how many objects are eligible for the garbage collector?**

A. 0
B. 1
C. 2
D. 3
E. 4
F. 6

**Correct Answer:** E
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 79**
**Given:**

```
class CardBoard {
   Short story = 200;
   CardBoard go(CardBoard cb) {
      cb = null;
      return cb;
   }
   public static void main(String[] args) {
       CardBoard c1 = new CardBoard();
       CardBoard c2 = new CardBoard();
       CardBoard c3 = c1.go(c2);
       c1 = null;
       // do Stuff
   }
}
```

**When // do Stuff is reached, how many objects are eligible for garbage collection?**

A. 0
B. 1
C. 2
D. Compilation fails
E. It is not possible to know
F. An exception is thrown at runtime

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 80**
**Given the following code, when the last line of main method is reached, how many objects are eligible for GC?**

```
public class ImmutableStrings {
    public static void main(String[] args) {
      String one = "someString";
      String two = new String("someString");
      one = null;
      two = null;
```

```
    }
}
```

A. 0 objects
B. 1 object
C. 2 objects
D. Compilation fails
E. It is not possible to know
F. An exception is thrown at runtime

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 81**
**How many String objects are created when the following method is invoked?**

```
11. public String makingStrings() {
12.    String s="Fred";
13.    s=s+"47";
14.    s=s.substring(2,5);
15.    s=s.toUpperCase();
16.    return s.toString();
17. }
```

A. 1
B. 2
C. 3
D. 4
E. 5
F. 6

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 82**
**How many String objects are created when the following method is invoked?**

```
11. public String makingStrings() {
12.    String s="FRED";
13.    s=s+"47";
14.    s=s.substring(2,5);
15.    s=s.toUpperCase();
16.    return s.toString();
17. }
```

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

F. 6

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 83**
**Given the code fragment:**

```
public class MarkList {
  int num;
  public static void graceMarks(MarkList obj4) {
     obj4.num += 10;
  }
  public static void main (String[] args) {
     MarkList obj1 = new MarkList();
     MarkList obj2 = obj1;
     MarkList obj3 = null;
     obj2.num = 60;
     graceMarks(obj2);
  }
}
```

**How many objects are created in the memory at runtime?**

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

**Correct Answer:** A
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 84**
**Given**:

```
04. public class Tahiti {
05.    Tahiti t;
06.
07.    public static void main(String[] args) {
08.       Tahiti t = new Tahiti();
09.       Tahiti t2 = t.go(t);
10.       t2 = null;
11.       // more code here
12.    }
13.
14.    Tahiti go(Tahiti t) {
15.       Tahiti t1 = new Tahiti();
16.       Tahiti t2 = new Tahiti();
17.       t1.t = t2;
18.       t2.t = t1;
19.       t.t = t2;
20.       return t1;
21.    }
22. }
```

**When line 11 is reached, how many objects are eligible for garbage collection?**

A. `0`
B. `1`
C. `2`
D. `3`
E. `Compilation fails.`

**Correct Answer:** A
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 85**
**Consider the following code snippet:**

```
public class Test {
   void test(){
       MyClass obj = new MyClass();
       obj.name = "jack";
       // 1 insert code here
    }
}

//In MyClass.java

public class MyClass{
    int value;
    String name;
}
```

**What can be inserted at // 1, which will make the object referred to by obj eligible for garbage collection?**

A. `obj.destroy();`
B. `Runtime.getRuntime().gc();`
C. `obj = null;`
D. `obj.finalize();`
E. `obj.name = null; as well as obj = null;`
F. `System.gc();`

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
Execution of garbage collector doesn't make an object eligible for garbage collection. So even if you try to invoke the garbage collector, it will not destroy the object that is not eligible for garbage collection.
Also remember that calling System.gc() or Runtime.getRuntime().gc() will not necessarily run the garbage collector. It only requests the JVM to perform garbage collection but there is no guarantee that the JVM will do it.

By the way, System.gc() is equivalent to Runtime.getRuntime().gc().

The official exam objectives now explicitly mention Garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.

2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling System.gc();

Nothing can ensure that an object will definitely be destroyed by the garbage collector. You can at most make an object eligible for GC by making sure that there are no references to it.

**QUESTION 86**
**Consider the following code:**

```
class MyClass { }

public class TestClass {
   MyClass getMyClassObject(){
      MyClass mc = new MyClass(); //1
      return mc; //2
   }

   public static void main(String[] args){
      TestClass tc = new TestClass(); //3
      MyClass x = tc.getMyClassObject(); //4
      System.out.println("got myclass object"); //5
      x = new MyClass(); //6
      System.out.println("done"); //7
   }
}
```

**After what line the MyClass object created at line 1 will be eligible for garbage collection?**

A. Line 2
B. Line 5
C. Line 6
D. Line 7
E. Never till the program ends.

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
At line 6, x starts pointing to a new MyClassObject and no reference to the original MyClass object is left.

The official exam objectives now explicitly mention Garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.
2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling System.gc();

**QUESTION 87**
**Which is the earliest line in the following code after which the object created on line // 1 can be garbage collected, assuming no compiler optimizations are done?**

```
public class NewClass{
   private Object o;
   void doSomething(Object s){  o = s;   }

   public static void main(String args[]){
      Object obj = new Object(); // 1
      NewClass tc = new NewClass(); //2
      tc.doSomething(obj); //3
      obj = new Object();    //4
      obj = null;    //5
      tc.doSomething(obj); //6
```

```
   }
}
```

A. Line 1
B. Line 2
C. Line 3
D. Line 4
E. Line 5
F. Line 6

**Correct Answer:** F
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
The official exam objectives now explicitly mention Garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.
2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling System.gc();

**QUESTION 88**
**Given:**

```
3. class Beta { }
4. class Alpha {
5.    static Beta b1;
6.    Beta b2;
7. }
8. public class Tester {
9.    public static void main(String[] args) {
10.     Beta b1 = new Beta(); Beta b2 = new Beta();
11.     Alpha a1 = new Alpha(); Alpha a2 = new Alpha();
12.     a1.b1 = b1;
13.     a1.b2 = b1;
14.     a2.b2 = b2;
15.     a1 = null; b1 = null; b2 = null;
16.     // do stuff
17.   }
18. }
```

**When line 16 is reached, how many objects will be eligible for garbage collection?**

A. 0
B. 1
C. 2
D. 3
E. 4
F. 5

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 89**
**Given:**

```
class Snoochy {
   Boochy booch;
   public Snoochy() {
       booch = new Boochy(this);
   }
}

class Boochy {
   Snoochy snooch;
   public Boochy(Snoochy s) {
       snooch = s;
   }
}
```

**And the statements:**

```
21. public static void main(String[] args) {
22.     Snoochy snoog = new Snoochy();
23.     snoog = null;
24.     // more code here
25. }
```

**Which statement is true about the objects referenced by snoog, snooch, and booch immediately after line 23 executes?**

A. None of these objects are eligible for garbage collection.
B. Only the object referenced by booch is eligible for garbage collection.
C. Only the object referenced by snoog is eligible for garbage collection.
D. Only the object referenced by snooch is eligible for garbage collection.
E. The objects referenced by snooch and booch are eligible for garbage collection.

**Correct Answer:** E
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 90**
**How many objects are eligible for garbage collection when control reaches (1)?**

```
public class Eligible {
   public static void main(String[] args) {
        for (int i = 0; i < 5; i++) {
            Eligible obj = new Eligible();
            new Eligible();
        }
        System.gc(); // (1);
   }
}
```

**Select the one correct answer.**

A. 0
B. 5
C. 10
D. Hard to say.

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 91**
**How many objects are eligible for garbage collection when control reaches (1)?**

```
public class Link {
    private Link next;
    Link(Link next) {
        this.next = next;
    }
    public void finialize() {
        System.out.print("X");
    }
    public static void main(String[] args) {
        Link p = null;
        for (int i = 0; i < 5; i++) {
           p = new Link(p);
        }
        System.gc(); // (1);
    }
}
```

**Select the one correct answer**

A. 0
B. 5
C. 10
D. Hard to say

**Correct Answer:** A
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 92**
**How many objects are eligible for garbage collection when control reaches (1)?**

```
public class Elements {
   public static void main(String[] args) {
       int[] array = new int[4];
       for (int i = 0; i < 4; i++) {
          array[i] = i;
       }
       array[0] = array[1] = array[2] = array[3] = 0;
       System.gc(); // (1);
   }
}
```

**Select the one correct answer.**

A. 0
B. 1
C. 4
D. Hard to say

**Correct Answer:** A
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 93**
**How many objects are reachable when control reaches (1)?**

```
public class Nullify {
    private static void nullify(Integer[] array) {
       array = null;
    }
    public static void main(String[] args) {
       Integer[] array = new Integer[4];
       for (int i = 0; i < 4; i++) {
         array[i] = i;
       }
       nullify(array);
       System.gc(); // (1);
    }
}
```

**Select the one correct answer.**

A. `0`
B. `1`
C. `4`
D. `5`
E. `Hard to say`

**Correct Answer:** A
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 94**
**Identify the location in the following program where the object, initially referenced with `arg1`, is eligible for garbage collection.**

```
public class MyClass {
   public static void main(String[] args) {
       String msg;
       String pre = "This program was called with ";
       String post = " as first argument.";
       String arg1 = new String((args.length > 0) ? "'" + args[0] + "'" : "<no
argument>");
       msg = arg1;
       arg1 = null; // (1)
       msg = pre + msg + post; // (2)
       pre = null; // (3)
       System.out.println(msg);
       msg = null; // (4)
       post = null; // (5)
       args = null; // (6)
    }
}
```

**Select the one correct answer.**

A. `After the line labeled (1).`
B. `After the line labeled (2).`
C. `After the line labeled (3).`

D. After the line labeled (4).
E. After the line labeled (5).
F. After the line labeled (6).

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 95**
**Given**:

```
class StaticField {
   static int i = 7;
   public static void main(String[] args) {
     StaticFied obj = new StaticField();
     obj.i++;
     StaticField.i++;
     obj.i++;
     System.out.println(StaticField.i + " "+ obj.i);
   }
}
```

**What is the result?**

A. 10 10
B. 8 9
C. 9 8
D. 7 10
E. Compilation fails
F. An exception is thrown

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 96**
**Given:**

```
class X {
    static int i;
    int j;
    public static void main(String[] args) {
         X x1 = new X();
         X x2 = new X();
         x1.i = 3;
         x1.j = 4;
         x2.i = 5;
         x2.j = 6;
         System.out.println(x1.i + " " + x1.j + " " + x2.i + " " + x2.j);
    }
}
```

**What is the result?**

A. 3 4 5 6

B. 3 4 3 6
C. 5 4 5 6
D. 3 6 4 6
E. Compilation fails
F. An exception is thrown

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 97**
**Given:**

```
class Alpha {
   int ns;
   static int s;

   Alpha(int ns) {
      if (s < ns) {
         s = ns;
         this.ns = ns;
      }
   }

   void doPrint() {
      System.out.println("ns = " + ns + " s = " + s);
   }
}
```

**And**

```
public class TestA {

   public static void main(String[] args) {
      Alpha ref1 = new Alpha(50);
      Alpha ref2 = new Alpha(125);
      Alpha ref3 = new Alpha(100);
      ref1.doPrint();
      ref2.doPrint();
      ref3.doPrint();
   }

}
```

**What is the result?**

A.
```
ns = 50 s = 125
ns = 125 s = 125
ns = 100 s = 125
```
B.
```
ns = 50 s = 125
ns = 125 s = 125
ns = 0 s = 125
```
C.
```
ns = 50 s = 50
ns = 125 s = 125
ns = 100 s = 100
```
D.
```
ns = 50 s = 50
ns = 125 s = 125
ns = 0 s = 125
```
E. Compilation fails

F. An exception is thrown

**Correct Answer:** B
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 98**
**Given:**

```
public class A{
   private int counter = 0;
   public static int getInstanceCount() {
     return counter;
   }
   public A() {
     counter++;
   }
}
```

**Given this code snippet from Class B:**

```
public class B {
   public static void main(String[] args) {
25.     A a1 = new A();
26.     A a2 = new A();
27.     A a3 = new A();
28.     System.out.println(A.getInstanceCount());
   }
}
```

**What is the result?**

A. Compilation of class A fails.
B. Line 28 prints the value 3 to System.out.
C. Line 28 prints the value 1 to System.out.
D. A runtime error occurs
E. An exception is thrown

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 99**
**Given the code fragment:**

```
class Test2 {
   int fvar;
   static int cvar;

   public static void main(String[] args) {
      Test2 t = new Test2();
      //insert code here to write field variables
   }
}
```

**Which code fragments, inserted independently, enable the code to compile?**

A.
```
t.fvar = 200;
t.cvar = 400;
```
B.
```
fvar = 200;
cvar = 400;
```
C.
```
this.fvar = 200;
this.cvar = 400;
```
D.
```
t.fvar = 200;
Test2.cvar = 400;
```
E.
```
this.fvar = 200;
Test2.cvar = 400;
```

**Correct Answer:** AD
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 100**
**Given this code:**

```
class Application {
  public static void main(String[] args) {
      SampleClass x = new SampleClass();
      SampleClass y = new SampleClass();
      System.out.println(x.increaseCount());
  }
}

public class SampleClass {
   private static int currentCount = 0;
   public SampleClass() {
      currentCount++;
   }
   public int increaseCount() {
      return currentCount++;
   }
}
```

**What is the result?**

A. `0`
B. `1`
C. `2`
D. `3`
E. `Compilation fails`
F. `An exception is thrown`

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 101**
**Given the code fragment:**

```
package p1;

public class Test {
   final double dvalue;
```

```
    final static Test ref;
    public static void main(String[] args) {
        System.out.println(ref);
        System.out.println(new Test().dvalue);
    }
}
```

**What is the result?**

A.
```
p1.Test.class
0.0
```
B.
```
<the summary address refrenced by ref>
0.000000
```
C.
```
null
0.0
```
D. `Compilation fails`

E. `A NullPointerException is thrown at runtime`

**Correct Answer:** D
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 102**
**Given the following code:**

```
class Main {

    static {
        System.out.println(Main.A);
    }

    static int A = 10;

    static {
        System.out.println(Main.A);
    }

    public static void main(String [] args)
    {
        System.out.println(Main.A);
    }

}
```

**What is the output?**

A. `Compilation fails`

B. `An exception is thrown`

C.
```
10
10
10
```
D.
```
0
0
10
```
E.
```
0
10
10
```
F.
```
0
0
0
```

**Correct Answer:** E
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 103**
**Given the following code:**

```
class Main {

    static int B = 5;
    static {
        System.out.println(B);
        System.out.println(Main.A);
    }

    static int A = 10;

    static {
        System.out.println(B);
        System.out.println(A);
    }

    public static void main(String [] args)
    {
    }

}
```

**What is the output?**

A. `Compilation fails`
B. `An exception is thrown`
C.
```
5
10
5
10
```
D.
```
0
0
5
5
```
E.
```
5
0
5
10
```
F.
```
0
0
5
10
```

**Correct Answer:** E
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 104**
**Given the following code:**

```
class Main {

    static {
        System.out.println(A);
    }

    final static int A = 10;

    static {
        System.out.println(Main.A);
    }

    public static void main(String [] args)
    {
        System.out.println(Main.A);
    }

}
```

**What is the output?**

A. `Compilation fails`
B. `An exception is thrown`
C. `10`
   `10`
   `10`
D. `0`
   `0`
   `10`
E. `0`
   `10`
   `10`
F. `0`
   `0`
   `0`

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
error: illegal forward reference
System.out.println(A);

**QUESTION 105**
**Given the following code:**

```
class Main {

    final static int B = 5;
    static {
        System.out.println(Main.B);
        System.out.println(Main.A);
    }

    final static int A = 10;

    static {
        System.out.println(Main.B);
        System.out.println(Main.A);
    }

    public static void main(String [] args)
    {
```

```
    }
}
```

**What is the output?**

A. `Compilation fails`

B. `An exception is thrown`

C.
```
5
10
5
10
```

D.
```
0
0
5
5
```

E.
```
5
0
5
10
```

F.
```
0
0
5
10
```

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 106**
**What is the output of the following code?**

```
public class NewClass3 {

    static int x = method("x");

    static {
        System.out.println("init 1");
    }

    static int y = method("y");

    static {
        System.out.println("init 2");
    }

    static int method(String name) {
        System.out.println(name);
        return 0;
    }

    public static void main(String[] args) {

    }
}
```

A. `Compilations fails`

B. `An exception is throw`

C.
```
x
init 1
```

```
   y
   init 2
```
D.
```
   x
   y
   init 1
   init 2
```
E.
```
   init 1
   init 2
   x
   y
```

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
From section 12.4.2 of the JLS:

The procedure for initializing a Class is then as follows:

1. First, initialize the final class fields whose values are compile-time constant expressions
2. Next, execute either the static initializers of the class in textual order, as though they were a single block.

**QUESTION 107**
**What is the output of the following code?**

```
public class NewClass
{
    static
    {
       System.out.println(NewClass.A + " " + " " + NewClass.B + " " +
NewClass.C + " " + NewClass.D);
    }

    static String A = "static";
    static Integer B = 0;
    static int C = 0;
    static boolean D = true;

    static
    {
       System.out.println(A + " " + " " + B + " " + C + " " + D);
    }

    int a = 1;

    {
        System.out.println(this.a + " " + this.b + " " + this.c);
    }

    double b = 2.0;
    boolean c = true;

    {
        System.out.println(this.a + " " + this.b + " " + this.c);
    }

    public static void main(String [] args)
    {
        System.out.println(A + " " + " " + B + " " + C + " " + D);
        NewClass nc = new NewClass();
        System.out.println(nc.a + " " + nc.b + " " + nc.c);
    }
```

```
}
```

A. Compilation fails

B. An exception is thrown

C.
```
null  null 0 false
static  0 0 true
static  0 0 true
1 0.0 false
1 2.0 true
1 2.0 true
```

D.
```
static  0 0 true
static  0 0 true
static  0 0 true
1 2.0 true
1 2.0 true
1 2.0 true
```

E.
```
null  null 0 false
null  null 0 false
static  0 0 true
1 0.0 false
1 0.0 false
1 2.0 true
```

F.
```
null  null 0 false
null  null 0 false
null  null 0 false
1 0.0 false
1 0.0 false
1 0.0 false
```

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
From section 12.4.2 of the JLS:

The procedure for initializing a Class is then as follows:

1. First, initialize the final class fields whose values are compile-time constant expressions
2. Next, execute either the static initializers of the class in textual order, as though they were a single block.

**QUESTION 108**
**What is the output of the following code?**

```
public class NewClass
{
    static
    {
       System.out.println(NewClass.A + " " + " " + NewClass.B + " " +
NewClass.C + " " + NewClass.D);
    }

    final static String A = "static";
    final static int B = 0;
    final static int C = 0;
    final static boolean D = true;

    static
    {
       System.out.println(A + " " + " " + B + " " + C + " " + D);
    }
```

```
    final a = 1;

    {
        System.out.println(this.a + " " + this.b + " " + this.c);
    }

    final double b = 2.0;
    final  c = true;

    {
        System.out.println(this.a + " " + this.b + " " + this.c);
    }

    public static void main(String [] args)
    {
        System.out.println(A + " " + " " + B + " " + C + " " + D);
        NewClass nc = new NewClass();
        System.out.println(nc.a + " " + nc.b + " " + nc.c);
    }
}
```

A. Compilation fails

B. An exception is thrown

C.
```
null  null 0 false
static  0 0 true
static  0 0 true
1 0.0 false
1 2.0 true
1 2.0 true
```

D.
```
static  0 0 true
static  0 0 true
static  0 0 true
1 2.0 true
1 2.0 true
1 2.0 true
```

E.
```
null  null 0 false
null  null 0 false
static  0 0 true
1 0.0 false
1 0.0 false
1 2.0 true
```

F.
```
null  null 0 false
null  null 0 false
null  null 0 false
1 0.0 false
1 0.0 false
1 0.0 false
```

**Correct Answer:** D
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
From section 12.4.2 of the JLS:

The procedure for initializing a Class is then as follows:

1. First, initialize the final class fields whose values are compile-time constant expressions
2. Next, execute either the static initializers of the class  in textual order, as though they were a single block.

**QUESTION 109**
**Consider the following program and predict the output:**

```
class Test {
   public static void main(String []args) {
     String s = new String("5");
     System.out.println(1+10+s+1+10);
   }
}
```

A. 11511
B. 1105110
C. 115110
D. 27

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 110**
**Given the code fragment:**

```
System.out.println("Result: " + 2 + 3 + 5);
System.out.println("Result: " + 2 + 3 * 5);
```

**What is the result?**

A. Result: 10
   Result: 30
B. Result: 10
   Result: 25
C. Result: 235
   Result: 215
D. Result: 215
   Result: 215
E. Compilation fails

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 111**
**Given the code fragment:**

```
System.out.println("Result: " + 3 + 5);
System.out.println("Result: " + (3 + 5));
```

**What is the result?**

A. Result: 8
   Result: 8
B. Result: 35
   Result: 8
C. Result: 8
   Result: 35
D. Result: 35
   Result: 35

**Correct Answer:** B
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 112**
**Given:**

```
StringBuffer b = new StringBuffer("3");
System.out.print(5+4+b+2+1);
```

**What is the result?**

A. 54321
B. 9321
C. 5433
D. 933
E. Output is similar to: 9java.lang.StringBuffer@100490121
F. Compilation fails

**Correct Answer:** F
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 113**
**Given:**

```
public class X {

     public static void main(String[] args) {
          String theString = "Hello World";
          System.out.println(theString.charAt(11));
     }
}
```

**What is the result?**

A. The program prints nothing
B. d
C. java.lang.StringIndexOutOfBoundsException is thrown at runtime.
D. java.lang.ArrayIndexOutOfBoundsException is thrown at runtime.
E. java.lang.NullPointerException is thrown at runtime.
F. java.lang.IndexOutOfBoundsException is thrown at runtime.
G. Compilation fails

**Correct Answer:** C
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 114**
**What is the result of compiling and running the following program.**

```
public class test {
   public static void main(String args[]) {
     String str1="abc";
     String str2="def";
     String str3=str1.concat(str2);
     str1.concat(str2);
     System.out.println(str1);
   }
}
```

A. abc
B. def
C. abcabc
D. abcdef
E. defabc
F. abcdefdef

**Correct Answer:** A
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 115**
**Given:**

```
public class Case {

     public static void main(String[] args) {
          String product = "Pen";
          product.toLowerCase();
          product.concat(" BOX".toLowerCase());
          System.out.println(product.substring(4, 6));
     }
}
```

**What is the result?**

A. box
B. nbo
C. bo
D. nb
E. An exception is thrown at runtime
F. Compilation fails

**Correct Answer:** E
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 116**
**Given the code fragment:**

```
String inputFromConsole = "   betaTEST   ";
String cleanedInput;
cleanedInput = inputFromConsole.toUpperCase();
cleanedInput = cleanedInput.trim();
```

```
System.out.println("[" + cleanedInput + "]");
```

**What is the result?**

A. [betaTest]
B. [ BETATEST ]
C. [BETATEST]
D. [ betaTEST ]

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 117**
**Given the code fragment:**

```
String digits = "0123456789";
System.out.println( digits.substring(digits.indexOf("4"), digits.indexOf("8"))
                        .concat("89"));
```

**What is the result?**

A. 3456789
B. 45678
C. 456789
D. 4567889

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 118**
**Which of the given options is the output produced by the code given below?**

```
String s1 = new String("hello");
String s2 = "hello";
String s3 = "hello";
System.out.println(s1==s3);
System.out.println(s2==s3);
System.out.println(s1.equals(s2));
```

A. true true false
B. false true true
C. false false true
D. false true false
E. true false false
F. false false false
G. true true true

**Correct Answer:** B
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 119**
**Consider the following program:**

```
class StrEqual {
   public static void main(String []args) {
      String s1 = "hi";
      String s2 = new String("hi");
      String s3 = "hi";
      if(s1 == s2) {
          System.out.println("s1 and s2 equal");
      } else {
          System.out.println("s1 and s2 not equal");
      }
      if(s1 == s3) {
          System.out.println("s1 and s3 equal");
      } else {
          System.out.println("s1 and s3 not equal");
      }
   }
}
```

**Which one of the following options provides the output of this program when executed?**

A.
```
s1 and s2 equal
s1 and s3 equal
```
B.
```
s1 and s2 equal
s1 and s3 not equal
```
C.
```
s1 and s2 not equal
s1 and s3 equal
```
D.
```
s1 and s2 not equal
s1 and s3 not equal
```

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 120**
**Given the code fragment:**

```
String s1 = "foo-bar";
String s2 = new String("foo-bar");
System.out.print(s1.equals(s2) + " ");
System.out.print(s1 == s2);
System.out.print(" " + s1.compareTo(s2));
```

**What is the result?**

A. `false false -1`
B. `true false 0`
C. `true true 0`
D. `false true -1`
E. `Compilation fails`
F. `An exception is thrown`

**Correct Answer:** B
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 121**
**What will happen when you attempt to compile and run the following code snippet?**

```
String str = "Java";
StringBuffer buffer = new StringBuffer(str);

if(str.equals(buffer))
{
  System.out.println("Both are equal");
}
else
{
  System.out.println("Both are not equal");
}
```

A. It will print - Both are not equal
B. It will print - Both are equal
C. Compile time error as you can not use equals for objects of different classes
D. Runtime error as you can not use equals for objects of different classes
E. None of these

**Correct Answer:** A
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 122**
**Given a code fragment:**

```
StringBuilder sb = new StringBuilder();
String h1 = "HelloWorld";
sb.append("Hello").append("world");
if (h1 == sb.toString()) {
    System.out.println("They match");
}
if (h1.equals(sb.toString())) {
    System.out.println("They really match");
}
```

**What is the result?**

A. They match
   They really match
B. They really match
C. They match
D. Nothing is printed to the screen
E. Compilation fails
F. An exception is thrown

**Correct Answer:** D
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 123**
**What is displayed when the following code is compiled and executed?**

```
StringBuilder s1 = new StringBuilder("Test");
StringBuilder s2 = new StringBuilder("Test");
if (s1==s2)
   System.out.println("Same");
if (s1.equals(s2))
   System.out.println("Equals");
```

A. Same
B. Equals
C. The code compiles, but nothing is displayed upon execution
D. The code fails to compile
E. Compilation fails
F. An exception is thrown

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 124**
**Given the following code fragment**:

```
class X {
    static void m(String s1) {
        s1 = "acting";
    }
    public static void main(String[] args) {
        String s2 = "action";
        m(s2);
        System.out.println(s2);
    }
}
```

**What is the result?**

A. acting
B. action
C. Compilation fails
D. An exception is thrown at runtime

**Correct Answer:** B
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 125**
**Given the code fragment:**

```
class X {
    static void m(StringBuilder sb1) {
         sb1.append("er");
    }
    public static void main (String[] args) {
         StringBuilder sb2 = new StringBuilder("moth");
```

```
        m(sb2);
        System.out.println(sb2);
    }
}
```

**What is the result?**

A. `moth`
B. `er`
C. `mother`
D. `Compilation fails`
E. `An exception is thrown at runtime`

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 126**
**What will the following class print when run?**

```
public class Sample {
   public static void main(String[] args) {
      String s1 = new String("java");
      StringBuilder s2 = new StringBuilder("java");
      replaceString(s1);
      replaceStringBuilder(s2);
      System.out.println(s1 + s2);
   }
   static void replaceString(String s) {
       s = s.replace('j', 'l');
   }
   static void replaceStringBuilder(StringBuilder s) {
       s.append("c");
   }
}
```

A. `javajava`
B. `lavajava`
C. `javajavac`
D. `lavajavac`
E. None of these
F. `Compilation fails`
G. `An exception is thrown`

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 127**
**Given the following code:**

```
2. public class Boot {
3.     static String s;
4.     static { s = ""; }
```

```
5.     { System.out.print("shinier "); }
6.     static { System.out.print(s.concat("better ")); }
7.     Boot() { System.out.print(s.concat("bigger ")); }
8.     public static void main(String[] args) {
9.        new Boot();
10.       System.out.println("boot");
11. } }
```

**What is the result?**

A. `better bigger boot`
B. `better bigger shinier boot`
C. `better shinier bigger boot`
D. `bigger shinier better boot`
E. `shinier better bigger boot`
F. `A NullPointerException is thrown at runtime.`
G. `An ExceptionInInitializationError is thrown at runtime.`

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 128**
**Given the following code:**

```
4. public class Hemlock {
5.    static StringBuffer sb;
6.    StringBuffer sb2;
7.    public static void main(String[] args) {
8.        sb = sb.append(new Hemlock().go(new StringBuffer("hey")));
9.        System.out.println(sb);
10.   }
11.   { sb2 = new StringBuffer("hi "); }
12.   StringBuffer go(StringBuffer s) {
13.       System.out.print(s + " oh " + sb2);
14.       return new StringBuffer("ey");
15.   }
16.   static { sb = new StringBuffer("yo "); }
17. }
```

**What is the result?**

A. `yo ey`
B. `hey oh hi`
C. `hey oh hi ey`
D. `oh hi hey`
E. `hey oh hi yo ey`
F. `yo hey oh hi ey`
G. `Compilation fails.`
H. `An exception is thrown at runtime`

**Correct Answer:** E
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 129**
**What output will be produced when the following code is executed?**

```
Integer i1 = new Integer(34);
Integer i2 = 34;
if (i1.equals(i2))
   System.out.println("equals() true");
if (i1 == i2)
   System.out.println("== true");
```

A. Compilation error
B. An exception is thrown
C. equals() true
D. == true
E. equals() true
   == true
F. Nothing is show in the output

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 130**
**Given**:

```
1. public class KungFu {
2.    public static void main(String[] args) {
3.      Integer x = 400;
4.      Integer y = x;
5.      x++;
6.      StringBuilder sb1 = new StringBuilder("123");
7.      StringBuilder sb2 = sb1;
8.      sb1.append("5");
9.      System.out.println((x == y) + " " + (sb1 == sb2));
10.  }
11.}
```

**What is the result?**

A. true true
B. false true
C. true false
D. false false
E. Compilation fails
F. An exception is thrown at runtime

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 131**
**Consider the following program:**

```
class Increment {
```

```
    public static void main(String []args) {
      Integer i = 10;
      Integer j = 11;
      Integer k = ++i; // INCR
      System.out.println("k == j is " + (k == j));
      System.out.println("k.equals(j) is " + k.equals(j));
   }
}
```

**Which one of the following options correctly describes the behavior of this program?**

A. When executed, this program prints
   `k == j is false`
   `k.equals(j) is false`
B. When executed, this program prints
   `k == j is true`
   `k.equals(j) is false`
C. When executed, this program prints
   `k == j is false`
   `k.equals(j) is true`
D. When executed, this program prints
   `k == j is true`
   `k.equals(j) is true`
E. When compiled, the program will result in a compiler error in the line marked with the comment INCR.

**Correct Answer:** D
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 132**
**Given:**

```
public class Spock {
   public static void main(String[] args) {
      Long tail = 2000L;
      Long distance = 1999L;
      Long story = 1000L;
      if ((tail > distance) ^ ((story * 2) == tail))
        System.out.print("1");
      if ((distance + 1 != tail) ^ ((story * 2) == distance))
        System.out.print("2");
   }
}
```

**What is the result?**

A. `1`
B. `2`
C. `12`
D. `Compilation fails`
E. `No output is produced`
F. `An exception is thrown at runtime`

**Correct Answer:** E
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 133**
**Given:**

```
4. public class SpecialOps {
5.    public static void main(String[] args) {
6.      String s = "";
7.      Boolean b1 = true;
8.      Boolean b2 = false;
9.      if((b2 = false) | (21%5) > 2) s += "x";
10.     if(b1 || (b2 = true)) s += "y";
11.     if(b2 == true) s += "z";
12.     System.out.println(s);
13.   }
14. }
```

**Which are true? (Choose all that apply.)**

A. Compilation fails
B. x will be included in the output
C. y will be included in the output
D. z will be included in the output
E. An exception is thrown at runtime

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 134**
**Given the following code fragment**:

```
2. public class Bunnies {
3.    static int count = 0;
4.    Bunnies() {
5.      while(count < 10) new Bunnies(++count);
6.    }
7.    Bunnies(int x) { }
8.    public static void main(String[] args) {
9.        new Bunnies();
10.       new Bunnies(count);
11.       System.out.println(count++);
12.   }
13. }
```

**What is the result?**

A. 9
B. 10
C. 11
D. 12
E. Compilation fails
F. An exception is thrown at runtime

**Correct Answer:** B
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 135**
**What output will be produced when the following code is executed?**

```
Integer my_Integer = new Integer(34);
long my_long = 34L;

if (my_Integer.equals(my_long))
   System.out.println("equals() true");

if (my_Integer == my_long)
   System.out.println("== true");
```

A. Compilation error
B. An exception is thrown
C. equals() true
D. == true
E. equals() true
   == true
F. Nothing is show in the output

**Correct Answer:** D
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 136**
**Given**:

```
class TKO {
    public static void main(String[] args) {
      String s = "-";
      Integer x = 343;
      long L343 = 343L;

      if(x.equals(L343)) s += ".e1 ";
      if(x.equals(343)) s += ".e2 ";
      Short s1 = (short)((new Short((short)343)) / (new Short((short)49)));
      if(s1 == 7) s += "=s ";
      if(s1 < new Integer(7+1)) s += "fly ";

      System.out.println(s);
   }
}
```

**Which of the following will be included in the output String s? (Choose all that apply.)**

A. .e1
B. .e2
C. =s
D. fly
E. None of the above
F. Compilation fails
G. An exception is thrown at runtime

**Correct Answer:** BCD
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 137**
**Given the following code:**

```
public class App {
   public static void main(String[] args) {
      Boolean[] bool = new Boolean[2];
      bool[0] = new Boolean(Boolean.parseBoolean("True"));
      bool[1] = new Boolean(null);

      System.out.println(bool[0] + " " + bool[1]);
  }
}
```

**What is the output?**

A. `true null`
B. `Compilation fails`
C. `A Exception is thrown at runtime`
D. `true false`
E. `false false`
F. `false true`

**Correct Answer:** D
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

```
Allocates a Boolean object representing the value true if the string argument
is not null and is equal, ignoring case, to the string "true". Otherwise,
allocate a Boolean object representing the value false.

Examples:
new Boolean("True") produces a Boolean object that represents true.
new Boolean("yes") produces a Boolean object that represents false.

public static boolean parseBoolean(String s)

Parses the string argument as a boolean. The boolean returned represents the
value true if the string argument is not null and is equal, ignoring case, to
the string "true".
Example: Boolean.parseBoolean("True") returns true.
Example: Boolean.parseBoolean("yes") returns false.
```

**QUESTION 138**
**What gets printed when the following code is compiled and run with the following command**

```
java Test 2

public class Test {
   public static void main(String args[]) {
      Integer intObj=Integer.valueOf(args[args.length-1]);
      int i = intObj.intValue();
      if(args.length > 1)
         System.out.println(i);
      if(args.length > 0)
         System.out.println(i - 1);
      else
         System.out.println(i - 2);
   }
}
```

**Select the one correct answer.**

A. `Test`
B. `Test -1`
C. `0`
D. `1`
E. `2`
F. `Compilation fails`
G. `An exception is thrown`

**Correct Answer:** D
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 139**
**Given:**

```
class Feline {
   public static void main(String[] args) {
     Long x = 42L;
     Long y = 44L;
     System.out.print(" " + 7 + 2 + " ");
     System.out.print(foo() + x + 5 + " ");
     System.out.println(x + y + foo());
   }
   static String foo() {
     return "foo";
   }
}
```

**What is the result?**

A. `9 foo47 86foo`
B. `9 foo47 4244foo`
C. `9 foo425 86foo`
D. `9 foo425 4244foo`
E. `72 foo47 86foo`
F. `72 foo47 4244foo`
G. `72 foo425 86foo`
H. `72 foo425 4244foo`
I. `Compilation fails`

**Correct Answer:** G
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 140**
**Consider the following program:**

```
class NullAccess {
   public static void main(String []args) {
      String str = null;
      System.out.println(str.valueOf(10));
```

```
    }
}
```

**Which of the following statements correctly describes the behavior of this program?**

A. `This program will result in a compiler error.`
B. `This program will throw a java.lang.NullPointerException`
C. `This program will print 10 in console.`
D. `This program will print null in console.`

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 141**
**Consider the following code segment:**

```
String str = "A.B.C!";
System.out.println(str.replaceAll(".", ",").replace("!", "?"));
```

**When executed, this code segment will print the following:**

A. `A,B,C!`
B. `A,B,C?`
C. `,,,,,,`
D. `A.B.C?`

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 142**
**Given the following code:**

```
public class App {
  public static void main(String[] args) {
    String str = " ";
    str.trim();
    System.out.println(str.equals("") + " " + str.isEmpty());
  }
}
```

**What is the output?**

A. `true false`
B. `false false`
C. `true true`
D. `false true`
E. `Compilation fails`

**Correct Answer:** B
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 143**
**Given the following code:**

```
public class Test {

   public static void main(String[] args) {
       String[] planets = {"Mercury", "Venus", "Earth", "Mars"};
       System.out.println(planets.length());
       System.out.println(planets[1].length());
   }
}
```

**What is the output?**

A. 4
   5
B. 4
   21
C. 5
   4
D. 4
   7
E. 3
   5
F. 4
   4
G. Compilation fails

**Correct Answer:** G
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 144**
**What will be the result of attempting to compile and run the following program?**

```
public class TestClass {
    public static void main(String args[ ] ){
         String s = "hello";
         StringBuilder sb = new StringBuilder( "hello" );
         sb.reverse();
         s.reverse();
         if( s == sb.toString() )  System.out.println( "Equal" );
         else System.out.println( "Not Equal" );
    }
}
```

A. Compilation error.
B. It will print 'Equal'.
C. It will print 'Not Equal'.
D. Runtime error.
E. None of the above.

**Correct Answer:** A
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 145**
**What will the following code print?**

```
String s = "blooper";
StringBuilder sb = new StringBuilder(s);
sb.append(s.substring(4)).delete(3, 5);
System.out.println(sb);
```

A. blorbloo
B. bloper
C. bloerper
D. blooperper
E. bloo
F. Compilation fials

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
s.substring(4) => "blooper".substring(4) => per
sb.append(s.substring(4)).delete(3, 5); => "blooperper".delete(3, 5) => bloerper

public String substring(int beginIndex)
Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.
Examples:
"unhappy".substring(2) returns "happy"
"Harbison".substring(3) returns "bison"
"emptiness".substring(9) returns "" (an empty string)

public StringBuilder delete(int start, int end)
Removes the characters in a substring of this sequence. The substring begins at the specified start and extends to the character at index end - 1 or to the end of the sequence if no such character exists. If start is equal to end, no changes are made.

**QUESTION 146**
**You are developing a banking module. You have developed a class named ccMask that has a maskcc method.**

**Given the code fragment:**

```
class CCMask {
    public static String maskCC(String creditCard) {
         String x = "XXXX XXXX XXXX ";
         //line n1
    }
    public static void main(String[] args) {
         System.out.println(maskCC("1234-5678-9101-1121"));
    }
}
```

**You must ensure that the maskcc method returns a string that hides all digits of the credit card number except the four last digits (and also hide the hyphens that separate each group of four digits).**

**Which code fragments should you use at line n1, independently, to achieve this requirement?**

A.
```
StringBuilder sb = new StringBuilder(creditCard);
sb.substring(15, 19);
return x + sb;
```
B.
```
return x + creditCard.substring(15, 19);
```
C.
```
StringBuilder sb = new StringBuilder(x);
sb.append(creditCard, 15, 19);
return sb.toString();
```
D.
```
StringBuilder sb = new StringBuilder(creditCard);
StringBuilder s = sb.insert(0,x);
return s.toString();
```
E. None of the above

**Correct Answer:** BC
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 147**
**Given:**

```
22. StringBuilder sb1 = new StringBuilder("123");
23. String s1 = "123";
24. // insert code here
25. System.out.println(sb1 + " " + s1);
```

**Which code fragment, inserted at line 24, outputs "123abc 123abc"?**

A.
```
sb1.append("abc");
s1.append("abc");
```
B.
```
sb1.append("abc");
s1.concat("abc");
```
C.
```
sb1.concat("abc");
s1.append("abc");
```
D.
```
sb1.concat("abc");
s1.concat("abc");
```
E.
```
sb1.append("abc");
s1 = s1.concat("abc");
```
F.
```
sb1.concat("abc");
s1 = s1.concat("abc");
```
G.
```
sb1.append("abc");
s1 = s1 + s1.concat("abc");
```
H.
```
sb1.concat("abc");
s1 = s1 + s1.concat("abc");
```

**Correct Answer:** E
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 148**
**Given:**

```
01. public class TestString3 {
02.     public static void main(String[] args) {
```

```
03.        // insert code here
04.        System.out.println(s);
05.     }
06. }
```

**Which two code fragments, inserted independently at line 3, generate the output 4247? (Choose two.)**

A.
```
String s = "123456789";
s = (s-"123").replace(1,3,"24") - "89";
```

B.
```
StringBuffer s = new StringBuffer("123456789");
s.delete(0,3).replace(1,3,"24").delete(4,6);
```

C.
```
StringBuffer s = new StringBuffer("123456789");
s.substring(3,6).delete(1,3).insert(1, "24");
```

D.
```
StringBuilder s = new StringBuilder("123456789");
s.substring(3,6).delete(1,2).insert(1, "24");
```

E.
```
StringBuilder s = new StringBuilder("123456789");
s.delete(0,3).delete(1,3).delete(2,5).insert(1, "24");
```

**Correct Answer:** BE
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 149**
**Given:**

```
class Polish {
    public static void main(String[] args) {
       int x = 4;
       StringBuffer sb = new StringBuffer("..fedcba");
       sb.delete(3,6);
       sb.insert(3, "az");
       if(sb.length() > 6) x = sb.indexOf("b");
       sb.delete((x-3), (x-2));
       System.out.println(sb);
    }
}
```

**What is the result?**

A. `.faza`
B. `.fzba`
C. `..azba`
D. `.fazba`
E. `..fezba`
F. `Compilation fails`
G. `An exception is thrown`

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 150**
**Given:**

```
3. public class Theory {
4.     public static void main(String[] args) {
5.        String s1 = "abc";
6.        String s2 = s1;
7.        s1 += "d";
8.        System.out.println(s1 + " " + s2 + " " + (s1==s2));
9.
10.       StringBuffer sb1 = new StringBuffer("abc");
11.       StringBuffer sb2 = sb1;
12.       sb1.append("d");
13.       System.out.println(sb1 + " " + sb2 + " " + (sb1==sb2));
14.    }
15. }
```

**Which are true? (Choose all that apply.)**

A. Compilation fails
B. The first line of output is abc abc true
C. The first line of output is abc abc false
D. The first line of output is abcd abc false
E. The second line of output is abcd abc false
F. The second line of output is abcd abcd true
G. The second line of output is abcd abcd false

**Correct Answer:** DF
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 151**
**Consider the following program:**

```
class SBAppend {
  public static void main(String []args) {
    Object nullObj = null;
    StringBuffer strBuffer = new StringBuffer(10);
    strBuffer.append("hello ");
    strBuffer.append("world ");
    strBuffer.append(nullObj);
    strBuffer.insert(11, '!');
    System.out.println(strBuffer);
  }
}
```

**Which one of the following options correctly describes the behavior of this program?**

A. This program prints the following: hello world!
B. This program prints the following: hello world! null
C. This program throws a java.lang.NullPointerException
D. This program throws an java.lang.InvalidArgumentException
E. This program throws an java.lang.ArrayIndexOutOfBoundsException

**Correct Answer:** B
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 152**
**Given the code fragment:**

```
class Lang {
   private String category = "procedura1";
   public static void main (String[] args) {
       Lang obj1 = new Lang();
       Lang obj2 = new Lang();
       if (obj1.category == obj2.category) {
           System.out.println("Equal");
       }
       else {
           System.out.println("Not equal");
       }
       if (obj1.category.equals(obj2.category)) {
           System.out.println("Equal");
       }
       else {
           System.out.println("Not equal");
       }
    }
}
```

**What is the result?**

A.
```
Equal
Not equal
```
B.
```
Not equal
Equal
```
C.
```
Equal
Equal
```
D.
```
Not equal
Not equal
```

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 153**
**Given the code fragment:**

```
class Prime {
   int num;
   Prime(int num) {
      this.num = num;
   }
}

public class Test {
    public static void main (String[] args) {
        Prime obj1 = new Prime(13);
        Prime obj2 = new Prime(13);
        if (obj1 == obj2) {
           System.out.println("Equal");
        }
        else {
           System.out.println("Not equal");
        }
        if (obj1.equals(obj2)) {
           System.out.println("Equal");
        }
        else {
```

```
            System.out.println("Not equal");
         }
      }
}
```

**What is the result?**

A.
```
Equal
Not equal
```
B.
```
Not equal
Equal
```
C.
```
Equal
Equal
```
D.
```
Not equal
Not equal
```

**Correct Answer:** D
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 154**
**Consider the following program:**

```
class ArrayCompare {
    public static void main(String []args) {
      int []arr1 = {1, 2, 3, 4, 5};
      int []arr2 = {1, 2, 3, 4, 5};
      System.out.println("arr1 == arr2 is " + (arr1 == arr2));
      System.out.println("arr1.equals(arr2) is " + arr1.equals(arr2));
      System.out.println("Arrays.equals(arr1, arr2) is " +
java.util.Arrays.equals(arr1, arr2));
   }
}
```

**Which one of the following options provides the output of this program when executed?**

A.
```
arr1 == arr2 is false
arr1.equals(arr2) is false
Arrays.equals(arr1, arr2) is true
```
B.
```
arr1 == arr2 is true
arr1.equals(arr2) is false
Arrays.equals(arr1, arr2) is true
```
C.
```
arr1 == arr2 is false
arr1.equals(arr2) is true
Arrays.equals(arr1, arr2) is true
```
D.
```
arr1 == arr2 is true
arr1.equals(arr2) is true
Arrays.equals(arr1, arr2) is false
```
E.
```
arr1 == arr2 is true
arr1.equals(arr2) is true
Arrays.equals(arr1, arr2) is true
```

**Correct Answer:** A
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 155**

**Which of the following are true regarding overloading of a method?**

A. An overloading method must have a different parameter list and same return type as that of the overloaded method.
B. If there is another method with the same name but with a different number of arguments in a class then that method can be called as overloaded.
C. If there is another method with the same name and same number and type of arguments but with a different return type in a class then that method can be called as overloaded.
D. An overloaded method means a method with the same name and same number and type of arguments exists in the super class and sub class.

**Correct Answer:** B
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
For overloading a method, the "signature" of the overloaded methods must be different.

In simple terms, a method signature includes method name and the number and type of arguments that it takes. So if the parameter list of the two methods with the same name are different either in terms of number or in terms of the types of the parameters, then they are overloaded. Note that return type is not considered a part of the method signature.

For example:
Method m1 is overloaded if you have two methods : void m1(int k); and void m1 (double d); or if you have: void m1(int k); and void m1(int k, double d);

**QUESTION 156**
**Given the following code, which method declarations can be inserted at line 1 without any problems?**

```
public class OverloadTest{
    public int sum(int i1, int i2) { return i1 + i2; }
    // 1
}
```

A. `public int sum(int a, int b) { return a + b; }`
B. `public int sum(long i1, long i2) { return (int) i1; }`
C. `public int sum(int i1, long i2) { return (int) i2; }`
D. `public long sum(long i1, int i2) { return i1 + i2; }`
E. `public long sum(int i1, int i2) { return i1 + i2; }`

**Correct Answer:** BCD
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 157**
**Consider the following method...**

```
public int setVar(int a, int b, float c) {
   ...
}
```

**Which of the following methods correctly overload the above method?**

A.
```
public int setVar(int a, float b, int c){
       return (int)(a + b + c);
```

```
   }
B. public int setVar(int a, float b, int c){
        return this(a, c, b);
   }
C. public int setVar(int x, int y, float z){
        return x+y;
   }
D. public float setVar(int a, int b, float c){
         return c*a;
   }
E. public float setVar(int a){
         return a;
   }
```

**Correct Answer:** AE
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 158**
**Consider the following code:**

```
public class Varargs {
    public void test() {
        test1(10);     //1
        test1(10, 20); //2
    }
    public static void main(String[] args){
        new Varargs().test();
    }
    //insert method here.
}
```

**Which of the following lines can be added independently to the above class so that it will run without any errors or exceptions?**

A. `public void test1(int i, int j) { }`
B. `public void test1(int i, int... j) { }`
C. `public void test1(int... i) { }`
D. `public void test1(int i...) { }`
E. `public void test1(int[] i) { }`

**Correct Answer:** BC
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 159**
**Given:**

```
1. public class A {
2.     public void doit() {
3.     }
4.
5.     public String doit() {
6.       return "a";
7.     }
8.
```

```
9.     public double doit(int x) {
10.      return 1.0;
11.    }
12.}
```

**What is the result?**

A. An exception is thrown at runtime.
B. Compilation fails because of an error in line 9.
C. Compilation fails because of an error in line 5.
D. Compilation succeeds and no runtime errors with class A occur.

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 160**
**Consider the following class definition:**

```
public class TestClass{
   public static void main(String[] args){  new TestClass().sayHello(); }   //1
   public static void sayHello() { System.out.println("Static Hello World"); }
 //2
   public void sayHello() { System.out.println("Hello World "); }  //3
}
```

**What will be the result of compiling and running the class?**

A. It will print Hello World.
B. It will print Static Hello World.
C. Compilation error at line 2.
D. Compilation error at line 3.
E. Runtime Error.

**Correct Answer:** D
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
You cannot have two methods with the same signature (name and parameter types) in one class.
Also, even if you put one sayHello() method in other class which is a subclass of this class, it won't compile because you cannot override/hide a static method with a non static method and vice versa.

**QUESTION 161**
**What will be printed when the following code is compiled and run?**

```
public class LoadTest {

    public static void main(String[] args) {
         LoadTest t = new LoadTest();
         int i = t.getLoad();
         double d = t.getLoad();
         System.out.println( i + d );
    }

    public int getLoad() {
```

```
        return 1;
    }

    public double getLoad(){
        return 3.0;
    }

}
```

A. 13.0
B. 4.0
C. 4
D. The code will not compile.

**Correct Answer:** D
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
You cannot have more than one method in a class with the same signature. Method signature includes method name and the argument list but does not include return type. Therefore, the two getLoad() methods have the same signature and will not compile.

This shows that method overloading cannot be done on the basis of the return types.

**QUESTION 162**
**Given the following code:**

```
public class MainTest {

    public static void main(int[] args) {
        System.out.println("int main " + args[0]);
    }

    public static void main(Object[] args) {
        System.out.println("Object main " + args[0]);
    }

    public static void main(String[] args) {
        System.out.println("String main " + args[0]);
    }

}
```

**and the commands:**

```
javac MainTest.java
java MainTest 1 2 3
```

**What is the result?**

A. String main 1
B. Object main 1
C. int main 1
D. Compilation fails
E. An error is thrown at runtime

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 163**
**Given**

```
1. public class A {
2.    public String doit(int x, int y){
3.       return "a";
4.    }
5.
6.    public String doit(int... vals){
7.       return "b";
8.    }
9. }
```

**and:**

```
25. A a = new A();
26. System.out.println(a.doit(4, 5));
```

**What is the result?**

A. Line 26 prints "a" to System.out.
B. Line 26 prints "b" to System.out.
C. An exception is thrown at line 26 at runtime.
D. Compilation of class A will fail due to an error in line 6.

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 164**
**Given**

```
1. public class A {
2.    public String doit(int[] vals){
3.       return "a";
4.    }
5.
6.    public String doit(int... vals){
7.       return "b";
8.    }
9. }
```

**and:**

```
25. A a = new A();
26. System.out.println(a.doit(4, 5));
```

**What is the result?**

A. Line 26 prints "a" to System.out.
B. Line 26 prints "b" to System.out.
C. An exception is thrown at line 26 at runtime.
D. Compilation of class A fails

**Correct Answer:** D
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
```
Main.java:6: error: cannot declare both doit(int...) and doit(int[]) in Main
    public String doit(int... vals){
```

**QUESTION 165**
**Given**

```
1. public class A {
2.    public String doit(int[] vals, int a){
3.       return "a";
4.    }
5.
6.    public String doit(int... vals){
7.       return "b";
8.    }
9. }
```

**and:**

```
25. A a = new A();
26. System.out.println(a.doit(4, 5));
```

**What is the result?**

A. Line 26 prints "a" to System.out.
B. Line 26 prints "b" to System.out.
C. An exception is thrown at line 26 at runtime.
D. Compilation of class A fails

**Correct Answer:** B
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 166**
**Given**

```
class Alien {
    String invade(short ships) { return "a few"; }
    String invade(short... ships) { return "many"; }
}
class Defender {
    public static void main(String [] args) {
        System.out.println(new Alien().invade(7));
    }
}
```

**What is the result?**

A. many
B. a few
C. Compilation fails
D. An exception is thrown
E. Output is not predictable

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

```
error: no suitable method found for invade(int)
    System.out.println(new Main().invade(7));
                                 ^
    method Main.invade(short) is not applicable
      (argument mismatch; possible lossy conversion from int to short)
    method Main.invade(short...) is not applicable
      (varargs mismatch; possible lossy conversion from int to short)
1 error
```

**QUESTION 167**
**Given:**

```
public class Barn {
   public static void main(String[] args) {
      new Barn().go("hi", 1);
      new Barn().go("hi", "world", 2);
   }
   public void go(String... y, int x) {
      System.out.print(y[y.length - 1] + " ");
   }
}
```

**What is the result?**

A. `hi hi`
B. `hi world`
C. `world world`
D. `Compilation fails`
E. `An exception is thrown at runtime`

**Correct Answer:** D
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 168**
**Given the following code:**

```
class OverloadingTest{
     void m1(int x){
         System.out.println("m1 int");
     }
     void m1(double x){
         System.out.println("m1 double");
     }
     void m1(String x){
         System.out.println("m1 String");
     }
}

public class TestClass {
     public static void main(String[] args) {
         OverloadingTest ot = new OverloadingTest();
         ot.m1(1.0);
     }
```

```
}
```

**What will be the output?**

A. `It will fail to compile.`
B. `m1 int`
C. `m1 double`
D. `m1 String`

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 169**
**Consider the following code:**

```
public class Varargs {

   public void test() {
      test1(10, 20);      //1
   }

   public void test1(int i, int... j) { System.out.println("1"); }
   public void test1(int... i )       { System.out.println("2"); }
   public void test1(int i, int j)    { System.out.println("3"); }

   public static void main(String[] args) {
     new Varargs().test();
   }
}
```

**What will the program print?**

A. `1`
B. `2`
C. `3`
D. `It will not compile.`
E. `Exception at runtime`

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 170**
**What will the following code print when run?**

```
public class Noobs {
    public void m(int a){
        System.out.println("In int ");
    }
    public void m(char c){
        System.out.println("In char ");
    }
    public static void main(String[] args) {
        Noobs n = new Noobs();
        int a = 'a';
```

```
        char c = 6;
        n.m(a);
        n.m(c);
    }
}
```

A. In int
   In char
B. In char
   In int
C. In int
   In int
D. In char
   In char
E. It will not compile.

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 171**
**Given:**

```
public class Yikes {
   public static void go(Long n) {
     System.out.print("Long ");
   }
   public static void go(Short n) {
     System.out.print("Short ");
   }
   public static void go(int n) {
     System.out.print("int ");
   }
   public static void main(String[] args) {
      short y = 6;
      long z = 7;
      go(y);
      go(z);
  }
}
```

**What is the result?**

A. int Long
B. Short Long
C. Compilation fails.
D. An exception is thrown at runtime.

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 172**
**Given:**

```
public class Yikes {
```

```
   public static void go(Object o) {
     System.out.print("Object ");
   }
   public static void go(Integer n) {
     System.out.print("Integer ");
   }
   public static void go(Short n) {
     System.out.print("Short ");
   }
   public static void go(int n) {
     System.out.print("int ");
   }
   public static void main(String[] args) {
      short y = 6;
      long z = 7;
      go(y);
      go(z);
  }

}
```

**What is the result?**

A. int Long
B. Short Long
C. Compilation fails.
D. An exception is thrown at runtime.
E. Short Object
F. int Object

**Correct Answer:** F
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 173**
**Given:**

```
public class Yikes {

   public static void go(Long l) {
     System.out.print("Long ");
   }
   public static void go(Short n) {
     System.out.print("Short ");
   }
   public static void go(short n) {
     System.out.print("short ");
   }
   public static void main(String[] args) {
      short y = 6;
      int z = 7;
      go(y);
      go(z);
  }
}
```

**What is the result?**

A. short Long

B. Short Long
C. Compilation fails.
D. An exception is thrown at runtime.

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 174**
**Given:**

```
public class Yikes {

   public static void go(Number n) {
     System.out.print("Number ");
   }

   public static void go(Long l) {
     System.out.print("Long ");
   }
   public static void go(Short n) {
     System.out.print("Short ");
   }
   public static void go(short n) {
     System.out.print("short ");
   }
   public static void main(String[] args) {
      short y = 6;
      int z = 7;
      go(y);
      go(z);
  }
}
```

**What is the result?**

A. short Long
B. Short Long
C. Compilation fails.
D. An exception is thrown at runtime.
E. short Number
F. Short Number

**Correct Answer:** E
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 175**
**Consider the following code:**

```
class TestClass {
   void probe(int... x)  { System.out.println("In ..."); }     //1
   void probe(Integer x) { System.out.println("In Integer"); } //2
   void probe(long x)    { System.out.println("In long"); }    //3
   void probe(Long x)    { System.out.println("In LONG"); }    //4
```

```
   public static void main(String[] args){
      Integer a = 4;
      new TestClass().probe(a); //5
      int b = 4;
      new TestClass().probe(b); //6
   }
}
```

**What will it print when compiled and run?**

A.
```
In Integer
In long
```
B.
```
In ...
In LONG
if //2 and //3 are commented.
```
C.
```
In Integer
In ...
if //4 is commented.
```
D. `It will not compile, if //1, //2, and //3 are commented.`

E.
```
In LONG
In long
if //1 and //2 are commented.
```
F. `Compilation fails`

**Correct Answer:** AD
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 176**
**Consider the following code:**

```
class TestClass{
   void probe(Integer x){ System.out.println("In Integer"); }       //2
   void probe(Object x) { System.out.println("In Object"); }        //3
   void probe(Long x)   { System.out.println("In Long"); }          //4

   public static void main(String[] args){
      String a = "hello";
      new TestClass().probe(a);
   }
}
```

**What will be printed?**

A. `In Integer`
B. `In Object`
C. `In Long`
D. `It will not compile`

**Correct Answer:** B
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 177**
**Given the following code:**

```
public class SumTest {

     public static void doSum(Integer x, Integer y) {
           System.out.println("Integer sum is " + (x + y));
     }

     public static void doSum(double x, double y) {
           System.out.println("double sum is " + (x + y));
     }

     public static void doSum(float x, float y) {
           System.out.println("float sum is " + (x + y));
     }

     public static void doSum(int x, int y) {
           System.out.println("int sum is " + (x + y));
     }

     public static void main(String[] args) {
           doSum(10, 20);
           doSum(10.0, 20.0);
     }
}
```

**What is the output?**

A. `int sum is 30`
   `float sum is 30.0`

B. `int sum is 30`
   `double sum is 30.0`

C. `Integer sum is 30`
   `double sum is 30.0`

D. `Integer sum is 30`
   `float sum is 30.0`

**Correct Answer:** B
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 178**
**Consider the following class...**

```
class TestClass {

    void probe(Object x) { System.out.println("In Object"); } //3

    void probe(Number x) { System.out.println("In Number"); } //2

    void probe(Integer x) { System.out.println("In Integer"); } //2

    void probe(Long x) { System.out.println("In Long"); } //4

    public static void main(String[] args){
        double a = 10;
        new TestClass().probe(a);
    }

}
```

**What will be printed?**

A. In Number
B. In Object
C. In Long
D. In Integer
E. It will not compile.

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
Here, we have four overloaded probe methods but there is no probe method that takes a double parameter. However, a double will be boxed into a Double and class Double extends Number. Therefore, a Double can be passed to the method that takes Number. A Double can also be passed to a method that takes Object, but Number is more specific than Object therefore probe(Number ) will be called.

We advise you to run this program and try out various combinations. The exam has questions on this pattern.

**QUESTION 179**
**Given**

```
class Eggs {

    int doX(Long x, Long y) {
       return 1;
    }
    int doX(long... x) {
       return 2;
    }
    int doX(Integer x, Integer y) {
       return 3;
    }
    int doX(Number n, Number m) {
       return 4;
    }
    public static void main(String[] args) {
        new Eggs().go();
    }
    void go() {
        short s = 7;
        System.out.print(doX(s,s) + " ");
        System.out.println(doX(7,7));
    }

}
```

**What is the result?**

A. 1 1
B. 2 1
C. 3 1
D. 4 1
E. 2 3
F. 3 3
G. 4 3
H. Compilation fails
I. An exception is thrown

**Correct Answer:** G
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 180**
**Given**

```
class Eggs {

    int doX(Long x, Long y) {
       return 1;
    }
    int doX(long... x) {
       return 2;
    }
    int doX(Object n, Object m) {
       return 3;
    }
    int doX(Number n, Number m) {
       return 4;
    }
    public static void main(String[] args) {
        new Eggs().go();
    }
    void go() {
        long val = 7;
        System.out.println(doX(val,val));
    }

}
```

**What is the result?**

A. 1
B. 2
C. 3
D. 4
E. Compilation fails
F. An exception is thrown

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 181**
**Given**

```
class Eggs {

    int doX(long... x) {
       return 2;
    }
    int doX(Object n, Object m) {
       return 3;
    }
    int doX(Number n, Number m) {
       return 4;
```

```
    }
    public static void main(String[] args) {
        new Eggs().go();
    }
    void go() {
        long val = 7;
        System.out.println(doX(val,val));
    }

}
```

**What is the result?**

A. 2
B. 3
C. 4
D. Compilation fails
E. An exception is thrown

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 182**
**Consider the following program:**

```
class Overload {

   private Overload(Object o) {
      System.out.println("Object");
   }
   private Overload(double [] arr) {
      System.out.println("double []");
   }
   private Overload() {
      System.out.println("void");
   }
   public static void main(String[]args) {
      new Overload(null); // MARKER
   }

}
```

**Which one of the following options correctly describes the behavior of this program?**

A. It throws a compiler error in the line marked with the comment MARKER for ambiguous overload.
B. When executed, the program prints the following: Object.
C. When executed, the program prints the following: double [].
D. When executed, the program prints the following: void.

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
The overload resolution matches to the most specific overload. When the argument null is passed, there are two candidates, Overload(Object) and Overload(double[]), and of these two, Overload(double[]) is the most specific overload, so the compiler resolves to calling that method.

If code was like above, compile error

```
class Overload {
   private Overload(String o) {
      System.out.println("String");
   }
   private Overload(double [] arr) {
      System.out.println("double []");
   }
   private Overload() {
      System.out.println("void");
   }
   public static void main(String[]args) {
      new Overload(null); // MARKER
   }
}
```

**QUESTION 183**
**Consider the following code:**

```
public class TestClass{
   public void method(Object o){
      System.out.println("Object Version");
   }
   public void method(java.io.FileNotFoundException s){
      System.out.println("java.io.FileNotFoundException Version");
   }
   public void method(java.io.IOException s){
      System.out.println("IOException Version");
   }
   public static void main(String args[]){
      TestClass tc = new TestClass();
      tc.method(null);
   }
}
```

**What would be the output when the above program is compiled and run?**

A. `It will print Object Version`
B. `It will print java.io.IOException Version`
C. `It will print java.io.FileNotFoundException Version`
D. `It will not compile.`
E. `It will throw an exception at runtime.`

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
The reason is quite simple, the most specific method depending upon the argument is called. Here, null can be passed to all the 3 methods but FileNotFoundException class is the subclass of IOException which in turn is the subclass of Object.
So, FileNotFoundException class is the most specific class.

**QUESTION 184**
**Consider the following code:**

```
public class TestClass {
    public void method(Object o){
        System.out.println("Object Version");
    }
    public void method(String s){
```

```
        System.out.println("String Version");
    }
    public void method(StringBuffer s){
        System.out.println("StringBuffer Version");
    }
    public static void main(String args[]){
        TestClass tc = new TestClass();
        tc.method(null);
    }
}
```

**What would be the output when the above program is compiled and run?**

A. `It will print Object Version`
B. `It will print String Version`
C. `It will print StringBuffer Version`
D. `It will not compile.`
E. `It will throw an exception at runtime.`

**Correct Answer:** D
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
Here, null can be passed as both StringBuffer and String and none is more specific than the other. So, it will not compile.

**QUESTION 185**
**What is meant by "encapsulation" ?**

A. There is no way to access member variable.
B. There are no member variables.
C. Member fields are declared private and public accessor/mutator methods are provided to access and change their values if needed.
D. Data fields are declared public and accessor methods are provided to access and change their values.

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
Encapsulation is one of the 4 fundamentals of OOP (Object Oriented Programming).

Encapsulation means that the internal representation of an object is generally hidden from view outside of the object's definition. Typically, only the object's own methods can directly inspect or manipulate its fields. Some languages like Smalltalk and Ruby only allow access via object methods, but most others (e.g. C++ or Java) offer the programmer a degree of control over what is hidden, typically via keywords like public and private.

Hiding the internals of the object protects its integrity by preventing users from setting the internal data of the component into an invalid or inconsistent state. A benefit of encapsulation is that it can reduce system complexity, and thus increases robustness, by allowing the developer to limit the interdependencies between software components.

**QUESTION 186**
**Encapsulation ensures that ...**

A. classes are able to inherit functionality from other classes.
B. classes expose only certain fields and methods to other classes for access.

C. classes designate certain methods to be abstract and let them be implemented by subclasses.
D. a method that takes a class X object as a parameter can be passed an object of a subclass of X.

**Correct Answer:** B
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 187**
**A programmer is designing a class to encapsulate the information about an inventory item. A JavaBeans component is needed to do this. The InventoryItem class has private instance variables to store the item information:**

```
10. private int itemId;
11. private String name;
12. private String description;
```

**Which method signature follows the JavaBeans naming standards for modifying the itemId instance variable?**

A. `itemID(int itemId)`
B. `update(int itemId)`
C. `setItemId(int itemId)`
D. `mutateItemId(int itemId)`
E. `updateItemID(int itemId)`

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 188**
**A JavaBeans component has the following field:**

```
11. private boolean enabled;
```

**Which two pairs of method declarations follow the JavaBeans standard for accessing this field? (Choose two.)**

A.
```
public void setEnabled( boolean enabled)
public boolean getEnabled()
```
B.
```
public void setEnabled( boolean enabled)
public void isEnabled()
```
C.
```
public void setEnabled( boolean enabled)
public boolean isEnabled()
```
D.
```
public boolean setEnabled( boolean enabled)
public boolean getEnabled()
```

**Correct Answer:** AC
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 189**
**Which statement is true about a mutator method?**

A. It must be declared private.
B. It replaces the default constructor.
C. It returns mutated instance members.
D. It can be used to assign data to instance members.

**Correct Answer:** D
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

```
//Accessor for firstName
public String getFirstName() {
   return firstName;
}

//Accessor for middleNames
public String getMiddlesNames() {
   return middleNames;
}


//Mutator for address
public void setAddress(String address) {
   this.address = address;
}

//Mutator for username
public void setUsername(String username){
   this.username = username;
}
```

**QUESTION 190**
**Which two actions will improve the encapsulation of a class?**

A. Changing the access modifier of a field from `public` to `private`.
B. Removing the `public` modifier from a `class` declaration.
C. Changing the return type of a method to `void`.
D. Returning a copy of the contents of an array or `ArrayList` instead of a direct reference.

**Correct Answer:** AD
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 191**
**The `protected` modifier on a field declaration within a `public` class means that the field**

_____

A. cannot be modified.
B. can be read but not written from outside the class.
C. can be read and written from this class and its subclasses only within the same package.
D. can be read and written from this class and its subclasses defined in any package.
E. can be read and written from this class and its subclasses defined in any package and any other class in the same package

**Correct Answer:** E
**Section: Working with Methods and Encapsulation**

**Explanation**

**Explanation/Reference:**

**QUESTION 192**
**Given:**

```
20. public class CreditCard {
21.
22.     private String cardID;
23.     private int limit;
24.     public String ownerName;
25.
26.     public void setCardlnformation(String cardID, String ownerName, int
limit) {
27.          this.cardID = cardID;
28.          this.ownerName = ownerName;
29.          this.limit = limit;
30.     }
31. }
```

**Which is true?**

A. The class is fully encapsulated.
B. The code demonstrates polymorphism
C. The `ownerName` variable breaks encapsulation.
D. The `cardlD` and limit variables break polymorphism
E. The `setCardlnformation` method breaks encapsulation

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 193**
**Consider the following two classes (in the same package but defined in different source files):**

```
public class Square {
    double side = 0;
    double area;

    public Square(double length) {
       this.side = length;
    }

    public double getSide() {
       return side;
    }

    public void setSide(double side) {
       this.side = side;
    }

    double getArea() {
       return area;
    }
}

public class TestClass {
    public static void main(String[] args) throws Exception {
        Square sq = new Square(10.0);
```

```
        sq.area = sq.getSide()*sq.getSide();
        System.out.println(sq.getArea());
    }
}
```

**You are assigned the task of refactoring the Square class to make it better in terms of encapsulation.**
**What changes will you make to this class?**

A. Make `setSide()` method private.
B. Make `getArea()` method private.
C. Make `side` and `area` fields private.
D. Make the `side` field private and remove the `area` field.
E. Change `getArea()` method to:

```
public double getArea(){
     return side * side;
}
```

F. Add a `setArea()` method.

**Correct Answer:** DE
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
There can be multiple ways to accomplish this. The exam asks you questions on the similar pattern.
The key is that your data variable should be private and the functionality that is to be exposed outside should be public. Further, your setter methods should be coded such that they don't leave the data members inconsistent with each other.

**QUESTION 194**
**Consider the following two classes (in the same package but defined in different source files)**

```
public class Square {
    double side = 0;
    double area;

    public Square(double length){
        this.side = length;
    }

    public double getSide() {
        return side;
    }

    public void setSide(double side) {
        this.side = side;
    }

    double getArea() {
        return area;
    }
}

public class TestClass {
    public static void main(String[] args) {
        Square sq = new Square(10.0);
        sq.area = sq.getSide() * sq.getSide();
        System.out.println(sq.getArea());
    }
}
```

**You are assigned the task of refactoring the Square class to make it better in terms of encapsulation.**
**What changes will you make to this class?**

A. Add a `calculateArea()` method:

```
private void calculateArea() {
   this.area = this.side * this.side;
}
```

B. Make `side` and `area` fields private.

C. Change `setSide()` method to:

```
public void setSide(double d){
   this.side = d;
   calculateArea();
}
```

D. Make the `getArea()` method public.

E. Add a `setArea()` method:

```
public void setArea(double d){
   area = d;
}
```

**Correct Answer:** ABCD
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
There can be multiple ways to accomplish this. The exam asks you questions on the similar pattern.
The key is that your data variable should be private and the functionality that is to be exposed outside should be public. Further, your setter methods should be coded such that they don't leave the data members inconsistent with each other.

**QUESTION 195**
**Consider the following code:**

```
import java.util.ArrayList;

public class Student{

    ArrayList<Integer> scores;
    private double average;

    public ArrayList<Integer> getScores(){
         return scores;
    }

    public double getAverage(){
         return average;
    }

    private void computeAverage(){
        //valid code to compute average
        average = //update average value
    }

    public Student(){
        computeAverage();
    }
}
```

**What can be done to improve the encapsulation of this class?**

A. Make the class private.
B. Make the `scores` instance field private.
C. Make `getScores()` protected.
D. Make `computeAverage()` public.
E. Change `getScores()` to return a copy of the scores list:

```
public ArrayList<Integer> getScores() {
    return new ArrayList(scores);
}
```

**Correct Answer:** BE
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
An important aspect of encapsulation is that other classes should not be able to modify the state fields of a class directly. Therefore, the data members should be private (or protected if you want to allow subclasses to inherit the field) and if the class wants to allow access to these fields, it should provide appropriate setters and getters with public access.

If you return the same scores list, the caller would be able to add or remove elements from it, thereby rendering the average incorrect. This can be prevented by returning a copy of the list.

**QUESTION 196**
**Given:**

```
public class Circle {
    double radius;
    public double area;
    public Circle(double r) {
        radius = r;
    }
    public double getRadius() {
        return radius;
    }
    public void setRadius(double r) {
        radius = r;
    }
    public double getArea() {
        return /* ??? */;
    }
}

class App {
    public static void main(String[] args) {
       Circle c1 = new Circle(17.4);
       c1.area = Math.PI * c1.getRadius() * c1.getRadius();
    }
}
```

**This class is poorly encapsulated. You need to change the Circle class to compute and return the area instead.**

**Which two modifications are necessary to ensure that the class is being properly encapsulated?**

A. Remove the area field.
B. Change the `getArea()` method as follows:

```
public double getArea() {
```

```
        return Math.PI * radius * radius;
    }
```

C. Add the following method:

```
    public void updateArea() {
        area = Math.PI * radius * radius;
    }
```

D. Change the access modifier of the `setRadius()` method to be `protected`.

**Correct Answer:** AB
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 197**
**You are writing a class named Bandwidth for an internet service provider that keeps track of number of bytes consumed by a user.**
**The following code illustrates the expected usage of this class**

```
class User {
    Bandwidth bw = new Bandwidth();
    public void consume(int bytesUsed) {
        bw.addUsage(bytesUsed);
    }

    ... other irrelevant code
}

class Bandwidth {
    private int totalUsage;
    private double totalBill;
    private double costPerByte;

    //add your code here

    ...other irrelevant code
}
```

**Your goal is to implement a method addUsage (and other methods, if required) in Bandwidth class such that all the bandwidth used by a User is reflected by the totalUsage field and totalBill is always equal to totalUsage*costPerByte.**

**Further, that a User should not be able to tamper with the totalBill value and is also not able to reduce it.**

**Which of the following implementation(s) accomplishes the above?**

A.
```
public void addUsage(int bytesUsed){
    if(bytesUsed>0){
       totalUsage = totalUsage + bytesUsed;
       totalBill = totalBill + bytesUsed*costPerByte;
    }
}
```

B.
```
protected void addUsage(int bytesUsed){
     totalUsage += bytesUsed;
     totalBill = totalBill + bytesUsed*costPerByte;
}
```

C.
```
private void addUsage(int bytesUsed){
     if(bytesUsed>0){
         totalUsage = totalUsage + bytesUsed;
         totalBill = totalUsage*costPerByte;
```

```
        }
    }
```

D.
```
public void addUsage(int bytesUsed){
     if(bytesUsed>0){
         totalUsage = totalUsage + bytesUsed;
     }
}

public void updateTotalBill(){
     totalBill = totalUsage*costPerByte;
}
```

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
D.
This is not a good approach because once the User class calls addUsage() method, totalBill field will not reflect the correct amount unless User also calls updateTotalBill, which means Bandwidth class is now dependent on some other class to keep its internal state consistent with the business logic.

B.
There is no validity check for bytesUsed argument. User will be able to tamper will the bill by suppling a negative number for bytesUsed.

C.
If this method is made private, User class will not be able to access it.

**QUESTION 198**
**Given:**

```
package test;

class Target {
   public String name = "hello";
}
```

**What can directly access and change the value of the variable name?**

A. any class
B. only the `Target` class
C. any class in the `test` package
D. any class that extends `Target`

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 199**
**Which of the following are benefits of polymorphism? (Select 2 options)**

A. It makes the code more reusable.
B. It makes the code more efficient.
C. It protects the code by preventing extension.
D. It makes the code more dynamic.

**Correct Answer:** AD

**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
Polymorphism allows the actual decision of which method is to be invoked to be taken at runtime based on the actual class of object. This is dynamic binding and makes the code more dynamic.

This option is a bit ambiguous because it is not clear which efficiency is it talking about - execution, memory, or maintenance. Our guess is that it is referring to execution efficiency. It is not true because polymorphism causes a very slight degradation due to dynamic binding at run time.

**QUESTION 200**
**Under what situations does a class get a default constructor?**

A. All classes in Java get a default constructor.
B. You have to define at least one constructor to get the default constructor.
C. If the class does not define any constructors explicitly.
D. All classes get default constructor from Object class.
E. None of the above.

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 201**
**Which of the following are true about the "default" constructor?**

A. It is provided by the compiler only if the class and any of its super classes does not define any constructor.
B. It takes no arguments.
C. A default constructor is used to return a default value.
D. To define a default constructor, you must use the default keyword.
E. It is always public.

**Correct Answer:** B
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
The access type of a default constructor is same as the access type of the class. Thus, if a class is public, the default constructor will be public.

**QUESTION 202**
**Which of these statements are true?**

A. All classes must explicitly define a constructor.
B. A constructor can be declared private.
C. A constructor can declare a return value.
D. A constructor must initialize all the member variables of a class.
E. A constructor can access the non-static members of a class.

**Correct Answer:** BE
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 203**
**Which of the following classes have a default constructor?**

```
class A {  }

class B {
  B(){ }
}

class C {
  C(String s){ }
}
```

A. A
B. A and B
C. B
D. C
E. B and C

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
There is only one rule regarding the "default" constructor:

The Java compiler automatically adds a constructor that takes no argument and has the same access as the class, if and only if the programmer does not define ANY constructor in the class. In this case, the programmer has not defined any constructor for class A, hence it will have the default constructor.

For class B, the programmer has defined a constructor that is exactly same as the default constructor that would have been provided automatically. It is a matter of interpretation whether it can be called a default constructor or not.

Based on Java Language Specification section 8.8.9, quoted below, our interpretation is that class B will not get a default constructor:
(http://docs.oracle.com/javase/specs/jls/se7/html/jls-8.html )

8.8.9 Default Constructor
If a class contains no constructor declarations, then a default constructor with no formal parameters and no throws clause is implicitly declared. If the class being declared is the primordial class Object, then the default constructor has an empty body.

Otherwise, the default constructor simply invokes the superclass constructor with no arguments. It is a compile-time error if a default constructor is implicitly declared but the superclass does not have an accessible constructor (6.6) that takes no arguments and has no throws clause.

It follows that if the nullary constructor of the superclass has a throws clause, then a compile-time error will occur.

**QUESTION 204**
**What will be the result of attempting to compile the following program?**

```
public class TestClass{
   long l1;
   public void TestClass(long pLong) { l1 = pLong ; }  //(1)
   public static void main(String args[]){
      TestClass a, b ;
      a = new TestClass();  //(2)
      b = new TestClass(5);  //(3)
```

```
   }
}
```

A. A compilation error will be encountered at (1), since constructors should not specify a return value.
B. A compilation error will be encountered at (2), since the class does not have a default constructor.
C. A compilation error will be encountered at (3).
D. The program will compile correctly.
E. It will not compile because parameter type of the constructor is different than the type of value passed to it.

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
The declaration at (1) declares a method, not a constructor because it has a return value. The method happens to have the same name as the class, but that is ok. The class has an implicit default constructor since the class contains no constructor declarations. This allows the instantiation at (2) to work.

**QUESTION 205**
**Given:**

```
class X {
   String str = "default";
   X(String s) {
      str = s;
   }
   void print() {
      System.out.println(str);
   }
   public static void main(String[] args) {
      new X("hello").print();
   }
}
```

**What is the result?**

A. hello
B. default
C. Compilations fails.
D. The program prinths nothing.
E. An exception is thrown at runtime.

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 206**
**What is the result?**

```
11. public class Person {
12.     String name = "No name";
13.     public Person(String nm) { name = nm; }
14. }
15.
16. public class Employee extends Person {
17.     String empID = "0000";
18.     public Employee(String id) { empID = id; }
```

```
19. }
20.
21. public class EmployeeTest {
22.     public static void main(String[] args){
23.         Employee e = new Employee("4321");
24.         System.out.println(e.empID);
25.     }
26. }
```

A. 4321
B. 0000
C. An exception is thrown at runtime.
D. Compilation fails because of an error in line 18.

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 207**
**Given:**

```
class ClassA {
    public int numberOfInstances;

    protected ClassA(int numberOfInstances) {
        this.numberOfInstances = numberOfInstances;
    }
}

public class ExtendedA extends ClassA {
    private ExtendedA(int numberOfInstances) {
        super(numberOfInstances);
    }
    public static void main(String[] args) {
       ExtendedA ext = new ExtendedA(420);
       System.out.print(ext.numberOfInstances);
    }
}
```

**Which statement is true?**

A. 420 is the output
B. An exception is thrown at runtime.
C. All constructors must be declared public.
D. Constructors CANNOT use the private modifier.
E. Constructors CANNOT use the protected modifier.

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 208**
**Which of the following are true about the "default" constructor?**

A. It is provided by the compiler only if the class does not define any constructor.

B. It initializes the instance members of the class.
C. It calls the no-args constructor of the super class.
D. It initializes instance as well as class fields of the class.
E. It is provided by the compiler if the class does not define a 'no-args' constructor.

**Correct Answer:** AC
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
The default constructor is provided by the compiler only when a class does not define ANY constructor explicitly. For example,

```
public class A{
  public A()  //This constructor is automatically inserted by the compiler
because there is no other constructor defined by the programmer explicitly.
  {
    super();  //Note that it calls the super class' default no-args
constructor.
  }
}

public class A{
  //Compiler will not generate any constructor because the programmer has
defined a constructor.
  public A(int i){
     //do something
  }
}
```

**QUESTION 209**
**Given:**

```
1. public class Plant {
2.   private String name;
3.
4.  public Plant(String name) {
5.    this.name = name;
6.  }
7.
8.  public String getName() {
9.    return name;
10.  }
11.}
```

```
1. public class Tree extends Plant {
2.   public void growFruit() {
3.   }
4.
5.  public void dropLeaves() {
6.  }
7. }
```

**Which statement is true?**

A. The code will compile without changes.
B. The code will compile if

```
public Tree() {
  Plant();
}
```

is added to the Tree class

C. The code will compile if

```
public Plant() {
   Tree();
}
```

is added to the Plant class.

D. The code will compile if

```
public Plant() {
   this("fern");
}
```

is added to the Plant class.

E. The code will compile if

```
public Plant() {
   Plant("fern");
}
```

is added to the Plant class.

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 210**
**Given:**

```
class Employee {
   String name;
   double baseSalary;
   Employee(String name, double baseSalary) {
     this.name = name;
     this.baseSalary = baseSalary;
   }
}
```

```
09. public class SalesPerson extends Employee {
10.    double commission;
11.
12.    public SalesPerson(String name, double baseSalary, double commission) {
13.       // insert code here
14.    }
15. }
```

**Which two code fragments, inserted independently at line 13, will compile? (Choose two.)**

A. `super(name, baseSalary);`

B. `this.commission = commission;`

C.
```
super();
this.commission = commission;
```

D.
```
this.commission = commission;
super();
```

E.
```
super(name, baseSalary);
this.commission = commission;
```

F.
```
this.commission = commission;
super(name, baseSalary);
```

G. `super(name, baseSalary, commission);`

**Correct Answer:** AE
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 211**
**Given:**

```
01. class Super {
02.     private int a;
03.     protected Super(int a) { this.a = a; }
04. }

11. class Sub extends Super {
12.     public Sub(int a) { super(a); }
13.     public Sub() { this.a = 5; }
14. }
```

**Which two, independently, will allow Sub to compile? (Choose two.)**

A. Change line 2 to:
```
public int a;
```
B. Change line 2 to:
```
protected int a;
```
C. Change line 13 to:
```
public Sub() {
  this(5);
}
```
D. Change line 13 to:
```
public Sub() {
   super(5);
}
```
E. Change line 13 to:
```
public Sub() {
   super(a);
}
```

**Correct Answer:** CD
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 212**
**Given the following code, which of the constructors shown in the options can be added to class B without causing a compilation to fail?**

```
class A {
   int i;
   public A(int x) {
      this.i = x;
   }
}

class B extends A {
```

```
   int j;
   public B(int x, int y) {
      super(x);
      this.j = y;
   }
}
```

A.
```
B( ) { }
```
B.
```
B(int y ) {
   j = y;
}
```
C.
```
B(int y ) {
   super(y*2);
   j = y;
}
```
D.
```
B(int y ) {
   i = y;
   j = y*2;
}
```
E.
```
B(int z ) {
   this(z, z);
}
```

**Correct Answer:** CE
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 213**
**Given:**

```
class Atom {
    Atom() { System.out.print("atom "); }
}

class Rock extends Atom {
    Rock(String type) { System.out.print(type); }
}

public class Mountain extends Rock {
    Mountain() {
       super("granite ");
       new Rock("granite ");
    }
    public static void main(String[] a) { new Mountain(); }
}
```

**What is the result?**

A. Compilation fails.
B. atom granite
C. granite granite
D. atom granite granite
E. An exception is thrown at runtime.
F. atom granite atom granite

**Correct Answer:** F
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 214**
**Given:**

```
1. class X {
2.   X() { System.out.print(1); }
3.
4.  X(int x) {
5.    this(); System.out.print(2);
6.  }
7. }
8.
9. public class Y extends X {
10.  Y() { super(6); System.out.print(3); }
11.
12.  Y(int y) {
13.    this(); System.out.println(4);
14.  }
15.
16.  public static void main(String[] a) { new Y(5); }
17.}
```

**What is the result?**

A. 13
B. 134
C. 1234
D. 2134
E. 2143
F. 4321

**Correct Answer:** C
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 215**
**Given:**

```
class C1 {
   public C1() { System.out.print(1); }
}

class C2 extends C1 {
   public C2() { System.out.print(2); }
}

class C3 extends C2 {
   public C3() { System.out.println(3); }
}

public class Ctest {
   public static void main(String[] a) { new C3(); }
}
```

**What is the result?**

A. 3
B. 23
C. 32
D. 123
E. 321
F. Compilation fails
G. An exception is thrown at runtime

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 216**
**Given:**

```
class A {
    public A() {
          System.out.print("A ");
    }
}

class B extends A {
    public B() {                                    //line n1
          System.out.print("B ");
    }
}

class C extends B {
     public C() {                                  //line n2
          System.out.print("C ");
     }

     public static void main(String[] args) {
          C c = new C();
     }
}
```

**What is the result?**

A. C B A
B. C
C. A B C
D. Compilation fails at line n1 and line n2

**Correct Answer:** C
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 217**
**What will the following code print when compiled and run?**

```
class X {
   public X(){
     System.out.println("In X");
   }
}
```

```
class Y extends X {
    public Y(){
      super();
      System.out.println("In Y");
    }
}

class Z extends Y {
     public Z(){
        System.out.println("In Z");
     }
}

public class Test {
     public static void main(String[] args) {
         Y y = new Z();
     }
}
```

A. `It will not compile.`

B.
```
In X
In Y
In Z
```

C.
```
In Z
In Y
In X
```

D.
```
In Y
In X
In Z
```

E.
```
In Z
In X
In Y
```

**Correct Answer:** B
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 218**
**Given:**

```
public class Hello {
   String title;
   int value;
   public Hello() {
      title += " World";
   }
   public Hello(int value) {
      this.value = value;
      title = "Hello";
      Hello();
   }
}
```

**and:**

```
Hello c = new Hello(5);
System.out.println(c.title);
```

**What is the result?**

A. `Hello`
B. `Hello World`
C. `Compilation fails.`
D. `Hello World 5`
E. `The code runs with no output.`
F. `An exception is thrown at runtime.`

**Correct Answer:** C
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 219**
**Given this code:**

```
1. public class Car {
2.   private int wheelCount;
3.   private String vin;
4.   public Car(String vin){
5.     this.vin = vin;
6.     this.wheelCount = 4;
7.   }
8.   public String drive(){
9.     return "zoom-zoom";
10.  }
11.  public String getInfo() {
12.    return "VIN: " + vin + " wheels: " + wheelCount;
13.  }
14.}
```

**And**

```
1. public class MeGo extends Car {
2.   public MeGo(String vin) {
3.     this.wheelCount = 3;
4.   }
5. }
```

**What two must the programmer do to correct the compilation errors? (Choose two.)**

A. insert a call to `this()` in the Car constructor
B. insert a call to `this()` in the MeGo constructor
C. insert a call to `super()` in the MeGo constructor
D. insert a call to `super(vin)` in the MeGo constructor
E. change the `wheelCount` variable in Car to `protected`
F. change line 3 in the MeGo class to `super.wheelCount = 3;`

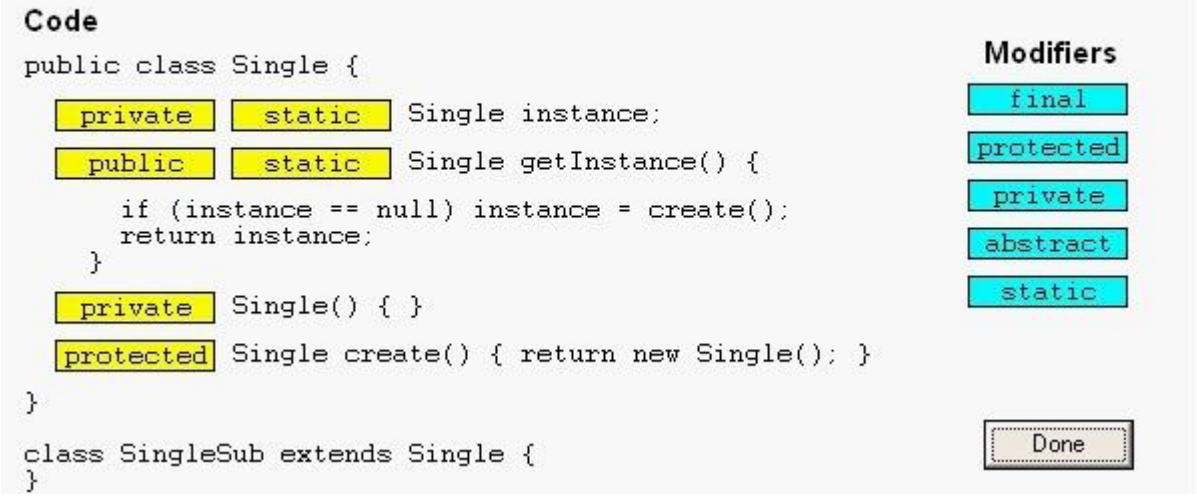
**Correct Answer:** DE
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 220**

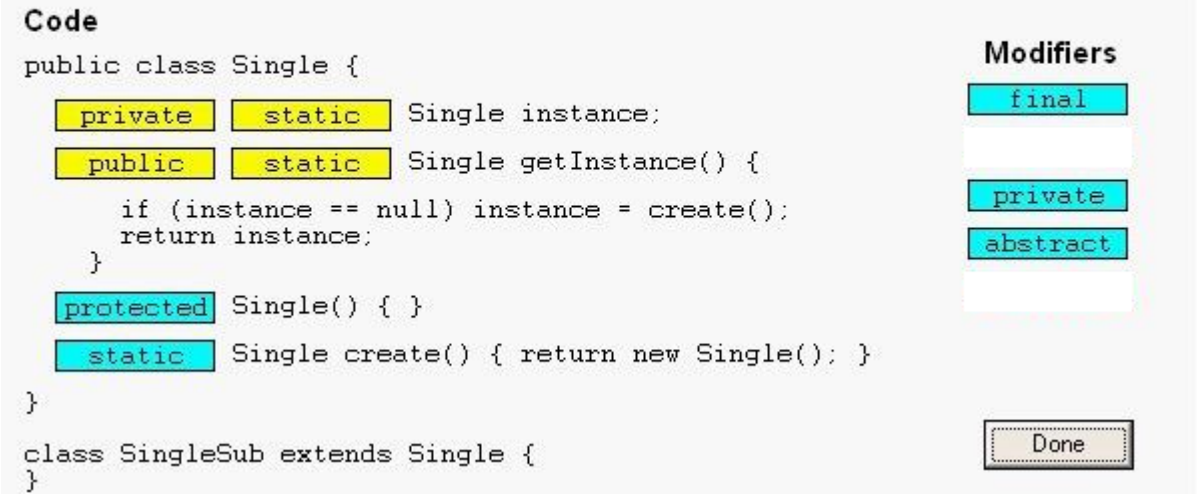
**Select and Place:**

Replace two of the Modifiers that appear in the `Single` class to make the code compile.
Note: Three modifiers will not be used and four modifiers in the code will remain unchanged.




**Correct Answer:**

Replace two of the Modifiers that appear in the `Single` class to make the code compile.
Note: Three modifiers will not be used and four modifiers in the code will remain unchanged.




**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 221**
**Given this code in a file Application.java:**

```
public class Application {
    static void main(String[] args) {
       System.out.print("Hello World! " + args[1]);
    }
}
```

**And the commands:**

```
javac Application.java
java Application Java Duke
```

**What is the result?**

A. `Hello World! Duke`
B. `Hello World! Java`
C. `An exception is thrown at runtime.`
D. `The program fails to execute due to a runtime error.`
E. `The code does not compile`

**Correct Answer:** D
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 222**
**Given:**

```
package beans.plans;

public class WirelessPlan {
    public void doCall() { }
    public void doMessage() { }
}
```

**And:**

```
package beans.handlers;

//line n1

public class Handler {
    public static void main(String[] args) {
       WirelessPlan wp = new WirelessPlan();
       wp.doMessage();
       wp.doCall();
    }
}
```

**Which code fragments, when inserted at line n1 independently, enables the code to compile successfully?**

A. `import plans.WirelessPlan;`
B. `import beans.*.*;`
C. `import beans.plans.WirelessPlan;`
D. `import beans.WirlessPlan.*;`
E. `import beans.plans.*;`
F. The code compile without any change.
G. None of the above

**Correct Answer:** CE
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 223**
**Given the following contents of two java source files:**

```
package commons.text;
```

```
public class StringUtils  {
   static String capitalize(String msg){
      // return code here
   }
}
```

and

```
package ui;

//Line 1
class TestClass {
    public static void main(String[] args) {
        String output = StringUtils.capitalize("jason donovan");
        System.out.println(output);
    }
}
```

**What changes, when made independently, will enable the code to compile and run?**

A. The code compile without any change.
B. Replace `StringUtils.capitalize("jason donovan")` to `commons.text.StringUtils.capitalize("jason donovan")`
C. Add the following import on Line 1: `import commons.text;`
D. Add the following import on Line 1: `import commons.text.StringUtils;`
E. Add the following import on Line 1: `import commons.text.*;`
F. Add the following import on Line 1: `import StringUtils;`
G. None of the above

**Correct Answer:** G
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
```
capitalize(String) is not public in StringUtils; cannot be accessed from
outside package
String output = StringUtils.capitalize("jason donovan");
```

**QUESTION 224**
**Consider the following two classes defined in two java source files.**

```
//in file /root/com/foo/X.java
package com.foo;

public class X {
  public static int LOGICID = 10;
  public void apply(int i){
    System.out.println("applied");
  }
}


//in file /root/com/bar/Y.java
package com.bar;

//1 <== INSERT STATEMENT(s) HERE

public class Y{
    public static void main(String[] args){
       X x = new X();
```

```
        x.apply(LOGICID);
    }
}
```

**What should be inserted at //1 so that Y.java can compile without any error?**

A. `import static X;`
B. `import static com.foo.*;`
C. `import static com.foo.X.*;`
D. `import com.foo.*;`
E. `import com.foo.X.LOGICID;`
F. The code compile without any change.
G. None of the above

**Correct Answer:** CD
**Section: Java Basics**
**Explanation**

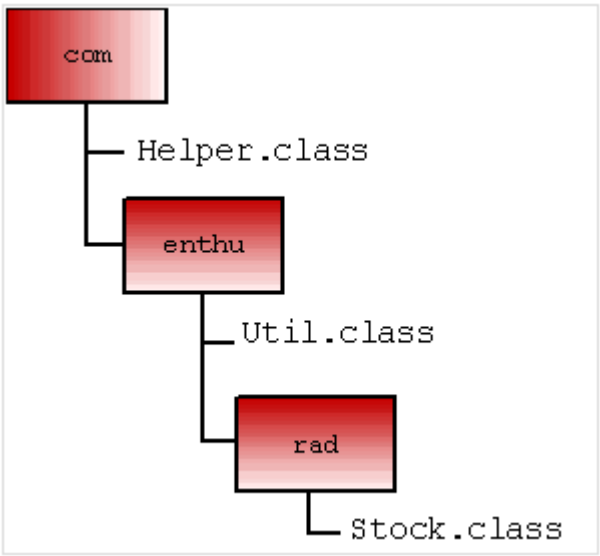
**Explanation/Reference:**

**QUESTION 225**
**Consider the directory structure shown in Image 1 that displays available folders and classes and the code given below:**

```
class StockQuote{

    public static void main(String[] args){
        Stock stock = new Stock();
        System.out.println(Util.capitalize(stock.getCustomer()));
        System.out.println(Helper.getLocaleTime());
    }
}
```

**Assuming that the code uses valid method calls, what statements MUST be added to the above class?**




A. `package com.enthu.rad.*;`
B. `import com.enthu.*;`

C. `package com.enthu.rad;`
D. `import com.*;`
E. `import java.io.*;`
F. It is not required to `import java.io.*` or import `java.io.IOException` because `java.io` package is imported automatically.

**Correct Answer:** BCD
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 226**
**Consider the following two classes defined in two .java files.**

```
//in file /root/com/foo/X.java
package com.foo;

public class X{
  public static int LOGICID = 10;
}


//in file /root/com/bar/Y.java
package com.bar;

//1  <== INSERT STATEMENT(s) HERE

public class Y{
    public static void main(String[] args){
       System.out.println(X.LOGICID);
    }
}
```

**What should be inserted at //1 so that Y.java can compile without any error?**

A. `import static X;`
B. `import static com.foo.*;`
C. `import static com.foo.X.LOGICID;`
D. `import com.foo.*;`
E. `import com.foo.X.LOGICID;`
F. The code compile without any change.
G. None of the above

**Correct Answer:** D
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 227**
**Given:**

```
package entities;

class Student {
    public int getAge() {
       //return code goes here
    }
    public String getName() {
```

```
      //return code goes here
    }
}
```

**And:**

```
package ui;

//line n1

public class Application {
    public static void main(String[] args) {
       Student st = new Student();
       System.out.println(st.getAge());
       System.out.println(st.getName());
    }
}
```

**Which code fragments, when inserted at line n1 independently, enables the code to compile successfully?**

A. `import entities.Student;`
B. `import entities;`
C. `import entities.*;`
D. `import entities.Student.*;`
E. `None of the above`
F. `The code compile without any change`

**Correct Answer:** E
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
```
Application.java:4: error: Student is not public in entities; cannot be
accessed from outside package
import entities.Student;
```

**QUESTION 228**
**Given the following contents of two java source files:**

```
package util.log4j;

public class Logger  {
   public void log(String msg){
      System.out.println(msg);
   }
}

and

package util;

public class TestClass {
    public static void main(String[] args) {
        Logger logger = new Logger();
        logger.log("hello");
    }
}
```

**What changes, when made independently, will enable the code to compile and run?**

A. Replace `Logger logger = new Logger();` with:
`log4j.Logger logger = new log4j.Logger();`

B. Replace `package util.log4j;` with
   `package util;`
C. Replace `Logger logger = new Logger();` with:
   `util.log4j.Logger logger = new util.log4j.Logger();`
D. Remove `package util.log4j;` from Logger.
E. Add `import log4j;` to TestClass.
F. The code compile without any change.
G. None of the above

**Correct Answer:** BC
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 229**
**Consider the following two java files:**

```
//in file SM.java
package x.y;

public class SM{
    public static void foo(){ };
}

//in file TestClass.java
//insert import statement here //1
public class TestClass{
   public static void main(String[] args){
       foo();
   }
}
```

**What should be inserted at //1 so that TestClass will compile and run?**

A. `import static x.y.*;`
B. `import static x.y.SM;`
C. `import static x.y.SM.foo;`
D. `import static x.y.SM.foo();`
E. `import static x.y.SM.*;`
F. The code compile without any change.
G. None of the above

**Correct Answer:** CE
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

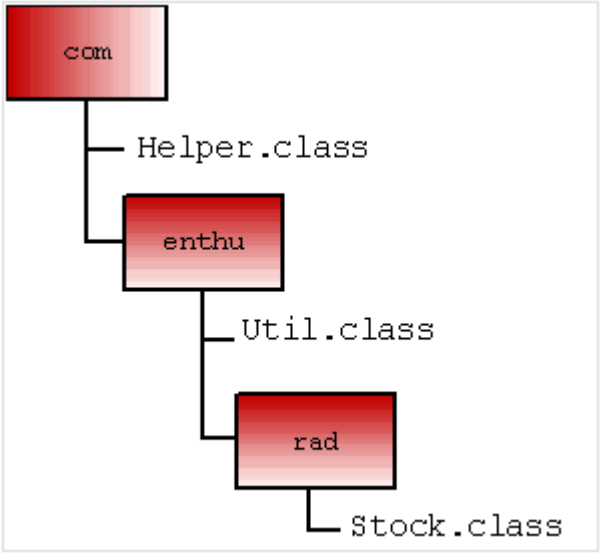
**QUESTION 230**
**Consider the directory structure shown in Image 1 that displays available folders and classes and the code given below:**

```
class StockQuote{

    public static void main(String[] args){
        Stock stock = new Stock();
        System.out.println(Helper.getLocaleTime());
    }
```

```
}
```

**Assuming that the code uses valid method calls, what statements MUST be added to the above class?**




A. `import com.enthu.*;`
B. `import com.*.*;`
C. `import *.*.*;`
D. `import com.*;`
E. `import com.enthu.rad.*;`
F. `import all;`

**Correct Answer:** DE
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 231**
**Which of the following is not a primitive data value in Java?**

A. `"x"`
B. `'x'`
C. `10.2F`
D. `Object`
E. `false`

**Correct Answer:** AD
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
Java has only the following primitive data types:
boolean, byte, short, char, int, long, float and double.

**QUESTION 232**
**What will the following program print?**

```
public class TestClass{
  public static void main(String[] args){
    unsigned byte b = 0;
    b--;
    System.out.println(b);
  }
}
```

A. `0`
B. `-1`
C. `255`
D. `-128`
E. `It will not compile.`

**Correct Answer:** E
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
There no unsigned keyword in java! A char can be used as an unsigned integer.

**QUESTION 233**
**What happens when you try to compile and run the following program?**

```
public class CastTest{
   public static void main(String args[ ] ){
      byte b = -128;
      int i = b ;
      b = (byte) i;
      System.out.println(i + " " + b);
   }
}
```

A. The compiler will refuse to compile it because i and b are of different types cannot be assigned to each other.
B. The program will compile and will print -128 and -128 when run .
C. The compiler will refuse to compile it because -128 is outside the legal range of values for a byte.
D. The program will compile and will print 128 and -128 when run .
E. The program will compile and will print 255 and -128 when run .

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
byte and int both hold signed values. So when b is assigned to i, the sign is preserved.

**QUESTION 234**
**Given the following code:**

```
public class TestClass {
     public static void main(String[] args) {
         //INSERT CODE HERE
         System.out.println(x);
     }
}
```

**What can be inserted in the above code so that it will compile and run without any problem?**

A. `double x = 0xb10_000;`

B. `float x = 0b10_000;`
C. `float x = 0b20_000;`
D. `float x = 0b10_000f;`
E. `long x = 0b10000L;`
F. `double d = 0b10_000D;`

**Correct Answer:** ABE
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
The real exam contains a few questions that test you on how to write numbers in binary. You might want to go through Section 3.10.1 and 3.10.2 of Java Language Specification to understand how this works.

`double x = 0xb10_000;`
0x implies the following digits must be interpreted as Hexadecimal digits and b is a valid Hexadecimal digit.

`float x = 0b10_000;`
A number starting with 0b (or 0B) implies that it is written in binary.
Since 10000 can fit into a float, an explicit cast is not required.
Note that when you specify the bit pattern using binary or hex, an explicit cast is not required even if the number specified using the bit pattern is larger than what a float can hold.

`float x = 0b20_000;`
Since it starts with 0b, that means you are writing the number in binary digits (i.e. 0 or 1). But 2 is not a valid binary digit.

`float x = 0b10_000f;`
This is invalid because the floating point suffices f, F, d, and D are used only when using decimal system and not while using binary. However, since f is a valid digit in hexadecimal system, a hex number may end with an f although it will not be interpreted as float but as the digit f. Thus, float x = 0x10_000f; and float x = 10_000f; are valid because they are written in hex and decimal respectively but float x = 0b10_000f; is invalid because is written in binary. Note that a floating point number cannot be written in Octal. Therefore, float x = 010_000f; is valid but it is not octal even though it starts with a 0. It is interpreted in decimal.

`long x = 0b10000L;`

`double d = 0b10_000D;`
A floating point number written in binary or hex cannot use any suffix for float. But a floating point number written in decimal can use the floating point suffices f, F, d, and D.
Thus, float dx = 0xff; is valid but the f here is not for indicating that it is a float but is interpreted as the hex digit F.

**QUESTION 235**
**Given the following code:**

```
1. int i1 = 1, i2 = 2, i3 = 3;
2. int i4 = i1 + (i2=i3 );
3. System.out.println(i4);
```

**What is the result?**

A. Compilation fails on line 1
B. Compilation fails on line 2
C. 4
D. 3

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 236**
**Given the following code:**

```
class Main {
  public static void main(String[] args) {
    float f1 = 1f;
    float f2 = 0010f;
    float f3 = 0x001f;
    float f4 = 0x001;
    int i1 = 010;

    System.out.println(f1);
    System.out.println(f2);
    System.out.println(f3);
    System.out.println(f4);
    System.out.println(i1);
  }
}
```

**What is the result?**

A. Compilation error
B.
```
1.0
10.0
31.0
1.0
10
```
C.
```
1.0
10.0
31.0
1.0
8
```
D.
```
1.0
10.0
1.0
1.0
8
```
E.
```
1.0
10.0
1.0
1.0
8
10
```

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 237**
**Given this code in a file Application.java:**

```
public class Helper {
    public static String dummy(String text) {
       return text;
    }
}

class Application {
    public static void main(String[] args) {
```

```
        System.out.print("Hello World! " + Helper.dummy(args[1]));
    }
}
```

**And the commands:**

```
javac Application.java
java Application Java Duke
```

**What is the result?**

A. `Hello World! Duke`
B. `Hello World! Java`
C. `An exception is thrown at runtime.`
D. `The program fails to execute due to a runtime error.`
E. `The code does not compile`

**Correct Answer:** E
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
```
Application.java:1: error: class Helper is public, should be declared in a file
named Helper.java
public class Helper {
```

**QUESTION 238**
**Given the following program, which statement is true?**

```
class SomeClass{
     public static void main( String args[ ] ){
         if (args.length == 0 ){
              System.out.println("no arguments") ;
         }
         else{
              System.out.println( args.length + " arguments") ;
         }
      }
}
```

A. The program will fail to compile.
B. The program will throw a NullPointerException when run with zero arguments.
C. The program will print no arguments when called with zero arguments and 1 arguments when called with one argument.
D. The program will print no arguments and 2 arguments when called with zero and one arguments.
E. The program will print no arguments and 3 arguments when called with zero and one arguments.

**Correct Answer:** C
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 239**
**Consider the following class:**

```
public class ArgsPrinter {
    public static void main(String args){
        for(int i=0; i<3; i++){
            System.out.print(args+" ");
```

```
        }
    }
}
```

**What will be printed when the above class is run using the following command line:**
`java ArgsPrinter 1 2 3 4`

A. `1 2 3`
B. `ArgsPrinter 1 2`
C. `java ArgsPrinter 1 2`
D. `1 1 1`
E. `None of these.`
F. `Compilation fails`

**Correct Answer:** E
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
To run a class from the command line, you need a main(String[] ) method that takes an array of Strings array not just a String. Therefore, an exception will be thrown at runtime saying no main(String[] ) method found. Note that String[] and String... are equivalent and so parameter type of String... is also valid for main method. When you use String... the compiler allows you to pass any number of String arguments to that method but internally, compiler converts String... to String[]. It also wraps the arguments into a String[] and invokes the String[] method. The JVM has no idea about String.... It sees only String[].

**QUESTION 240**
**What does the zeroth element of the string array passed to the standard main method contain?**

A. The name of the class.
B. The string "java".
C. The number of arguments.
D. The first argument of the argument list, if present.
E. None of the above.

**Correct Answer:** D
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
Note that if no argument is passed to the program, the args parameter is NOT null but a non-null array of Strings of length zero.

**QUESTION 241**
**Given:**

```
public class GameOfThrones {
    static private String args[] = {"Valar", "Morgulhis", "!"};
    static public void main(String[] args) {
       System.out.println(args[0] + " " + args[1] + " " + args[2]);
    }
}
```

**And the commands:**

```
javac GameOfThrones.java
java GameOfThrones Winter is coming
```

**What is the result?**

A. `Valar Morgulhis !`

B. `Winter is coming !`
C. `Compilation fails.`
D. `An exception is thrown at runtime.`
E. `Winter is coming`

**Correct Answer:** E
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 242**
**Given this code in a file Traveler.java:**

```
public class Traveler {
    public static void main(String[] args) {
       Tours.printf(args);
    }
}

public class Tours {
    public static void printf(String[] args) {
       System.out.print("Happy Journey! " + args[1]);
    }
}
```

**And the commands:**

```
javac Traveler.java
java Traveler Java Duke
```

**What is the result?**

A. `Happy Journey! Duke`
B. `Happy Journey! Java`
C. `An exception is thrown at runtime.`
D. `The program fails to execute due to a runtime error.`
E. `A compilation error occurs`

**Correct Answer:** E
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 243**
**Given the following contents of two java source files:**

```
class Logger  {
   public void log(String msg){
      System.out.println(msg);
   }
}

and

package util;

public class TestClass {
    public static void main(String[] args) {
```

```
        Logger logger = new Logger();
        logger.log("hello");
    }
}
```

**Which statements are true?**

A. `The code prints: hello`
B. A compilation error occurs
C. An exception is thrown at runtime
D. The code compiles correctly and prints hello if we remove `package util;` from TestClass
E. Nothing is printed
F. The code compiles correctly if we add `package util;` to Logger class.
G. None of the above

**Correct Answer:** BDF
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 244**
**When is the Object created at line //1 eligible for garbage collection?**

```
public class TestClass {
   public Object getObject(){
       Object obj = new String("aaaaa");   //1
       Object objArr[] = new Object[1];    //2
       objArr[0] = obj;                    //3
       obj = null;                         //4
       objArr[0] = null;                   //5
       return obj;                         //6
    }
}
```

A. `Just after line 2.`
B. `Just after line 3.`
C. `Just after line 4.`
D. `Just after line 5.`
E. `Just after line 6.`

**Correct Answer:** D
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 245**
**What will be the output of the following program?**

```
public class EqualTest {
     public static void main(String args[]) {
         Integer i = new Integer(1);
         Long m = new Long(1);
         if (i.equals(m)) System.out.println("equal");
         else System.out.println("not equal");
      }
}
```

A. `equal`
B. `not equal`
C. `Compilation error`
D. `An exception is thrown`

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
Signature of equals method is : boolean equals(Object o); So it can take any object. The equals methods of all wrapper classes first check if the two object are of same class or not. If not, they immediately return false. Hence it will print not equal.

**QUESTION 246**
**Which of the following declarations are valid?**

A. `float f1 = 5.5;`
B. `float f2 = 43e10;`
C. `float f3 = -1;`
D. `float f4 = 0x0123;`
E. `float f5 = 4;`
F. `int i1 = 2e5;`
G. `int i2 = 0xDeadCafeL;`
H. `double d2 = 0b000101D;`
I. `long l1 = 0xDeadCafeL;`

**Correct Answer:** CDEI
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
```
Error: incompatible types: possible lossy conversion from double to float
    float f2 = 43e10;

Error: incompatible types: possible lossy conversion from double to int
    int i1 = 2e5;

error: incompatible types: possible lossy conversion from long to int
    int i2 = 0xDeadCafeL;
```

**QUESTION 247**
**What will the following code print when run?**

```
public class TestClass{
    public static Integer wiggler(Integer x){
       Integer y = x + 10;
       x++;
       System.out.println(x);
       return y;
    }

    public static void main(String[] args){
       Integer dataWrapper = new Integer(5);
       Integer value = wiggler(dataWrapper);
       System.out.println(dataWrapper+value);
    }
}
```

A. `5 and 20`

B. 6 and 515
C. 6 and 20
D. 6 and 615
E. It will not compile.
F. An exception is thrown

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
1. Wrapper objects are always immutable. Therefore, when dataWrapper is passed into wiggler() method, it is never changed even when x++; is executed. However, x, which was pointing to the same object as dataWrapper, is assigned a new Integer object (different from dataWrapper) containing 6.

2. If both the operands of the + operator are numeric, it adds the two operands. Here, the two operands are Integer 5 and Integer 15, so it unboxes them, adds them, and prints 20.

**QUESTION 248**
**Given**:

```
public class ScopeTest {
   int z;
   public static void main(String[] args){
      ScopeTest myScope = new ScopeTest();
      int z = 6;
      System.out.println(z);
      myScope.doStuff();
      System.out.println(z);
      System.out.println(myScope.z);
   }
   void doStuff() {
      int z = 5;
      doStuff2();
      System.out.println(z);
   }
   void doStuff2() {
      z=4;
   }
}
```

**What is the result?**

A. 6 5 6 4
B. 6 5 5 4
C. 6 5 6 6
D. 6 5 6 5

**Correct Answer:** A
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 249**
**Given**:

```
public class Batman {
  int squares = 81;
```

```
  public static void main(String[] args) {
     new Batman().go();
  }
  void go() {
     incr(++squares);
     System.out.println(squares);
  }
  void incr(int squares) {
     squares += 10;
  }
}
```

**What is the result?**

A. 81
B. 82
C. 91
D. 92
E. Compilation fails.
F. An exception is thrown at runtime.

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 250**
**Given**:

```
06. public class Employee {
07.    Integer salary = 128;
08.    Boolean active = true;
09.    public static void main(String[] args) {
10.        Employee e1 = new Employee();
11.        Employee e2 = new Employee();
12.        Employee e3 = new Employee();
13.        e3 = e1; e1 = e2; e2 = e1; e1 = e3; e3 = null;
14.    }
15. }
```

**When line 14 is reached, how many objects are eligible for the garbage collector?**

A. 0
B. 1
C. 2
D. 3
E. 4
F. 6
G. 7
H. 8
I. 9

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 251**
**Given the code fragment:**

```
public class MyClass{
   public static int a1;
   public int a2;
   public MyClass (int v) {
     a1++;
     a2 = v;
   }
   public static void main(String[] args) {
     MyClass a1 = new MyClass(10);
     MyClass a2 = new MyClass(20);
     a1.a1 = 5;
     a2.a1 = 10;
     a1.a2 = 20;
     a2.a2 = 20;
     MyClass.a1++;
     System.out.println(a1.a1 + " " + a1.a2 + " " + a2.a1 + " " + a2.a2 + " " +
MyClass.a1);
   }
}
```

**What is the result?**

A. `Compilation fails`
B. `An exception is thrown at runtime`
C. `5 20 10 20`
D. `11 20 11 20 11`
E. `10 20 10 20 10`

**Correct Answer:** D
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 252**
**Given the following code fragment**:

```
3. class Dozens {
4.     int[] dz = {1,2,3,4,5,6,7,8,9,10,11,12};
5. }
6. public class Eggs {
7.     public static void main(String[] args) {
8.     Dozens [] da = new Dozens[3];
9.     da[0] = new Dozens();
10.    Dozens d = new Dozens();
11.    da[1] = d;
12.    d = null;
13.    da[1] = null;
14.    // do stuff
15. }
16. }
```

**Which two are true about the objects created within main(), and eligible for garbage collection when line 14 is reached?**

A. Three objects were created
B. Four objects were created
C. Five objects were created

D. Zero objects are eligible for GC
E. One object is eligible for GC
F. Two objects are eligible for GC
G. Three objects are eligible for GC

**Correct Answer:** CF
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 253**
**Given the code fragment:**

```
3. public class Ebb {
4.    static int x = 7;
5.    public static void main(String[] args) {
6.      String s = "";
7.      for(int y = 0; y < 3; y++) {
8.          x++;
9.          switch(x) {
10.            case 8: s += "8 ";
11.            case 9: s += "9 ";
12.            case 10: { s+= "10 "; break; }
13.            default: s += "d ";
14.            case 13: s+= "13 ";
15.         }
16.      }
17.      System.out.println(s);
18.   }
19.   static { x++; }
20. }
```

**What is the result?**

A. `9 10 d`
B. `8 9 10 d`
C. `9 10 10 d`
D. `9 10 10 d 13`
E. `8 9 10 10 d 13`
F. `8 9 10 9 10 10 d 13`
G. `Compilation fails`

**Correct Answer:** D
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 254**
**Given the code fragment:**

```
public static void main(String[] args) {
    int x = 353;
    int j = x++;
    switch (j) {
       case 317:
       case 353:
       case 367:
          System.out.println("Is a prime number");
```

```
       case 353:
       case 363:
          System.out.println("Is a palindrome");
          break;
       default:
          System.out.println("Invalid value");
     }
}
```

**What is the result?**

A. `Compilation fails`
B. `Is a prime number`
C. `Is a palindrome`
D. `Is a prime number`
   `Is a palindrome`
E. `Invalid value`

**Correct Answer:** A
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 255**
**Given the following code fragment**:

```
class X {
    public void method(String s1) {
        s1 = "acting";
    }
    public static void main(String[] args) {
        String s2 = "action";
        method(s2);
        System.out.println(s2);
    }
}
```

**What is the result?**

A. `acting`
B. `action`
C. `Compilation fails`
D. `An exception is thrown at runtime`

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
```
error: non-static method method(String) cannot be referenced from a static
context
        method(s2);
```

**QUESTION 256**
**Given:**

```
int i, j = 0;
i = (3 * 2 + 4 + 5) ;
j = (3 * ((2 + 4) + 5));
System.out.println("i: " + i + "\nj: " + j);
```

**What is the result?**

A.
```
i: 16
j: 33
```
B.
```
i: 15
j: 33
```
C.
```
i: 33
j: 23
```
D.
```
i: 15
j: 23
```
E. `Compilation fails`

F. `An exception is thrown`

**Correct Answer:** B
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 257**
**Given:**

```
boolean log3 = ( 5.0 != 6.0) && ( 4 != 5);
boolean log4 = (4 != 4) | (4 == 4);
boolean log5 = (0 == -0) & (0 == 0.0);
System.out.println("log3: " + log3 + "\nlog4: " + log4 + "\nlog5: " + log5);
```

**What is the result?**

A.
```
log3: false
log4: true
log5: true
```
B.
```
log3: true
log4: true
log5: true
```
C.
```
log3: true
log4: true
log5: false
```
D.
```
log3: false
log4: false
log5: false
```
E. `Compilation fails`

F. `An exception is thrown`

**Correct Answer:** B
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 258**
**Given:**

```
public class ArithmeticResultsOutput {
   public static void main(String[] args) {
      int i, j = 0;
      if (i++ == ++j) {
        System.out.println("True: i=" + i + ", j=" + j);
```

```
      }
      else {
        System.out.println("False: i=" + i + ", j=" + j);
      }
   }
}
```

**What will be printed to standard out?**

A. `True: i=0, j=1`
B. `True: i=1, j=1`
C. `False: i=0, j=1`
D. `False: i=1, j=1`
E. `Compilation fails`
F. `An exception is thrown`

**Correct Answer:** E
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**
```
error: variable i might not have been initialized
      if (i++ == ++j) {
```

**QUESTION 259**
**Consider the following program:**

```
class Increment {
    public static void main(String []args) {
      Integer i = 127;
      Integer j = 128;
      Integer k = ++i;
      System.out.println("k == j is " + (k == j));
      System.out.println("k.equals(j) is " + k.equals(j));
   }
}
```

**Which one of the following options correctly describes the behavior of this program?**

A. `When executed, this program prints`
   `k == j is false`
   `k.equals(j) is false`
B. `When executed, this program prints`
   `k == j is true`
   `k.equals(j) is false`
C. `When executed, this program prints`
   `k == j is false`
   `k.equals(j) is true`
D. `When executed, this program prints`
   `k == j is true`
   `k.equals(j) is true`
E. Compilation error

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 260**
**Given the code fragment:**

```
System.out.println("Result: " + 2 + 3 + 5);
System.out.println("Result: " + 2 + 3 * 5);
```

**What is the result?**

A.
```
Result: 10
Result: 30
```
B.
```
Result: 10
Result: 25
```
C.
```
Result: 235
Result: 215
```
D.
```
Result: 215
Result: 215
```
E. `Compilation fails`

**Correct Answer:** C
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 261**
**Given:**

```
class A { }
class B { }

interface X { }
interface Y { }
```

**Which two definitions of class C are valid?**

A. `class C extends A implements X { }`
B. `class C implements Y extends B { }`
C. `class C extends A, B { }`
D. `class C implements X, Y extends B { }`
E. `class C extends B implements X, Y { }`

**Correct Answer:** AE
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 262**
**Given:**

```
abstract class X {
    public abstract void methodX();
}

interface Y {
    public void methodY();
}
```

**Which two code fragments are valid?**

A.
```
class Z extends X implements Y {
   public void methodZ() { }
}
```
B.
```
abstract class Z extends X implements Y {
   public void methodZ() { }
}
```
C.
```
class Z extends X implements Y {
   public void methodX() { }
}
```
D.
```
abstract class Z extends X implements Y {
}
```
E.
```
class Z extends X implements Y {
   public void methodY() { }
}
```

**Correct Answer:** BD
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 263**
**Given:**

```
interface Readable {
   public void readBook();
   public void setBookMark();
}

abstract class Book implements Readable {   // line n1

    public void readBook() {

    }
    // line n2
}

class EBook extends Book {    // line n3

     public void readBook() {

     }
     // line n4
}
```

**Which options enables the code to compile? Select 2 options**

A. Replace the code fragment at line n1 with:

```
class Book implements Readable {
```

B. At line n2 insert:

```
public abstract void setBookMark();
```

C. Replace the code fragment at line n3 with:

```
abstract class EBook extends Book {
```

D. At line n4 insert:

```
public void setBookMark() { }
```

**Correct Answer:** CD
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 264**
**Given:**

```
public abstract class Shape {

   private int x;
   private int y;

   public abstract void draw();

   public void setAnchor(int x, int y) {
      this.x = x;
      this.y = y;
   }

}
```

**Which two classes use the Shape class correctly?**

A.
```
public class Circle implements Shape {
    private int radius;
}
```
B.
```
public abstract class Circle extends Shape {
    private int radius;
}
```
C.
```
public class Circle extends Shape {
    private int radius;
    public void draw();
}
```
D.
```
public abstract class Circle implements Shape {
    private int radius;
    public void draw();
}
```
E.
```
public class Circle extends Shape {
    private int radius;
    public void draw() {
       /* code here */
    }
}
```
F.
```
public abstract class Circle implements Shape {
    private int radius;
    public void draw() {
       /* code here */
    }
}
```

**Correct Answer:** BE
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 265**

**Which one of these is a proper definition of a class Car that cannot be sub-classed?**

A. `class Car { }`
B. `abstract class Car { }`
C. `native class Car { }`
D. `static class Car { }`
E. `final class Car { }`

**Correct Answer:** E
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
A class can be extended unless it is declared final.
An inner class can be declared static and still be extended. Notice the distinction. For classes, final means it cannot be extended, while for methods, final means it cannot be overridden in a subclass.
The native keyword can only be used on methods, not on classes and instance variables.

**QUESTION 266**
**Given:**

```
public interface A { public void m1(); }
class B implements A { }
class C implements A { public void m1() { } }
class D implements A { public void m1(int x) { } }
abstract class E implements A { }
abstract class F implements A { public void m1() { } }
abstract class G implements A { public void m1(int x) { } }
```

**What is the result?**

A. Compilation succeeds.
B. Exactly one class does do NOT compile.
C. Exactly two classes do NOT compile.
D. Exactly four classes do NOT compile.
E. Exactly three classes do NOT compile.

**Correct Answer:** C
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 267**
**Which of the following statements are correct? Select 3 options**

A. An abstract class can be extended by an abstract or a concrete class.
B. A concrete class can be extended by an abstract or a concrete class.
C. An interface can be extended by another interface.
D. An interface can be extended by an abstract class.
E. An interface can be extended by a concrete class.
F. An abstract class cannot implement an interface.

**Correct Answer:** ABC
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
A class "implements" an interface. It does not "extend" an interface.

**QUESTION 268**
**Which of the following statements are true? Select 2 options**

A. Private methods cannot be overridden in subclasses.
B. A subclass can override any method in a non-final superclass.
C. An overriding method can declare that it throws a wider spectrum of checked exceptions than the method it is overriding.
D. The parameter list of an overriding method must be a subset of the parameter list of the method that it is overriding.
E. The overriding method may opt not to declare any throws clause even if the original method has a throws clause.

**Correct Answer:** AE
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
- Only methods that are inherited can be overridden and private methods are not inherited.
- Only the methods that are not declared to be final can be overridden. Further, private methods are not inherited so they cannot be overridden either.
- An overriding method (the method that is trying to override the base class's method) must have the same parameters.
- No exception (i.e. an empty set of exceptions) is a valid subset of the set of exceptions thrown by the original method so an overriding method can choose to not have any throws clause.

1. A method can be overridden by defining a method with the same signature(i.e. name and parameter list) and return type as the method in a superclass. The return type can also be a subclass of the original method's return type.
2. Only methods that are accessible can be overridden. A private method cannot, therefore, be overridden in subclasses, but the subclasses are allowed to define a new method with exactly the same signature.
3. A final method cannot be overridden.
4. An overriding method cannot exhibit behavior that contradicts the declaration of the original method. An overriding method therefore cannot return a different type (except a subtype) or throw a wider spectrum of exceptions than the original method in the superclass. This, of course, applies only to checked exceptions because unchecked exceptions are not required to be declared at all.
5. A subclass may have a static method with the same signature as a static method in the base class but it is not called overriding. It is called hiding because the concept of polymorphism doesn't apply to static members.

**QUESTION 269**
**Which of the following statements are true? Select 2 options**

A. The `extends` keyword is used to specify inheritance.
B. subclass of a non-abstract class cannot be declared `abstract`.
C. subclass of an `abstract` class can be declared `abstract`.
D. subclass of a `final` class cannot be `abstract`.
E. A class, in which all the members are declared `private`, cannot be declared `public`.

**Correct Answer:** AC
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
final class cannot be subclassed

The extends clause is used to specify that a class extends another class and thereby inherits all non-private instance members of that class. A subclass can be declared abstract regardless of whether the superclass was declared abstract. A class cannot be declared abstract and final at the same time. This restriction makes sense because abstract classes need to be subclassed to be useful and final forbids subclasses. The visibility of the class is not limited by the visibility of its members. A class with all the members declared private can still be declared public or a class having all public members may be declared private.

**QUESTION 270**
**Which of these statements concerning interfaces are true? Select 2 options**

A. An interface may extend an interface.
B. An interface may extend a class and may implement an interface.
C. A class can implement an interface and extend a class.
D. A class can extend an interface and can implement a class.
E. An interface can only be implemented and cannot be extended.

**Correct Answer:** AC
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
Unlike a class, an interface can extend from multiple interfaces.
An interface cannot implement another interface. It can extend another interface but not a class.

The keyword implements is used when a class inherits method prototypes from an interface. The keyword extends is used when an interface inherits from another interface, or a class inherits from another class.

**QUESTION 271**
**Given:**

```
interface Fish {
}

class Perch implements Fish {
}

class Walleye extends Perch {
}

class Bluegill {
}

public class Fisherman {
   public static void main(String[] args) {
      Fish f = new Walleye();
      Walleye w = new Walleye();
      Bluegill b = new Bluegill();
      if (f instanceof Perch)
         System.out.print("f-p ");
      if (w instanceof Fish)
         System.out.print("w-f ");
      if (b instanceof Fish)
         System.out.print("b-f ");
   }
```

```
}
```

**What is the result?**

A. `w-f`
B. `f-p w-f`
C. `w-f b-f`
D. `f-p w-f b-f`
E. `Compilation fails.`
F. `An exception is thrown at runtime`

**Correct Answer:** B
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 272**
**Consider the following classes in one file named A.java...**

```
abstract class A {
    protected int m1(){
        return 0;
    }
}

class B extends A {
    int m1(){
        return 1;
    }
}
```

**Which of the following statements are correct...**

A. The code will not compile as you cannot have more than 1 class in 1 file.
B. The code will not compile because class B does not override the method m1() correctly.
C. The code will not compile as A is an abstract class.
D. The code will not compile as A does not have any abstract method.
E. The code will compile fine.

**Correct Answer:** B
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
The concept here is that an overriding method cannot make the overridden method more private.
The access hierarchy in increasing levels of accessibility is:
private->'no modifier'->protected->public ( public is accessible to all and private is accessible to none except itself.)
Here, class B has no modifier for m1() so it is trying to reduce the accessibility of protected to default.
'protected' means the method will be accessible to all the classes in the same package and all the subclasses (even if the subclass is in a different package).
No modifier (which is the default level) means the method will be accessible only to all the classes in the same package. (i.e. not even to the subclass if the subclass is in a different package.)

**QUESTION 273**
**Given:**

```
08. abstract public class Employee {
09.     protected abstract double getSalesAmount();
10.
11.     public double getCommision() {
12.        return getSalesAmount() * 0.15;
13.     }
14. }
15
16. class Sales extends Employee {
17.     // insert method here
18. }
```

**Which two methods, inserted independently at line 17, correctly complete the Sales class? (Choose two.)**

A. `double getSalesAmount() { return 1230.45; }`
B. `public double getSalesAmount() { return 1230.45; }`
C. `private double getSalesAmount() { return 1230.45; }`
D. `protected double getSalesAmount() { return 1230.45; }`

**Correct Answer:** BD
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 274**
**Given:**

```
09. class One {
10.    void foo() { }
11. }
12.
13. class Two extends One {
14.    //insert method here
15. }
```

**Which three methods, inserted individually at line 14, will correctly complete class Two? (Choose three.)**

A. `int foo() { /* more code here */ }`
B. `void foo() { /* more code here */ }`
C. `public void foo() { /* more code here */ }`
D. `private void foo() { /* more code here */ }`
E. `protected void foo() { /* more code here */ }`

**Correct Answer:** BCE
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 275**
**Given:**

```
01. public class Blip {
02.      protected int blipvert(int x) { return 0; }
03. }
```

```
04. class Vert extends Blip {
05.       // insert code here
06. }
```

**Which five methods, inserted independently at line 5, will compile? (Choose five.)**

A. `public int blipvert(int x) { return 0; }`
B. `private int blipvert(int x) { return 0; }`
C. `private int blipvert(long x) { return 0; }`
D. `protected long blipvert(int x) { return 0; }`
E. `protected int blipvert(long x) { return 0; }`
F. `protected long blipvert(long x) { return 0; }`
G. `protected long blipvert(int x, int y) { return 0; }`

**Correct Answer:** ACEFG
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 276**
**Given:**

```
class One {
   public One foo() {
     return this;
   }
}

class Two extends One {
   public One foo() {
     return this;
   }
}

class Three extends Two {
   // insert method here
}
```

**Which two methods, inserted individually, correctly complete the Three class? (Choose two.)**

A. `public void foo() {}`
B. `public int foo() { return 3; }`
C. `public Two foo() { return this; }`
D. `public One foo() { return this; }`
E. `public Object foo() { return this; }`

**Correct Answer:** CD
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 277**
**Given:**

```
01. public class Hi {
02.    void m1() { }
```

```
03.    protected void() m2 { }
04. }
05.
06. class Lois extends Hi {
07.    // insert code here
08. }
```

**Which four code fragments, inserted independently at line 7, will compile? (Choose four.)**

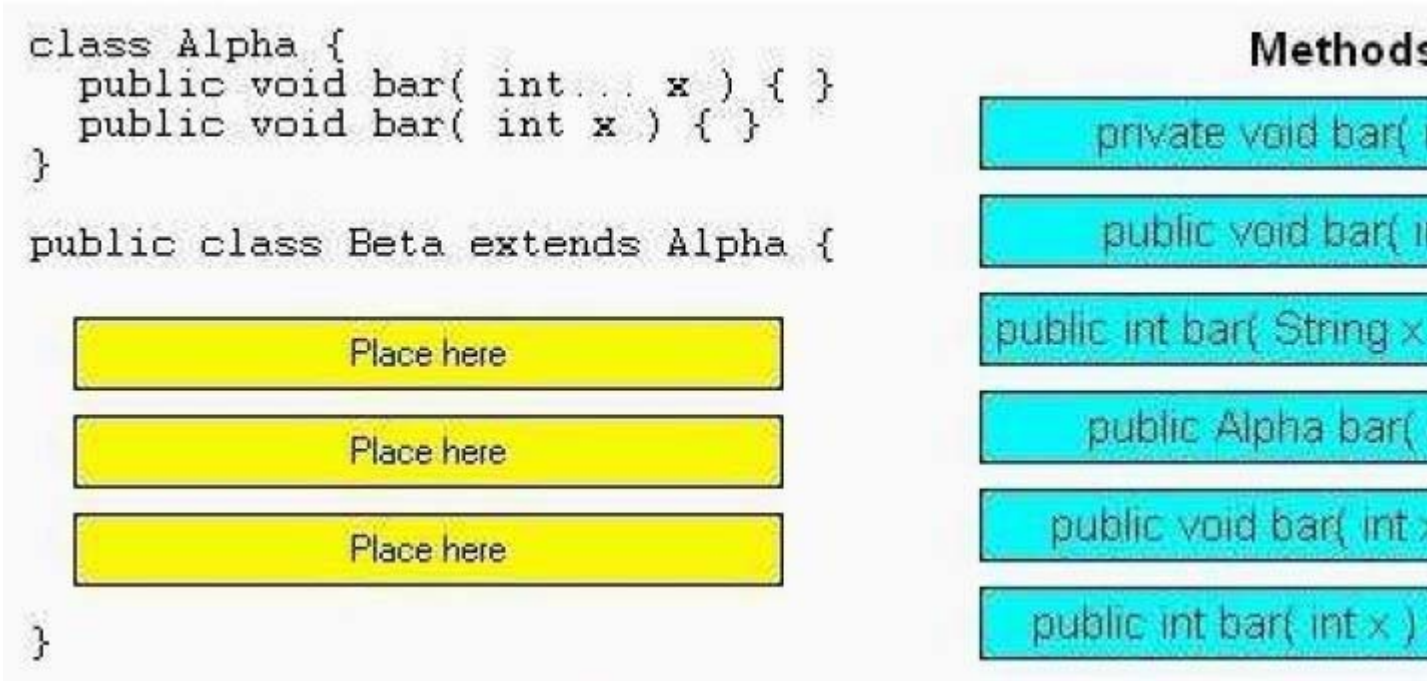
A. `public void m1() { }`
B. `protected void m1() { }`
C. `private void m1() { }`
D. `void m2() { }`
E. `public void m2() { }`
F. `protected void m2() { }`
G. `private void m2() { }`

**Correct Answer:** ABEF
**Section: Working with Inheritance**
**Explanation**

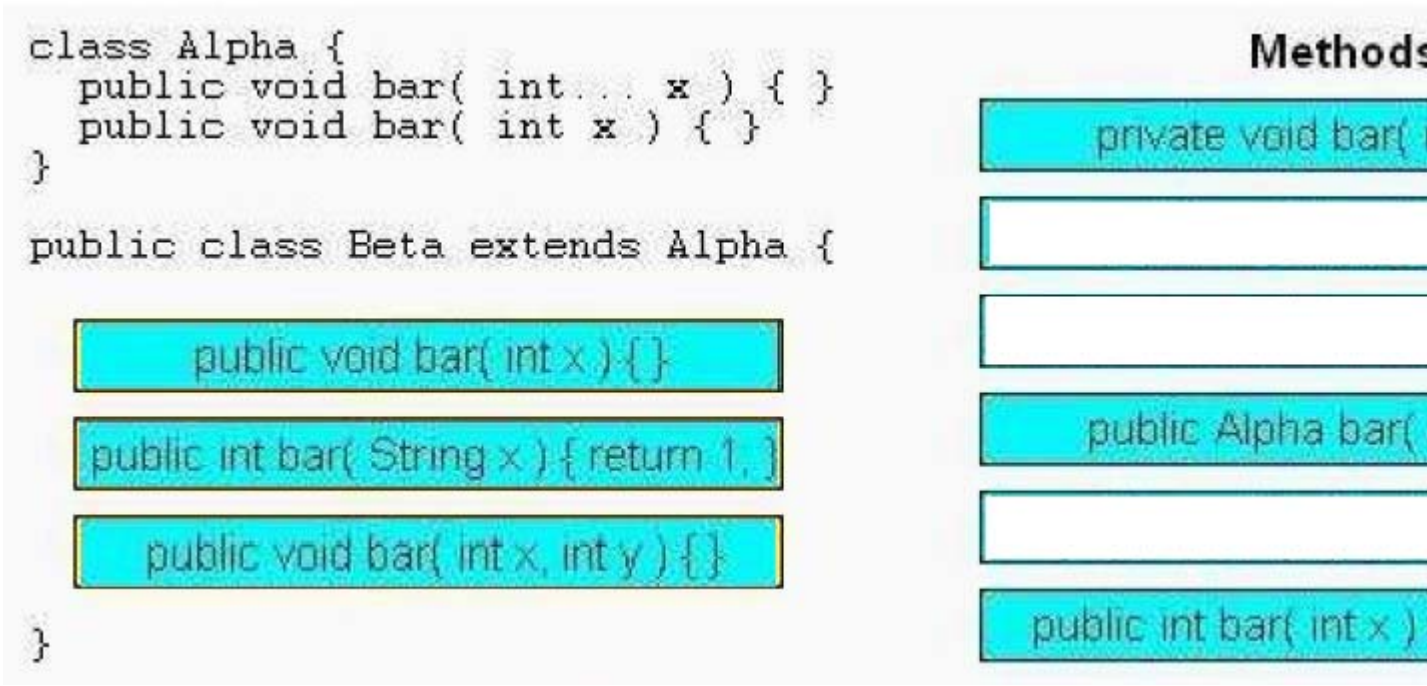
**Explanation/Reference:**

**QUESTION 278**
**Add methods to the Beta class to make it compile correctly.**

**Select and Place:**




**Correct Answer:**




**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 279**
**Given:**

```
class ClassA {}
class ClassB extends ClassA {}
class ClassC extends ClassA {}
```

**and:**

```
ClassA p0 = new ClassA();
ClassB p1 = new ClassB();
ClassC p2 = new ClassC();
ClassA p3 = new ClassB();
ClassA p4 = new ClassC();
```

**Which three are valid? (Choose three.)**

A. `p0 = p1;`
B. `p1 = p2;`
C. `p2 = p4;`
D. `p2 = (ClassC)p1;`
E. `p1 = (ClassB)p3;`
F. `p2 = (ClassC)p4;`

**Correct Answer:** AEF
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 280**

**Given:**

```
05. class Building { }
06. public class Barn extends Building {
07.    public static void main(String[] args) {
08.       Building build1 = new Building();
09.       Barn barn1 = new Barn();
10.       Barn barn2 = (Barn) build1;
11.       Object obj1 = (Object) build1;
12.       String str1 = (String) build1;
13.       Building build2 = (Building) barn1;
14.    }
15. }
```

**Which is true?**

A. If line 10 is removed, the compilation succeeds.
B. If line 11 is removed, the compilation succeeds.
C. If line 12 is removed, the compilation succeeds.
D. If line 13 is removed, the compilation succeeds.
E. More than one line must be removed for compilation to succeed.

**Correct Answer:** C
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 281**
**Given:**

```
class Animal {
  public String noise() {
     return "peep";
  }
}

class Dog extends Animal {
  public String noise() {
     return "bark";
  }
}

class Cat extends Animal {
  public String noise() {
     return "meow";
  }
}

30. Animal animal = new Dog();
31. Cat cat = (Cat)animal;
32. System.out.println(cat.noise());
```

**What is the result?**

A. peep
B. bark
C. meow
D. Compilation fails
E. An exception is thrown at runtime.

**Correct Answer:** E
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 282**
**Given:**

```
10. interface Foo {}
11. class Alpha implements Foo {}
12. class Beta extends Alpha {}
13. class Delta extends Beta {
14.    public static void main( String[] args ) {
15.      Beta x = new Beta();
16.      //insert code here
17.    }
18. }
```

**Which code, inserted at line 16, will cause a java.lang.ClassCastException?**

A. `Alpha a = x;`
B. `Foo f = (Delta)x;`
C. `Foo f = (Alpha)x;`
D. `Beta b = (Beta)(Alpha)x;`

**Correct Answer:** B
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 283**
**Given:**

```
class TestA {
    public void start() {
      System.out.println("TestA");
    }
}

public class TestB extends TestA {
    public void start() {
       System.out.println("TestB");
    }
    public static void main(String[] args) {
       ((TestA)new TestB()).start();
    }
}
```

**What is the result?**

A. `TestA`
B. `TestB`
C. `Compilation fails`
D. `An exception is thrown at runtime.`

**Correct Answer:** B
**Section: Working with Inheritance**

**Explanation**

**Explanation/Reference:**

**QUESTION 284**
**Given:**

```
class Alpha {
    public void foo() {
      System.out.print("Afoo ");
    }
}

public class Beta extends Alpha {
    public void foo() {
       System.out.print("Bfoo ");
    }
    public static void main(String[] args) {
       Alpha a = new Beta();
       Beta b = (Beta)a;
       a.foo();
       b.foo();
    }
}
```

**What is the result?**

A. Afoo Afoo
B. Afoo Bfoo
C. Bfoo Afoo
D. Bfoo Bfoo
E. Compilation fails.
F. An exception is thrown at runtime.

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 285**
**Given:**

```
public class Base {
    public static final String FOO = "foo";
    public static void main(String[] args) {
       Base b = new Base();
       Sub s = new Sub();
       System.out.print(Base.FOO);
       System.out.print(Sub.FOO);
       System.out.print(b.FOO);
       System.out.print(s.FOO);
       System.out.print(((Base) s).FOO);
    }
}

class Sub extends Base {
   public static final String FOO = "bar";
}
```

**What is the result?**

A. `foofoofoofoofoo`
B. `foobarfoobarbar`
C. `foobarfoofoofoo`
D. `foobarfoobarfoo`
E. `barbarbarbarbar`
F. `foofoofoobarbar`
G. `foofoofoobarfoo`

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 286**
**Given:**

```
class A {
   String name = "A";
   String getName() {
     return name;
   }
   String greeting() {
     return "class A";
   }
}

class B extends A {
   String name = "B";
   String greeting() {
      return "class B";
   }
}

public class Client {
   public static void main(String[] args) {
     A a = new A();
     A b = new B();
     System.out.println(a.greeting() + " has name " + a.getName());
     System.out.println(b.greeting() + " has name " + b.getName());
   }
}
```

**Select and Place:**


**Correct Answer:**


**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 287**
**Given:**

```
01. public class GoTest {
02.   public static void main(String[] args) {
03.     Sente a = new Sente(); a.go();
04.     Goban b = new Goban(); b.go();
05.     Stone c = new Stone(); c.go();
06.   }
07. }
08.
09. class Sente implements Go {
10.   public void go(){
11.     System.out.println("go in Sente");
12.   }
13. }
14.
15. class Goban extends Sente {
16.    public void go(){
17.     System.out.println("go in Goban");
18.   }
19.
20. }
21. class Stone extends Goban implements Go {
22. }
23.
24. interface Go { public void go(); }
```

**What is the result?**

A. `go in Goban go in Sente go in Sente`
B. `go in Sente go in Sente go in Goban`
C. `go in Sente go in Goban go in Goban`
D. `go in Goban go in Goban go in Sente`
E. `Compilation fails because of an error in line 17.`

**Correct Answer:** C
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 288**
**Given:**

```
class X {
   public void mX() {
       System.out.println("Xm1");
   }
}

class Y extends X {
   public void mX() {
        System.out.println("Xm2");
   }
   public void mY() {
        System.out.println("Ym");
   }
}

public class Test {
   public static void main(String[] args) {
      X xRef = new Y();
      Y yRef = (Y)xRef;
      yRef.mY();
      xRef.mX();
   }
}
```

**What is the result?**

A. Ym
   Xm2
B. Ym
   Xm1
C. Compilation fails.
D. A ClassCastException is thrown at runtime.

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 289**
**Given:**

```
11. abstract class Vehicle { public int speed() { return 0; }
12. class Car extends Vehicle { public int speed() { return 60; } .... }
13. class RaceCar extends Car { public int speed() { return 150; } ... }

21. RaceCar racer = new RaceCar();
22. Car car = new RaceCar();
23. Vehicle vehicle = new RaceCar();
24. System.out.println(racer.speed() + ", " + car.speed() + ", " +
vehicle.speed());
```

**What is the result?**

A. 0, 0, 0
B. 150, 60, 0
C. Compilation fails
D. 150, 150, 150
E. An exception is thrown at runtime

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 290**
**Given:**

```
class Star {
   public void doStuff() {
      System.out.println("Twinkling Star");
   }
}

interface Universe {
   public void doStuff();
}

class Sun extends Star implements Universe {
   public void doStuff() {
      System.out.println("Shining Sun");
   }
}

public class Bob {
   public static void main(String[] args) {
      Sun obj2 = new Sun();
      Star obj3 = obj2;
      ((Sun)obj3).doStuff();
      ((Star)obj2).doStuff();
      ((Universe)obj2).doStuff();
   }
}
```

**What is the result?**

A.
```
Shining Sun
Shining Sun
Shining Sun
```

B.
```
Shining Sun
Twinkling Star
Shining Sun
```

C. `Compilation fails.`

D. `A ClassCastException is thrown at runtime.`

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 291**
**Given:**

```
21. class Money {
22.     private String country = "Canada";
23.     public String getC() { return country; }
24. }

25. class Yen extends Money {
26.     public String getC() { return super.country; }
```

```
27. }

28. public class Euro extends Money {
29.     public String getC(int x) { return super.getC(); }
30.     public static void main(String[] args) {
31.         System.out.print(new Yen().getC() + " " + new Euro().getC());
32.     }
33. }
```

**What is the result?**

A. Canada
B. null Canada
C. Canada null
D. Canada Canada
E. Compilation fails due to an error on line 26.
F. Compilation fails due to an error on line 29.

**Correct Answer:** E
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 292**
**Given:**

```
class Foo {
    public int a = 3;
    public void addFive() {
       a += 5;
       System.out.print("f ");
    }
}

class Bar extends Foo {
    public int a = 8;
    public void addFive() {
       this.a += 5;
       System.out.print("b " );
    }
}
```

**Invoked with:**

```
Foo f = new Bar();
f.addFive();
System.out.println(f.a);
```

**What is the result?**

A. b 3
B. b 8
C. b 13
D. f 3
E. f 8
F. f 13
G. Compilation fails.
H. An exception is thrown at runtime.

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 293**
**What is the output of class TestCafe4Java?**

```
class SuperCafe4Java {
   public Object get () {
    return ("SuperCafe4Java");
   }
}

class SubCafe4Java extends SuperCafe4Java {
   public String get () {
     return ("SubCafe4Java");
   }
}

class TestCafe4Java {
   public static void main (String[] arguments) {
      SuperCafe4Java superFoo;
      SubCafe4Java subFoo;
      superFoo = new SubCafe4Java();
      System.out.println (superFoo.get());
      subFoo = new SubCafe4Java();
      superFoo = subFoo;
      System.out.println (superFoo.get());
   }
}
```

A. `SubCafe4Java`
   `SubCafe4Java`
B. `SuperCafe4Java`
   `SuperCafe4Java`
C. `SubCafe4Java`
   `SuperCafe4Java`
D. `SuperCafe4Java`
   `SubCafe4Java`
E. `Compilation Error`
F. `An exception is throw at runtime`

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 294**
**What is the output of class TestCafe4Java, if the class SubCafe4Java is compiled as follows (using JDK 1.5)?**

```
javac -source 1.4 SubCafe4Java.java

class SuperCafe4Java {
   public Object get () {
     return ("SuperCafe4Java");
   }
}
```

```
class SubCafe4Java extends SuperCafe4Java {
   public String get () {
      return ("SubCafe4Java");
   }
}

class TestCafe4Java {
   public static void main (String[] arguments) {
      SuperCafe4Java superFoo;
      SubCafe4Java subFoo;
      superFoo = new SubCafe4Java();
      System.out.println (superFoo.get());
      subFoo = new SubCafe4Java();
      superFoo = subFoo;
      System.out.println (superFoo.get());
   }
}
```

A. SubCafe4Java
   SubCafe4Java
B. SuperCafe4Java
   SuperCafe4Java
C. SubCafe4Java
   SuperCafe4Java
D. SuperCafe4Java
   SubCafe4Java
E. Compilation Error
F. Runtime Exception

**Correct Answer:** E
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 295**
**What is the output of class TestCafe4Java?**

```
class SuperCafe4Java {
   public Object get () {
     return ("SuperCafe4Java");
   }
}

class SubCafe4Java extends SuperCafe4Java {
   public String get () {
      return ("SubCafe4Java");
   }
   public Object get () {
      return ("SubCafe4JavaObject");
   }
}

class TestCafe4Java {
   public static void main (String[] arguments) {
      SuperCafe4Java superFoo;
      SubCafe4Java subFoo;
      superFoo = new SubCafe4Java();
      System.out.println (superFoo.get());
      subFoo = new SubCafe4Java();
      superFoo = subFoo;
      System.out.println (superFoo.get());
  }
```

```
}
```

A.
```
SubCafe4Java
SubCafe4Java
```

B.
```
SuperCafe4Java
SuperCafe4Java
```

C.
```
SubCafe4Java
SuperCafe4Java
```

D.
```
SuperCafe4Java
SubCafe4Java
```

E. `Compilation Error`

F. `Runtime Exception`

**Correct Answer:** E
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 296**
**What is the output of class TestCafe4Java?**

```
class SuperCafe4Java {
   public Object get (Object o) {
     return ("SuperCafe4Java");
   }
}

class SubCafe4Java extends SuperCafe4Java {
   public Object get (String o) {
      return ("SubCafe4Java");
   }
}

class TestCafe4Java {
   public static void main (String[] arguments) {
      SuperCafe4Java superFoo;
      SubCafe4Java subFoo;
      superFoo = new SubCafe4Java();
      System.out.println (superFoo.get("super"));
      subFoo = new SubCafe4Java();
      superFoo = subFoo;
      System.out.println (superFoo.get("super"));
  }
}
```

A.
```
SubCafe4Java
SubCafe4Java
```

B.
```
SuperCafe4Java
SuperCafe4Java
```

C.
```
SubCafe4Java
SuperCafe4Java
```

D.
```
SuperCafe4Java
SubCafe4Java
```

E. `Compilation Error`

F. `Runtime Exception`

**Correct Answer:** B
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 297**
**Given:**

```
1. interface DeclareStuff {
2.    public static final int EASY = 3;
3.
4.   void doStuff(int t);
5. }
6.
7. public class TestDeclare implements DeclareStuff {
8.   public static void main(String[] args) {
9.     int x = 5;
10.    new TestDeclare().doStuff(++x);
11.  }
12.
13.  void doStuff(int s) {
14.    s += EASY + ++s;
15.    System.out.println("s " + s);
16.  }
17.}
```

**What is the result?**

A. `s 14`
B. `s 16`
C. `s 10`
D. `Compilation fails.`
E. `An exception is thrown at runtime`

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 298**
**Which statement is true about the following classes and interfaces?**

```
01. public interface A {
02.     public void doSomething(String thing);
03. }

01. public class AImpl implements A {
02.     public void doSomething(String msg) {}
03. }

01. public class B {
02.     public A doit(){
03.        //more code here
04.     }
05.     public String execute(){
06.       //more code here
07.     }
08. }

01. public class C extends B {
02.     public AImpl doit(){
03.        //more code here
```

```
04.     }
05.
06.     public Object execute() {
07.        //more code here
08.     }
09. }
```

A. Compilation will succeed for all classes and interfaces.
B. Compilation of class C will fail because of an error in line 2.
C. Compilation of class C will fail because of an error in line 6.
D. Compilation of class AImpl will fail because of an error in line 2.

**Correct Answer:** C
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 299**
**Given:**

```
interface DoStuff2 {
    float getRange(int low, int high);
}

interface DoMore {
    float getAvg(int a, int b, int c);
}

abstract class DoAbstract implements DoStuff2, DoMore {
}

06. class DoStuff implements DoStuff2 {
07.    public float getRange(int x, int y) {
08.      return 3.14f;
09.    }
10. }
11.
12. interface DoAll extends DoMore {
13.    float getAvg(int a, int b, int c, int d);
14. }
```

**What is the result?**

A. The file will compile without error
B. Compilation fails. Only line 7 contains an error
C. Compilation fails. Only line 12 contains an error.
D. Compilation fails. Only line 13 contains an error.
E. Compilation fails. Only lines 7 and 12 contain errors
F. Compilation fails. Only lines 7 and 13 contain errors.
G. Compilation fails. Lines 7, 12, and 13 contain errors.

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 300**
**Given:**

```
public class X implements Z {
    public String toString() {
       return "X ";
    }
    public static void main(String[] args) {
       Y myY = new Y();
       X myX = myY;
       Z myZ = myX;
       System.out.print(myX);
      System.out.print((Y)myX);
       System.out.print(myZ);
    }
}

class Y extends X {
    public String toString() {
       return "Y ";
    }
}

interface Z { }
```

**What is the result?**

A. X X X
B. X Y X
C. Y Y X
D. Y Y Y
E. Compilation fails.
F. An exception is thrown at runtime.

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 301**
**What would be the result of attempting to compile and run the following code?**

```
public class TestClass{
   public static void main(String args[]) {
      B c = new C();
      System.out.println(c.max(10, 20));
   }
}

class A{
   int max(int x, int y)  { if (x>y) return x; else return y; }
}

class B extends A{
  int max(int x, int y)  {  return 2 * super.max(x, y) ; }
}

class C extends B{
  int max(int x, int y)  {  return super.max( 2*x, 2*y); }
}
```

A. The code will fail to compile.
B. Runtime error.
C. The code will compile without errors and will print 80 when run.
D. The code will compile without errors and will print 40 when run.
E. The code will compile without errors and will print 20 when run.

**Correct Answer:** C
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
When the program is run, the main() method will call the max() method in C with parameters 10 and 20 because the actual object referenced by 'c' is of class C. This method will call the max() method in B with the parameters 20 and 40. The max() method in B will in turn call the max() method in A with the parameters 20 and 40. The max() method in A will return 40 to the max() method in B. The max() method in B will return 80 to the max() method in C. And finally the max () of C will return 80 in main() which will be printed out.

**QUESTION 302**
**Given:**

```
01. interface TestA { String toString(); }
02.
03. public class Test {
04.    public static void main(String[] args) {
05.       System.out.println(new TestA() {
06.          public String toString() { return "test"; }
07.       });
08.    }
09. }
```

**What is the result?**

A. test
B. null
C. An exception is thrown at runtime
D. Compilation fails because of an error in line 1.
E. Compilation fails because of an error in line 5.
F. Compilation fails because of an error in line 6.

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 303**
**Given:**

```
public class SimpleCalc {
  public int value;
  public void calculate() {
     value += 7;
  }
}
```

**and:**

```
public class MultiCalc extends SimpleCalc {
```

```
   public void calculate() {
      value -= 3;
   }
   public void calculate(int multiplier) {
      calculate();
      super.calculate();
      value *= multiplier;
   }
   public static void main(String[] args) {
       MultiCalc calculator = new MultiCalc();
       calculator.calculate(2);
       System.out.println("Value is: " + calculator.value);
   }
}
```

**What is the result?**

A. Value is: 8
B. Compilation fails.
C. Value is: 12
D. Value is: -12
E. The code runs with no output.
F. An exception is thrown at runtime

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 304**
**Given:**

```
10. public class SuperCalc {
11.     protected static int multiply(int a, int b) { return a * b;}
12. }
```

**and:**

```
20. public class SubCalc extends SuperCalc{
21.    public static int multiply(int a, int b) {
22.       int c = super.multiply(a, b);
23.       return c;
24.    }
25. }
```

**and:**

```
30. SubCalc sc = new SubCalc ();
31. System.out.println(sc.multiply(3,4));
32. System.out.println(SubCalc.multiply(2,2));
```

**What is the result?**

A. 12
B. The code runs with no output.
C. An exception is thrown at runtime.
D. Compilation fails because of an error in line 21.
E. Compilation fails because of an error in line 22.
F. Compilation fails because of an error in line 31.

**Correct Answer:** E
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 305**
**Given:**

```
public class ItemTest {
   private final int id;
   public ItemTest(int id) {
     this.id = id;
   }
   public void updateId(int newId) {
      id = newId;
   }
   public static void main(String[] args) {
      ItemTest fa = new ItemTest(42);
      fa.updateId(69);
      System.out.println(fa.id);
   }
}
```

**What is the result?**

A. Compilation fails.
B. An exception is thrown at runtime.
C. The attribute id in the ItemTest object remains unchanged
D. The attribute id in the ItemTest object is modified to the new value.
E. A new ItemTest object is created with the preferred value in the id attribute.

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 306**
**Given:**

```
1. public interface A {
2.     String DEFAULT_GREETING = "Hello World";
3.     public void method1();
4. }
```

**A programmer wants to create an interface called B that has A as its parent. Which interface declaration is correct?**

A. public interface B extends A { }
B. public interface B implements A { }
C. public interface B instanceOf A { }
D. public interface B inheritsFrom A { }

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 307**
**Given:**

```
public class X {

     public static void main(String[] args) {
          String myString = "Good Day!";
          System.out.println(myString.charAt(9));
     }
}
```

**What is the result?**

A. The program prints nothing
B. !
C. java.lang.StringIndexOutOfBoundsException is thrown at runtime.
D. java.lang.ArrayIndexOutOfBoundsException is thrown at runtime.
E. java.lang.NullPointerException is thrown at runtime.

**Correct Answer:** C
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

```
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String
index out of range: 9
    at java.lang.String.charAt(String.java:658)
    at Main.main(Main.java:5)
```

**QUESTION 308**
**Given:**

```
public class X {

     public static void main(String[] args) {
          int[] numbers = {1,2,3,4,5,6,7};
          System.out.println(numbers[7]);
     }
}
```

**What is the result?**

A. The program prints nothing
B. 7
C. java.lang.StringIndexOutOfBoundsException is thrown at runtime.
D. java.lang.ArrayIndexOutOfBoundsException is thrown at runtime.
E. java.lang.NullPointerException is thrown at runtime.

**Correct Answer:** D
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 7
    at Main.main(Main.java:5)
```

**QUESTION 309**
**Given the following code:**

```
public class MyClass {
    static int[] numbers = {1,2,3,4};
    static {
       numbers[4] = 5;
    }
    public static void main(String[] args ) {

    }
}
```

**Which exception or error will be thrown when a programmer attempts to run this code?**

A. java.lang.StackOverflowError
B. java.lang.IllegalStateException
C. java.lang.ExceptionInInitializerError
D. java.lang.ArrayIndexOutOfBoundsException
E. java.lang.NullPointerException
F. Compilation fails

**Correct Answer:** C
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
```
Exception in thread "main" java.lang.ExceptionInInitializerError
Caused by: java.lang.ArrayIndexOutOfBoundsException: 4
    at Main.<clinit>(Main.java:4)
```

**QUESTION 310**
**Given the code fragment:**

```
static public void infected() throws Exception {
   System.out.print("before ");
   try {
      int i = 1/0;
      System.out.print("try ");
   }
   catch(Exception e) {
      System.out.print("catch ");
      throw e;
   }
   finally {
       System.out.print("finally ");
   }
   System.out.print("after ");
}
```

**What is the result when infected() is invoked?**

A. before try catch finally after
B. before catch finally after
C. before catch after
D. before catch finally
E. before catch

**Correct Answer:** D
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 311**
**What will be the output when the following code is compiled and run?**

```
//in file Test.java
class E1 extends Exception { }
class E2 extends E1 { }

class Test {
   public static void main(String[] args) {
      try {
          throw new E2();
      }
      catch(E1 e) {
          System.out.println("E1");
      }
      catch(Exception e) {
          System.out.println("E");
      }
      finally {
          System.out.println("Finally");
      }
   }
}
```

A. It will not compile.
B. It will print E1 and Finally.
C. It will print E1, E and Finally.
D. It will print E and Finally.
E. It will print Finally.

**Correct Answer:** B
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
Since E2 is a sub class of E1, catch(E1 e) will be able to catch exceptions of class E2. Therefore, E1 is printed. Once the exception is caught the rest of the catch blocks at the same level (that is the ones associated with the same try block) are ignored. So E is not printed. finally is always executed (except in case of System.exit()), so Finally is also printed.

**QUESTION 312**
**What will be the output of the following program?**

```
class TestClass {

    public static void main(String[] args) throws Exception {
        try {
           amethod();
           System.out.println("try ");
        }
        catch(Exception e) {
            System.out.print("catch ");
        }
        finally  {
            System.out.print("finally ");
        }
        System.out.print("out ");
     }

     public static void amethod(){ }

}
```

A. try finally
B. try finally out
C. try out
D. catch finally out
E. It will not compile because amethod() does not throw any exception.

**Correct Answer:** B
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
Since the method amethod() does not throw any exception, try is printed and the control goes to finally which prints finally. After that out is printed.

**QUESTION 313**
**Given the following code:**

```
class EHBehavior {
    public static void main(String []args) {
       try {
          int i = 10/0; // LINE A
          System.out.print("after throw -> ");
       }
       catch(ArithmeticException ae) {
          System.out.print("in catch -> ");
          return;
       }
       finally {
          System.out.print("in finally -> ");
       }
       System.out.print("after everything");
   }
}
```

**Which one of the following options best describes the behavior of this program?**

A. The program prints the following: in catch -> in finally -> after everything.
B. The program prints the following: after throw -> in catch -> in finally -> after everything.
C. The program prints the following: in catch.
D. The program prints the following: in catch -> after everything
E. The program prints the following: in catch -> in finally ->.
F. When compiled, the program results in a compiler error in line marked with comment in LINE A for divide-by-zero.

**Correct Answer:** E
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 314**
**What will be the result of attempting to compile and run the following program?**

```
public class TestClass {
    public static void main(String args[]){
        Exception e = null;
        throw e;
    }
```

```
}
```

A. The code will fail to compile.
B. The program will fail to compile, since it cannot throw a null.
C. The program will compile without error and will throw an Exception when run.
D. The program will compile without error and will throw java.lang.NullPointerException when run.
E. The program will compile without error and will run and terminate without any output.

**Correct Answer:** A
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
You are throwing an exception and there is no try or catch block, or a throws clause. So it will not compile. If you either put a try catch block or declare a throws clause for the method then it will throw a NullPointerException at run time because e is null.

A method that throws a 'checked' exception i.e. an exception that is not a subclass of Error or RuntimeException, either must declare it in throws clause or put the code that throws the exception in a try/catch block.

**QUESTION 315**
**What will the following program print when run?**

```
public class TestClass {
   public static void main(String[] args){
       try {
          System.exit(0);
       }
       finally {
          System.out.println("finally is always executed!");
       }
    }
}
```

A. It will print  "finally is always executed!"
B. It will not compile as there is no catch block.
C. It will not print anything.
D. An exception will be thrown
E. None of the above.

**Correct Answer:** C
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
finally is always executed (even if you throw an exception in try or catch) but this is the exception to the rule.
When you call System.exit(...); The JVM exits so there is no way to execute the finally block.

**QUESTION 316**
**Which of the following are standard Java exception classes? Please select  2 options**

A. java.io.FileNotFoundException
B. java.io.InputException
C. java.lang.CPUError
D. java.lang.MemoryException

E. `java.lang.SecurityException`

**Correct Answer:** AE
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
There is an `java.io.IOException` but no `InputException` or `OutputException`.

Java has a `java.lang.SecurityException`. This exception extends `RuntimeException` and is thrown by the security manager upon security violation. For example, when a java program runs in a sandbox (such as an applet) and it tries to use prohibited APIs such as File I/O, the security manager throws this exception. Since this exception is explicitly thrown using the new keyword by a security manager class, it can be considered to be thrown by the application programmer.

There is a `java.lang.OutOfMemoryError` but no `MemoryException`. There is also a `java.lang.StackOverflowError`.

**QUESTION 317**
**Given the classes:**

- `java.lang.AssertionError`
- `java.lang.ArithmeticException`
- `java.lang.ArrayIndexOutofBoundsException`
- `java.io.FileNotFoundException`
- `java.lang.IllegalArgumentException`
- `java.io.IOError`
- `java.io.IOException`
- `java.lang.NumberFormatException`
- `java.sql.SQLException`

**Which option lists only those classes that belong to the unchecked exception category?**

A. `NumberFormatException, ArrayIndexOutOfBoundsException, ArithmeticException`
B. `AssertionError, IOError, IOException`
C. `ArithmeticException, FileNotFoundException, NumberFormatException`
D. `FileNotFoundException, IOException, SQLException`
E. `ArrayIndexOutOfBoundException, IllegalArgumentException, FileNotFoundException`

**Correct Answer:** A
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 318**
**Java's Exception mechanism helps in which of the following ways? Select 2 options**

A. It allows creation of new exceptions that are custom to a particular application domain.
B. It improves code because error handling code is clearly separated from the main program logic.
C. It enhances the security of the application by reporting errors in the logs
D. It improves the code because the exception is handled right at the place where it occured.
E. It provides a vast set of standard exceptions that covers all possible exceptions.

**Correct Answer:** AB
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
It allows creation of new exceptions that are custom to a particular application domain.
You can define your own exceptions based of your application business domain. For example, in a banking application, you might want to create a InsufficientFundsException. This increases code clarity as compared to having a single (or a few standard) exception class(es) and looking at the exception code to determine what happened.

It improves code because error handling code is clearly separated from the main program logic.
The error handling logic is put in the catch block, which makes the main flow of the program clean and easily understandable.

It enhances the security of the application by reporting errors in the logs
Exception handling as such has nothing to do with the security of the application but good exception handling in an application can prevent security holes.

It improves the code because the exception is handled right at the place where it occured.
Just the opposite is true. It improves the code because the code does not have to include error handling code if it is not capable of handling it. It can propagate the exception up the chain and it can be handled at a somewhere at a more appropriate place.

It provides a vast set of standard exceptions that covers all possible exceptions.
Although it does provide a vast set of standard exceptions, they cannot cover all scenarios. But you can always create new exceptions tailored for your application.

**QUESTION 319**
**Which three are advantatges of Java exception mechanism?**

A. Improves the program structure because the error handling code is separated from the normal program function.
B. Provides a set of standard exceptions that covers all the possible errors.
C. Improves the program structure because the programmer can choose where to handle exceptions.
D. Improves the program structure because exceptions must be handled in the method in which they occurred.
E. Allows the creation of new exceptions that are tailored to the particular program being created.

**Correct Answer:** ACE
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 320**
**Checked exceptions are meant for...**

A. exceptional conditions external to an application that a well written application should anticipate and from which it can recover.
B. exceptional conditions external to the program that a well written program cannot anticipate but should recover from.
C. exceptional conditions from which recovery is difficult or impossible.
D. exceptional situations internal to an application that the application can anticipate but cannot recover from.

**Correct Answer:** A

**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
There are multiple view points regarding checked and and unchecked exceptions. As per the official Java tutorial ( http://docs.oracle.com/javase/tutorial/essential/exceptions/runtime.html ) : If a client can reasonably be expected to recover from an exception, make it a checked exception. If a client cannot do anything to recover from the exception, make it an unchecked exception. Here, the client basically means the caller of a method.

Another way to look at exceptions is to see the cause of the exception in terms of whether it is internal or external to the program's code. For example, an incorrectly written code may try to access a reference pointing to null, or it may try to access an array beyond its length. These are internal sources of exception. Here, using runtime exceptions is appropriate because ideally these problems should be identified while testing and should not occur when the program is ready for deployment.

On the other hand, a program interacting with files may not be able to do its job at all if a file is not available but it should anticipate this situation. This is an external source of an exception and has nothing to do with a program's code as such. It is therefore appropriate to use a checked exception here.

**QUESTION 321**
**Given the following code:**

```
public static void main(String[] args) {
   try {
     args = null;
     args[0] = "test";
     System.out.println(args[0]);
   }
   catch (Exception ex) {
     System.out.println("Exception");
   }
   catch (NullPointerException npe) {
     System.out.println("NullPointerException");
   }
}
```

**What is the result?**

A. test
B. Exception
C. Compilation fails.
D. NullPointerException

**Correct Answer:** C
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 322**
**Given the following code:**

```
public static void parse(String str) {
  try {
     float f = Float.parseFloat(str);
  }
  catch (NumberFormatException nfe) {
```

```
     f = 0;
  }
  finally {
     System.out.println(f);
  }
}

public static void main(String[] args) {
   parse("invalid");
}
```

**What is the result?**

A. `0.0`
B. `Compilation fails`
C. `A ParseException is thrown by the parse method at runtime.`
D. `A NumberFormatException is thrown by the parse method at runtime.`

**Correct Answer:** B
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 323**
**Given**

```
class Scoop {
    static int thrower() throws Exception { return 42; }
    public static void main(String [] args) {
      try {
          int x = thrower();
      }
      catch (Exception e) {
         x++;
      }
      finally {
          System.out.println("x = " + ++x);
      }
    }
}
```

**What is the result?**

A. `x = 42`
B. `x = 43`
C. `x = 44`
D. `Compilation fails`
E. `No output`

**Correct Answer:** D
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 324**
**Given:**

```
public class Test {
```

```
   public static void main(String[] args) {
       int ax = 10, az = 30;
       int aw = 1, ay = 1;

       try {
          aw = ax % 2;
          ay = az / aw;
       } catch (ArithmeticException e1) {
          System.out.println("Invalid Divisor");
       } catch (Exception e2) {
          aw = 1;
          System.out.println("Divisor Changed");
       }

       ay = az /aw; // Line 14
       System.out.println("Succesful Division " + ay);
   }

}
```

**What is the result?**

A.
```
Invalid Divisor
Divisor Changed
Successful Division 30
```
B.
```
Invalid Divisor
Successful Division 30
```
C.
```
Invalid Divisor
Exception in thread "main" java.lang.ArithmeticException: / by zero
```
D.
```
Invalid Divisor
Exception in thread "main" java.lang.ArithmeticException: / by zero
Successful Division 1
```

**Correct Answer:** C
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 325**
**Given:**

```
3. public class Chopper {
4.    String a = "12b";
5.    public static void main(String[] args) {
6.        System.out.println(new Chopper().chop(args[0]));
7.    }
8.    int chop(String a) {
9.        if(a == null) throw new IllegalArgumentException();
10.         return Integer.parseInt(a);
11.   } }
```

**And, if the code compiles, the invocation:**
`java Chopper`

**What is the result?**

A. `12`
B. `12b`
C. `Compilation fails.`
D. `A NullPointerException is thrown.`

E. A NumberFormatException is thrown.
F. An IllegalArgumentException is thrown.
G. An ArrayIndexOutOfBoundsException is thrown.

**Correct Answer:** G
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 326**
**Given:**

```
public class Test3 {

     public static void main(String[] args) {
          String[] names = new String[3];
          names[0] = "Mary Brown";
          names[1] = "Nancy Red";
          names[2] = "Jessy Orange";
          try {
                for(String n : names) {
                    try {
                        String pwd = n.substring(0, 3) + n.substring(6,10);
                        System.out.println(pwd);
                    }
                    catch(StringIndexOutOfBoundsException ex) {
                        System.out.println("String out of limits");
                    }
                }

          }catch(ArrayIndexOutOfBoundsException ex) {
                System.out.println("Array out of limits");
          }
     }
}
```

**What is the result?**

A.
```
Marrown
String out of limits
JesOran
```
B.
```
Marrown
String out of limits
Array out of limits
```
C.
```
Marrown
String out of limits
```
D.
```
Marrown
NanRed
JesOran
```

**Correct Answer:** A
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 327**
**Given:**

```
public class TestTry {
```

```
    public static void main(String[] args) {
        StringBuilder message = new StringBuilder("hello java!");
        int pos = 0;
        try {
             for(pos = 0; pos < 12; pos++) {
               switch(message.charAt(pos)) {
                    case 'a':
                    case 'e':
                    case 'o':
                     String utc = Character.toString(message.charAt
                     (pos)).toUpperCase();
                      message.replace(pos, pos+1, utc);
               }
             }
        } catch (Exception ex) {
            System.out.println("Out of limits");
        }
        System.out.println(message);

    }

}
```

**What is the result?**

A. `hEllO jAvA!`
B. `Hello java!`
C. `Out of limits`
   `hEllO jAvA!`
D. `Out of limits`

**Correct Answer:** C
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 328**
**Given the following code:**

```
1. public class A {
2.   public void method1(){
3.     B b = new B();
4.     b.method2();
5.     // more code here
6.   }
7. }
```

```
1. public class B{
2.   public void method2() {
3.     C c = new C();
4.     c.method3();
5.     // more code here
6.   }
7. }
1. public class C {
2.   public void method3(){
3.     // more code here
4.   }
5. }
```

**Given**:

```
25. try {
26.    A a = new A();
27.    a.method1();
28. } catch (Exception e) {
29.    System.out.print("an error occurred");
30. }
```

**Which two statements are true if a NullPointerException is thrown on line 3 of class C?**

A. The application will crash.
B. The code on line 29 will be executed.
C. The code on line 5 of class A will execute.
D. The code on line 5 of class B will execute.
E. The exception will be propagated back to line 27.

**Correct Answer:** BE
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 329**
**Given the following code:**

```
01. public class A{
02.     public void method1() {
03.        try {
04.          B b = new B();
05.          b.method2();
06.          //more code here
07.        } catch (TestException te){
08.           throw new RuntimeException(te);
09.        }
10.    }
11. }
```

```
01. public class B{
02.     public void method2() throws TestException {
03.          //more code here
04     }
05. }
01. class TestException extends Exception {
02. }
```

```
31. public void method() {
32.    A a = new A();
33.    a.method1();
34. }
```

**Which statement is true if a TestException is thrown on line 3 of class B?**

A. Line 33 must be called within a try block.
B. The exception thrown by method1 in class A is not required to be caught.
C. The method declared on line 31 must be declared to throw a RuntimeException.
D. On line 5 of class A, the call to method2 of class B does not need to be placed in a try/catch block.

**Correct Answer:** B
**Section: Handling Exceptions**

**Explanation**

**Explanation/Reference:**

**QUESTION 330**
**Given the following code:**

```
public class A {

   static void test() {
      try {
         String x = null;
         System.out.print(x.toString() + " ");
      }
      finally {
         System.out.print("finally ");
      }
   }

   public static void main(String[] args) {
      try {
        test();
      }
      catch (Exception ex) {
        System.out.print("exception ");
      }
   }

}
```

**What is the result?**

A. `null`
B. `finally`
C. `null finally`
D. `Compilation fails`
E. `finally exception`

**Correct Answer:** E
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 331**
**Given the following code:**

```
public class A {

   static void test() throws Error {
       if (true) throw new AssertionError();
       System.out.print("test ");
   }

   public static void main(String[] args) {
       try {
         test();
       }
       catch (Exception ex) {
         System.out.print("exception ");
       }
```

```
        System.out.print("end ");
    }
}
```

**What is the result?**

A. `end`
B. `Compilation fails.`
C. `exception end`
D. `exception test end`
E. `A Throwable is thrown by main`
F. `An Exception is thrown by main`

**Correct Answer:** E
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 332**
**Given the following code:**

```
01. class TestException extends Exception { }
02. class A {
03.     public String sayHello(String name) throws TestException {
04.         if(name == null) throw new TestException();
05          return "Hello " + name;
06.     }
07. }
08. public class TestA {
09.     public static void main(String[] args) {
10.        new A().sayHello("Aiko");
11.    }
12. }
```

A. `Compilation succeeds.`
B. `Class A does not compile.`
C. `The method declared on line 9 cannot be modified to throw TestException.`
D. `TestA compiles if line 10 is enclosed in a try/catch block that catches TestException.`

**Correct Answer:** D
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 333**
**Given the following code:**

```
11. class A {
12.     void process() throws Exception { throw new Exception(); }
13. }

14. class B extends A {
15.     void process() { System.out.println("B"); }
16.
17.     public static void main(String[] args) {
18.          new B().process();
19.     }
```

```
20. }
```

**What is the result?**

A. B
B. The code runs with no output.
C. Compilation fails because of an error in line 12.
D. Compilation fails because of an error in line 15.
E. Compilation fails because of an error in line 18.

**Correct Answer:** A
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 334**
**Given the following code:**

```
11. class X { public void foo() { System.out.print("X "); } }
12.
13. public class SubB extends X {
14.      public void foo() throws RuntimeException {
15.        super.foo();
16.        if (true) throw new RuntimeException();
17.        System.out.print("B ");
18.      }
19.      public static void main(String[] args) {
20.        new SubB().foo();
21.      }
22. }
```

**What is the result?**

A. X, followed by an Exception
B. No output, and an Exception is thrown.
C. Compilation fails due to an error on line 14.
D. Compilation fails due to an error on line 16.
E. Compilation fails due to an error on line 17.
F. X, followed by an Exception, followed by B.

**Correct Answer:** A
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 335**
**Given the following code:**

```
05. class A {
06.      void foo() throws Exception { throw new Exception(); }
07. }
08. class SubB2 extends A {
09.      void foo() { System.out.println("B "); }
10. }
11. class Tester {
12.      public static void main(String[] args) {
13.         A a = new SubB2();
```

```
14.         a.foo();
15.      }
16. }
```

**What is the result?**

A. B
B. B, followed by an Exception
C. Compilation fails due to an error on line 9.
D. Compilation fails due to an error on line 14.
E. An Exception is thrown with no other output.

**Correct Answer:** D
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 336**
**Given the following code:**

```
import java.io.IOException;

class A {
   public void process() {
     System.out.print("A,");
   }
}

13. class B extends A {
14.     public void process() throws IOException {
15.        super.process();
16.        System.out.print("B,");
17.        throw new IOException();
18.     }
19.
20.     public static void main(String[] args) {
21.       try {
22.          new B().process();
23.       } catch (IOException e) {
24.          System.out.println("Exception");
25.       }
26.     }
27. }
```

**What is the result?**

A. Exception
B. A,B,Exception
C. Compilation fails because of an error in line 20.
D. Compilation fails because of an error in line 14.
E. A NullPointerException is thrown at runtime.

**Correct Answer:** D
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 337**

**Given:**

```
public class Test {

   static void dispResult(int[] num) {
      try {
         System.out.println(num[0] / (num[0] - num[1]));
      }
      catch (ArithmeticException ex) {
         System.out.println("First exception");
      }
      System.out.println("Done");
   }

   public static void main(String[] args) {
      try {
         int[] arr = {100, 100};
         dispResult(arr);
      }
      catch (IllegalArgumentException ex)  {
         System.out.println("Second exception");
      }
      catch(Exception ex) {
         System.out.println("Third exception");
      }
   }

}
```

**What is the result?**

A. 0
   Done
B. First exception
   Done
C. Second exception
D. Done
   Third exception
E. Third exception

**Correct Answer:** B
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 338**
**Given:**

```
class MissingInfoException extends Exception { }

class AgeOutOfRangeException extends Exception { }

class Candidate {
   String name;
   int age;

   Candidate(String name, int age) throws Exception {
      if (name == null) {
         throw new MissingInfoException();
      }
      else if (age <= 10 || age >= 100) {
         throw new AgeOutOfRangeException();
```

```
     }
     else {
        this.name= name;
        this.age = age;
     }
   }

   public String toString() {
      return name + " age: " + age;
   }
}
```

**And**

```
4. public class Test {
5.    public static void main(String[] args) {
6.      Candidate c = new Candidate("James", 20);
7.      Candidate c1 = new Candidate("Williams", 32);
8.      System.out.println(c);
9.      System.out.println(c1);
10.   }
11. }
```

**Which change enables the code to print the following?**

```
James age: 20
Williams age: 32
```

A. Replacing line 5 with
   `public static void main (String [] args) throws MissingInfoException, AgeOutofRangeException {`
B. Replacing line 5 with
   `public static void main (String [] args) throws Exception {`
C. Enclosing line 6 and line 7 within a try block and adding:
   ```
   catch(Exception e1) { //code goes here}
   catch (MissingInfoException e2) { //code goes here}
   catch (AgeOutofRangeException e3) {//code goes here}
   ```
D. Enclosing line 6 and line 7 within a try block and adding:
   ```
   catch (MissingInfoException e2) { //code goes here}
   catch (AgeOutofRangeException e3) {//code goes here}
   ```

**Correct Answer:** B
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 339**
**Given**

```
class Plane {
    static String s = "-";
    public static void main(String[] args) {
        new Plane().s1();
        System.out.println(s);
    }
    void s1() {
        try {
           s2();
        }
        catch (Exception e) {
           s += "c";
        }
```

```
    }
    void s2() throws Exception {
        s3();
        s += "2";
        s3();
        s += "2b";
    }
    void s3() throws Exception {
        throw new Exception();
    }
}
```

**What is the result?**

A. -
B. -c
C. -c2
D. -2c
E. -c22b
F. -2c2b
G. -2c2bc
H. Compilation fails

**Correct Answer:** B
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 340**
**Given**

```
class Emu {
    static String s = "-";
    public static void main(String[] args) {
        try {
            throw new Exception();
        }
        catch (Exception e) {
             try {
                 try {
                    throw new Exception();
                 }
                 catch (Exception ex) {
                    s += "ic ";
                 }
                 throw new Exception();
              }
              catch (Exception x) {
                 s += "mc ";
              }
              finally {
                 s += "mf ";
              }
        }
        finally {
            s += "of ";
        }
        System.out.println(s);
    }
}
```

**What is the result?**

A. `-ic of`
B. `-mf of`
C. `-mc mf`
D. `-ic mf of`
E. `-ic mc mf of`
F. `-ic mc of mf`
G. `Compilation fails`

**Correct Answer:** E
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 341**
**Consider the following program:**

```
import java.lang.*;

class InvalidValueException extends IllegalArgumentException {}
class InvalidKeyException extends IllegalArgumentException {}

class BaseClass {
   void foo() throws IllegalArgumentException {
        throw new IllegalArgumentException();
   }
}

class DeriClass extends BaseClass {
    public void foo() throws IllegalArgumentException {
        throw new InvalidValueException();
    }
}

class DeriDeriClass extends DeriClass {
     public void foo() { // LINE A
        throw new InvalidKeyException();
     }
}

class EHTest {
     public static void main(String []args) {
        try {
           BaseClass base = new DeriDeriClass();
           base.foo();
        }
        catch(RuntimeException e) {
           System.out.println(e);
        }
     }
}
```

**Which one of the following options correctly describes the behavior of this program?**

A. `The program prints the following: InvalidKeyException.`
B. `The program prints the following: RuntimeException.`
C. `The program prints the following: IllegalArgumentException.`
D. `The program prints the following: InvalidValueException.`

E. When compiled, the program will result in a compiler error in line marked with comment Line A due to missing throws clause.

**Correct Answer:** A
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
The program prints the following: InvalidKeyException.
It is not necessary to provide an Exception thrown by a method when the method is overriding a method defined with an exception (using the throws clause). Hence, the given program will compile successfully and it will print InvalidKeyException.

**QUESTION 342**
**Given:**

```
class MarkOutOfBoundsException extends ArrayIndexOutOfBoundsException {}

public class Test {
    public void verify(int[] arr) throws ArrayIndexOutOfBoundsException {
       for (int i = 1; i <= 3; i++) {
         if(arr[i] > 100)
             throw new MarkOutOfBoundsException();
         System.out.println(arr[i]);
       }
    }
    public static void main(String[] args) {
       int[] arr = {105,78,56};
       try {
          new Test().verify(arr);
       }
       catch (ArrayIndexOutOfBoundsException | MarkOutOfBoundsException e) {
          System.out.print(e.getClass());
       }
    }
}
```

**What is the result?**

A. Compilation fails.
B. 78
class java.lang.ArrayIndexOutOfBoundException
C. class MarkOutOfBoundException
D. class java.lang.ArrayIndexOutOfBoundException

**Correct Answer:** A
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 343**
**Given:**

```
public class Test {
   void display(String[] arr) {
      try {
         System.out.print(arr[2]);
      }
      catch (ArrayIndexOutOfBoundsException | NullPointerException e) {
         e = new RuntimeException();
         throw e;
      }
```

```
    }
    public static void main(String[] args) throws Exception {
      try {
         String[] arr = {"Unix","Solaris",null};
         new Test().display(arr);
      }
      catch(Exception e) {
         System.err.print(e.getClass());
      }
    }
}
```

**What is the result?**

A. `null`
B. `java.lang.ArraylndexOutOfBoundsException`
C. `java.lang.NullPointerException`
D. `java.lang.Exception`
E. `Compilation fails`

**Correct Answer:** E
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 344**
**Given:**

```
class InvalidAgeException extends IllegalArgumentException { }

public class Tracker {
    void verify (int age) throws IllegalArgumentException {
       if (age < 12)
             throw new InvalidAgeException ();
       if (age >= 12 && age <= 60)
             System.out.print("General category");
       else
             System.out.print("Senior citizen category");
    }
    public static void main(String[] args) {
       int age = Integer.parseInt(args[1]);
       try {
          new Tracker().verify(age);
       }
       catch (Exception e) {
          System.out.print(e.getClass());
       }
    }
}
```

**And the command-line invocation:**
`java Tracker 12 11`

**What is the result?**

A. `General category`
B. `class InvalidAgeException`
C. `class java.lang.IllegalArgumentException`
D. `class java.lang.RuntimeException`

**Correct Answer:** B

**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 345**
**Given:**

```
public class FunWithArgs {
   public static void main(String[][] args) {
       System.out.println(args[0][1]);
   }
   public static void main(String[] args) {
       FunWithArgs fwa = new FunWithArgs();
       String[][] newargs = {args};
       fwa.main(newargs);
   }
}
```

**The above program is compiled with the command line:**
`javac FunWithArgs.java`

**and then run with:**
`java FunWithArgs a b c`

**What will be the output?**

A. `It will not compile.`
B. `It will throw java.lang.ArrayIndexOutOfBoundsException at run time.`
C. `It will print b`
D. `It will print null`

**Correct Answer:** C
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
There is no problem with the code. The main method is just overloaded. When it is run, the main method with String[] will be called. This method then calls the main with String[][] with an argument { {"a", "b", "c"} }  Thus, args[0][1] refers to "b", which is what is printed.

**QUESTION 346**
**Given the following code:**

```
public class MainClass {
     public static void main(String[] args) {
         //INSERT CODE HERE
         System.out.println(var);
     }
}
```

**What can be inserted in the above code so that it will compile and run without any problem? Select 3 options**

A. `double var = 0xb10_000;`
B. `float var = 0b10_000;`
C. `float var = 0b20_000;`
D. `float var = 0b10_000f;`
E. `long var = 0b10000L;`
F. `double var = 0b10_000D;`

**Correct Answer:** ABE
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
The real exam contains a few questions that test you on how to write numbers in binary. You might want to go through Section 3.10.1 and 3.10.2 of Java Language Specification to understand how this works.

`double x = 0xb10_000;`
0x implies the following digits must be interpreted as Hexadecimal digits and b is a valid Hexadecimal digit.

`float x = 0b10_000;`
A number starting with 0b (or 0B) implies that it is written in binary.
Since 10000 can fit into a float, an explicit cast is not required.
Note that when you specify the bit pattern using binary or hex, an explicit cast is not required even if the number specified using the bit pattern is larger than what a float can hold.

`float x = 0b20_000;`
Since it starts with 0b, that means you are writing the number in binary digits (i.e. 0 or 1). But 2 is not a valid binary digit.

`float x = 0b10_000f;`
This is invalid because the floating point suffices f, F, d, and D are used only when using decimal system and not while using binary. However, since f is a valid digit in hexadecimal system, a hex number may end with an f although it will not be interpreted as float but as the digit f. Thus, float x = 0x10_000f; and float x = 10_000f; are valid because they are written in hex and decimal respectively but float x = 0b10_000f; is invalid because is written in binary. Note that a floating point number cannot be written in Octal. Therefore, float x = 010_000f; is valid but it is not octal even though it starts with a 0. It is interpreted in decimal.

`long x = 0b10000L;`

`double d = 0b10_000D;`
A floating point number written in binary or hex cannot use any suffix for float. But a floating point number written in decimal can use the floating point suffices f, F, d, and D.
Thus, float dx = 0xff; is valid but the f here is not for indicating that it is a float but is interpreted as the hex digit F.

**QUESTION 347**
**What will the following code print when run?**

```
public class TestClass {
    public void switchString(String input){
        switch(input) {
          case "a" : System.out.println( "apple" );
          case "b" : System.out.println( "bat" );
                     break;
          case "c" : System.out.println( "cat" );
          default :  System.out.println( "none" );
        }
     }

     public static void main(String[] args) {
         TestClass tc = new TestClass();
         tc.switchString("c");
     }
}
```

A.
```
apple
cat
none
```
B.
```
apple
cat
```
C.
```
cat
```

none

D. cat

E. Compilation fails

**Correct Answer:** C
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**
Since there is a case condition that matches the input string "c", that case statement will be executed directly. This prints "cat". Since there is no break after this case statement and the next case statement, the control will fall through the next one (which is default : ) and so "none" will be printed as well.

In the JDK 7 release, you can use a String object in the expression of a switch statement.

**QUESTION 348**
**What will be the output when the following program is run?**

```
package exceptions;

public class TestClass{
    public static void main(String[] args) {
        try{
            hello();
        }
        catch(MyException me){
            System.out.println(me);
        }
    }

    static void hello() throws MyException{
        int[] dear = new int[7];
        dear[0] = 747;
        foo();
    }

    static void foo() throws MyException{
        throw new MyException("Exception from foo");
    }
}

class MyException extends Exception {
    public MyException(String msg){
        super(msg);
    }
}
```

**(Assume that line numbers printed in the messages given below are correct.)**

A.
```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
        at exceptions.TestClass.doTest(TestClass.java:24)
        at exceptions.TestClass.main(TestClass.java:14)
```

B. Error in thread "main" java.lang.ArrayIndexOutOfBoundsException

C. exceptions.MyException: Exception from foo

D.
```
exceptions.MyException: Exception from foo
        at exceptions.TestClass.foo(TestClass.java:29)
        at exceptions.TestClass.hello(TestClass.java:25)
        at exceptions.TestClass.main(TestClass.java:14)
```

**Correct Answer:** C
**Section: Handling Exceptions**

**Explanation**

**Explanation/Reference:**
Note that there are a few questions in the exam that test your knowledge about how exception messages are printed. When you use System.out.println (exception), a stack trace is not printed. Just the name of the exception class and the message is printed.

When you use exception.printStackTrace(), a complete chain of the names of the methods called, along with the line numbers, is printed. It contains the names of the methods in the chain of method calls that led to the place where the exception was created going back up to the point where the thread, in which the exception was created, was started.

**QUESTION 349**
**Given the code fragment:**

```
int[] lst = {1, 2, 3, 4, 5, 4, 3, 2, 1};
int sum = 0;
for (int frnt = 0, rear = lst.length - 1;
     frnt < 5 && rear >= 5;
     frnt++, rear--) {
   sum = sum + lst[frnt] + lst[rear];
}
System.out.print(sum);
```

**What is the result?**

A. 20
B. 25
C. 29
D. Compilation fails
E. An ArrayIndexOutOfBoundsException is thrown at runtime

**Correct Answer:** A
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 350**
**Consider the contents of following two files:**

```
//In file A.java
package a;

public class A {
   A(){ }
   public void  print() {
      System.out.println("A");
   }
}

//In file B.java
package b;

import a.*;

public class B extends A {
    B(){ }
    public void  print() {
       System.out.println("B");
    }
```

```
    public static void main(String[] args){
       new B().print();
    }
}
```

**What will be printed when you try to compile and run class B?**

A. It will print A.
B. It will print B.
C. It will not compile.
D. It will compile but will not run.
E. None of the above.

**Correct Answer:** C
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
Because A() is not accessible in B.

Note that there is no modifier for A's constructor. So it has default access. This means only classes in package a can use it. Also note that class B is in a different package and is extending from A. In B's constructor the compiler will automatically add super() as the first line. But since A() is not accessible in B, this code will not compile.

**QUESTION 351**
**Given:**

```
public class DoCompare1 {
   public static void main(String[] args) {
      String[] table = {"aa", "bb", "cc"};
      for(String ss : table) {
         int ii = 0;
         while (ii < table.length) {
            System.out.println(ss + ", " + ii);
            ii++;
         }
      }
   }
}
```

**How many time(s) is 2 printed?**

A. The program prints nothing.
B. One time.
C. Two times.
D. Three times.
E. Compilation fails.

**Correct Answer:** D
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**
```
aa, 0
aa, 1
aa, 2
bb, 0
bb, 1
bb, 2
cc, 0
```

```
cc, 1
cc, 2
```

**QUESTION 352**
**What will the following code print when compiled and run?**

```
public class TestClass {
   public static void main(String[] args) {
      String s = "blooper";
      StringBuilder sb = new StringBuilder(s);
      s.append("whopper");
      sb.append("shopper");
      System.out.println(s);
      System.out.println(sb);
    }
}
```

A. `blooper and bloopershopper`
B. `blooperwhopper and bloopershopper`
C. `blooper and blooperwhoppershopper`
D. `It will not compile.`

**Correct Answer:** D
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
`append() method does not exist in String class. It exits only in StringBuffer and StringBuilder. The value of sb will be bloopershopper though.`

**QUESTION 353**
**What is the result of compiling and running this code?**

```
class MyException extends Throwable {}
class MyException1 extends MyException {}
class MyException2 extends MyException {}
class MyException3 extends MyException2 {}

public class ExceptionTest {
    void myMethod() throws MyException {
         throw new MyException3();
    }
    public static void main(String[] args) {
         ExceptionTest et = new ExceptionTest();
         try {
            et.myMethod();
         }
         catch(MyException me){
            System.out.println("MyException thrown");
         }
         catch(MyException3 me3) {
            System.out.println("MyException3 thrown");
         }
         finally {
            System.out.println(" Done");
         }
     }
}
```

A. `MyException thrown`
B. `MyException3 thrown`
C. `MyException thrown Done`
D. `MyException3 thrown Done`

E. It fails to compile

**Correct Answer:** E
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
You can have multiple catch blocks to catch different kinds of exceptions, including exceptions that are subclasses of other exceptions. However, the catch clause for more specific exceptions (i.e. a SubClassException) should come before the catch clause for more general exceptions ( i.e. a SuperClassException). Failure to do so results in a compiler error as the more specific exception is unreachable.

In this case, catch for MyException3 cannot follow catch for MyException because if MyException3 is thrown, it will be caught by the catch clause for MyException. And so, there is no way the catch clause for MyException3 can ever execute. And so it becomes an unreachable statement.

**QUESTION 354**
**Consider the following program:**

```
class MyClass {

   private void m1(Object o) {
      System.out.println("Object");
   }
   private void m1(double [] arr) {
      System.out.println("double []");
   }

   private void m1(int o) {
      System.out.println("int");
   }

   private void m1() {
      System.out.println("void");
   }
   public static void main(String[]args) {
      MyClass c = new MyClass();
      c.m1(null);
   }
}
```

**Which one of the following options correctly describes the behavior of this program?**

A. It throws a compiler error for ambiguous overload.
B. When executed, the program prints the following: Object.
C. When executed, the program prints the following: double [].
D. When executed, the program prints the following: void.
E. When executed, the program prints the following: int.

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 355**
**What is the result of compiling and running the following program?**

```
public class Learner {
```

```
    public static void main(String[] args) {
        String[] dataArr = new String[4];
        dataArr[1] = "Bill";
        dataArr[2] = "Steve";
        dataArr[3] = "Larry";
        try {
           for(String data : dataArr) {
              System.out.print(data + " ");
           }
        } catch(Exception e) {
           System.out.println(e.getClass());
        }
     }
}
```

A. `Bill Steve Larry null`
B. `Bill Steve Larry class java.lang.NullPointerException`
C. `class java.lang.Exception Bill Steve Larry`
D. `Bill Steve Larry class java.lang.Exception`
E. `null Bill Steve Larry`

**Correct Answer:** E
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**
`Array indexing starts with 0. The first element therefore is at dataArr[0], which is not set in this code. It is initialized by default to null. Hence, the code prints null Bill Steve Larry.`

**QUESTION 356**
**Given the code fragment:**

```
String p = "nature*beautiful";
int vowel = 0;
int num = 0;
for (int idx = p.length() - 1, c = 7;
     idx > 0 && p.charAt(idx) != '*';
     idx--) {

   switch(p.charAt(idx)) {
       case 'a':
       case 'e':
       case 'i':
       case 'o':
       case 'u':
           vowel = --c;
   }
   num = idx;
}
System.out.println(vowel + ", " + num);
```

**What is the result?**

A. `2, 7`
B. `1, 6`
C. `0, 7`
D. `An IndexOutOfBoundsException is thrown at runtime.`
E. `Compilation fails.`

**Correct Answer:** A
**Section: Using Loops Constructs**

**Explanation**

**Explanation/Reference:**

**QUESTION 357**
**What will the following code print when compiled and run?**

```
public class OrderTest {
    public void initData(String[] arr){
       int ind = 0;
       for(String str : arr) {
          str.concat(str + " " + ind);
          ind++;
       }
    }

    public void printData(String[] arr){
       for(String str : arr){
          System.out.println(str);
       }
    }

    public static void main(String[] args) {
       OrderTest ot = new OrderTest();
       String[] arr = new String[2];
       ot.initData(arr);
       ot.printData(arr);
    }
}
```

A.
```
null 0
null 1
```

B.
```
0
1
```

C.
```
   0
   1
```

D.
```
null
null
```

E. `It will throw a RuntimeException at run time.`

**Correct Answer:** E
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**
When you do new String[2], you create a String array of two elements. arr is therefore not null. But each element of the array is not given any value and is therefore null. When you call a method on that element (i.e. str.concat(str+" "+ind); in initData), it will generate a NullPointerException, which is a RuntimeException.

**QUESTION 358**
**What can be inserted at //1 and //2 in the code below so that it can compile without errors:**

```
class Doll {
    String name;
    Doll(String nm){
        this.name = nm;
    }
}

class Barbie extends Doll {
    Barbie(){
```

```
        //1
    }
    Barbie(String nm){
        //2
    }
}

public class TestClass {
    public static void main(String[] args) {
        Barbie b = new Barbie("mydoll");
    }
}
```

**Please select 2 options**

A. this("unknown"); at 1 and super(nm); at 2
B. super("unknown"); at 1 and super(nm); at 2
C. super(); at 1 and super(nm); at 2
D. super(); at 1 and Doll(nm); at 2
E. super("unknown"); at 1 and this(nm); at 2
F. Doll(); at 1 and Doll(nm); at 2

**Correct Answer:** AB
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 359**
**Given:**

```
public class MyFor1 {
   public static void main(String[] args) {
      int[] x = {6, 7, 8};
      for(int i : x) {
         System.out.print(i + " ");
         i++;
      }
   }
}
```

**What is the result?**

A. 6 7 8
B. 7 8 9
C. 0 1 2
D. 6 8 10
E. Compilation fails.
F. An exception is thrown at runtime.

**Correct Answer:** A
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 360**
**What will the following code print when run?**

```
public class TestClass {
    public void switchString(String input){
        switch(input){
            case "a" : System.out.println( "apple" );
            case "b" : System.out.println( "bat" );
                break;
            case "B" : System.out.println( "big bat" );
            default : System.out.println( "none" );
        }
    }

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.switchString("B");
    }
}
```

A. bat
   big bat
B. big bat
   none
C. big bat
D. bat
E. The code will not compile.

**Correct Answer:** B
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 361**
**Consider the following code:**

```
public class Logger {
   private StringBuilder sb = new StringBuilder();

   public void logMsg(String location, String message) {
       sb.append(location);
       sb.append("-");
       sb.append(message);
   }

   public void dumpLog() {
       System.out.println(sb.toString());
       //Empty the contents of sb here
   }
}
```

**Which of the following options will empty the contents of the StringBuilder referred to by variable sb in method dumpLog()?**

A. `sb.delete(0, sb.length());`
B. `sb.clear();`
C. `sb.empty();`
D. `sb.removeAll();`
E. `sb.deleteAll();`

**Correct Answer:** A
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
public StringBuilder delete(int start, int end)
Removes the characters in a substring of this sequence. The substring begins at the specified start and extends to the character at index end - 1 or to the end of the sequence if no such character exists. If start is equal to end, no changes are made.

Parameters:
start - The beginning index, inclusive.
end - The ending index, exclusive.
Returns: This object.
Throws:
StringIndexOutOfBoundsException - if start is negative, greater than length(), or greater than end.

**QUESTION 362**
**Consider the following method**

```
public float parseFloat( String s ) {
    float f = 0.0f;
    try {
        f = Float.valueOf(s).floatValue();
        return f;
    }
    catch(NumberFormatException nfe){
        f = Float.NaN;
        return f;
    }
    finally {
        f = 10.0f;
        return f;
    }
}
```

**What will it return if the method is called with the input "0.0" ?**

A. It will not compile.
B. It will return 10.0
C. It will return Float.Nan
D. It will return 0.0
E. None of the above.

**Correct Answer:** B
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
finally block will always execute (except when there is a System.exit() in try or catch). And inside the finally block, it is setting f to 10.0. So no matter what the input is, this method will always return 10.0.

**QUESTION 363**
**Which statements about the following code are correct?**

```
interface House {
   public default String getAddress() {
        return "101 Main Str";
   }
}

interface Bungalow extends House {
   public default String getAddress() {
        return "101 Smart Str";
```

```
   }
}

class MyHouse implements Bungalow, House {  }

public class TestClass {
    public static void main(String[] args) {
         House ci = new MyHouse();  //1
         System.out.println(ci.getAddress()); //2
    }
}
```

A. Code for interface House will cause compilation to fail.
B. Code for interface Bungalow will cause compilation to fail.
C. Code for class MyHouse will cause compilation to fail.
D. Line at //1 will cause compilation to fail.
E. Line at //2 will cause compilation to fail.
F. The code will compile successfully.

**Correct Answer:** F
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
There is no problem with the code. It is perfectly valid for a subinterface to override a default method of the base interface. A class that implements an interface can also override a default method. It is valid for MyHouse to say that it implements Bungalow as well as House even though House is redundant because Bungalow is a House anyway.

It will actually print 101 Smart str.

**QUESTION 364**
**Given:**

```
public class Test2 {
   public static void doChange(int[] arr) {
      for(int pos = 0; pos < arr.length; pos++) {
         arr[pos] = arr[pos] + 1;
      }
   }
   public static void main(String[] args) {
      int[] arr = {10, 20, 30};
      doChange(arr);
      for(int x : arr) {
         System.out.print(x + ", ");
      }
      doChange(arr[0], arr[1], arr[2]);
      System.out.print(arr[0] + ", " + arr[1] + ", " + arr[2]);
   }
}
```

**What is the result?**

A. 11, 21, 31, 11, 21, 31
B. 11, 21, 31, 12, 22, 32
C. 12, 22, 32, 12, 22, 32
D. 10, 20, 30, 10, 20, 30
E. Compilation fails.
F. An exception is thrown at runtime.

**Correct Answer:** E

**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**
```
error: method doChange in class Main cannot be applied to given types;
      doChange(arr[0], arr[1], arr[2]);
      ^
  required: int[]
  found: int,int,int
  reason: actual and formal argument lists differ in length
```

**QUESTION 365**
**What is the result of executing the following code when the value of i is 5:**

```
switch (i) {
   default:
   case 1:
       System.out.println(1);
   case 0:
       System.out.println(0);
   case 2:
       System.out.println(2);
       break;
   case 3:
       System.out.println(3);
}
```

A. It will print 1 0 2
B. It will print 1 0 2 3
C. It will print 1 0
D. It will print 1
E. Nothing will be printed.

**Correct Answer:** A
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**
Here are the rules:

The type of the switch expression must be String, char, byte, short, or int (and their wrapper classes), or an enum or a compile-time error occurs.

All of the following must be true, or a compile-time error will result:
1. Every case constant expression associated with a switch statement must be assignable (5.2) to the type of the switch Expression.
2. No two of the case constant expressions associated with a switch statement may have the same value.
3. At most one default label may be associated with the same switch statement.

Basically it looks for a matching case or if no match is found it goes to default. (If default is also not found it does nothing)
Then it executes the statements till it reaches a break or end of the switch statement.
Here, it goes to default and executes till it reaches first break. So it prints 1 0 2.

Note that the switch statement compares the String object in its expression with the expressions associated with each case label as if it were using the String.equals method; consequently, the comparison of String objects in switch statements is case sensitive. The Java compiler generates generally more efficient bytecode from switch statements that use String objects than from chained if-then-else statements.

**QUESTION 366**
**Given:**

```
public class MyFor {
   public static void main(String[] args) {
       int[] arr = null;
       for (int num : arr) {
          System.out.println(num);
       }
   }
}
```

**What is the result?**

A. `null`
B. `Compilation fails.`
C. `An exception is thrown at runtime.`
D. `0`

**Correct Answer:** C
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

```
Exception in thread "main" java.lang.NullPointerException
    at Main.main(Main.java:4)
```

**QUESTION 367**
**Given:**

```
package strings;

public class StringFromChar {
     public static void main(String[] args) {
          String myStr = "good";
          char[] myCharArr = {'g', 'o', 'o', 'd' };
          String newStr = null;
          for(char ch : myCharArr){
              newStr = newStr + ch;
          }
          System.out.println((newStr == myStr)+ " " + (newStr.equals(myStr)));

     }
}
```

**What will it print when compiled and run?**

A. `true true`
B. `true false`
C. `false true`
D. `false false`

**Correct Answer:** D
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

```
Since newStr is null at the beginning, the first iteration of the loop assigns
"nullg" to newStr. Therefore, at the end of the loop, myStr is actually
"nullgood".

Had newStr been defined as String newStr = ""; then the program would have
printed false for newStr == myStr because both the references are pointing to
```

two different objects, and true for newStr.equals(myStr) because both the objects contain the exact same String.

**QUESTION 368**
**What will the following program print when run?**

```
// Filename: TestClass.java
public class TestClass {
   public static void main(String args[] ){ A b = new B("good bye");  }
}

class A {
   A() { this("hello", " world");  }
   A(String s) { System.out.println(s); }
   A(String s1, String s2){ this(s1 + s2); }
}

class B extends A{
   B(){ super("good bye"); };
   B(String s){ super(s, " world"); }
   B(String s1, String s2){ this(s1 + s2 + " ! "); }
}
```

A. It will print "good bye".
B. It will print "hello world".
C. It will print "good bye world".
D. It will print "good bye" followed by "hello world".
E. It will print "hello world" followed by "good bye".

**Correct Answer:** C
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
new B("good bye"); will call class B's one args constructor which in turn calls super(s, " world"); (i.e. class A's two args constructor) which in turn calls this(s1 + s2); (i.e. class A's one arg constructor with parameter "good bye world") which prints it.

**QUESTION 369**
**Given:**

```
public class Test2 {
   public static void main(String[] args) {
       int ar1[] = {2, 4, 6, 8};
       int ar2[] = {1, 3, 5, 7, 9};
       ar2 = ar1;
       for (int e2 : ar2) {
          System.out.print(" " + e2);
       }
   }
}
```

**What is the result?**

A. 2 4 6 8
B. 2 4 6 8 9
C. 1 3 5 7
D. 1 3 5 7 9
E. Compilation fails.
F. An exception is thrown at runtime.

**Correct Answer:** A
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 370**
**Given:**

```
public class Calculator {
   public static void main(String[] args) {
       int num = 5;
       int sum;
       do {
         sum += num;
       } while ((num--) > 1);

       System.out.println("The sum is " + sum + ".");
   }
}
```

**What is the result?**

A. `The sum is 2.`
B. `The sum is 14.`
C. `The sum is 15.`
D. `The loop executes infinite times.`
E. `Compilation fails.`

**Correct Answer:** E
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

```
Main.java:6: error: variable sum might not have been initialized
         sum += num;
```

**QUESTION 371**
**What will be the output of the following class:**

```
public class TestClass{
  public void testRefs(String str, StringBuilder sb){
    str = str + sb.toString();
    sb.append(str);
    str = null;
    sb = null;
  }
  public static void main(String[] args){
    String s = "aaa";
    StringBuilder sb = new StringBuilder("bbb");
    new TestClass().testRefs(s, sb);
    System.out.println("s="+s+" sb="+sb);
  }
}
```

A. `s=aaa sb=bbb`
B. `s=null sb=null`
C. `s=aaa sb=null`
D. `s=null sb=bbbaaa`
E. `s=aaa sb=bbbaaabbb`

**Correct Answer:** E
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
Always remember that Strings are immutable, you cannot change them. In this case, s refers to "aaa", and no matter what testRefs() method does, the variable s of main() will keep pointing to the same string "aaa".

StringBuilder on the other hand, is mutable. So, initially sb is pointing to a StringBuilder object containing "bbb". Its reference is passed to the testRefs () method. In that method, we change the local variable str to point to a new string "aaa"+"bbb" = "aaabbb". Then we append this to sb. Therefore sb now contains "bbbaaabbb".

Setting the local reference str and sb (in method testRefs()) to null, does not affect the variables s and sb of the main() method.

**QUESTION 372**
**What will be the result of attempting to run the following program?**

```
public class StringArrayTest {
    public static void main(String args[]){
        String[][][] arr = {{ {"a", "b" , "c"}, {"d", "e", null}},{ {"x"}, null
},{{"y"}},{{"z","p"}, {}}};
        System.out.println(arr[0][1][2]);
    }
}
```

A. It will throw NullPointerBoundsException.
B. It will throw ArrayIndexOutOfBoundsException.
C. It will print null.
D. It will run without any error but will print nothing.
E. None of the above.

**Correct Answer:** C
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**
arr[0][1][2] => [0] = { { "a", "b" , "c"}, { "d", "e", null } }, [1] = { "d", "e", null } and [2] = null.
So it will print null.

**QUESTION 373**
**Given:**

```
class Caller {

   private void init() {
     System.out.println("Initialized");
   }

    public void start() {
      init();
      System.out.println("Started");
    }
}

public class TestCall {
   public static void main(String[] args) {
    Caller c = new Caller();
    c.start();
    c.init();
```

```
    }
}
```

**What is the result?**

A.
```
Initialized
Started
```

B.
```
Initialized
Started
Initialized
```

C. `Compilation fails`

D. `An exception is thrown at runtime`

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 374**
**What will the following code print when compiled and run? (Assume that MySpecialException is an unchecked exception.)**

```
1. public class ExceptionTest {
2.    public static void main(String[] args) {
3.        try {
4.            doSomething();
5.        } catch (MySpecialException e) {
6.            System.out.println(e);
7.        }
8.    }
9.
10.    static void doSomething() {
11.        int[] array = new int[4];
12.        array[4] = 4;
13.        doSomethingElse();
14.    }
15.
16.    static void doSomethingElse() {
17.        throw new MySpecialException("Sorry, can't do something else");
18.    } }
```

A. `It will not compile.`

B.
```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4
    at ExceptionTest.doSomething(ExceptionTest.java:12)
    at ExceptionTest.main(ExceptionTest.java:4)
```

C.
```
Exception in thread "main" MySpecialException: 4
    at ExceptionTest.doSomethingElse(ExceptionTest.java:17)
    at ExceptionTest.doSomething(ExceptionTest.java:12)
    at ExceptionTest.main(ExceptionTest.java:4)
```

D.
```
Exception in thread "main" MySpecialException: Sorry, can't do something
else
    at ExceptionTest.doSomethingElse(ExceptionTest.java:17)
    at ExceptionTest.doSomething(ExceptionTest.java:12)
    at ExceptionTest.main(ExceptionTest.java:4)
```

E.
```
Exception in thread "main" MySpecialException: Sorry, can't do something
else
    at ExceptionTest.doSomethingElse(ExceptionTest.java:17)
    at ExceptionTest.doSomething(ExceptionTest.java:13)
    at ExceptionTest.main(ExceptionTest.java:4)
```

**Correct Answer:** B
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 375**
**Given:**

```
String[] colors = {"red", "blue", "green", "yellow", "maroon", "cyan"};
```

**Which code fragment prints `blue, cyan,` ?**

A.
```
for (String c : colors) {
      if (c.length() != 4) {
         continue;
      }
      System.out.print(c + ", ");
  }
```
B.
```
for (String c : colors[]) {
      if (c.length() <= 4) {
         continue;
      }
      System.out.print(c + ", ");
  }
```
C.
```
for (String c : String[] colors) {
      if (c.length() >= 3) {
         continue;
      }
      System.out.print(c + ", ");
  }
```
D.
```
for (String c : colors) {
      if (c.length() != 4) {
         System.out.print(c + ", ");
         continue;
      }
  }
```

**Correct Answer:** A
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 376**
**Given the code fragment:**

```
import java.util.*;

public class Test {
  public static void main(String[] args) {
     List<String> names = new ArrayList<>();
     names.add("Robb");
     names.add("Bran");
     names.add("Rick");
     names.add("Bran");

     if (names.remove("Bran")) {
           names.remove("Jon");
     }
     System.out.println(names);
  }
```

```
}
```

**What is the result?**

A. `[Robb, Rick, Bran]`
B. `[Robb, Rick]`
C. `[Robb, Bran, Rick, Bran]`
D. `An exception is thrown at runtime`

**Correct Answer:** A
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 377**
**Given:**

```
//in file Movable.java
package p1;

public interface Movable {
   int location = 0;
   void move(int by);
   public void moveBack(int by);
}

//in file Donkey.java
package p2;

import p1.Movable;

public class Donkey implements Movable {
    int location = 200;
    public void move(int by) {
        location = location + by;
    }
    public void moveBack(int by) {
        location = location - by;
    }
}

//in file TestClass.java
package px;

import p1.Movable;
import p2.Donkey;

public class TestClass {
     public static void main(String[] args) {
          Movable m = new Donkey();
          m.move(10);
          m.moveBack(20);
          System.out.println(m.location);
     }
}
```

**Identify the correct statement.**

A. `Donkey.java will not compile.`
B. `TestClass.java will not compile.`
C. `Movable.java will not compile.`

D. It will print 190 when TestClass is run.
E. It will print 0 when TestClass is run.

**Correct Answer:** E
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
There is no problem with the code. All variables in an interface are implicitly public, static, and final. All methods in an interface are public. There is no need to define them so explicitly.

Therefore, the location variable in Movable is public and static and the move() method is public. Now, when you call m.move(10) and m.moveBack(20), the instance member location of Donkey is updated to 190 because the reference m refers to a Donkey at run time and so move and moveBack methods of Donkey are invoked at runtime. However, when you print m.location, it is the Movable's location (which is never updated) that is printed.

**QUESTION 378**
**Given:**

```
public class Series {
    private boolean flag;

    public void displaySeries() {
       int num = 2;
       while (flag) {
          if (num % 7 == 0)
              flag = false;
          System.out.println(num);
          num += 2;
       }
    }

    public static void main(String[] args) {
       new Series().displaySeries();
    }
}
```

**What is the result?**

A. 2 4 6 8 10 12
B. 2 4 6 8 10 12 14
C. Compilation fails.
D. The program prints multiples of 2 infinite times.
E. The program prints nothing.

**Correct Answer:** E
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 379**
**Given the code fragment:**

```
 9. int a = -10;
10. int b = 17;
11. int c = expression1;
12. int d = expression2;
13. c++;
```

```
14. d--;
15. System.out.print(c + ", " + d);
```

**What could expression1 and expression2 be, respectively, in order to produce output `-8, 16`?**

A. `++a, --b`
B. `++a, b--`
C. `a++, --b`
D. `a++, b--`

**Correct Answer:** B
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 380**
**Given the code fragment:**

```
import java.util.*;

public class Test {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();

    list.add("SE");
    list.add("EE");
    list.add("ME");
    list.add("SE");
    list.add("EE");

    list.remove("SE");

    System.out.print("Values are : "  + list);
  }
}
```

**What is the result?**

A. `Values are: [EE, ME]`
B. `Values are: [EE, EE, ME]`
C. `Values are: [EE, ME, EE]`
D. `Values are: [SE, EE, ME, EE]`
E. `Values are: [EE, ME, SE, EE]`
F. `Compilation fails.`
G. `None of the above.`

**Correct Answer:** E
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 381**
**Given:**

```
3. public class OffRamp {
4.    public static void main(String[] args) {
5.       int [] exits = {0,0,0,0,0,0};
```

```
6.       int x1 = 0;
7.
8.       for(int x = 0; x < 4; x++) exits[0] = x;
9.       for(int x = 0; x < 4; ++x) exits[1] = x;
10.
11.      x1 = 0; while(x1++ < 3) exits[2] = x1;
12.      x1 = 0; while(++x1 < 3) exits[3] = x1;
13.
14.      x1 = 0; do { exits[4] = x1; } while(x1++ < 7);
15.      x1 = 0; do { exits[5] = x1; } while(++x1 < 7);
16.
17.      for(int x: exits)
18.         System.out.print(x + " ");
19. } }
```

**What is the result?**

A. `3 3 2 2 6 6`
B. `3 3 3 2 7 6`
C. `3 3 3 2 7 7`
D. `4 3 3 2 7 6`
E. `4 3 3 2 7 7`
F. `Compilation fails.`

**Correct Answer:** B
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 382**
**Given:**

```
public class Test {
   public static void main(String[] args) {
       int ax = 10, az = 30;
       int aw = 1, ay = 1;
       try {
          aw = ax % 2;
          ay = az / aw;
       } catch (ArithmeticException e1) {
          System.out.println("Invalid Divisor");
       } catch (Exception e2) {
          aw = 1;
          System.out.println("Divisor Changed");
       }
       ay = az /aw; // Line 14
       System.out.println("Succesful Division " + ay);
   }
}
```

**What is the result?**

A.
```
Invalid Divisor
Divisor Changed
Successful Division 30
```
B.
```
Invalid Divisor
Successful Division 30
```
C.
```
Invalid Divisor
Exception in thread "main" java.lang.ArithmeticException: / by zero
```
D.
```
Invalid Divisor
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

```
   Successful Division 1
```

**Correct Answer:** C
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 383**
**What will the following code snippet print?**

```
int index = 1;
String[] strArr = new String[5];
String   myStr  = strArr[index];
System.out.println(myStr);
```

A. nothing
B. null
C. It will throw ArrayIndexOutOfBounds at runtime.
D. It will print some junk value.
E. None of the above.

**Correct Answer:** B
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**
When you create an array of Objects ( here, Strings) all the elements are initialized to null. So in the line 3, null is assigned to myStr.
Note that. empty string is "" ( String str = ""; ) and is not same as null.

**QUESTION 384**
**What will the following code print?**

```
public class Test {
   public int luckyNumber(int seed) {
        if(seed > 10) return seed%10;
        int x = 0;
        try {
           if (seed%2 == 0) throw new Exception("No Even no.");
           else return x;
        }
        catch(Exception e) {
           return 3;
        }
        finally {
           return 7;
        }
    }

    public static void main(String args[]){
        int amount = 100, seed = 6;
        switch( new Test().luckyNumber(6) ){
            case 3: amount = amount * 2;
            case 7: amount = amount * 2;
            case 6: amount = amount + amount;
            default :
        }
        System.out.println(amount);
    }
}
```

A. It will not compile.
B. It will throw an exception at runtime.
C. 800
D. 200
E. 400

**Correct Answer:** E
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 385**
**What will the following code print when compiled and run?**

```
class Base{
   void methodA() {
      System.out.println("base - MethodA");
   }
}

class Sub extends Base {
   public void methodA(){
      System.out.println("sub - MethodA");
   }
   public void methodB(){
      System.out.println("sub - MethodB");
   }
   public static void main(String args[]){
      Base b=new Sub(); //1
      b.methodA(); //2
      b.methodB(); //3
   }
}
```

A. sub - MethodA and sub - MethodB
B. base - MethodA and sub - MethodB
C. Compile time error at //1
D. Compile time error at //2
E. Compile time error at //3

**Correct Answer:** E
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
The point to understand here is, b is declared to be a reference of class Base and methodB() is not defined in Base. So the compiler cannot accept the statement b.methodB() because it only verifies the validity of a call by looking at the declared class of the reference.

For example, the compiler is able to verify that b.methodA() is a valid call because class Base has method methodA. But it does not "bind" the call. Call binding is done at runtime by the jvm and the jvm looks for the actual class of object referenced by the variable before invoking the method.

**QUESTION 386**
**Given:**

```
public class Series {
    public static void main(String[] args) {
```

```
        int arr[] = {1, 2, 3};

        for(int var : arr) {
            int i = 1;
            while ( i <= var);
                System.out.println(i++);
        }
    }
}
```

**What is the result?**

A.
```
1
1
1
```

B.
```
1
2
3
```

C.
```
2
3
4
```

D. Compilation fails.

E. The loop executes infinite times.

**Correct Answer:** E
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 387**
**What two changes can you do, independent of each other, to make the following code compile:**

```
//assume appropriate imports
class PortConnector {
    public PortConnector(int port) {
        if (Math.random() > 0.5) {
            throw new IOException();
        }
        throw new RuntimeException();
    }
}

public class TestClass {
    public static void main(String[] args) {
         try {
             PortConnector pc = new PortConnector(10);
         } catch (RuntimeException re) {
             re.printStackTrace();
         }
    }
}
```

A. add throws IOException to the main method.

B. add throws IOException to PortConnector constructor.

C. add throws IOException to the main method as well as to PortConnector constructor.

D. Change RuntimeException to java.io.IOException.

E. add throws Exception to PortConnector constructor and change catch (RuntimeException re) to catch(Exception re) in the main method.

**Correct Answer:** CE

**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
IOException is a checked exception and since the PortConnector constructor throws IOException, this exception (or its superclass) must be present in the throws clause of the constructor.

Now, the main method has two options, either catch IOException (or whatever exception PortConnector throws) in its catch block (i.e. option 5) or put that exception in its throws clause (i.e. option 3).

**QUESTION 388**
**Given:**

```
class Main {

  public static void method(Object o) {
     System.out.print("Object ");
   }

   public static void method(Number n) {
     System.out.print("Number ");
   }

   public static void method(Long l) {
     System.out.print("Long ");
   }

   public static void method(Short n) {
     System.out.print("Short ");
   }
   public static void method(Integer i) {
     System.out.print("Integer ");
   }
   public static void main(String[] args) {
      char c = 'a';
      short y = 12;
      double d = 15.3d;
      method(c);
      method(y);
      method(d);
  }
}
```

**What is the result?**

A. Integer Short Number
B. Integer Integer Number
C. Compilation fails.
D. Number Short Number
E. Object Short Number
F. Object Integer Number
G. Object Short Object
H. Object Integer Object

**Correct Answer:** E
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 389**
**What will be the output when the following code is compiled and run?**

```
//in file Test.java
class E1 extends Exception{ }
class E2 extends E1 { }

class Test {
   public static void main(String[] args){
      try {
         throw new E2();
      }
      catch(E1 e) {
         System.out.println("E1");
      }
      catch(Exception e){
         System.out.println("E");
      }
      finally {
         System.out.println("Finally");
      }
   }
}
```

A. It will not compile.
B. It will print E1 and Finally.
C. It will print E1, E and Finally
D. It will print E and Finally.
E. It will print Finally.

**Correct Answer:** B
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
Since E2 is a sub class of E1, catch(E1 e) will be able to catch exceptions of class E2. Therefore, E1 is printed. Once the exception is caught the rest of the catch blocks at the same level (that is the ones associated with the same try block) are ignored. So E is not printed. finally is always executed (except in case of System.exit()), so Finally is also printed.

**QUESTION 390**
**Which line will print the string "MUM"?**

```
public class TestClass {
   public static void main(String args []) {
        String s = "MINIMUM";
        System.out.println(s.substring(4, 7));              //1
        System.out.println(s.substring(5));                 //2
        System.out.println(s.substring(s.indexOf('I', 3))); //3
        System.out.println(s.substring(s.indexOf('I', 4))); //4
   }
}
```

A. 1
B. 2
C. 3
D. 4
E. None of these.

**Correct Answer:** A
**Section: Working with Selected classes from the Java API**

**Explanation**

**Explanation/Reference:**
```
Line 2 => It will print UM.
Line 3 => It will print IMUM. as s.indexOf('I', 3) will return 3.
Line 4 => It will throw an exception as s.indexOf('I', 4) will return -1.
```

**QUESTION 391**
**Given:**

```
public class Msg {

   public static String doMsg(char x) {
       return "Good Day!";
   }

   public static String doMsg(int y) {
       return "Good Luck!";
   }

   public static void main(String[] args) {
       char x = 8;
       int z = '8';
       System.out.println(doMsg(x));
       System.out.println(doMsg(z));

   }

}
```

**What is the result?**

A.
```
Good Day!
Good Luck!
```
B.
```
Good Day!
Good Day!
```
C.
```
Good Luck!
Good Day!
```
D.
```
Good Luck!
Good Luck!
```
E. `Compilation fails`

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 392**
**Given:**

```
1. public class Twisty {
2.     { index = 1; }
3.     int index;
4.     public static void main(String[] args) {
5.         new Twisty().go();
6.     }
7.     void go() {
8.         int [][] dd = {{9,8,7}, {6,5,4}, {3,2,1,0}};
9.         System.out.println(dd[index++][index++]);
10.    }
11.}
```

**What is the result?**

A. 1
B. 2
C. 4
D. 6
E. 8
F. Compilation fails
G. An exception is thrown at runtime

**Correct Answer:** C
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 393**
**What will the following code print?**

```
public class Test {
   public static void testInts(Integer obj, int var){
       obj = var++;
       obj++;
   }
   public static void main(String[] args) {
       Integer val1 = new Integer(5);
       int val2 = 9;
       testInts(val1++, ++val2);
       System.out.println(val1+" "+val2);
   }
}
```

A. 10 9
B. 10 10
C. 6 9
D. 6 10
E. 5 11

**Correct Answer:** D
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 394**
**Given:**

```
class Bird {
  { System.out.print("b1 "); }

  public Bird() { System.out.print("b2 "); }
}

class Raptor extends Bird {
  static { System.out.print("r1 "); }

  public Raptor() { System.out.print("r2 "); }

  { System.out.print("r3 "); }
```

```
  static { System.out.print("r4 "); }
}

class Hawk extends Raptor {

  public static void main(String[] args) {
    System.out.print("pre ");
    new Hawk();
    System.out.println("hawk ");
  }

}
```

**What is the result?**

A. `pre b1 b2 r3 r2 hawk`
B. `pre b2 b1 r2 r3 hawk`
C. `pre b2 b1 r2 r3 hawk r1 r4`
D. `r1 r4 pre b1 b2 r3 r2 hawk`
E. `r1 r4 pre b2 b1 r2 r3 hawk`
F. `pre r1 r4 b1 b2 r3 r2 hawk`
G. `pre r1 r4 b2 b1 r2 r3 hawk`
H. `The order is not predictable`
I. `Compilation fails`

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 395**
**Given:**

```
public class Sphere {

     double radius;
     public double area;

     public Sphere(double r) { radius = r;}

     public double getRadius() { return radius;}
     public void setRadius(double r) { radius = r;}
     public double getArea() { return //*??*//}
}

class App {
      public static void main(String[] args) {
           Sphere s1 = new Sphere(17.4);
           s1.area = 4 * Math.PI * Math.pow(c1.getRadius(), 2);
      }
}
```

**The class is poorly encapsulated. You need to change the Sphere class to compute and return the area instead.**
**Which two modifications are necessary to ensure that the class is being properly encapsulated?**

A. Remove the area field.
B. Change the getArea( ) method as follows:

```
public double getArea ( ) {
    return 4 * Math.PI * Math.pow(radius, 2);
}
```

C. Change the getArea( ) method as follows:

```
public double getArea ( ) {
    area = 4 * Math.PI * Math.pow(radius, 2);
}
```

D. Change the access modifier of the setRadius ( ) method to be protected.

**Correct Answer:** AB
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 396**
**Given the code fragment:**

```
public class Test {
  public static void main(String[] args) {
     String[] arr = {"A", "B", "C", "D"};
     for (int i = 0; i < arr.length; i++) {
        System.out.print(arr[i] + " ");
        if (arr[i].equals("C")) {
            continue;
        }
        System.out.println("Work done");
        break;
     }
  }
}
```

**What is the result?**

A. `A B C Work done`
B. `A B C D Work done`
C. `A Work done`
D. `Compilation fails`

**Correct Answer:** C
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 397**
**Given:**

```
String s = "1234";
StringBuilder sb = new StringBuilder(s.substring(2).concat("56").replace
("7","6"));
System.out.println(sb.append("89").insert(3,"x"));
```

**What is the result?**

A. `34x5689`
B. `345x689`
C. `345x789`

D. 23x45689
E. 23x45789
F. Compilation fails.

**Correct Answer:** B
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 398**
**Given**

```
class Mixer {
    Mixer() { }
    Mixer(Mixer m) { m1 = m; }
    Mixer m1;
    public static void main(String[] args) {
        Mixer m2 = new Mixer();
        Mixer m3 = new Mixer(m2); m3.go();
        Mixer m4 = m3.m1; m4.go();
        Mixer m5 = m2.m1; m5.go();
    }
    void go() { System.out.print("hi "); }
}
```

**What is the result?**

A. hi
B. hi hi
C. hi hi hi
D. Compilation fails
E. hi followed by an Exception
F. hi hi followed by an Exception

**Correct Answer:** F
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 399**
**Given:**

```
#1
package handy.dandy;

public class KeyStroke {
   public void typeExclamation() {
     System.out.println("!")
   }
}


#2
package handy;                                  /* Line 1 */

public class Greet {                            /* Line 2 */
   public static void main(String[] args) {     /* Line 3 */
      String greeting = "Hello";                /* Line 4 */
```

```
      System.out.print(greeting);             /* Line 5 */
      Keystroke stroke = new Keystroke();     /* Line 6 */
      stroke.typeExclamation();               /* Line 7 */
   }                                          /* Line 8 */
}                                             /* Line 9 */
```

**What three modifications, made independently, made to class greet, enable the code to compile and run?**

A. `Line 6 replaced with handy.dandy.keystroke stroke = new KeyStroke ( );`
B. `Line 6 replaced with handy.*.KeyStroke = new KeyStroke ( );`
C. `Line 6 replaced with handy.dandy.KeyStroke stroke = new handy.dandy.KeyStroke();`
D. `import handy.*; added before line 1`
E. `import handy.dandy.*; added after line 1`
F. `import handy.dandy,KeyStroke; added after line 1`
G. `import handy.dandy.KeyStroke.typeException(); added before line 1`

**Correct Answer:** CEF
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 400**
**Given:**

```
class A {
   int a = 5;
   String doA() {
      return "a1";
   }
   protected static String doA2 () {
      return "a2";
   }
}

class B extends A {
   int a = 7;
   String doA() {
      return "b1";
   }
   public static String doA2() {
      return "b2";
   }
   void go() {
      A myA = new B();
      System.out.print(myA.doA() + myA.doA2() + myA.a);
   }
   public static void main (String[] args) {
      new B().go();
   }
}
```

**Which three values will appear in the output?**

A. `5`
B. `7`
C. `a1`
D. `a2`
E. `b1`

F. b2

**Correct Answer:** ADE
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 401**
**Given**

```
class Plane {
    static String s = "-";
    public static void main(String[] args) {
        new Plane().s1();
        System.out.println(s);
    }
    void s1() {
        try {
           s2();
        }
        catch (Exception e) {
           s += "c";
        }
    }
    void s2() throws Exception {
        s3();
        s += "2";
        s3();
        s += "2b";
    }
    void s3() throws Exception {
        throw new Exception();
    }
}
```

**What is the result?**

A. -
B. -c
C. -c2
D. -2c
E. -c22b
F. -2c2b
G. -2c2bc
H. Compilation fails

**Correct Answer:** B
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 402**
**Given**

```
public class OrtegorumFunction {
    public int computeDiscontinuous(int x) {
        int r = 1;
        r += x;
```

```
        if ((x > 4) && (x < 10)) {
          r += 2 * x;
        } else if (x <= 4) {
          r += 3 * x;
        } else {
          r += 4 * x;
        }
        r += 5 * x;
        return r;
    }
    public static void main(String [] args) {
        OrtegorumFunction o = new OrtegorumFunction();
        System.out.println("OF(11) is: " + o.computeDiscontinuous(11));
    }
}
```

**What is the result?**

A. `OF(11) is: 45`
B. `OF(11) is: 56`
C. `OF(11) is: 89`
D. `OF(11) is: 111`
E. `Compilation fails`
F. `An exception is thrown`

**Correct Answer:** D
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 403**
**Given:**

```
class Mineral { }

class Gem extends Mineral { }

class Miner {
    static int x = 7;
    static String s = null;
    public static void getWeight(Mineral m) {
        int y = 0 / x;
        System.out.print(s + " ");
    }
    public static void main(String[] args) {
        Mineral[] ma = {new Mineral(), new Gem()};
        for(Object o : ma)
            getWeight((Mineral) o);
    }
}
```

**and the command line**
`java Miner.java`

**What is the result?**

A. `null`
B. `null null`
C. `A ClassCastException is thrown`
D. `A NullPointerException is thrown`

E. A NoClassDefFoundError is thrown
F. An ArithmeticException is thrown
G. An IllegalArgumentException is thrown
H. An ArrayIndexOutOfBoundsException is thrown

**Correct Answer:** B
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 404**
**Given**

```
class Emu {
    static String s = "-";

    public static void main(String[] args) {
        try {
            throw new Exception();
        } catch (Exception e) {
        try {
            try { throw new Exception();
            } catch (Exception ex) { s += "ic "; }
            throw new Exception(); }
            catch (Exception x) { s += "mc "; }
            finally { s += "mf "; }
        } finally { s += "of "; }
        System.out.println(s);
    }
}
```

**What is the result?**

A. -ic of
B. -mf of
C. -mc mf
D. -ic mf of
E. -ic mc mf of
F. -ic mc of mf
G. Compilation fails

**Correct Answer:** E
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 405**
**Assuming that the following method will always be called with a phone number in the format ddd-ddd-dddd (where d stands for a digit), what can be inserted at //1 so that it will return a String containing "xxx-xxx-"+dddd, where dddd represents the same four digits in the original number?**

```
public static String hidePhone(String fullPhoneNumber) {
   //1 Insert code here
}
```

**Please select 2 options**

A.
```
String mask = "xxx-xxx-";
mask.append(fullPhoneNumber.substring(8));
return mask;
```

B.
```
return new StringBuilder("xxx-xxx-")+fullPhoneNumber.substring(8);
```

C.
```
return new StringBuilder(fullPhoneNumber).replace(0, 7, "xxx-xxx-").toString
();
```

D.
```
return "xxx-xxx-"+fullPhoneNumber.substring(8, 12);
```

**Correct Answer:** BD
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
Remember that String class doesn't have append (and insert) method because a String cannot be mutated.

For all of the methods in String and StringBuilder that take two int parameters for specifying a range, remember that the first index is included but the last index is not.

For example, as in this case, the arguments given are 0 and 7, which means it will include the characters with index 0 to 6, that is, a total of 7 characters 0, 1, 2, 3, 4, 5, and 6. Therefore, this will actually produce "xxx-xxx--dddd". The same pattern is used for almost all other methods in standard java library classes. The first index is included but the last one is not.

**QUESTION 406**
**Given the class definitions:**

```
class Alpha {
   public String doStuff(String msg) {
       return msg;
   }
}

class Beta extends Alpha {
   public String doStuff(String msg) {
       return msg.replace('a', 'e');
   }
}

class Gamma extends Beta {
   public String doStuff(String msg) {
       return msg.substring(2);
   }
}
```

**And the code fragment of the main() method,**

```
List<Alpha> strs = new ArrayList<>();
strs.add(new Alpha());
strs.add(new Beta());
strs.add(new Gamma());
for (Alpha t: strs)  {
   System.out.println(t.doStuff("Java"));
}
```

**What is the result?**

A.
```
Java
Java
Java
```

B.
```
Java
```

```
Jeve
va
```

C.
```
Java
Jeve
ve
```

D. `Compilation fails`

**Correct Answer:** B
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 407**
**Consider the following program:**

```
interface Side {
     String getSide();
}

class Head implements Side {
     public String getSide() {
         return "Head ";
     }
}

class Tail implements Side {
      public String getSide() {
          return "Tail ";
      }
}

class Coin {
   public static void overload(Head side) {
      System.out.print(side.getSide());
   }
   public static void overload(Tail side) {
      System.out.print(side.getSide());
   }
   public static void overload(Side side) {
      System.out.print("Side ");
   }
   public static void overload(Object side) {
      System.out.print("Object ");
   }

   public static void main(String []args) {
        Side firstAttempt = new Head();
        Tail secondAttempt = new Tail();
        overload(firstAttempt);
        overload((Object)firstAttempt);
        overload(secondAttempt);
        overload((Side)secondAttempt);
   }
}
```

**What is the output of this program when executed?**

A. `Head Head Tail Tail`
B. `Side Object Tail Side`
C. `Head Object Tail Side`
D. `Side Head Tail Side`

**Correct Answer:** B
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
Overloading is based on the static type of the objects (while overriding and runtime resolution resolves to the dynamic type of the objects). Here is how the calls to the overload() method are resolved:

• overload(firstAttempt); --> firstAttempt is of type Side, hence it resolves to overload(Side).
• overload((Object)firstAttempt); -> firstAttempt is casted to Object, hence it resolves to overload(Object).
• overload(secondAttempt); -> secondAttempt is of type Tail, hence it resolves to overload(Tail).
• overload((Side)secondAttempt); -> secondAttempt is casted to Side, hence it resolves to overload(Side).

**QUESTION 408**
**Given the classes:**

```
class Pupil {
   String name = "unknown";
   public String getName() {
     return name;
   }
}

class John extends Pupil {
   String name = "John";
}

class Harry extends Pupil {
   String name = "Harry";
   public String getName() {
     return name;
   }
}

public class Director {
   public static void main(String[] args) {
      Pupil p1 = new John();
      Pupil p2 = new Harry();
      System.out.print(p1.getName() + " ");
      System.out.print(p2.getName());
  }
}
```

**What is the result?**

A. `John Harry`
B. `unknown Harry`
C. `john unknown`
D. `unknown unknown`
E. `Compilation fails.`
F. `An exception is thrown at runtime.`

**Correct Answer:** B
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 409**
**Given the following declaration:**

```
int[][] twoD = { { 1, 2, 3} , { 4, 5, 6, 7}, { 8, 9, 10 } };
```

**What will the following lines of code print?**

```
System.out.println(twoD[1].length);
System.out.println(twoD[2].getClass().isArray());
System.out.println(twoD[1][2]);
```

A.
```
4
true
6
```
B.
```
3
true
3
```
C.
```
3
false
3
```
D.
```
4
false
6
```

**Correct Answer:** A
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**
In Java, array numbering starts from 0. So in this case, twoD is an array containing 3 other arrays.
twoD[0] is { 1, 2, 3} , twoD[1] is { 4, 5, 6, 7}, and twoD[2] is { 8, 9, 10 }.

Thus, twoD[1].length is 4 and twoD[1][2] is the third element in { 4, 5, 6, 7}, which is 6.

In Java, arrays are just like regular Objects and arrays of different types have different class names. For example, the class name of an int[] is [I and the class name for int[][] is [[I.

For array classes, the isArray() method of a Class returns true. For example, twoD.getClass().isArray() will return true.

**QUESTION 410**
**Given**

```
class Uber {
   static int y = 2;
   Uber(int x) {
      this();
      y = y * 2;
   }
   Uber() {
      y++;
   }
}
class Minor extends Uber {
   Minor() {
       super(y);
       y = y + 3;
   }
   public static void main(String [] args) {
      new Minor();
      System.out.println(y);
   }
}
```

**What is the result?**

A. 6
B. 7
C. 8
D. 9
E. Compilation fails
F. An exception is thrown

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 411**
**What will be the output of the following lines?**

```
System.out.println("" + 5 + 6);   // 1
System.out.println(5 + "" + 6);   // 2
System.out.println(5 + 6 + "");   // 3
System.out.println(5 + 6);        // 4
```

A.
```
56
56
11
11
```
B.
```
11
56
11
11
```
C.
```
56
56
56
11
```
D.
```
56
56
56
56
```
E.
```
56
56
11
56
```

**Correct Answer:** A
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
```
In line 1, "" + 5 + 6 => "5"+6 => "56"
In line 2, 5 + "" +6  => "5"+6 => "56"
In line 3, 5 + 6 +"" => 11+"" => "11"
In line 4, 5 + 6 => 11 => "11"
```

**QUESTION 412**
**What will the following program print?**

```
public class TestClass {
   static String str = "Hello World";
   public static void changeIt(String s){
       s = "Good bye world";
   }
```

```
   public static void main(String[] args){
       changeIt(str);
       System.out.println(str);
   }
}
```

A. "Hello World"
B. "Good bye world"
C. It will not compile.
D. It will throw an exception at runtime.
E. None of the above

**Correct Answer:** A
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 413**
**What will the following code print when run?**

```
public class Test {
   static String s = "";
   public static void m0(int a, int b) {
       s += a;
       m2();
       m1(b);
   }
   public static void m1(int i) {
       s += i;
   }
   public static void m2() {
       throw new NullPointerException("aa");
   }
   public static void m() {
       m0(1, 2);
       m1(3);
   }
   public static void main(String args[]) {
       try {
          m();
       } catch (Exception e) {
       }
       System.out.println(s);
   }
}
```

A. 1
B. 12
C. 123
D. 2
E. It will throw exception at runtime.
F. Compilation fails

**Correct Answer:** A
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
Try to follow the control flow:

1. m() calls m0(1, 2).
2. m0(1, 2) first executes s += 1 (so s is now 1) and then calls m2().
3. Now, m2() throws an exception which is not caught by m2() so it is propagated back to m0(1, 2). Since, within m0 method, the call to m2() is not wrapped in a try catch block, this exception further propagates up to m().
( The next line in m0(1, 2), which is m1(2), is not executed ).
4. Again, m() also does not have the try catch block so the same exception is further propagated up to the main() method. (The next line in m(), which is a call to m1(3) is not called).
4. In main method, the call to m() is wrapped in a try catch block and so the exception is handled here.
5. Finally, s stays as "1".

The point to note here is that if you do not catch an exception, it is propagated up the stack of method calls until it is handled. If nobody handles it, the JVM handles that exception and kills the thread. If that thread is the only user thread running, the program ends.

**QUESTION 414**
**Which of the following declarations are valid? Select 4 options**

A. `float var1 = 8.3;`
B. `float var2 = 59e11;`
C. `float var3 = -10;`
D. `float var4 = 0x0321;`
E. `float var5 = 9;`
F. `int var6 = 3e9;`
G. `int var7 = 0xF0F0L;`
H. `double var8 = 0b000101D;`
I. `long var9 = F0F0;`

**Correct Answer:** CDEI
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
```
Error: incompatible types: possible lossy conversion from double to float
    float var2 = 59e11;

Error: incompatible types: possible lossy conversion from double to int
    int var6 = 3e9;

error: incompatible types: possible lossy conversion from long to int
    int i2 = 0xF0F0L;
```

**QUESTION 415**
**Given the following code:**

```
3. public class Honcho {
4.    static boolean b1;
5.    static int z = 7;
6.    static Long y;
7.    public static void main(String[] args) {
8.      for(int i = 0; i < 4; i++)
9.         go(i);
10.   }
11.   static void go(int x) {
12.     try {
13.        if((x == 0) && (!b1 && z == 7)) System.out.print("0 ");
14.        if(x < 2 ^ x < 10) System.out.print("1 ");
15.        if((x == 2) && (y == null | (y == 0))) System.out.print("2 ");
16.        if(z <= (x + 4)) System.out.print("3 ");
17.     }
```

```
18.     catch(Exception e) { System.out.print("e "); }
19. } }
```

**What is the result?**

A. `0 1 2 3`
B. `1 e 1 3`
C. `0 1 e 1 3`
D. `0 1 1 1 1 3`
E. `1 1 1 2 1 3`
F. `0 1 1 1 2 1 3`
G. `Compilation fails.`

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 416**
**Given this code in a method:**

```
3. int y, count = 0;
4. for(int x = 3; x < 6; x++) {
5.   try {
6.     switch(x) {
7.       case 3: count++;
8.       case 4: count++;
9.       case 7: count++;
10.      case 9: { y = 7 / (x - 4); count += 10; }
11.    }
12.   } catch (Exception ex) { count++; }
13. }
14. System.out.println(count);
```

**What is the result?**

A. `2`
B. `15`
C. `16`
D. `25`
E. `26`
F. `Compilation fails`
G. `An exception is thrown with no other output.`

**Correct Answer:** C
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 417**
**Given the following code:**

```
2. public class Swanky {
3.    Swanky s;
4.    public static void main(String[] args) {
5.       Swanky s1 = new Swanky();
```

```
6.       s1.s = new Swanky();
7.       go(s1.s);
8.       Swanky s2 = go(s1);
9.       if(s2.s.equals(s1)) System.out.print("1 ");
10.      if(s2.equals(s1)) System.out.print("2 ");
11.      if(s1.s.equals(s1)) System.out.print("3 ");
12.      if(s2.s.equals(s2)) System.out.print("4 ");
13.   }
14.   static Swanky go(Swanky s) {
15.      Swanky gs = new Swanky();
16.      gs.s = s;
17.      return gs;
18. } }
```

**What is the result?**

A. 1
B. 1 2
C. 1 2 4
D. Compilation fails.
E. "1 ", followed by an exception.
F. "1 2 ", followed by an exception.

**Correct Answer:** A
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 418**
**What will the following code print?**

```
public class Test {
   public static void stringTest(String s) {
       s.replace('h', 's');
   }
   public static void stringBuilderTest(StringBuilder s){
       s.append("o");
   }
   public static void main(String[] args){
       String s = "hell";
       StringBuilder sb = new StringBuilder("well");
       stringTest(s);
       stringBuilderTest(sb);
       System.out.println(s + sb);
   }
}
```

A. sellwello
B. hellwello
C. hellwell
D. sellwell
E. None of these.

**Correct Answer:** B
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
A String is immutable while a StringBuilder is not. So in stringTest(), "hell".replace('h', 's') will produce a new String "sell" but will not affect

the original String that was passed to the method. However, the append() method of StringBuilder appends to the original String object. So, "well" becomes "wello".

**QUESTION 419**
**Given the code fragment:**

```
public static void main(String[] args) {
    String[] arr = {"A", "B", "C", "D"};
    for (int i = 0; i < arr.length; i++) {
        System.out.print(arr[i] + " ");
        if (arr[i].equals("A")) {
            continue;
        }
        System.out.println("Work done");
        break;
    }
}
```

**What is the result?**

A. `A B C Work done`
B. `A B C D Work done`
C. `A Work done`
D. `Compilation fails`
E. `A B Work done`

**Correct Answer:** E
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 420**
**How many objects have been created by the time the main method reaches its end in the following code?**

```
public class Noobs {
  public Noobs() {
       try {
           throw new MyException();
       } catch(Exception e) {
       }
  }

  public static void main(String[] args) {
       Noobs a = new Noobs();
       Noobs b = new Noobs();
       Noobs c = a;
  }
}

class MyException extends Exception {
}
```

A. 2
B. 3
C. 4
D. 5
E. 6

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**
When a Noobs object is created, a MyException object is also created. Therefore a total of 4 objects are created. The line Noobs c = a; just assigns an existing Noobs object to c. No new object is created.

Note: Some candidates have reported getting a similar question.
The question is ambiguous because two Class objects (one for Noobs and one for MyException) are also created. If you consider those, then the answer would be 6. Further, several Thread objects are also created (although not directly by this code.) Since this is out of scope for the exam, it is best to ignore these kind of objects and consider only the objects created directly by the code.

**QUESTION 421**
**Given the following code**:

```
3. public class Spock {
4.    public static void main(String[] args) {
5.       int mask = 0;
6.       int count = 0;
7.       if( (5<7) || (++count < 10) | mask++ < 10 ) mask = mask + 1;
8.       if( (6 > 8) ^ false) mask = mask + 10;
9.       if( !(mask > 1) && ++count > 1) mask = mask + 100;
10.      System.out.println(mask + " " + count);
11.   }
12. }
```

**Which two answers are true about the value of mask and the value of count at line 10? (Choose two.)**

A. mask is 0
B. mask is 1
C. mask is 2
D. mask is 10
E. mask is greater than 10
F. count is 0
G. count is 1

**Correct Answer:** BG
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 422**
**What will be the output of the following program:**

```
public class TestClass {
    public static void main(String args[]) {
        try {
           m1();
        } catch(IndexOutOfBoundsException e) {
             System.out.print("1 ");
             throw new NullPointerException();
        } catch(NullPointerException e){
             System.out.print("2 ");
             return;
        } catch (Exception e) {
```

```
            System.out.print("3 ");
        } finally {
            System.out.print("4 ");
        }
        System.out.print("END");
    }

    static void m1(){
        System.out.print("m1 Starts ");
        throw new IndexOutOfBoundsException( "Big Bang " );
    }
}
```

A. The program will print: m1 Starts 1 4.
B. The program will print: m1 Starts 1 4 followed by an Exception.
C. The program will print: m1 Starts 1 2.
D. The program will print m1 Starts 1 2 4.
E. The program will print: m1 Starts 1 4 END.

**Correct Answer:** B
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
The IndexOutOfBoundsException is handled by the first catch block. Inside this block, a new NullPointerException is thrown. As this exception is not thrown inside the try block, it will not be caught by any of the remaining catch blocks. It will actually be sent to the caller of the main() method after the finally block is executed. (Hence '4' in the output.) The code that prints END is never reached, since the NullPointerException remains uncaught after the execution of the finally block. At the end a stack trace for the NullPointerException will be printed.

**QUESTION 423**
**Which digits and in what order will be printed when the following program is run?**

```
public class TestClass{
   public static void main(String args[]){
      int k = 0;
      try{
         int i = 5/k;
      }
      catch (ArithmeticException e){
         System.out.print("1 ");
      }
      catch (RuntimeException e){
         System.out.print("2 ");
         return ;
      }
      catch (Exception e){
         System.out.print("3 ");
      }
      finally{
         System.out.print("4 ");
      }
      System.out.print("5 ");
   }
}
```

A. The program will print 5.
B. The program will print: 1 4.
C. The program will print: 1 2 4.

D. The program will print: 1 4 5.
E. The program will print 1 2 4 5.

**Correct Answer:** D
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
Division by 0 throws a java.lang.ArithmeticException, which is a RuntimeException. This is caught by the first catch clause because it is the first block that can handle ArithmeticException. This prints 1. Now, as the exception is already handled, control goes to finally which prints 4 and then the try/catch/finally ends and 5 is printed. Remember : finally is always executed even if try or catch return; (Except when there is System.exit() in try or catch.)

**QUESTION 424**
**Given:**

```
int x = 0;
int y = 10;
do {
   y--;
   ++x;
} while (x < 5);
System.out.print(x + "," + y);
```

**What is the result?**

A. 5,6
B. 5,5
C. 6,5
D. 6,6

**Correct Answer:** B
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 425**
**Given the following code:**

```
for (int i = 0; i < 10; i++) {
   if (i == 5)
      continue;
   System.out.print(i);
}
```

**What is the output?**

A. Compilation fails
B. An exception is thrown at runtime
C. 0123456789
D. 012346789

**Correct Answer:** D
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 426**
**What all gets printed when the following program is compiled and run.**

```
public class test {
    public static void main(String args[]) {
        int i=0, j=2;
        do {
          i=++i;
          j--;
        } while(j>0);
        System.out.println(i);
   }
}
```

A. 0
B. 1
C. 2
D. Compilation fails
E. An exception is thrown

**Correct Answer:** C
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 427**
**Given the following code:**

```
boolean isTrue = true;
for (int i=0; i <5; i++) {
   while(isTrue) {
     System.out.println("Hello");
     break;
   }
   System.out.println("Outer loop");
}
System.out.println("Good-Bye");
```

**What is the output?**

A. Compilation fails
B. An exception is thrown at runtime
C.
```
Hello
Outer loop
Hello
Outer loop
Hello
Outer loop
Hello
Outer loop
Hello
Outer loop
Good-Bye
```
D.
```
Hello
Good-Bye
```

**Correct Answer:** C

**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 428**
**Given:**

```
public void go() {
   String o = "";
   z:
   for(int x = 0; x < 3; x++) {
      for(int y = 0; y < 2; y++) {
         if(x==1) break;
         if(x==2 && y==1) break z;
         o = o + x + y;
      }
   }
   System.out.println(o);
}
```

**What is the result when the go() method is invoked?**

A. `00`
B. `0001`
C. `000120`
D. `00012021`
E. `Compilation fails.`
F. `An exception is thrown at runtime`

**Correct Answer:** C
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 429**
**Given the following code:**

```
outer:
   for (int i=0; i<5; i++) {
     for(int j=0; j<5; j++) {
        System.out.println("Hello");
        continue outer;
     }
     System.out.println("outer");
   }
   System.out.println("Good-Bye");
```

**What is the output?**

A. `Compilation fails`
B. `An exception is thrown at runtime`
C.
```
Hello
Hello
Hello
Hello
Hello
outer
Hello
```

```
Hello
Hello
Hello
Hello
outer
Hello
Hello
Hello
Hello
Hello
outer
Hello
Hello
Hello
Hello
Hello
outer
Hello
Hello
Hello
Hello
Hello
outer
Good-Bye
```

D.
```
Hello
Hello
Hello
Hello
Hello
Good-Bye
```

**Correct Answer:** D
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 430**
**Given the following code:**

```
boolean isTrue = true;
outer:
for (int i=0; i <5; i++) {
   while(isTrue) {
    System.out.println("Hello");
    break outer;
   }
   System.out.println("Outer loop");
}
System.out.println("Good-Bye");
```

**What is the output?**

A. `Compilation fails`
B. `An exception is thrown at runtime`
C.
```
Hello
Outer loop
Hello
Outer loop
Hello
Outer loop
Hello
Outer loop
```

```
Hello
Outer loop
Good-Bye
```

D.
```
Hello
Good-Bye
```

**Correct Answer:** D
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 431**
**Which two are valid array declaration?**

A. `Object array[];`
B. `Boolean array[3];`
C. `int[] array;`
D. `Float[2] array;`

**Correct Answer:** AC
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 432**
**Given the code fragment:**

```
String[] colors = {"red", "blue", "green", "yellow", "maroon", "cyan"};
```

**Which code fragment prints blue, cyan, ?**

A.
```
for (String c : colors) {
      if (c.length() != 4) {
         continue;
      }
      System.out.print(c + ", ");
}
```

B.
```
for (String c : colors[]) {
      if (c.length() <= 4) {
          continue;
      }
      System.out.print(c + ", ");
}
```

C.
```
for (String c : String[] colors) {
      if (c.length() >= 3) {
           continue;
      }
      System.out.print(c + ", ");
}
```

D.
```
for (String c : colors) {
      if (c.length() != 4) {
           System.out.print(c + ", ");
           continue;
      }
}
```

**Correct Answer:** A
**Section: Creating and Using Arrays**

**Explanation**

**Explanation/Reference:**

**QUESTION 433**
**Given the code fragment:**

```
class X {
   static void m (int[] i) {
     i[0] += 7;
   }

   public static void main (String[] args) {
     int[] j = new int[1];
     j[0] = 12;
     m(j);
     System.out.println(j[0]);
   }
}
```

**What is the result?**

A. 7
B. 12
C. 19
D. Compilation fails.
E. An exception is thrown at runtime

**Correct Answer:** C
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 434**
**Given the code fragment:**

```
public class Test {
   public static void main (String[] args) {
     char[] arr = {97, '\t', 'e', '\n', 'i', '\t', 'o'};
     for (char var: arr) {
        System.out.print(var);
     }
     System.out.print("\nThe length is :" + arr.length);
   }
}
```

**What is the result?**

A.
```
a     e
The length is : 2
```
B.
```
a     e
i     o
The length is : 4
```
C.
```
aeio
The length is : 4
```
D.
```
a     e
i     o
The length is : 7
```
E. Compilation fails

**Correct Answer:** D
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 435**
**Given the code fragment:**

```
public class Test2 {
   public static void main (String[] args) {
     int ar1[] = {2, 4, 6, 8};
     int ar2[] = {1, 3, 5, 7, 9};
     ar2 = ar1;
     for (int e2 : ar2) {
        System.out.print(" " + e2);
     }
   }
}
```

**What is the result?**

A. `2 4 6 8`
B. `2 4 6 8 9`
C. `1 3 5 7`
D. `1 3 5 7 9`
E. `Compilation fails`
F. `An exception is thrown at runtime`

**Correct Answer:** A
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 436**
**Given the code fragment:**

```
public class Test2 {
   public static void doChange(int[] arr) {
     for (int pos=0; pos < arr.length; pos++) {
        arr[pos] = arr[pos] + 1;
     }
  }
  public static void main (String[] args) {
     int[] arr = {10, 20, 30};
     doChange(arr);
     for (int x : arr) {
         System.out.print(x + ", ");
     }
     doChange(arr);
     System.out.print(arr[0] + ", " + arr[1] + ", " + arr[2]);
  }
}
```

**What is the result?**

A. `11, 21, 31, 11, 21, 31`
B. `11, 21, 31, 12, 22, 32`

C. `12, 22, 32, 12, 22, 32`
D. `10, 20, 30, 10, 20, 30`

**Correct Answer:** B
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 437**
**Which two are valid declarations of a two-dimensional array?**

A. `int [][] array2D;`
B. `int [2][2] array2D;`
C. `int array2D[];`
D. `int[] array2D[];`
E. `int[][] array2D[];`

**Correct Answer:** AD
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 438**
**Given the code fragment:**

```
int [] [] array2D = {{0, 1, 2}, {3, 4, 5, 6}};
System.out.print (array2D[0].length + "" );
System.out.print(array2D[1].getClass(). isArray() + "");
System.out.println (array2D[0][1]);
```

**What is the result?**

A. `3false1`
B. `2true3`
C. `2false3`
D. `3true1`
E. `3false3`
F. `2true1`
G. `2false1`

**Correct Answer:** D
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 439**
**Given the code fragment:**

```
int [] [] array = {{0}, {0, 1}, {0, 2, 4}, {0, 3, 6, 9}, {0, 4, 8, 12, 16}};
System.out.println(array [4] [1]);
System.out.println(array[1][4]);
```

**What is the result?**

A. `4 null`
B. `null 4`
C. `An IllegalArgumentException is thrown at run time`
D. `4 An ArrayIndexOutOfBoundException is thrown at run time`

**Correct Answer:** D
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 440**
**Given:**

```
public class DoCompare1 {
    public static void main(String[] args) {
       String[] table = {"aa", "bb", "cc"};
       for (String ss: table) {
          int ii = 0;
          while (ii < table.length) {
             System.out.println(ss + ", " + ii); ii++;
          }
       }
   }
}
```

**How many times is 2 printed as a part of the output?**

A. `Zero`
B. `Once`
C. `Twice`
D. `Thrice`
E. `Compilation fails`

**Correct Answer:** D
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 441**
**Given:**

```
public class DoBreak1 {
    public static void main(String[] args) {
        String[] table = {"aa", "bb", "cc", "dd"};
        for (String ss: table) {
           if ( "bb".equals(ss)) {
                continue;
           }
           System.out.println(ss);
           if ( "cc".equals(ss)) {
                 break;
           }
         }
     }
}
```

**What is the result?**

A. `aa cc`
B. `aa bb cc`
C. `cc dd`
D. `cc`
E. `Compilation fails`

**Correct Answer:** A
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 442**
**What should keyword1 and keyword2 be respectively, in oreder to produce output 2345?**

```
int [] array = {1,2,3,4,5};
for (int i: array) {
   if ( i < 2) {
     keyword1 ;
   }
   System.out.println(i);
   if ( i == 3) {
      keyword2 ;
   }
}
```

A.
```
continue;
break;
```
B.
```
break;
break;
```
C.
```
break;
continue;
```
D.
```
continue;
continue;
```

**Correct Answer:** D
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 443**
**Given the code fragment:**

```
String[] cartoons = {"tom", "jerry", "micky", "tom"};
 int counter = 0;
```

**And,**

```
if ("tom".equals(cartoons[0])) {
    counter++;
} else if ("tom".equals(cartoons[1])) {
    counter++;
} else if ("tom".equals(cartoons[2])) {
    counter++;
} else if ("tom".equals(cartoons[3])) {
    counter++;
}
System.out.print(counter);
```

**What is the result?**

A. `1`
B. `2`
C. `4`
D. `0`

**Correct Answer:** A
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 444**
**Given**

```
public class App {
   public static void main(String[] args) {
      char[] arr = {'A', 'e', 'I', 'O', 'u'};
      int count = 0;
      for (int i = 0; i < arr.length; i++) {
         switch (arr[i]) {
            case 'A':
               continue;
            case 'a':
               count++;
               break;
            case 'E':
               count++;
               break;
            case 'I':
               count++;
               continue;
            case 'O':
               count++;
               break;
            case 'U':
               count++;
          }
       }
       System.out.print("Total match found: " + count);
   }
}
```

**What is the result?**

A. `Total match found: 1`
B. `Total match found: 2`
C. `Total match found: 3`
D. `Total match found: 5`

**Correct Answer:** B
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 445**
**Given the code fragment:**

```
5.  // insert code here
6.
7.  arr[0] = new int[3];
8.  arr[0][0] = 1;
9.  arr[0][1] = 2;
10. arr[0][2] = 3;
11.
12. arr[1] = new int[4];
13. arr[1][0] = 10;
14. arr[1][1] = 20;
15. arr[1][2] = 30;
16. arr[1][3] = 40;
```

**Which two statements, when inserted independently at line 5, enable the code to compile and run without problems?**

A. `int [][] arr = null;`
B. `int [][] arr = new int[2];`
C. `int [][] arr = new int[2][];`
D. `int [][] arr = new int[][4];`
E. `int [][] arr = new int[2][0];`
F. `int [][] arr = new int[0][4];`

**Correct Answer:** CE
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 446**
**Given the code fragment:**

```
public class Test3 {
  public static void main(String[] args) {
    double[] array = {10, 20.23, 'c', 300.00f};
    for (double d : array) {
       d = d + 10;
       System.out.print(d + " ");
    }
  }
}
```

**What is the result?**

A. `20.0 30.23 109.0 310.0`
B. `20.0 30.23 c10 310.0`
C. `Compilation fails.`
D. `An exception is thrown at runtime.`

**Correct Answer:** A
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 447**
**Given the following code fragment:**

```
if (value >= 0) {
   if (value != 0)
```

```
      System.out.print("The ");
   else
      System.out.print("Quick ");

   if (value < 10)
      System.out.print("brown ");

   if (value > 30)
      System.out.print("fox ");
   else if (value < 50)
      System.out.print("jumps ");
   else if (value < 10)
      System.out.print("over ");
   else
      System.out.print("the ");

   if (value > 10)
      System.out.print("lazy ");
}
else {
   System.out.print("dog ");
}
System.out.print("... ");
```

**What is the result if the integer value is 33?**

A. `The fox jump lazy ...`
B. `The fox lazy ...`
C. `Quick fox over lazy ...`
D. `Quick fox the ....`

**Correct Answer:** B
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 448**
**Given:**

```
public class ArithmeticResultsOutput {
   public static void main(String[] args) {
      int i = 0;
      int j = 0;
      if (i++ == ++j) {
        System.out.println("True: i=" + i + ", j=" + j);
      }
      else {
        System.out.println("False: i=" + i + ", j=" + j);
      }
   }
}
```

**What will be printed to standard out?**

A. `True: i=0, j=1`
B. `True: i=1, j=1`
C. `False: i=0, j=1`
D. `False: i=1, j=1`

**Correct Answer:** D

**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 449**
**Given:**

```
public class Dinner {
   public static void main(String[] args) {
      boolean isKeeperFish = false;
      if (isKeeperFish = true) {
         System.out.println("Fish for dinner");
      }
      else {
         System.out.println("Take out for dinner");
      }
   }
}
```

**What will be the result of the application's execution?**

A. A compilation error will occur
B. An exception is thrown at runtime
C. Fish for dinner will be printed
D. Take out for dinner will be printed

**Correct Answer:** C
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 450**
**What is the result of executing the following fragment of code:**

```
boolean b1 = false;
boolean b2 = false;
if (b2 != b1 = !b2)
{
    System.out.println("true");
}
else
{
    System.out.println("false");
}
```

A. Compile time error
B. It will print true
C. It will print false
D. Runtime error
E. It will print nothing

**Correct Answer:** A
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 451**
**Given:**

```
public class BooleanResultsOutput {
   public static void main(String[] args) {
      boolean booleanValue1 = true;
      boolean booleanValue2 = false;
      System.out.print(!(booleanValue1 & !booleanValue2) + ", ");
      System.out.print(!(booleanValue1 | !booleanValue2) + ", ");
      System.out.print(!(booleanValue1 ^ !booleanValue2));
   }
}
```

**What is the result?**

A. `false, false, true`
B. `false, true, true`
C. `true, false, false`
D. `true, true, true`

**Correct Answer:** A
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 452**
**Given:**

```
class Hexy {
   public static void main(String[] args) {
      Integer i = 42;
      String s = (i < 40) ? "life" : (i > 50) ? "universe" : "everything";
      System.out.println(s);
   }
}
```

**What is the result?**

A. `null`
B. `life`
C. `universe`
D. `everything`
E. `Compilation fails`
F. `An exception is thrown at runtime`

**Correct Answer:** D
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 453**
**Consider the following code segment:**

```
Boolean b = null;
System.out.println(b ? true : false);
```

**Which one of the following options correctly describes the behavior of this code segment?**

A. This code will result in a compiler error since a reference type (of type Boolean) cannot be used as part of expression for condition check.
B. This code will result in a throwing a NullPointerException.
C. This code will print true in console.
D. This code will print false in console.

**Correct Answer:** B
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 454**
**Given the following code**:

```
3. public class Spock {
4.    public static void main(String[] args) {
5.       int mask = 0;
6.       int count = 0;
7.       if( ((5<7) || (++count < 10)) | mask++ < 10 ) mask = mask + 1;
8.       if( (6 > 8) ^ false) mask = mask + 10;
9.       if( !(mask > 1) && ++count > 1) mask = mask + 100;
10.      System.out.println(mask + " " + count);
11.    }
12. }
```

**Which two answers are true about the value of mask and the value of count at line 10? (Choose two.)**

A. mask is 0
B. mask is 1
C. mask is 2
D. mask is 10
E. mask is greater than 10
F. count is 0
G. count is 1

**Correct Answer:** CF
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 455**
**Given the following code and assume that doStuff() returns true:**

```
int y = 5;
int x = 2;
if ( (x > 3) && (y < 2) | doStuff() )
{
   System.out.println("true");
}
else
{
   System.out.println("false");
}
```

**What is the result?**

A. Compilation fails
B. An exception is thrown at runtime
C. true
D. false

**Correct Answer:** D
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 456**
**Given:**

```
4. public class SpecialOps {
5.    public static void main(String[] args) {
6.      String s = "";
7.      Boolean b1 = true;
8.      Boolean b2 = false;
9.      if((b2 = false) | (21%5) > 2) s += "x";
10.     if(b1 || (b2 = true)) s += "y";
11.     if(b2 == true) s += "z";
12.     System.out.println(s);
13.   }
14. }
```

**Which are true? (Choose all that apply.)**

A. Compilation fails
B. x will be included in the output
C. y will be included in the output
D. z will be included in the output
E. An exception is thrown at runtime

**Correct Answer:** C
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 457**
**Given:**

```
public class Test {
   public static void main(String [] args) {
      int x =5;
      boolean b1 = true;
      boolean b2 = false;
      if((x==4) && !b2) {
        System.out.print("1 ");
      }
      System.out.print("2 ");
      if ((b2 = true) && b1) {
        System.out.print("3 ");
      }
   }
}
```

**What is the result?**

A. 2
B. 3
C. 1 2
D. 2 3
E. 1 2 3
F. Compilaton fails
G. An exception is thrown at runtime

**Correct Answer:** D
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 458**
**Given**

```
public class ArithmeticResultsOutput {
   public static void main(String[] args) {
        int i1 = 100;
        int j1 = 200;
        if ((i1 == 99) & (--j1 == 199)) {
                System.out.print("Value1: " + (i1 + j1) + " ");
        }
        else {
                System.out.print("Value2: " + (i1 + j1) + " ");
        }
        int i2 = 100;
        int j2 = 200;
        if ((i2 == 99) && (--j2 == 199)) {
                System.out.print("Value1: " + (i2 + j2) + " ");
        }
        else {
                System.out.print("Value2: " + (i2 + j2) + " ");
        }
        int i3 = 100;
        int j3 = 200;

        if ((i3 == 100) | (--j3 == 200)) {
                System.out.print("Value1: " + (i3 + j3) + " ");
        }
        else {
                System.out.print("Value2: " + (i3 + j3) + " ");
        }
        int i4 = 100;
        int j4 = 200;
        if ((i4 == 100) || (--j4 == 200)) {
                System.out.print("Value1: " + (i4 + j4) + " ");
        }
        else {
                System.out.print("Value2: " + (i4 + j4) + " ");
        }
    }
}
```

**What output will be printed to standard out?**

A. Value2: 300 Value2: 300 Value1: 300 Value1: 300
B. Value2: 299 Value2: 300 Value1: 299 Value1: 300
C. Value2: 299 Value2: 300 Value1: 299 Value1: 300

D. `Value2: 300 Value2: 299 Value1: 300 Value1: 299`

**Correct Answer:** B
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 459**
**What all gets printed when the following program is compiled and run.**

```
public class test {
   public static void main(String args[]) {
     int i, j=1;
     i = (j>1)?2:1;
     switch(i) {
       case 0: System.out.println(0); break;
       case 1: System.out.println(1);
       case 2: System.out.println(2); break;
       case 3: System.out.println(3); break;
     }
  }
}
```

A. `0`
B. `1`
C. `2`
D. `3`
E. `Compilation fails`
F. `An exception is thrown`

**Correct Answer:** BC
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 460**
**Given the code fragment:**

```
String color = "Red";
switch(color) {
   case "Red":
     System.out.println("Found Red");
   case "Blue":
     System.out.println("Found Blue");
     break;
   case "White":
     System.out.println("Found White");
     break;
   default:
      System.out.println("Found Default");
}
```

**What is the result?**

A. `Found Red`
B. `Found Red`
   `Found Blue`
C. `Found Red`

```
   Found Blue
   Found White
```
D.
```
   Found Red
   Found Blue
   Found White
   Found Default
```

**Correct Answer:** B
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 461**
**Consider the following program:**

```
class TestSwitch {
   public static void main(String []args) {
      String [] cards = { "Club", "spade", " diamond ", "hearts" };

      for(String card : cards) {
         switch(card) {
             case "Club" : System.out.print(" club "); break;
             case "Spade" : System.out.print(" spade "); break;
             case "diamond" : System.out.print(" diamond "); break;
             case "heart" : System.out.print(" heart "); break;
             default: System.out.print(" none ");
         }
      }
   }
}
```

**Which one of the following options shows the output of this program?**

A. `none none none none`
B. `club none none none`
C. `club spade none none`
D. `club spade diamond none`
E. `club spade diamond heart`

**Correct Answer:** B
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 462**
**Which exact exception class will the following class throw when compiled and run?**

```
class Test {
     public static void main(String[] args) throws Exception{
         int[] a = null;
         int i = a [ m1() ];
     }

     public static int m1() throws Exception{
         throw new Exception("Some Exception");
     }
}
```

A. `NullPointerException`

B. `ArrayIndexOutOfBoundsException`
C. `Exception`
D. `RuntimeException`

**Correct Answer:** C
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
A `NullPointerException` never occurs because the index expression must be completely evaluated before any part of the indexing operation occurs, and that includes the check as to whether the value of the left-hand operand is null. If the array reference expression produces null instead of a reference to an array, then a `NullPointerException` is thrown at runtime, but only after all parts of the array reference expression have been evaluated and only if these evaluations completed normally.

In an array access, the expression to the left of the brackets appears to be fully evaluated before any part of the expression within the brackets is evaluated. Note that if evaluation of the expression to the left of the brackets completes abruptly, no part of the expression within the brackets will appear to have been evaluated.

Here, `m1()` is called first, which throws `Exception` and so a is never accessed and `NullPointerException` is never thrown.

**QUESTION 463**
**Which of the following is/are valid instantiations and initializations of a multi dimensional array?**

A. `int[][] array2D = new int[][] { { 0, 1, 2, 4} {5, 6}};`
B. `int[][][] array3D = {{0, 1}, {2, 3}, {4, 5}};`
C.
```
int[] array2D[] = new int [2] [2];
array2D[0][0] = 1;
array2D[0][1] = 2;
array2D[1][0] = 3;
```
D. `int[][] array2D = new int[][]{0, 1};`
E.
```
int[] arr = {1, 2};
int[][] arr2 = {arr, {1, 2}, arr};
int[][][] arr3 = {arr2};
```

**Correct Answer:** CE
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 464**
**Given:**

```
interface Rideable {
   String ride() ;
}
class Horse implements Rideable {
   String ride() {
      return "cantering ";
   }
}

class Icelandic extends Horse {
   String ride() {
       return "tolting ";
   }
```

```
}

public class Test1 {
   public static void main(String[] args) {
      Rideable r1 = new Icelandic();
      Rideable r2 = new Horse();
      Horse h1 = new Icelandic();
      System.out.println(r1.ride() + r2.ride() + h1.ride());
   }
}
```

**What is the result?**

A. tolting cantering tolting
B. cantering cantering cantering
C. Compilation fails
D. An exception is thrown at runtime

**Correct Answer:** C
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

```
Main.java:5: error: ride() in Horse cannot implement ride() in Rideable
   String ride() {
          ^
  attempting to assign weaker access privileges; was public
Main.java:11: error: ride() in Icelandic cannot implement ride() in Rideable
   String ride() {
          ^
  attempting to assign weaker access privileges; was public
2 errors
```

**QUESTION 465**
**What will the following code snippet print?**

```
List s1 = new ArrayList();
try {
   while(true) {
      s1.add("sdfa");
   }
}
catch(RuntimeException e) {
     e.printStackTrace();
}
System.out.println(s1.size());
```

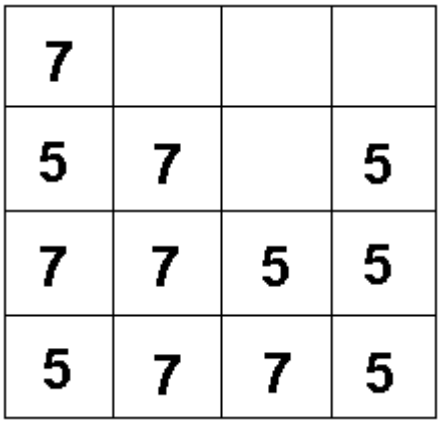
A. It will not compile.
B. It will print a RuntimeException stack trace from the catch clause.
C. It will throw an error at runtime that will not be caught by the catch block.
D. It will print a stack trace from the catch clause and a number depending on the memory available in the system.
E. It will only print a number depending on the memory available in the system

**Correct Answer:** C
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
It will throw a java.lang.OutOfMemoryError. Note that this is not a subclass of RuntimeException or even Exception. It is a subclass of java.lang.Error.

**QUESTION 466**
Assuming that the following array and image represents variation of Connect4 game, where a player wins if she places same number in a row or column:

```
char[][] grid = {{'7', ' ',' ',' '}, {'5', '7', ' ', '5'}, {'7','7','5','5'},
{'5','7','7','5'}};
```




Which of the following assignments would enable a player with number 7 to win? (Choose 2 options)

A. `grid[0] = new char[]{'7','7',' ', ' '};`
B. `grid[1] = new char[]{'7','7',' ', ' '};`
C. `grid[0] = {'7','7',' ', ' '};`
D. `grid[1] = {'7','7',' ', ' '};`
E. `grid[0][1] = '7';`
F. `grid[1][2] = '7';`
G. `grid[0] = new char[4]{'7','7',' ', ' '};`
H. `grid[1] = new char[4]{'7','7',' ', ' '};`

**Correct Answer:** AE
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 467**
**Consider following two classes:**

```
//in file A.java
package p1;

public class A {
     protected int i = 10;
     public int getI() { return i; }
}

//in file B.java
package p2;

import p1.*;
public class B extends p1.A {
     public void process(A a) {
          a.i = a.i*2;
     }
     public static void main(String[] args) {
          A a = new B();
```

```
        B b = new B();
        b.process(a);
        System.out.println( a.getI() );
    }
}
```

**What will be the output of compiling and running class B ?**

A. It will print 10.
B. It will print 20.
C. It will not compile.
D. It will throw an exception at runtime.
E. None of the above.

**Correct Answer:** C
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
Although, class B extends class A and 'i' is a protected member of A, B still cannot access i, (now this is imp) through A's reference because B is not involved in the implementation of A.
Had the process() method been defined as process(B b); b.i would have been accessible as B is involved in the implementation of B.

**QUESTION 468**
**Consider the following program:**

```
class Base {
   public static void foo(Base bObj) {
      System.out.println("In Base.foo()");
      bObj.bar();
   }

   public void bar() {
      System.out.println("In Base.bar()");
   }
}

class Derived extends Base {
   public static void foo(Base bObj) {
       System.out.println("In Derived.foo()");
       bObj.bar();
   }
   public void bar() {
       System.out.println("In Derived.bar()");
   }
}

class OverrideTest {
   public static void main(String []args) {
       Base bObj = new Derived();
       bObj.foo(bObj);
   }
}
```

**What is the output of this program when executed?**

A.
```
In Base.foo()
In Base.bar()
```
B.
```
In Base.foo()
In Derived.bar()
```
C.
```
In Derived.foo()
```

```
In Base.bar()
```

D.
```
In Derived.foo()
In Derived.bar()
```

E.
```
Compilation fails
```

**Correct Answer:** B
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 469**
**Consider the following code:**

```
public class TestClass {
   public static void doStuff() throws Exception {
        System.out.println("Doing stuff...");
        if(Math.random() > 0.4) {
            throw new Exception("Too high!");
        }
        System.out.println("Done stuff.");
   }

   public static void main(String[] args) throws Exception {
        doStuff();
        System.out.println("Over");
   }
}
```

**Which two of the following are possible outputs when the above program is compiled and run? Assume that Math.random() returns a double between 0.0 and 1.0 not including 1.0. Further assume that there is no mistake in the line numbers printed in the output shown in the options.**

A.
```
Doing stuff...
Exception in thread "main" java.lang.Exception: Too high!
    at TestClass.doStuff(TestClass.java:29)
    at TestClass.main(TestClass.java:41)
```

B.
```
Doing stuff...
Exception in thread "main" java.lang.Exception: Too high!
    at TestClass.doStuff(TestClass.java:29)
    at TestClass.main(TestClass.java:41)
Over
```

C.
```
Doing stuff...
Done stuff.
Over
```

D.
```
Doing stuff...
Exception in thread "main" java.lang.Exception: Too high!
    at TestClass.doStuff(TestClass.java:29)
    at TestClass.main(TestClass.java:41)
Done stuff.
```

**Correct Answer:** AC
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 470**
**What will the following program print?**

```
public class TestClass{
  static String str;
  public static void main(String[] args){
     System.out.println(str);
  }
}
```

A. It will not compile.
B. It will compile but throw an exception at runtime.
C. It will print null.
D. It will print nothing.
E. None of the above.

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 471**
**Given the definitions of the MyString class and the Test class:**

**MyString.java:**
```
package p1;

class MyString {
      String msg;
      MyString(String msg) {
            this.msg = msg;
      }
}
```

**Test.java:**
```
package p1;

public class Test {
      public static void main(String[] args) {
            System.out.println("Hello " + new StringBuilder("Java SE 8 "));
            System.out.println("Hello " + new MyString("Java SE 8"));
     }
}
```

**What is the result?**

A. Hello Java SE 8
   Hello Java SE 8
B. Hello java.lang.StringBuilder@<<hashcode>>
   Hello p1.MyString@<<hashcode>>
C. Hello Java SE 8
   Hello p1.MyString@<<hashcode>>
D. Compilation fails

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 472**
**You are asked to develop a program for a shopping application, and you are given the following information:**

**1. The application must contain the classes `Toy`, `EduToy`, and `ConsToy`.**
**The `Toy` class is the superclass of the other two classes.**
**2. The `int calculatePrice (Toy t)` method calculates the price of a toy.**
**3. The `void printToy (Toy t)` method prints the details of a toy.**

**Which definition of the Toy class adds a valid layer of abstraction to the class hierarchy?**

A.
```
public abstract class Toy {
    public abstract int calculatePrice(Toy t);
    public void printToy(Toy t) { /* code goes here */ }
}
```
B.
```
public abstract class Toy {
    public int calculatePrice(Toy t);
    public void printToy(Toy t);
}
```
C.
```
public abstract class Toy {
    public int calculatePrice(Toy t);
    public final void printToy(Toy t) { /* code goes here */ }
}
```
D.
```
public abstract class Toy {
    public abstract int calculatePrice(Toy t) { /* code goes here */ }
    public abstract void printToy(Toy t) { /* code goes here */ }
}
```

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 473**
**Given:**

```
public class TestScope {
    public static void main(String[] args) {
           int var1 = 200;
           System.out.print(doCalc(var1));
           System.out.print(" " + var1);
    }
    static int doCalc(int var1) {
           var1 = var1 * 2;
           return var1;
    }
}
```

**What is the result?**

A. `400 200`
B. `200 200`
C. `400 400`
D. `Compilation fails`

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 474**
**Consider the following program:**

```
class CannotFlyException extends Exception {}

interface Birdie {
    public abstract void fly() throws CannotFlyException;
}

interface Biped {
    public void walk();
}

abstract class NonFlyer {
    public void fly() { System.out.print("cannot fly "); } // LINE A
}

class Penguin extends NonFlyer implements Birdie, Biped { // LINE B
    public void walk() { System.out.print("walk "); }
}

class PenguinTest {
    public static void main(String []args) {
       Penguin pingu = new Penguin();
       pingu.walk();
       pingu.fly();
    }
}
```

**Which one of the following options correctly describes the behavior of this program?**

A. Compiler error in line with comment LINE A
B. Compiler error in line with comment LINE B
C. It crashes after throwing the exception CannotFlyException
D. When executed, the program prints “walk cannot fly”.

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 475**
**Given:**

```
public class Triangle {
    static double area;
    int b = 2, h = 3;
    public static void main(String[] args) {
           double p, b, h;
           if (area == 0) {
                 b = 3;
                 h = 4;
                 p = 0.5;
           }
           area = p * b * h;
           System.out.println("Area is " + area);
    }
}
```

**What is the result?**

A. `Area is 6.0`
B. `Area is 3.0`
C. `Compilation fails`
D. `An exception is thrown`

**Correct Answer:** C
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 476**
**Given the code fragment:**

```
public class Test {
     public static void main(String[] args) {
          //line n1
          switch(x) {
              case 1:
                    System.out.println("One");
                    break;
              case 2:
                    System.out.println("Two");
                    break;
           }
     }
}
```

**Which three code fragments can be independently inserted at line n1 to enable the code to print one?**

A. `Byte x = 1;`
B. `short x = 1;`
C. `String x = "1";`
D. `Long x = 1;`
E. `Double x = 1;`
F. `Integer x = new Integer ("1");`

**Correct Answer:** ABF
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 477**
**Consider the following program:**

```
abstract class AbstractBook {
   public String name;
}

interface Sleepy {
   public String name = "undefined"; //LINE A
}

class Book extends AbstractBook implements Sleepy {
   public Book(String name) {
      this.name = name; // LINE B
   }
```

```
   public static void main(String []args) {
      AbstractBook philosophyBook = new Book("Principia Mathematica");
      System.out.println("The name of the book is " + philosophyBook.name); //
LINE B
   }
}
```

**Which one of the following options correctly describes the behavior of this program?**

A. The program will print the output “The name of the book is Principia Mathematica”.
B. The program will print the output “The name of the book is null”.
C. Compilation fails on LINE A
D. Compilation fails on LINE B
E. Compilation fails on LINE C

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

```
Main.java:13: error: reference to name is ambiguous
      this.name = name; // LINE B
           ^
  both variable name in AbstractBook and variable name in Sleepy match
```

**QUESTION 478**
**Given the following code:**

package planets;

```
public class Planet {
     public String name;
     public int moons;

     public Planet(String name, int moons) {
          this.name = name;
          this.moons = moons;
     }
}
```

**And the following main method:**

```
public static void main(String[] args) {
    Planet[] planets = {
                new Planet("Mercury", 0),
                new Planet("Venus", 0),
                new Planet("Earth", 1),
                new Planet("Mars", 2)
    };

    System.out.println(planets);
    System.out.println(planets[2]);
    System.out.println(planets[2].moons);
}
```

**What is the output?**

A.
```
planets
Earth
1
```
B. [Lplanets.Planet;@15db9742

```
   Earth
   1
```
C.
```
   [Lplanets.Planet;@15db9742
   planets.Planet@6d06d69c
   1
```
D.
```
   [Lplanets.Planet;@15db9742
   planets.Planet@6d06d69c
   [Lplanets.Moon@7852e922
```
E.
```
   [Lplanets.Planet;@15db9742
   Venus
   0
```

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 479**
**Given:**

```
package p1;

public interface DoInterface {
   void m1(int n);        // line n1
   public void m2(int n);
}


package p3;
import p1.DoInterface;

public class DoClass implements DoInterface {
    int x1, x2;
    DoClass() {
        this.x1 = 0;
        this.x2 = 10;
    }

    public void m1(int p1) {x1 += p1; System.out.println(x1);} //line n2
    public void m2(int p1) {x2 += p1; System.out.println(x2);}

}

package p2;
import p1.*;
import p3.*;

class Test {

     public static void main(String[] args) {
          DoInterface doi = new DoClass();  // line n3
          doi.m1(100);
          doi.m2(200);
     }
}
```

**What is the result?**

A.
```
   100
   210
```
B. `Compilation fails due to an error in line n1`
C. `Compilation fails due to an error at line n2`

D. Compilation fails due to an error at line n3

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 480**
**Given:**

```
class Vehicle {
     int x;
     Vehicle() {
          this(10);
     }
     Vehicle(int x) {
          this.x = x;
     }
}

class Car extends Vehicle {
     int y;
     Car() {
          super();
          this(20);
     }
     Car(int y) {
          this.y = y;
     }
     public String toString() {
          return super.x + ":" + this.y;
     }
}
```

**And given the code fragment:**

```
Vehicle y = new Car();
System.out.println(y);
```

**What is the result?**

A. 10:20
B. 0:20
C. Compilation fails
D. An exception is thrown

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**
```
Main.java:24: error: call to this must be first statement in constructor
          this(20);
```

**QUESTION 481**
**Given:**

```
public class Test {

  public static void main(String[] args) {
      if (args[0].equals("Hello") ? false : true) {
```

```
            System.out.println("Success");
        } else {
            System.out.println("Failure");
        }
}
```

**And given the commands:**
```
javac Test.java
java Test Hello
```

**What is the result?**

A. `Success`
B. `Failure`
C. `Compilation fails`
D. `An exception is thrown at runtime`

**Correct Answer:** B
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 482**
**Given:**

```
interface Event {
   String type = "Event";
   public void details();
}

class Quiz {
   static String type = "Quiz";
}

public class PracticeQuiz extends Quiz implements Event {
   public void details() {
      System.out.print(type);
   }
   public static void main(String[] args) {
      new PracticeQuiz().details();
      System.out.print(" " + type);
   }
}
```

**What is the result?**

A. `Event Quiz`
B. `Event Event`
C. `Quiz Quiz`
D. `Quiz Event`
E. `Compilation fails`

**Correct Answer:** E
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
```
Main.java:12: error: reference to type is ambiguous
      System.out.print(type);
                       ^
```

```
  both variable type in Quiz and variable type in Event match
Main.java:16: error: reference to type is ambiguous
      System.out.print(" " + type);
                             ^
  both variable type in Quiz and variable type in Event match
2 errors
```

**QUESTION 483**
**Given:**

```
public class Test {
   public void doChange(StringBuilder sb) {
      sb.append(" Moliday");
   }

   public static void main(String[] args) {
      StringBuilder sb = new StringBuilder("Sunday");
      doChange(sb);
      System.out.println(sb);
   }
}
```

**What is the result?**

A. `Sunday Moliday`
B. `Sunday`
C. Compilation fails
D. An exception is thrown at runtime

**Correct Answer:** C
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 484**
**Given:**

```
public class Test2 {

  public static void main(String[] args) {
     int[] arr = {10, 20, 30};
     doChange(arr);
     for (int x : arr) {
        System.out.print(x + ", ");
     }
     doChange(arr[0], arr[1], arr[2]);
     System.out.print(arr[0] + ", " + arr[1] + ", " + arr[2]);
  }

  void doChange(int... arr) {
     for (int pos = 0; pos < arr.length; pos++) {
        arr[pos] = arr[pos] + 1;
     }
  }

}
```

A. `11, 21, 31, 11, 21, 31`
B. `11, 21, 31, 12, 22, 32`
C. `12, 22, 32, 12, 22, 32`
D. `10, 20, 30, 10, 20, 30`

E. Compilation fails

**Correct Answer:** E
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**
```
error: non-static method doChange(int...) cannot be referenced from a static
context
     doChange(arr);
```

**QUESTION 485**
**What will the following code print when compiled and run?**

```
import java.util.*;

interface Birdie {
    void fly();
}

class Dino implements Birdie {
    public void fly(){ System.out.println("Dino flies"); }
    public void eat(){ System.out.println("Dino eats");}
}

class Bino extends Dino {
    public void fly(){ System.out.println("Bino flies"); }
    public void eat(){ System.out.println("Bino eats");}
}

public class TestClass {
    public static void main(String[] args) {
         List<Birdie> m = new ArrayList<>();
         m.add(new Dino());
         m.add(new Bino());
         for(Birdie b : m) {
             b.fly();
             b.eat();
         }
    }
}
```

A.
```
Dino flies
Dino eats
Bino flies
Bino eats
```
B.
```
Bino flies
Bino eats
```
C.
```
Dino flies
Bino eats
```
D. The code will not compile.
E. Exception at run time.

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
Note that in the for loop b has been declared to be of type Birdie. But Birdie doesn't define the method eat(), so the compiler will not allow b.eat() even though the actual class of the object referred to by b does have an eat() method.

**QUESTION 486**
**Given:**

```
class Alpha {
    public String[] main = new String[2];
    Alpha(String[] main) {
         for(int ii=0; ii < main.length; ii++) {
              this.main[ii] = main[ii] + 5;
         }
    }
    public void main() {
              System.out.println(main[0] + main[1]);
    }
}

public class Test {
    public static void main(String[] args) {
          Alpha main = new Alpha(args);
          main.main();
    }
}
```

**And the commands:**

```
javac Test.java
java Test 1 2
```

**What is the result?**

A. 1525
B. 13
C. Compilation fails
D. An exception is thrown at runtime
E. The program fails to execute due a runtime error.

**Correct Answer:** A
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 487**
**Given the following code:**

```
2. class Explode {
3.    static String s = "";
4.    static { s += "sb1 "; }
5.    Explode() { s += "e "; }
6. }
7. public class C4 extends Explode {
8.    C4() {
9.      s += "c4 ";
10.     new Explode();
11.   }
12.   static {
13.     new C4();
14.     System.out.print(s);
15.   }
16.   { s += "i "; }
17.   public static void main(String[] args) { }
18. }
```

**And given the command-line invocation "java C4", what is the result?**

A. `e c4 i`
B. `e i c4`
C. `e sb1 i c4`
D. `sb1 e i c4 e`
E. `sb1 e c4 i e`
F. `Compilation fails.`
G. `A StackOverflowError is thrown`
H. `An exception other than StackOverflowError is thrown`

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 488**
**Given the code fragments:**

```
public class Person {

    String name;
    int age;

    public Person(String n, int a) {
        name = n;
        age = a;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }
}

public class Test {

    public static void checkAge(List<Person> list, Predicate<Person> predicate)
{
        for (Person p : list) {
            if (predicate.test(p)) {
                System.out.println(p.name + " ");
            }
        }
    }

    public static void main(String[] args) {
        List<Person> iList = Arrays.asList(new Person("Hank", 45), new Person
("Charlie", 40), new Person("Smith", 38));
        // line n1
    }
}
```

**Which code fragment, when inserted at line n1, enables the code to print Hank?**

A. `checkAge(iList, () -> p.getAge() > 40);`
B. `checkAge(iList, Person p -> p.getAge() > 40);`
C. `checkAge(iList, p -> p.getAge() > 40);`
D. `checkAge(iList, (Person p) -> { p.getAge() -> 40; });`

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

```
@FunctionalInterface
public interface Predicate<T> {

    boolean test(T t);
}
```

**QUESTION 489**
**Given the following code:**

```
3. public class Gauntlet {
4.    public static void main(String[] args) {
5.       String r = "0";
6.       int x = 3, y = 4;
7.       boolean test = false;
8.       if((x > 2) || (test = true))
9.       if((y > 5) || (++x == 4))
10.      if((test == true) || (++y == 4))
11.      r += "1";
12.      else if(y == 5) r += "2";
13.      else r += "3";
14.      else r += "4";
15.      // else r += "5";
16.      System.out.println(r);
17. } }
```

**And given that, if necessary you can add line 15 to make the code compile, what is the result? (Choose three)**

A. At line 10, test will equal true
B. At line 10, test will equal false
C. The output will be 02
D. The output will be 03
E. The output will be 023
F. The code will compile as is.
G. The code will only compile if line 15 is added

**Correct Answer:** BCF
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 490**
**Given the following code fragment**:

```
public class Access {
   private int x = 0;
   private int y = 0;
   public static void main(String[] args) {
       Access accApp = new Access();
       accApp.printThis(1, 2);
       accApp.printThat(3, 4);
   }
   public void printThis(int x, int y) {
       x = x;
       y = y;
```

```
        System.out.println("x:" + this.x + " y:" + this.y);
    }
    public void printThat(int x, int y) {
        this.x = x;
        this.y = y;
        System.out.println("x:" + this.x + " y:" + this.y);
    }
}
```

**What is the result?**

A.
```
x:1 y:2
x:3 y:4
```
B.
```
x:0 y:0
x:3 y:4
```
C.
```
x:3 y:4
x:0 y:0
```
D.
```
x:3 y:4
x:1 y:2
```

**Correct Answer:** B
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 491**
**Given the following code fragment**:

```
class Fizz {
    int x = 5;
    public static void main(String[] args) {
        final Fizz f1 = new Fizz();
        Fizz f2 = new Fizz();
        Fizz f3 = FizzSwitch(f1,f2);
        System.out.println((f1 == f3) + " " + (f1.x == f3.x));
    }
    static Fizz FizzSwitch(Fizz x, Fizz y) {
        final Fizz z = x;
        z.x = 6;
        return z;
    }
}
```

**What is the result?**

A. `true true`
B. `false true`
C. `true false`
D. `false false`
E. `Compilation fails`
F. `An exception is thrown at runtime`

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 492**

**Given the following code fragment**:

```
class Cake {
   int model;
   String flavor;
   Cake() {
     model = 0;
     flavor = "Unknown";
   }
}

public class Test {
    public static void main(String[] args) {
        Cake c = new Cake();
        bake1(c);
        System.out.println(c.model + " " + c.flavor);
        bake2(c);
        System.out.println(c.model + " " + c.flavor);
    }
    public static Cake bake1(Cake c) {
        c.flavor = "Strawberry";
        c.model = 1200;
        return c;
    }
    public static void bake2(Cake c) {
        c.flavor = "Chocolate";
        c.model = 1230;
        return;
    }
}
```

**What is the result?**

A.
```
0 Unknown
0 Unknown
```

B.
```
1200 Strawberry
1200 Strawberry
```

C.
```
1200 Strawberry
1230 Chocolate
```

D.
```
Compilation fails
```

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 493**
**Given the following code fragment**:

```
public class Painting {
     private String type;
     public String getType() {
         return type;
     }
     public void setType(String type) {
          this.type = type;
     }
     public static void main (String[] args) {
          Painting obj1 = new Painting();
          Painting obj2 = new Painting();
          obj1.setType(null);
          obj2.setType("Fresco");
          System.out.print(obj1.getType() + " : " + obj2.getType());
```

```
    }
}
```

**What is the result?**

A. `: Fresco`
B. `null : Fresco`
C. `Fresco : Fresco`
D. `A NullPointerException is thrown at runtime`

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 494**
**Given:**

```
public class TestApp {

  public static void main(String[] args) {
     TestApp t = new TestApp();
     try {
         t.doPrint();
         t.doList();
     }
     catch(Exception ex) {
         System.out.println("Caught " + ex);
     }
  }

  public void doList() throws Exception {
      throw new Error("Error");
  }

  public void doPrint() throws Exception {
      throw new RuntimeException("Exception");
  }

}
```

**What is the result?**

A.
```
Caught java.lang.RuntimeException: Exception
Exception in thread "main" java.lang.Error: Error
    at TestApp.doList(TestApp.java: 14)
    at TestApp.main(TestApp.java: 6)
```

B.
```
Exception in thread "main" java.lang.Error: Error
    at TestApp.doList(TestApp.java: 14)
    at TestApp.main(TestApp.java: 6)
```

C.
```
Caught java.lang.RuntimeException: Exception
Caught java.lang.Error: Error
```

D.
```
Caught java.lang.RuntimeException: Exception
```

**Correct Answer:** D
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 495**
**Given the code fragment**

```
interface Contract {}
class Super implements Contract {}
class Sub extends Super {}

public class Ref {
    public static void main(String[] args) {
       List objs = new ArrayList();
       Contract c1 = new Super();
       Contract c2 = new Sub();  //Line n1
       Super s1 = new Sub();
       objs.add(c1);
       objs.add(c2);
       objs.add(s1);            //Line n2

       for(Object itm : objs) {
            System.out.println(itm.getClass().getName());
       }
    }
}
```

**What is the result?**

A.
```
Super
Sub
Sub
```

B.
```
Contract
Contract
Super
```

C. `Compilation fails at line n1`

D. `Compilation fails at line n2`

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 496**
**Given**

```
1.     class Ring {
2.         final static int x2 = 7;
3.         final static Integer x4 = 8;
4.         public static void main(String[] args) {
5.             Integer x1 = 5;
6.             String s = "a";
7.             if(x1 < 9) s += "b";
8.             switch(x1) {
9.                 case 5: s += "c";
10.                case x2: s += "d";
11.                case x4: s += "e";
12.            }
13.            System.out.println(s);
14.         }
15.     }
```

**What is the result?**

A. `abc`

B. `abcde`

C. Compilation fails on line 8
D. Compilation fails on line 9
E. Compilation fails on line 10
F. Compilation fails on line 11

**Correct Answer:** F
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

```
Main.java:11: error: constant expression required
                case x4: s += "e";
```

**QUESTION 497**
**Given**

```
class Knowing {
    static final long tooth = 343L;
    static long doIt(long tooth) {
        System.out.print(++tooth + " ");
        return ++tooth;
    }
    public static void main(String[] args) {
        System.out.print(tooth + " ");
        final long tooth = 340L;
        new Knowing().doIt(tooth);
        System.out.println(tooth);
    }
}
```

**What is the result?**

A. 343 340 340
B. 343 340 342
C. 343 341 342
D. 343 341 340
E. 343 341 343
F. Compilation fails
G. An exception is thrown

**Correct Answer:** D
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 498**
**Given a java source file:**

```
class X {
   X () {}
   private void one() {}
}
public class Y extends X {
   Y () {}
   private void two() {one();}
   public static void main (String[] args) {
      new Y().two();
   }
}
```

**What changes will make this code compile?**

A. `adding the public modifier to the declaration of class X`
B. `adding the protected modifier to the X() constructor`
C. `changing the private modifier on the declaration of the one() method to protected`
D. `removing the Y() constructor`
E. `removing the private modifier from the two() method`

**Correct Answer:** C
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 499**
**Given:**

```
1. import java.util.*;

2. public class MyPancake implements Pancake {
3.     public static void main(String[] args) {
4.         List<String> x = new ArrayList<String>();
5.         x.add("3"); x.add("7"); x.add("5");
6.         List<String> y = new MyPancake().doStuff(x);
7.         y.add("1");
8.         System.out.println(x);
9.     }
10.    List<String> doStuff(List<String> z) {
11.        z.add("9");
12.        return z;
13.    }
14. }

15. interface Pancake {
16.     List<String> doStuff(List<String> s);
17. }
```

**What is the result?**

A. `[3, 7, 5]`
B. `[3, 7, 5, 9]`
C. `[3, 7, 5, 9, 1]`
D. `Compilation fails.`
E. `An exception is thrown at runtime.`

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 500**
**What will be the output when the following program is run?**

```
package exceptions;
public class TestClass{
    public static void main(String[] args) {
        try{
```

```
            hello();
        }
        catch(MyException me){
            System.out.println(me);
        }
    }

    static void hello() throws MyException{
        int[] dear = new int[7];
        dear[0] = 747;
        foo();
    }

    static void foo() throws MyException{
        throw new MyException("Exception from foo");
    }
}

class MyException extends Exception {
    public MyException(String msg){
        super(msg);
    }
}
```

A. 
```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
            at exceptions.TestClass.doTest(TestClass.java:24)
            at exceptions.TestClass.main(TestClass.java:14)
```

B. 
```
Error in thread "main" java.lang.ArrayIndexOutOfBoundsException
```

C. 
```
exceptions.MyException: Exception from foo
```

D. 
```
exceptions.MyException: Exception from foo
        at exceptions.TestClass.foo(TestClass.java:29)
        at exceptions.TestClass.hello(TestClass.java:25)
        at exceptions.TestClass.main(TestClass.java:14)
```

**Correct Answer:** C
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 501**
**Given**

```
2. import java.util.*;
3. public class HR {
4.     public static void main(String[] args) {
5.         List<Integer> i = new ArrayList<Integer>();
6.         i.add(3); i.add(2); i.add(5);
7.         int ref = 1;
8.         doStuff(ref);
9.         System.out.println(i.get(ref));
10.    }
11.    static int doStuff(int x) {
12.        return ++x;
13.  } }
```

**What is the result?**

A. 2
B. 3
C. 5
D. Compilation fails.

E. An exception is thrown at runtime.

**Correct Answer:** A
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 502**
**Given:**

```
2. class RainCatcher {
3.    static StringBuffer s;
4.    public static void main(String[] args) {
5.       Integer i = new Integer(42);
6.       for(int j = 40; j < i; i--)
7.         switch(i) {
8.           case 41: s.append("41 ");
9.           default: s.append("def ");
10.          case 42: s.append("42 ");
11.        }
12.      System.out.println(s);
13. } }
```

**What is the result?**

A. 41 def
B. 41 def 42
C. 42 41 def 42
D. An exception is thrown at runtime.
E. Compilation fails due to an error on line 6.
F. Compilation fails due to an error on line 7.

**Correct Answer:** D
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

```
Exception in thread "main" java.lang.NullPointerException
    at Main.main(Main.java:11)
```

**QUESTION 503**
**View the following code:**

```
class Foo {
   private int x;
   public Foo( int x ){ this.x = x;}
   public void setX( int x ) { this.x = x; }
   public int getX(){ return x;}
}

public class Gamma {
   static Foo fooBar(Foo foo) {
     foo = new Foo(100);
     return foo;
   }

   public static void main(String[] args) {
     Foo foo = new Foo( 300 );
     System.out.print( foo.getX() + "-");
     Foo fooFoo = fooBar(foo);
     System.out.print(foo.getX() + "-");
```

```
      System.out.print(fooFoo.getX() + "-");
      foo = fooBar( fooFoo);
      System.out.print( foo.getX() + "-");
      System.out.print(fooFoo.getX());
   }
}
```

**What is the output of the program shown in the exhibit?**

A. 300-100-100-100-100
B. 300-300-100-100-100
C. 300-300-300-100-100
D. 300-300-300-300-100

**Correct Answer:** B
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 504**
**Given the code fragment:**

```
public class Test {
   static void main(String[] args) {
        String[] array = {"a", "b", "c", "d"};
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
            if (array[i].equals("c")) {
               continue;
            }
            System.out.println("Finish");
            break;
        }
   }
}
```

**What is the result?**

A. a b c Finish
B. a b c d Finish
C. a Finish
D. Compilation fails
E. An error is thrown at runtime

**Correct Answer:** E
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**

**QUESTION 505**
**Given:**

```
class X {
  int x1, x2, x3;
}

class Y extends X {
  int y1;
```

```
  Y() {
     x1 = 1;
     x2 = 2;
     y1 = 10;
  }
}

class Z extends Y {
   int z1;
   Z() {
     x1 = 3;
     y1 = 20;
     z1 = 100;
   }
}
```

**And,**

```
public class Test3  {
   public static void main(String[] args) {
      Z obj = new Z();
      System.out.println(obj.x3 + ", " + obj.y1 + ", " + obj.z1);
   }
}
```

**Which constructor initializes the attribute x3?**

A. only the default constructor of class X
B. only the no-argument constructor of class Y
C. only the no-argument constructor of class Z
D. only the default constructor of Object class

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 506**
**Given the code fragment:**

```
LocalDate date1 = LocalDate.now();
LocalDate date2 = LocalDate.of(2014, 6, 20);
LocalDate date3 = LocalDate.parse("2014-06-20", DateTimeFormatter.ISO_DATE);
System.out.println("date1 = " + date1);
System.out.println("date2 = " + date2);
System.out.println("date3 = " + date3);
```

**Assume that the system date is June 20, 2014. What is the result?**

A.
```
date1 = 2014-06-20
date2 = 2014-06-20
date3 = 2014-06-20
```
B.
```
date1 = 06/20/2014
date2 = 2014-06-20
date3 = June 20, 2014
```
C. Compilation fails
D. A DateParseException is thrown at runtime

**Correct Answer:** A
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 507**
**Given:**

```
public class SumTest {

    public static void doSum(Integer x, Integer y) {
         System.out.println("Integer sum is " + (x + y));
    }

    public static void doSum(Number x, Number y) {
         System.out.println("Number sum is " + (x + y));
    }


    public static void doSum(float x, float y) {
         System.out.println("float sum is " + (x + y));
    }

    public static void doSum(int x, int y) {
         System.out.println("int sum is " + (x + y));
    }

    public static void main(String[] args) {
         doSum(10, 20);
         doSum(10.0, 20.0);
    }
}
```

**What is the result?**

A.
```
int sum is 30
float sum is 30.0
```
B.
```
int sum is 30
Number sum is 30.0
```
C.
```
Integer sum is 30
Number sum is 30.0
```
D.
```
Integer sum is 30
float sum is 30.0
```
E. `Compilation fails`

**Correct Answer:** E
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

```
Main.java:9: error: bad operand types for binary operator '+'
         System.out.println("Number sum is " + (x + y));
                                                 ^
  first type:  Number
  second type: Number
```

**QUESTION 508**
**What is displayed when the following code is compiled and executed?**

```
StringBuilder s1 = new StringBuilder("Hello");
StringBuilder s2 = new StringBuilder("Hello");

if (s1==s2)
   System.out.println("Same");
if (s1.equals(s2))
```

```
   System.out.println("Equals");
```

A. Same
B. Equals
C. Nothing is printed
D. Same
   Equals
E. The code fails to compile
F. An exception is thrown

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 509**
**Given:**

```
class Test {
   public static void main(String args[]) {
      Long a = 3000L;
      Long b = 2999L;
      Long c = 2000L;
      if ((a > b) ^ ((c * 2) == a))
        System.out.print("x");
      if ((b + 1 != a) ^ ((c * 2) == b))
        System.out.print("y");
   }
}
```

**What is the result?**

A. x
B. y
C. xy
D. Compilation fails
E. No output is produced
F. An exception is thrown at runtime

**Correct Answer:** A
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 510**
**Given:**

```
public class TableTest {
    static String[][] table;
    public static void main(String[] args) {
          String[] x = { "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun" };

          String[] y1 = { "1", "2", "3", "4", "5" };
          String[] y2 = { "a", "b", "c" };
          table = new String[3][];
          table[0] = x;
          table[1] = y1;
```

```
            table[2] = y2;
            //INSERT CODE HERE
        }
}
```

**What can be inserted in the above code to make it print Sun5c?**

A.
```
for(String[] row : table){
    System.out.print(row[row.length]);
}
```
B.
```
int i = 0;
for(String[] col : table){
    i++;
    if(i==col.length){
        System.out.print(table[col.length][i]);
    }
}
```
C.
```
for(String[] row : table){
     System.out.print(row[row.length-1]);
}
```
D.
```
for(int i=0; i<table.length-1; i++){
     int j = table[i].length-1;
     System.out.print(table[i][j]);
}
```

**Correct Answer:** C
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 511**
**What will the following code print when compiled and run?**

```
abstract class Widget {
    String data = "data";
    public void doWidgetStuff() {
    }
}

class GoodWidget extends Widget{
    String data = "big data";
    public void doWidgetStuff() {
         System.out.println(data);
    }
}

public class WidgetUser{
     public static void main(String[] args) {
          Widget w = new GoodWidget();
          w.doWidgetStuff();
     }
}
```

A. `data`
B. `big data`
C. `It will not print anything.`
D. `It will not compile.`

**Correct Answer:** B
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 512**
**Which of the following is correct about Java byte code?**

A. It can run on any OS and chip architecture.
B. It can run on any OS and chip architecture if there is a JRE available for that OS and chip architecture.
C. It can run only any OS and chip architecture if that platform has a JRE built for it and the Java code was compiled ON that platform.
D. It can run only any OS and chip architecture if that platform has a JRE built for it and the Java code was compiled FOR that platform.

**Correct Answer:** B
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
Java byte code is basically just a set of instructions that are intepreted by a virtual machine and is independent of the actual machine and OS i.e. the platform. JRE (Java Runtime Environment) is the virtual machine that interprets the given byte code and converts it into the acutal platform understandable instructions. Therefore, all you need to run the byte code is the virtual machine (JRE) for that specific platform on which you want to run it.

Since the byte code itself is platform independent, you can compile your java code on any platform because no matter where you compile your code, the same byte code will be produced. Therefore, you don't need a java compiler for a particular platform. You just need the JRE for that platform. Oracle provides JRE for several platforms including Windows and Unix.

**QUESTION 513**
**What will be the output of the following program?**

```
class Test {
    static int i1, i2, i3;
    public static void main(String[] args) {
        try {
           test(i1 = 1, oops(i2=2), i3 = 3);
        } catch (Exception e)  {
           System.out.println(i1 + " " + i2 + " " +i3);
        }
    }
    static int oops(int i) throws Exception {
          throw new Exception("oops");
    }
    static int test(int a, int b, int c) {
          return a + b + c;
    }
}
```

A. `1 0 0`
B. `1 2 0`
C. `1 2 3`
D. `0 0 0`
E. `It will not compile.`

**Correct Answer:** B
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 514**
**Given the code fragment:**

```
String shirts[][] = new String[2][2];
shirts[0][0] = "red";
shirts[0][1] = "blue";
shirts[1][0] = "small";
shirts[1][1] = "medium";
```

**Which code fragment prints red: blue: small: medium?**

A.
```
for (int index = 1; index < 2; index++) {
      for (int idx = 1; idx < 2; idx++) {
          System.out.print(shirts[index][idx] + ":");
      }
  }
```
B.
```
for (int index = 0; index < 2; index++) {
      for (int idx = 0; idx < index; idx++) {
          System.out.print(shirts[index][idx] + ":");
      }
  }
```
C.
```
for (String sizes : shirts) {
      for (String s : sizes) {
          System.out.print(s + ":");
      }
  }
```
D.
```
for (int index = 0; index < 2;) {
      for (int idx = 0; idx < 2;) {
          System.out.print(shirts[index][idx] + ":");
          idx++;
      }
      index++;
  }
```

**Correct Answer:** D
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 515**
**Which of the following are valid ways to create a LocalDateTime?**

A. `java.time.LocalDateTime.of(2015, 10, 1, 10, 10);`
B. `java.time.LocalDate.parse("2015-01-02");`
C. `java.time.LocalDateTime.of(2015, "January", 1, 10, 10);`
D. `java.time.LocalDateTime.parse("2015-01-02");`

**Correct Answer:** A
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 516**
**Given:**

```
//In Data.java
public class Data {
    int value;
    Data(int value){
        this.value = value;
    }
}
```

**and the following code fragments:**

```
public void printUsefulData(ArrayList<Data> dataList, Predicate<Data> p){
     for(Data d: dataList) {
         if(p.test(d))
             System.out.println(d.value);
     }
}

....
ArrayList<Data> al = new ArrayList<Data>();
al.add(new Data(1));
al.add(new Data(2));
al.add(new Data(3));
//INSERT METHOD CALL HERE
```

**Which of the following options can be inserted above so that it will print 3? (Please select 2 options)**

A. `printUsefulData(al, (Data d)-> { return d.value>2; }  );`
B. `printUsefulData(al, d-> d.value > 2 );`
C. `printUsefulData(al, (d)-> return d.value > 2;  );`
D. `printUsefulData(al, Data d -> d.value>2  );`
E. `printUsefulData(al, d -> d.value > 2;   );`

**Correct Answer:** AB
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
`printUsefulData(al, (d)-> return d.value > 2;  );`
If you write return, the compiler assumes that you are writing the complete method body and so it expects the curly braces as well as the semi-colon.

`printUsefulData(al, Data d -> d.value>2  );`
If you write parameter type, the compiler assumes that you are writing the complete parameter list of the method and so it expects the brackets i.e. (Data d) instead of just Data d.

`printUsefulData(al, d -> d.value > 2;   );`
The semi-colon in the method body should not be there because the line of code is not enclosed within curly braces.

There is a simple trick to identify invalid lambda constructs. When you write a lambda expression for a functional interface, you are essentially providing an implementation of the method declared in that interface but in a very concise manner.  Therefore, the lambda expression code that you write must contain all the pieces of the regular method code except the ones that the compiler can easily figure out on its own such as the parameter types, return keyword, and brackets. So, in a lambda expression, just check that all the information is there and that the expression follows the basic syntax -

**QUESTION 517**
**What will the following code print?**

```
int x = 1;
int y = 2;
int z = x++;
int a = --y;
int b = z--;
b += ++z;
int answ = x>a?y>b?y:b:x>z?x:z;
System.out.println(answ);
```

A. 0
B. 1
C. 2
D. -1
E. -2
F. 3

**Correct Answer:** C
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 518**
**Given:**

```
package strings;

public class StringFromChar {
   public static void main(String[] args) {
      String myStr = "good";
      char[] myCharArr = {'g', 'o', 'o', 'd' };
      String newStr = "";
      for(char ch : myCharArr){
          newStr = newStr + ch;
      }
      boolean b1 = newStr == myStr;
      boolean b2 = newStr.equals(myStr);
      System.out.println(b1+ " " + b2);
    }
}
```

**What will it print when compiled and run?**

A. true true
B. true false
C. false true
D. false false
E. Compilation fails

**Correct Answer:** C
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 519**
**What will be the output of the following class:**

```
public class TestClass{
```

```
  public void testRefs(String str, StringBuilder sb){
    str = str + sb.toString();
    sb.append(str);
    str = null;
    sb = null;
  }
  public static void main(String[] args){
    String s = "aaa";
    StringBuilder sb = new StringBuilder("bbb");
    new TestClass().testRefs(s, sb);
    System.out.println("s="+s+" sb="+sb);
  }
}
```

A. `s=aaa sb=bbb`
B. `s=null sb=null`
C. `s=aaa sb=null`
D. `s=null sb=bbbaaa`
E. `s=aaa sb=bbbaaabbb`

**Correct Answer:** E
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 520**
**Given the following code fragment**:

```
3. class A {
4.     char[] dz = {'a','b','c','d','e'};
5. }
6. public class B {
7.     public static void main(String[] args) {
8.        A[] da = new A[3];
9.        da[0] = new A();
10.       A d = new A();
11.       da[1] = d;
12.       d = null;
13.       da[1] = null;
14.       // do stuff
15.   }
16. }
```

**Which two are true about the objects created within main(), and eligible for garbage collection when line 14 is reached?**

A. Three objects were created
B. Four objects were created
C. Five objects were created
D. Zero objects are eligible for GC
E. One object is eligible for GC
F. Two objects are eligible for GC
G. Three objects are eligible for GC

**Correct Answer:** CF
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 521**
**What will the following code print when compiled and run?**

```
abstract class Switch {
    protected abstract void switchON();
    protected abstract void switchOFF();
}

class SimpleSwitch extends Switch {
    public final void switchON() { System.out.println("Switched ON"); }
    public final void switchOFF() { System.out.println("Switched OFF"); }
}

class Fan {
     Switch sw = new SimpleSwitch();
     public void test(){
         sw.switchON();
         sw.switchOFF();
     }
}

public class Q1763 {
     public static void main(String[] args) {
         Fan f = new Fan();
         f.test();
     }
}
```

A. `Switched ON`
   `Switched OFF`
B. `class Switch will fail to compile`
C. `class SimpleSwitch will fail to compile`
D. `class Fan will fail to compile`

**Correct Answer:** A
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 522**
**Given the following contents of two java source files:**

```
package commons.text;

public class StringUtils  {
   static String capitalize(String msg){
      // return code here
   }
}
```

and

```
package ui;

//Line 1
class TestClass {
    public static void main(String[] args) {
        String output = StringUtils.capitalize("jason donovan");
        System.out.println(output);
    }
```

```
}
```

**What changes, when made independently, will enable the code to compile and run?**

A. The code compile without any change.
B. Replace `StringUtils.capitalize("jason donovan")` to `commons.text.StringUtils.capitalize("jason donovan")`
C. Add the following import on Line 1:
   `import commons.text;`
D. Add the following import on Line 1:
   `import commons.text.StringUtils;`
E. Add the following import on Line 1:
   `import commons.text.*;`
F. Add the following import on Line 1:
   `import StringUtils;`
G. None of the above

**Correct Answer:** G
**Section: Java Basics**
**Explanation**

**Explanation/Reference:**
```
capitalize(String) is not public in StringUtils; cannot be accessed from
outside package
String output = StringUtils.capitalize("jason donovan");
```

**QUESTION 523**
**Given:**

```
class Sports {

    int num_players;
    String name, ground_condition;

    Sports(int np, String sname, String sground) {
        num_players = np;
        name = sname;
        ground_condition = sground;
    }
}

class Cricket extends Sports {

    int num_umpires;
    int num_substitutes;

    //insert code here
}
```

**Which code fragment can be inserted at line "//insert code here" to enable the code to compile?**

A.
```
Cricket() {
      super(11, "Cricket", "Condidtion OK");
      num_umpires = 3;
      num_substitutes= 2;
  }
```
B.
```
Cricket() {
      super.ground_condition = "Condition OK";
      super.name= "Cricket";
      super.num_players = 11;
      num_umpires = 3;
      num_substitutes= 2;
  }
```

C.
```
Cricket() {
      this(3,2);
      super(11, "Cricket", "Condidtion OK");
  }
  Cricket(int nu, int ns) {
      this.num_umpires = nu;
      this.num_substitutes= ns;
  }
```
D.
```
Cricket() {
      this.num_umpires = 3;
      this.num_substitutes = 2;
      super(11, "Cricket", "Condidtion OK");
  }
```

**Correct Answer:** A
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 524**
**Given the fragment:**

```
String[][] arra = new String[3][];
arra[0] = new String[]{"rose", "lily"};
arra[1] = new String[]{"apple", "berry", "cherry", "grapes"};
arra[2] = new String[]{"beans", "carrot", "potato"};
// insert code fragment here
```

**Which code fragment when inserted at line '// insert code fragment here', enables the code to successfully change arra elements to uppercase?**

A.
```
for(int i = 0; i < arra.length; i++){
      for(int j = 0; j < arra[i].length; j++){
          arra[i][j] = arra[i][j].toUpperCase();
      }
  }
```
B.
```
for(int i = 0;i < 3; i++){
      for(int j=0; j < 4; j++){
          arra[i][j] = arra[i][j].toUpperCase();
      }
  }
```
C.
```
for(String a[] : arra[][]){
      for(String x : a[]){
          x.toUpperCase();
      }
  }
```
D.
```
for(int i : arra.length){
      for(String x : arra){
          arra[i].toUpperCase();
      }
  }
```

**Correct Answer:** A
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 525**
**Given:**

```
public class Test {
   static boolean bVar;
   public static void main(String[] args) {
        boolean bVar1 = true;
        int count = 8;
        do {
           System.out.println("Hello Java! " + count);
           if (count >= 7) {  bVar1 = false;}
        } while (bVar != bVar1 && count > 4);

        count -= 2;
    }
}
```

**What is the result?**

A.
```
Hello Java! 8
Hello Java! 6
Hello Java! 4
```

B.
```
Hello Java! 8
Hello Java! 6
```

C.
```
Hello Java! 8
```

D.
```
Compilation fails
```

**Correct Answer:** C
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 526**
**What will the following program print?**

```
public class TestClass{
  static boolean b;
  static int[] ia = new int[1];
  static char ch;
  static boolean[] ba = new boolean[1];
  public static void main(String args[]) throws Exception{
    boolean x = false;
    if( b ){
      x = ( ch == ia[ch]);
    }
    else x = ( ba[ch] = b );
    System.out.println(x+" "+ba[ch]);
  }
}
```

A.
```
true true
```

B.
```
true false
```

C.
```
false true
```

D.
```
false false
```

E.
```
It will not compile.
```

**Correct Answer:** D
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

**QUESTION 527**
**Given:**

```
public class TestClass{
  public static int getSwitch(String str){
      return (int) Math.round( Double.parseDouble(str.substring(1, str.length
()-1)) );
  }
  public static void main(String args []){
    switch(getSwitch(args[0])){
      case 0 : System.out.print("Hello ");
      case 1 : System.out.print("World"); break;
      default : System.out.print("Good Bye");
    }
  }
}
```

**What will be printed by the above code if it is run with command line:**

```
java TestClass --0.50
```

A. `Hello`
B. `World`
C. `Hello World`
D. `Hello World Good Bye`
E. `Good Bye`

**Correct Answer:** C
**Section: Using Operators and Decision Constructs**
**Explanation**

**Explanation/Reference:**
`str.substring(1, str.length()-1) => "--0.50".substring(1, (6-1) ) => -0.5`
`Math.round(-0.5) = 0.0` so `getSwitch(...)` returns 0 if passed an argument of "--0.50". Now, there is no "break" in case 0 of switch. so the control falls through to the next case ( i.e. case 1) after printing Hello. At case 1, it prints World. And since there is a break. default is not executed.

Observe that rounding is a standard mathematical procedure where the number that lies exactly between two numbers always rounds up to the higher one. So .5 rounds to 1 and -.5 rounds to 0.

**QUESTION 528**
**What will the following program print?**

```
class LoopTest{
    public static void main(String args[]) {
        int counter = 0;
        outer:
        for (int i = 0; i < 3; i++) {
            middle:
            for (int j = 0; j < 3; j++) {
                inner:
                for (int k = 0; k < 3; k++) {
                    if (k - j > 0) {
                        break middle;
                    }
                    counter++;
                }
            }
        }
        System.out.println(counter);
```

```
    }
}
```

A. 2
B. 3
C. 6
D. 7
E. 9

**Correct Answer:** B
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 529**
**Consider the following code:**

```
public class MyClass{

   public static void myMethod(Object o){
      System.out.println("Object Version");
   }
   public static void myMethod(java.lang.Throwable t){
      System.out.println("java.lang.Throwable Version");
   }
   public static void myMethod(java.lang.Exception e){
      System.out.println("java.lang.Exception Version");
   }
   public static void myMethod(java.lang.RuntimeException e){
      System.out.println("java.lang.RuntimeException Version");
   }
   public static void myMethod(java.lang.Error e){
      System.out.println("java.lang.Error Version");
   }

   public static void main(String args[]){
      myMethod(null);
   }
}
```

**What would be the output when the above program is compiled and run?**

A. `Object Version`
B. `java.lang.Throwable Version`
C. `java.lang.Exception Version`
D. `java.lang.Error Version`
E. `java.lang.RuntimeException Version`
F. `Compilation fails`
G. `An exception is thrown`

**Correct Answer:** F
**Section: Working with Methods and Encapsulation**
**Explanation**

**Explanation/Reference:**

```
Main.java:18: error: reference to myMethod is ambiguous
    myMethod(null);
    ^
  both method myMethod(Error) in Main and method myMethod(RuntimeException) in
Main match
```

```
1 error
```

**QUESTION 530**
**What will the following code print?**

```
public class Test {
    public int luckyNumber(int seed){
         if(seed > 10) return seed%10;
         int x = 0;
         try {
             if(seed%2 == 0) throw new Exception("No Even no.");

             else return x;
         }
         catch(Exception e){
             return 3;
         }
         finally {
             return 7;
         }<
    }
    public static void main(String args[]){
         int amount = 100, seed = 6;
         switch( new Test().luckyNumber(6) ){
              case 3: amount = amount * 2;
              case 7: amount = amount * 2;
              case 6: amount = amount + amount;
              default :
         }
         System.out.println(amount);
    }
}
```

A. It will not compile.
B. It will throw an exception at runtime.
C. 800
D. 200
E. 400

**Correct Answer:** E
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 531**
**What will the following program print?**

```
public class InitTest{
   public InitTest(){
      s1 = sM1("1");
   }
   static String s1 = sM1("a");
   String s3 = sM1("2");{
      s1 = sM1("3");
   }
   static{
      s1 = sM1("b");
   }
   static String s2 = sM1("c");
   String s4 = sM1("4");
    public static void main(String args[]){
        InitTest it = new InitTest();
```

```
    }
    private static String sM1(String s){
       System.out.println(s);  return s;
    }
}
```

A. The program will not compile.
B. It will print : a b c 2 3 4 1
C. It will print : 2 3 4 1 a b c
D. It will print : 1 a 2 3 b c 4
E. It will print : 1 a b c 2 3 4

**Correct Answer:** B
**Section: Working With Java Data Types**
**Explanation**

**Explanation/Reference:**

**QUESTION 532**
**Consider that you are writing a set of classes related to a new Data Transmission Protocol and have created your own exception hierarchy derived from java.lang.Exception as follows:**

```
enthu.trans.ChannelException
       +-- enthu.trans.DataFloodingException,
enthu.trans.FrameCollisionException
```

**You have a TransSocket class that has the following method:**

```
long connect(String ipAddr) throws ChannelException
```

**Now, you also want to write another "AdvancedTransSocket" class, derived from "TransSocket" which overrides the above mentioned method. Which of the following are valid declaration of the overriding method?**

**Please select 2 options**

A. int connect(String ipAddr) throws DataFloodingException
B. int connect(String ipAddr) throws ChannelException
C. long connect(String ipAddr) throws FrameCollisionException
D. long connect(String ipAddr) throws Exception
E. long connect(String str)

**Correct Answer:** CE
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
There are 2 important concepts involved here:
1. The overriding method must have same return type in case of primitives (a subclass is allowed in case of classes)  (Therefore, the choices returning int are not valid.) and the parameter list must be the same (The name of the parameter does not matter, just the Type is important).

2. The overriding method can throw a subset of the exception or subclass of the exceptions thrown by the overridden class. Having no throws clause is also valid since an empty set is a valid subset.

**QUESTION 533**
**Which statement, when inserted below, will cause a runtime exception ?**

```
class B {}
class B1 extends B {}
class B2 extends B {}
public class ExtendsTest {
   public static void main(String args[])   {
        B b = new B();
        B1 b1 = new B1();
        B2 b2 = new B2();
        // insert statement here
   }
}
```

A. `b = b1;`
B. `b2 = b;`
C. `b1 = (B1) b;`
D. `b2 = (B2) b1;`
E. `b1 = (B) b1;`

**Correct Answer:** C
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
`b1 = (B1) b;`
It will pass at compile time but fail at run time as the actual object referenced by b is not a B1.

`b2 = b;`
It fails at Compile time as an object referenced by b may not be a B2, so an explicit cast will be needed.

`b2 = (B2) b1;`
It will not compile because b1 can never point to an object of class B2.

`b1 = (B) b1;`
This won't compile. Another cast is needed. i.e. `b1 = (B1) (B) b1;`

**QUESTION 534**
**Given the following code:**

```
1. class X {
2.     public void printFileContent() {
3.         /* code goes here */
4.         throw new java.io.IOException();
5.     }
6. }

7. public class Test {
8.    public static void main(String[] args) {
9.         X xobj = new X();
10.        xobj.printFileContent();
11.   }
12.}
```

**Which two modifications should you make so that the code compiles successfully?**

A. Replace line 8 with:
```
public static void main(String[] args) throws Exception {
```
B. Replace line 10 with:
```
try {
   xobj.printFileContent();
}
catch(Exception ex){}
```

```
catch(java.io.Exception ex) {}
```

C. Replace line 2 with:
```
public void printFileContent() throws java.io.IOException {
```

D. Replace line 14 with:
```
throw new java.io.IOException("Exception raised");
```

E. At line 11, insert:
```
throw new java.io.IOException();
```

**Correct Answer:** AC
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 535**
**Given:**

```
public class Test {

    public static void main(String[] args) {
        String[][] chs = new String[2][];
        chs[0] = new String[2];
        chs[1] = new String[5];
        int i = 97;

        for (int a = 0; a < chs.length; a++) {
            for (int b = 0; b < chs.length; b++) {
                chs[a][b] = "" + i;
                i++;
            }
        }

        for (String[] ca : chs) {
            for (String c : ca) {
                System.out.print(c + " ");
            }
            System.out.println();
        }
    }
}
```

**What is the result?**

A.
```
97 98
99 100 null null null
```

B.
```
91 98
99 100 101 102 103
```

C. `Compilation fails.`

D. `A NullPointerException is thrown at runtime.`

E. `An ArraylndexOutOfBoundsException is thrown at runtime.`

**Correct Answer:** A
**Section: Creating and Using Arrays**
**Explanation**

**Explanation/Reference:**

**QUESTION 536**
**What can be inserted in the code below so that it will print true when run?**

```
public class TestClass {
    public static boolean checkList(List list, Predicate<List> p){
          return p.test(list);
    }
    public static void main(String[] args) {
          boolean b = //WRITE CODE HERE
          System.out.println(b);
    }
}
```

A. `checkList(new ArrayList(), al -> al.isEmpty());`
B. `checkList(new ArrayList(), ArrayList al -> al.isEmpty());`
C. `checkList(new ArrayList(), al -> return al.size() == 0);`
D. `checkList(new ArrayList(), al -> al.add("hello"));`
E. `checkList(new ArrayList(), (ArrayList al) -> al.isEmpty());`

**Correct Answer:** AD
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

`checkList(new ArrayList(), al -> al.isEmpty());`
The test method of Predicate returns a boolean. So all you need for your body part in your lambda expression is an expression that returns a boolean. isEmpty () is a valid method of ArrayList, which returns true if there are no elements in the list. Therefore, al.isEmpty() constitutes a valid body for the lambda expression in this case.

`checkList(new ArrayList(), ArrayList al -> al.isEmpty());`
You need to put the parameter list of the lambda expression in brackets if you want to use the parameter type. For example, checkList(new ArrayList(), (List al) -> al.isEmpty()); Remember that specifying the parameter type is optional ( as shown in option 1) because the compiler can figure out the parameter types by looking at the signature of the abstract method of any functional interface (here, Predicate's test method).

`checkList(new ArrayList(), al -> return al.size() == 0);`
You need to put the body withing curly braces if you want to use the return keyword. For example, checkList(new ArrayList(), al -> { return al.size() == 0; });

`checkList(new ArrayList(), al -> al.add("hello"));`
The add method of ArrayList returns a boolean. Further, it returns true if the list is altered because of the call to add. In this case, al.add("hello") indeed alters the list because a new element is added to the list.

`checkList(new ArrayList(), (ArrayList al) -> al.isEmpty());`
Predicate is typed to List (not ArrayList) in the checkList method, therefore, the parameter type in the lambda expression must also be List. It cannot be ArrayList.

**QUESTION 537**
**Consider the following**

```
public class TestClass {
    public static void main(String[] args) {
        TestClass tc = new TestClass();
        tc.myMethod();
    }

    public void myMethod() {
        yourMethod();
    }
```

```
    public void yourMethod() {
        throw new Exception();
    }
}
```

**What changes can be done to make the above code compile?**

A. Change declaration of main to:
`public static void main(String[] args) throws Exception`
B. Change declaration of myMethod to
`public void myMethod throws Exception`
C. Change declaration of yourMethod to
`public void yourMethod throws Exception`
D. Change declaration of main and yourMethod to :
`public static void main(String[] args) throws Exception`
and
`public void yourMethod throws Exception`
E. Change declaration of all the three method to include `throws Exception.`

**Correct Answer:** E
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**

**QUESTION 538**
**Consider the following program...**

```
class ArrayTest {
   public static void main(String[] args){
        int ia[][] = { {1, 2}, null };
        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 2; j++)
                System.out.println(ia[i][j]);
   }
}
```

**Which of the following statements are true?**

A. `It will not compile.`
B. `It will throw an ArrayIndexOutOfBoundsException at Runtime.`
C. `It will throw a NullPointerException at runtime.`
D. `It will compile and run without throwing any exceptions.`
E. `None of the above`

**Correct Answer:** C
**Section: Handling Exceptions**
**Explanation**

**Explanation/Reference:**
`It will throw a NullPointerException for ia[1][0] because ia[1] is null. Note that null is not same as having less number of elements in an array than expected. If you try to access ia[2][0], it would have thrown ArrayIndexOutOfBoundsException because the length of ia is only 2 and so ia[2] tries to access an element out of that range. ia[2] is not null, it simply does not exist.`

**QUESTION 539**
**Which of the following statements are correct regarding a functional interface?**

A. It has exactly one method and it must be abstract.
B. It has exactly one method and it may or may not be abstract.
C. It must have exactly one abstract method and may have other default or static methods.
D. It must have exactly one static method and may have other default or abstract methods.

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
A functional interface is an interface that contains exactly one abstract method. It may contain zero or more default methods and/or static methods. Because a functional interface contains exactly one abstract method, you can omit the name of that method when you implement it using a lambda expression.

For example, consider the following interface -

```
interface Predicate<T> {
   boolean test(T t);
}
```

**QUESTION 540**
**Which of the following are correct about java.util.function.Predicate?**

A. It is an interface that has only one method with the signature -
   `public void test(T t);`
B. It is an interface that has only one method with the signature -
   `public boolean test(T t);`
C. It is an abstract class that has only one abstract method with the signature -
   `public abstract void test(T t);`
D. It is an abstract class that has only one abstract method with the signature -
   `public abstract boolean test(T t);`

**Correct Answer:** B
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 541**
**Which of the following lambda expressions can be used to invoke a method that accepts a java.util.function.Predicate as an argument? (Select 2 options)**

A. `x -> System.out.println(x)`
B. `x -> System.out.println(x);`
C. `x -> x == null`
D. `() -> true`
E. `x->true`

**Correct Answer:** CE
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 542**
**Given the following functional interface:**

```
interface Runner {
    public void run();
}
```

**Which of the following are valid lambda expressions? Select 2 options**

A. `-> System.out.println("running...");`
B. `void -> System.out.println("running...")`
C. `() -> System.out.println("running...")`
D. `() -> { System.out.println("running..."); return; }`
E. `(void) -> System.out.println("running...")`
F. `  -> System.out.println("running...")`

**Correct Answer:** CD
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
Runner is a valid functional interface because it has exactly one abstract method. Since this method does not take any parameter, the parameter list part of the lambda expression must be (). Further, since it does not return anything, the body part should ideally be such that it does not return anything either. Thus, you can either use a method call that returns void or some code enclosed within { and } that does not return anything.

**QUESTION 543**
**Which of the following lines can be inserted at line 11 to print true?**

```
10: public static void main(String[] args) {
11:     // INSERT CODE HERE
12: }
13: private static boolean test(Predicate<Integer> p) {
14:     return p.test(5);
15: }
```

A. `System.out.println(test(i -> i == 5));`
B. `System.out.println(test(i -> {i == 5;}));`
C. `System.out.println(test((i) -> i == 5));`
D. `System.out.println(test((int i) -> i == 5);`
E. `System.out.println(test((int i) -> {return i == 5;}));`
F. `System.out.println(test((i) -> {return i == 5;}));`

**Correct Answer:** ACF
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
- The only functional programming interface you need to memorize for the exam is Predicate. It takes a single parameter and returns a boolean.
- Lambda expressions with one parameter are allowed to omit the parentheses around the parameter list, making options A and C correct.
- The return statement is optional when a single statement is in the body, making option F correct.
- Option B is incorrect because a return statement must be used if braces are included around the body.
- Options D and E are incorrect because the type is Integer in the predicate and int in the lambda.

**QUESTION 544**
**Given the code fragments:**

```
public class Employee {

    String name;
    double salary;

    public Employee(String n, double s) {
        name = n;
        salary = s;
    }

    public String getName() {
        return name;
    }
    public double getSalary() {
        return salary;
    }
}

import java.util.*;
import java.util.function.*;

public class Test {

    public static void main(String[] args) {
        List<Employee> list = Arrays.asList(new Employee("Jason", 23000), new
Employee("Donovan", 18000), new Employee("James", 35000));
        // line n1
    }

    public static void showEmployeesThat(List<Employee> list,
Predicate<Employee> eval) {
        for (Employee e : list) {
            if (eval.test(e)) {
                System.out.println(e.getName() + " ");
            }
        }
    }
}
```

**Which code fragment, when inserted at line n1, enables the code to compile?**

A. `showEmployeesThat(list, () -> e.getSalary() > 30000);`
B. `showEmployeesThat(list, Employee e -> e.getSalary() > 30000);`
C. `showEmployeesThat(list, e -> e.getSalary() > 30000);`
D. `showEmployeesThat(list, (Employee e) -> { e.getSalary() -> 30000; });`

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

```
package java.util.function;

@FunctionalInterface
public interface Predicate<T> {

    boolean test(T t);
}
```

**QUESTION 545**
**Given the code fragments:**

```
public class Animal {
    private String species;
```

```
    private boolean canSwim;

    public Animal(String speciesName, boolean swimmer) {
        species = speciesName;
        canSwim = swimmer;
    }
    public boolean canSwim() { return canSwim; }
    public String toString() { return species; }
}


import java.util.*;
import java.util.function.*;

public class PredicateSearch {

    public static void main(String[] args) {
        List<Animal> animals = new ArrayList<>();
        animals.add(new Animal("fish", true));
        animals.add(new Animal("kangaroo", false));
        animals.add(new Animal("rabbit", false));
        animals.add(new Animal("turtle", true));
        // line n1
    }

    private static void printAnimalsThat(List<Animal> animals,
Predicate<Animal> checker) {
        for (Animal animal : animals) {
           if (checker.test(animal))
               System.out.print(animal + " ");
        }
        System.out.println();
    }
}
```

**Which code fragment, when inserted at line n1, enables the code to compile? (Select 4 options)**

A. `printAnimalsThat(animals, () -> a.canSwim());`
B. `printAnimalsThat(animals, Animal a -> a.canSwim());`
C. `printAnimalsThat(animals, (a) -> a.canSwim());`
D. `printAnimalsThat(animals, (Animal a) -> { a.canSwim(); });`
E. `printAnimalsThat(animals, (Animal a) -> a.canSwim());`
F. `printAnimalsThat(animals, a -> a.canSwim());`
G. `printAnimalsThat(animals, a -> {return a.canSwim();});`

**Correct Answer:** CEFG
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

```
package java.util.function;

@FunctionalInterface
public interface Predicate<T> {

    boolean test(T t);
}
```

**QUESTION 546**
**Given :**

```
//In Data.java
public class Data {
```

```
   int value;
   Data(int value){
      this.value = value;
   }
   public String toString(){
      return "" + value;
   }
}
```

**and the following code fragments:**

```
public  void filterData(ArrayList<Data> dataList, Predicate<Data> p) {
    Iterator<Data> i = dataList.iterator();
    while(i.hasNext()){
        if(p.test(i.next())){
           i.remove();
        }
    }
}
....
List<Data> a1 = new ArrayList<>();
Data d = new Data(1);
al.add(d);
d = new Data(2);
al.add(d);
d = new Data(3);
al.add(d);
//INSERT METHOD CALL HERE
System.out.println(al);
```

**Which of the following options can be inserted above so that it will print [1, 3]?**

A. `filterData(al, d -> d.value % 2 == 0 );`
B. `filterData(al, (Data x) -> x.value % 2 == 0 );`
C. `filterData(al, (Data y) -> y.value % 2  );`
D. `filterData(al, d -> return d.value % 2 );`

**Correct Answer:** B
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
Syntactically, this lambda expression is correct. However, remember that a lambda expression does not create a new scope for variables. Therefore, you cannot reuse the variable names that have already been used to define new variables in your argument list .
Here, observe that the variable d is already defined so your argument list cannot use d as a variable name. It would be like defining the same variable twice in the same scope.

**QUESTION 547**
**Given:**

```
interface Process {
    public void process(int a, int b);
}

public class Data {
    int value;
    Data(int value){
        this.value = value;
    }
}
```

**and the following code fragments:**

```
public void processList(ArrayList<Data> dataList, Process p){
    for(Data d: dataList){
        p.process(d.value, d.value);
    }
}

....
ArrayList<Data> al = new ArrayList<Data>();
al.add(new Data(1));
al.add(new Data(2));
al.add(new Data(3));
//INSERT METHOD CALL HERE
```

**Which of the following options can be inserted above so that it will print 1 4 9? Select 3 options**

A. `processList(al, a, b->System.out.println(a*b));`
B. `processList(al, (int a, int b)->System.out.println(a*b) );`
C. `processList(al, (int a, int b)->System.out.println(a*b); );`
D. `processList(al, (a, b)->System.out.println(a*b));`
E. `processList(al, (a, b) ->{  System.out.println(a*b); } );`

**Correct Answer:** BDE
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 548**
**What will the following code print when run?**

```
import java.util.function.Predicate;

class Employee {
   int age;   //1
}

public class TestClass {
   public static boolean validateEmployee(Employee e, Predicate<Employee> p)
{
      return p.test(e);
   }
   public static void main(String[] args) {
      Employee e = new Employee(); //2
      System.out.println(validateEmployee(e, e->e.age<10000)); //3
   }
}
```

A. It will fail to compile at line marked //1
B. It will fail to compile at line marked //2
C. It will fail to compile at line marked //3
D. It will compile fine and print true when run.
E. It will compile fine and print false when run.

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
Remember that the parameter list part of a lambda expression declares new

variables that are used in the body part of that lambda expression. However, a lambda expression does not create a new scope for variables. Therefore, you cannot reuse the local variable names that have already been used in the enclosing method to declare the variables in you lambda expression. It would be like declaring the same variable twice. Here, the main method has already declared a variable named e. Therefore, the parameter list part of the lambda expression must not declare another variable with the same name. You need to use another name.

For example, if you change //3 to the following, it will work.
System.out.println(validateEmployee(e, x->x.age<10000));

It would print true.

**QUESTION 549**
**Which of the following classes should you use to represent just a date without any time or zone information?**

A. java.util.Date
B. java.sql.Date
C. java.time.Date
D. java.time.LocalDate

**Correct Answer:** D
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
Java 8 introduces a new package java.time to deal with dates. The old classes such as java.util.Date are not recommended anymore.

Briefly:
java.time Package: This is the base package of new Java Date Time API. All the commonly used classes such as LocalDate, LocalTime, LocalDateTime, Instant, Period, Duration are part of this package. All of these classes are immutable and thread safe.

java.time.format Package: This package contains classes used for formatting and parsing date time objects such as java.time.format.DateTimeFormatter.

(The following two are not important for the exam.)

java.time.zone Package: This package contains classes for supporting different time zones and their rules.

java.time.chrono Package: This package defines generic APIs for non ISO calendar systems. We can extend AbstractChronology class to create our own calendar system.

java.time.temporal Package: This package contains temporal objects and we can use it for find out specific date or time related to date/time object. For example, we can use these to find out the first or last day of the month. You can identify these methods easily because they always have format “withXXX”.

**QUESTION 550**
**Which of the following print out a date representing April 1, 2015?**

A. System.out.println(LocalDate.of(2015, Calendar.APRIL, 1));
B. System.out.println(LocalDate.of(2015, Month.APRIL, 1));
C. System.out.println(LocalDate.of(2015, 3, 1));
D. System.out.println(LocalDate.of(2015, 4, 1));
E. System.out.println(new LocalDate(2015, 3, 1));

F. `System.out.println(new LocalDate(2015, 4, 1));`

**Correct Answer:** BD
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
The new date APIs added in Java 8 use static methods rather than a constructor to create a new date, making options E and F incorrect. The months are indexed starting with 1 in these APIs, making options A and C incorrect.

Option A uses the old Calendar constants which are indexed from 0. Therefore, options B and D are correct.

**QUESTION 551**
**Which of the following are valid ways to create a LocalDateTime?**

A. `java.time.LocalDate.parse("2015-01-02");`
B. `java.time.LocalDateTime.parse("2015-01-02");`
C. `java.time.LocalDateTime.of(2015, 10, 1, 10, 10);`
D. `java.time.LocalDateTime.of(2015, "January", 1, 10, 10);`

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
To create an instance of LocalDateTime, you need to use the methods in LocalDateTime class. Methods in LocalDate class create LocalDate instances. Similarly, methods in LocalTime class create LocalTime instances.

LocalDateTime requires date as well as time. Here, you just have a date in the input so it will throw a java.time.format.DateTimeParseException.

`java.time.LocalDateTime.parse("2015-01-02T17:13:50");` would be valid.

All parameters should be ints. For the month argument, you can either pass the numbers 1 to 12 (and not 0 to 11) or use constants such as `java.time.Month.JANUARY`.

**QUESTION 552**
**Given:**

```
LocalDate d1 = LocalDate.parse("2015-02-05");
LocalDate d2 = LocalDate.of(2015, 2, 5);
LocalDate d3 = LocalDate.now();
System.out.println(d1);
System.out.println(d2);
System.out.println(d3);
```

**Assuming that the current date on the system is 5th Feb, 2015, which of the following will be a part of the output?**

A. `5th Feb, 2015`
B. `2015-02-05T00:00:00`
C. `02/05/2015`
D. `05/02/2015`
E. `java.time.format.DateTimeParseException`
F. `None of the above.`

**Correct Answer:** F

**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
All the three printlns will produce 2015-02-05.

**QUESTION 553**
**What will the following code print when compiled and run?**

```
import java.time.*;
import java.time.format.*;

public class DateTest {
     public static void main(String[] args){  //1
         LocalDateTime greatDay = LocalDateTime.parse("2015-01-01"); //2

         String greatDayStr = greatDay.format(DateTimeFormatter.ISO_DATE_TIME);
//3
         System.out.println(greatDayStr); //4
     }
}
```

A. //1 will not compile because of lack of throws clause.
B. //2 will not compile because of invalid date string.
C. //2 will throw an exception at run time.
D. It will print 2015-01-01T00:00:00
E. It will print null.

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
It will throw a DateTimeException because it doesn't have time component.
Exception in thread "main" java.time.format.DateTimeParseException:
    Text '2015-01-01' could not be parsed at index 10.

A String such as 2015-01-01T17:13:50 would have worked.

Operations in the new date/time related classes throw
java.time.DateTimeException, which extends from RuntimeException. Therefore,
this exception is not required to be caught or declared in the throws clause.

**QUESTION 554**
**What will the following lines of code print**

```
LocalDate dt = LocalDate.parse("2015-01-01");
dt.minusMonths(1);
dt.minusDays(1);
dt.plusYears(1);
System.out.println(dt);
```

A. Compilation error.
B. Exception at run time.
C. 2015-12-31
D. 2015-11-30
E. 2015-01-01

**Correct Answer:** E
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
The numbering for days and months starts with 1. Rest is simple math.

Observe that most of the methods of LocalDate (as well as LocalTime and LocalDateTime) return an object of the same class. This allows you to chain the calls as done in this question. However, these methods return a new object. They don't modify the object on which the method is called.

**QUESTION 555**
**What will the following lines of code print**

```
LocalDate dt = LocalDate.parse("2015-01-01").minusMonths(1).minusDays
(1).plusYears(1);
System.out.println(dt);
```

A. Compilation error.
B. Exception at run time.
C. 2015-12-31
D. 2015-11-30

**Correct Answer:** D
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
The numbering for days and months starts with 1. Rest is simple math.

Observe that most of the methods of LocalDate (as well as LocalTime and LocalDateTime) return an object of the same class. This allows you to chain the calls as done in this question. However, these methods return a new object. They don't modify the object on which the method is called.

**QUESTION 556**
**What will the following line of code print?**

```
LocalDate d = LocalDate.of(2015, Month.JANUARY, 01);
System.out.println(d.format(DateTimeFormatter.ISO_DATE_TIME));
```

A. 01 Jan 2015
B. 01 January 2015 00:00:00
C. 2015-01-01
D. 2015-01-01T00:00:00
E. Exception at run time.

**Correct Answer:** E
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
Observe that you are creating a LocalDate and not a LocalDateTime. LocalDate doesn't have time component and therefore, you cannot format it with a formatter that expects time component such as DateTimeFormatter.ISO_DATE_TIME.

Thus, it will print java.time.temporal.UnsupportedTemporalTypeException: Unsupported field: HourOfDay exception message.

If you use DateTimeFormatter.ISO_DATE, it will print 2015-01-01 .

Also, remember that a LocalDateTime object can be formatted using a DateTimeFormatter.ISO_DATE though.

**QUESTION 557**

**Which of the following options correctly add 1 month and 1 day to a given LocalDate -**

```
public LocalDate process(LocalDate ld) {
    //INSERT CODE HERE
    return ld2;
}
```

A. `LocalDate ld2 = ld.plus(Period.ofMonths(1).ofDays(1));`
B. `LocalDate ld2 = ld.plus(new Period(0, 1, 1));`
C. `LocalDate ld2 = ld.plus(new Period(31)).plus(new Period(1));`
D. `LocalDate ld2 = ld.plus(Period.of(0, 1, 1));`

**Correct Answer:** D
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
public static Period of(int years, int months, int days)
Obtains a Period representing a number of years, months and days. This creates an instance based on years, months and days.

**QUESTION 558**
**Given the code fragment:**

```
public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();

    list.add("Java SE");
    list.add("Java EE");
    list.add("Java ME");
    list.add("Java SE");
    list.add("Java EE");

    list.remove("Java SE");

    System.out.print("Values are : "  + list);
}
```

**What is the result?**

A. `Values are: [Java EE, Java ME]`
B. `Values are: [Java EE, Java EE, Java ME]`
C. `Values are: [Java EE, Java ME, Java EE]`
D. `Values are: [Java SE, Java EE, Java ME, Java EE]`
E. `Values are: [Java EE, Java ME, Java SE, Java EE]`
F. `Compilation fails.`
G. `None of the above.`

**Correct Answer:** E
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 559**
**Given the code fragment:**

```
public static void main(String[] args) {
     List<String> names = new ArrayList<>();
     names.add("Robb");
```

```
      names.add("Bran");
      names.add("Rick");
      names.add("Bran");

      if (names.remove("Bran")) {
            names.remove("Jon");
      }
      System.out.println(names);
}
```

**What is the result?**

A. `[Robb, Rick, Bran]`
B. `[Robb, Rick]`
C. `[Robb, Bran, Rick, Bran]`
D. `An exception is thrown at runtime`

**Correct Answer:** A
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 560**
**Given the code fragment:**

```
List<String> fls = new ArrayList<>();
fls.add("jasmine");
fls.add("rose");
fls.add("lotus");
fls.remove(2);
fls.set(2, "lily");
System.out.println(fls);
```

**What is the result?**

A. `A runtime exception is thrown.`
B. `[jasmine, lily, lotus]`
C. `[jasmine, rose, lotus, lily]`
D. `[jasmine, rose, lily]`

**Correct Answer:** A
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
`Exception in thread "main" java.lang.IndexOutOfBoundsException: Index: 2, Size: 2`

**QUESTION 561**
**Given the code fragment:**

```
List<String> names = new ArrayList<>();
names.add("Joel");
names.add("Paul");
names.remove(0);
names.remove(0);
System.out.println(names.isEmpty());
names.add("Joel");
names.add("Paul");
names.clear();
```

```
System.out.println(names.isEmpty());
```

**What is the result?**

A.
```
false
false
```
B.
```
false
true
```
C.
```
true
true
```
D.
```
A runtime exception is thrown
```

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 562**
**In which sequence will the characters a, b, and, c be printed by the following program?**

```
import java.util.* ;

public class ListTest {
   public static void main(String args[]){
      List s1 = new ArrayList( );
      s1.add("a");
      s1.add("b");
      s1.add(1, "c");
      List s2 = new ArrayList(  s1.subList(1, 1) );
      s1.addAll(s2);
      System.out.println(s1);
   }
}
```

A. The sequence a, b, c is printed.
B. The sequence a, b, c, b is printed.
C. The sequence a, c, b, c is printed.
D. The sequence a, c, b is printed.
E. None of the above.

**Correct Answer:** D
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
add(1, "c") will insert 'c' between 'a' and 'b' . subList(1 , 1) will return an empty list.

```
First, "a" and "b" are appended to an empty list. Next, "c" is added between
"a" and "b".
Then a new list s2 is created using the sublist view allowing access to
elements from index 1 to index 1(exclusive) (i.e. no elements ).
Now, s2 is added to s1.
So s1 remains :a, c, b
```

**QUESTION 563**
**What will the following code print?**

```
List s1 = new ArrayList( );
s1.add("a");
s1.add("b");
```

```
s1.add("c");
s1.add("a");
System.out.println(s1.remove("a") + " " + s1.remove("x"));
```

A. 1 0
B. 2 -1
C. 2 0
D. 1 -1
E. true false

**Correct Answer:** E
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
ArrayList's remove(Object ) method returns a boolean. It returns true if the element is found in the list and false otherwise. The JavaDoc API description of this method is important for the exam -

**QUESTION 564**
**What will the following code print?**

```
List s1 = new ArrayList( );
s1.add("a");
s1.add("b");
s1.add("c");
s1.add("a");
if(s1.remove("a")){
    if(s1.remove("a")){
       s1.remove("b");
    }
    else {
       s1.remove("c");
    }
}
System.out.println(s1);
```

A. [b]
B. [c]
C. [b, c, a]
D. [a, b, c, a]
E. Exception at runtime

**Correct Answer:** B
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 565**
**What will the following code print when compiled and run?**

```
import java.util.*;

public class TestClass {
    public static void main(String[] args) throws Exception {
           List list = new ArrayList();
           list.add("val1"); //1
           list.add(2, "val2"); //2
           list.add(1, "val3"); //3
           System.out.println(list);
```

```
    }
}
```

A. It will not compile.
B. It will throw an exception at run time because of line //1
C. It will throw an exception at run time because of line //2
D. It will throw an exception at run time because of line //3
E. null

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
This line is trying to put a value in an ArrayList at index 2 (i.e. 3rd position). To be able to put a value at index 2, the ArrayList must have atleast 2 values already. Therefore, it will throw java.lang.IndexOutOfBoundsException exception.

**QUESTION 566**
**What will the following code print when compiled and run?**

```
import java.util.*;

public class TestClass {
    public static void main(String[] args) throws Exception {
        ArrayList<String> al = new ArrayList<String>();
        al.add("111");
        al.add("222");
        System.out.println(al.get(al.size()));
     }
}
```

A. It will not compile.
B. It will throw a NullPointerException at run time.
C. It will throw an IndexOutOfBoundsException at run time.
D. 222
E. null

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
size() method of ArrayList returns the number of elements. Here, it returns 2. Since numbering in ArrayList starts with 0. al.get(2) will cause an IndexOutOfBoundsException to be thrown because only 0 and 1 are valid indexes for a list of size 2.

**QUESTION 567**
**What will the following code print when compiled and run?**

```
import java.util.*;

public class TestClass {
    public static void main(String[] args) throws Exception {
        List  al = new ArrayList(); //1
        al.add(111); //2
        System.out.println(al.get(al.size()));  //3
     }
}
```

A. It will not compile.
B. It will throw an exception at run time because of line //1
C. It will throw an exception at run time because of line //2
D. It will throw an exception at run time because of line //3
E. null.

**Correct Answer:** D
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
It will throw an IndexOutOfBoundsException at run time because of this line. The size() method of ArrayList returns the number of elements. Here, it returns 1. Since numbering in ArrayList starts with 0. al.get(1) will cause an IndexOutOfBoundsException to be thrown because only 0 is a valid index for a list of size 1.

**QUESTION 568**
**Given the code fragment:**

```
int num[] = new int[3];
num[1] = 10;
num[2] = 15;
List<Integer> lst = new ArrayList<>(3);
lst.add(10);
lst.add(15);
System.out.println(num);
System.out.println(lst);
```

**What is the result?**

A. 10, 15
   [10, 15]
B. 0, 10, 15
   [10, 15, null]
C. a memory address in hexadecimal number format
   [10, 15]
D. a memory address1 in hexadecimal number format
   a memory address2 in hexadecimal number format

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 569**
**Given the code fragment:**

```
int[] num = new int[2];
num[0] = 10;
num[1] = 15;

List<Integer> lst = new ArrayList<>(2);
lst.add(10);
lst.add(15);

num[1] = 20;
lst.add(20);

for (int x : num) { System.out.print(x + " "); }
```

```
System.out.println("");
for (int y : lst) { System.out.print(y + " "); }
```

**What is the result?**

A. A compilation error occurs.
B. 10 20
   10 20
C. 10 20
   10 15 20
D. A runtime exception is thrown.

**Correct Answer:** C
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**

**QUESTION 570**
**Given:**

```
abstract class Vehicle{ }
interface Drivable{ }
class Car extends Vehicle implements Drivable{ }
class SUV extends Car { }
```

**Which of the following options will fail to compile?**

A.
```
ArrayList<Vehicle> al1 = new ArrayList<>();
al1.add(new SUV());
```
B.
```
ArrayList<Drivable> al2 = new ArrayList<>();
al2.add(new Car());
```
C.
```
ArrayList<Drivable> al3 = new ArrayList<>();
al3.add(new SUV());
```
D.
```
ArrayList<SUV> al4 = new ArrayList<>();
al4.add(new Car());
```
E.
```
ArrayList<Vehicle> al5 = new ArrayList<>();
al5.add(new Car());
```

**Correct Answer:** D
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 571**
**Given:**

```
abstract class Vehicle{ }
interface Drivable{ }
class Car extends Vehicle implements Drivable{ }
class SUV extends Car { }
```

**Which of the following options will compile? Please select 2 options**

A.
```
ArrayList<Vehicle> al1 = new ArrayList<>();
SUV s = al1.get(0);
```
B.
```
ArrayList<Drivable> al2 = new ArrayList<>();
Car c1 = al2.get(0);
```

C.
```
ArrayList<SUV> al3 = new ArrayList<>();
Drivable d1 = al3.get(0);
```
D.
```
ArrayList<SUV> al4 = new ArrayList<>();
Car c2 = al4.get(0);
```
E.
```
ArrayList<Vehicle> al5 = new ArrayList<>();
Drivable d2 = al5.get(0);
```

**Correct Answer:** CD
**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**
This question is based on your understand of is-a relationship. When class A extends or implements B directly or indirectly, you can say that A is-a B.
Here, Car directly extends Vehicle and directly implements Drivable. Therefore, a Car is-a Vehicle and a Car is-a Drivable.
Similarly, an SUV is-a Car and since Car is-a Vehicle and is-a Drivable, SUV is also a Vehicle and a Drivable.

**QUESTION 572**
**Given the class definitions:**

```
class Alpha {
     public String doStuff(String msg) {
         return msg;
     }
}

class Beta extends Alpha {
     public String doStuff(String msg) {
         return msg.replace('a', 'e');
     }
}

class Gamma extends Beta {
     public String doStuff(String msg) {
          return msg.substring(2);
     }
}
```

**And the code fragment of the main() method:**

```
12. List<Alpha> strs = new ArrayList<Alpha>();
13. strs.add(new Alpha());
14. strs.add(new Beta());
15. strs.add(new Gamma());
16. for (Alpha t : strs) {
17.     System.out.println(t.doStuff("Java"));
18. }
```

**What is the result?**

A.
```
Java
Java
Java
```
B.
```
Java
Jeve
va
```
C.
```
Java
Jeve
ve
```
D.
```
Compilation fails.
```

**Correct Answer:** B

**Section: Working with Inheritance**
**Explanation**

**Explanation/Reference:**

**QUESTION 573**
**Which of the following are benefits of an array over an ArrayList? Select 2 options**

A. It consumes less memory.
B. Accessing an element in an array is faster than in ArrayList.
C. You do not have to worry about thread safety.
D. It implements Collection interface and can thus be passed where ever a Collection is required.

**Correct Answer:** AB
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
It consumes less memory.
This is an ambiguous option because in certain situation an ArrayList may consume a little bit more memory than an array (because of additional internal data structure and pointers), while in some other situation it may consume less (when your array is only half full).

Accessing an element in an array is faster than in ArrayList.
Although very little, but a direct array access using an index is faster than calling a method on ArrayList.

You do not have to worry about thread safety.
Neither an ArrayList nor an array is thread safe. If you have multiple threads trying to add and remove elements from an ArrayList or an array, you have to write additional code to ensure thread safety.

It implements Collection interface and can thus be passed where ever a Collection is required.
arrays do not implement Collection interface. ArrayList does. This is actually an advantage of an ArrayList over an array.

An ArrayList resized dynamically at run time as per the situation. An array cannot be resized once created. This reduces the amount of boiler plate code that is required to do the same task using an array. Some candidates have reported getting a similar question with ambiguous options such as "An ArrayList implements Collection API". It is anybody's guess as to what is the correct answer.

**QUESTION 574**
**Which of the following are benefits of ArrayList over an array?**

A. You do not have to worry about the size of the ArrayList while appending elements.
B. It consumes less memory space.
C. You do not have to worry about thread safety.
D. It allows you to write type safe code.

**Correct Answer:** A
**Section: Working with Selected classes from the Java API**
**Explanation**

**Explanation/Reference:**
An ArrayList resized dynamically at run time as per the situation. An array

cannot be resized once created. This reduces the amount of boiler plate code that is required to do the same task using an array.

An ArrayList, just like an array is not thread safe. If you have multiple threads trying to add and remove elements from an ArrayList, you have to write additional code to ensure thread safety.

Since ArrayList is a generics enabled class, it helps you write type safe code. For example, if you have: ArrayList<String> al = new ArrayList<>();
al.add(new Integer(10));

will not compile because the compiler knows that al can only contain Strings.

However, this is not an advantage over an array because arrays are also type safe. For example, if you have: String[] sa = new String[10]; you cannot do sa [0] = new Integer(10); either. But you can do Object[] oa = sa; and oa[0] = new Integer(10); This will compile fine but will fail at runtime. This is a hole in the type safety provided by arrays.

**QUESTION 575**
**What will the following code print when compiled and run?**

```
int[][] ab = { {1, 2, 3}, {4, 5} };
for(int i=0; i<ab.length; i++) {
    for(int j=0; j<ab[i].length; j++){
         System.out.print(ab[i][j]+" ");
         if(ab[i][j] == 2){
             break;
         }
    }
    continue;
}
```

A. 1 2 3 4 5
B. 1 2
C. 1 3 4 5
D. 1 2 4 5
E. 2 3 5
F. It will not compile.

**Correct Answer:** D
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**
For answering such questions, it is best to use a pen and a paper and start executing the code line by line.
The i of the outer loop runs from 0 to < ab.length, which is 2. i.e. i will be 0 and then 1.
The j of the inner loop runs from 0 to < ab[i].length, which is 3 for the first iteration of the outer for loop and 2 for the second iteration of the outer for loop.
Thus, for the first iteration of the outer for loop - the inner for loop prints ab[0][0] i.e. 1 , ab[0][1] i.e. 2 and then since the if condition is satisfied, the inner loop ends and thesecond iteration of the outer for loop begins.
For the second iteration of the outer for loop - the inner for loop prints ab [1][0] i.e. 4 , ab[1][1] i.e. 5.

The continue statement in this case is redundant because there is no statement left to execute after continue in the for loop anyway.

**QUESTION 576**
**What will the following program snippet print?**

```
int i=0, j=11;
do{
  if(i > j) { break; }
  j--;
}while( ++i < 5);

System.out.println(i+"  "+j);
```

A. 5 5
B. 5 6
C. 6 6
D. 6 5
E. 4 5

**Correct Answer:** B
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**
++i < 5 means, increment the value of i and then compare with 5. Now, Try to work out the values of i and j at every iteration. To start with, i=0 and j=11. At the time of evaluation of the while condition, i and j are as follows:
j = 10 and i=1 (loop will continue because i<5) (Remember that comparison will happen AFTER increment i because it is ++i and not i++.
j = 9 and i=2 (loop will continue because i<5).
j = 8 and i=3 (loop will continue because i<5).
j = 7 and i=4 (loop will continue because i<5).
j = 6 and i=5 (loop will NOT continue because i not <5).

So it will print 5 6. (It is print i first and then j).

**QUESTION 577**
**What will the following program print?**

```
class Test{
   public static void main(String args[]){
      int var = 20, i=0;
      do{
         while(true){
             if( i++ > var) break;
         }
      } while(i < var--);
      System.out.println(var);
   }
}
```

A. 19
B. 20
C. 21
D. 22
E. It will enter an infinite loop.

**Correct Answer:** A
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**
When the first iteration of outer do-while loop starts, var is 20. Now, the inner loop executes till i becomes 21.
Now, the condition for outer do-while is checked, while( 22 < 20 ), [i is 22 because of the last i++>var check], thereby making var 19. And as the condition is false, the outer loop also ends.

So, 19 is printed.

**QUESTION 578**
**What will the following program print?**

```
class Test{
    public static void main(String args[]){
        int i=0, j=0;
        X1: for(i = 0; i < 3; i++){
            X2: for(j = 3; j > 0; j--){
                if(i < j) continue X1;
                else break X2;
            }
        }
        System.out.println(i+" "+j);
    }
}
```

A. 0 3
B. 0 2
C. 3 0
D. 3 3
E. 2 2

**Correct Answer:** D
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

**QUESTION 579**
**What will the following program print?**

```
class LoopTest{
    public static void main(String args[]) {
        int counter = 0;
        outer:
        for (int i = 0; i < 3; i++) {
            middle:
            for (int j = 0; j < 3; j++) {
                inner:
                for (int k = 0; k < 3; k++) {
                    if (k - j > 0) {
                        break middle;
                    }
                    counter++;
                }
            }
        }
        System.out.println(counter);
    }
}
```

A. 2
B. 3
C. 6
D. 7
E. 9

**Correct Answer:** B

**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**
To understand how this loop works let us put some extra print statements in the innermost loop:

```
System.out.println("i="+i+" j="+j+" k="+k);
if(k-j>0){
     System.out.println("breaking middle "+j);
     break middle;
}
counter++;
```

This is what it prints:

```
i=0 j=0 k=0
i=0 j=0 k=1
breaking middle 0
i=1 j=0 k=0
i=1 j=0 k=1
breaking middle 0
i=2 j=0 k=0
i=2 j=0 k=1
breaking middle 0
3
```

The key is that the middle loop is broken as soon as k-j becomes > 0. This happens on every second iteration of inner loop when k is 1 and j is 0. Now, when middle is broken inner cannot continue. So the next iteration of outer starts.

**QUESTION 580**
**What will be the result of attempting to compile and run the following program?**

```
public class TestClass{
   public static void main(String args[]){
      int x  = 0;
      labelA:   for (int i=10; i<0; i--){
         int j = 0;
         labelB:
         while (j < 10){
            if (j > i) break labelB;
            if (i == j){
               x++;
               continue labelA;
            }
            j++;
         }
         x--;
      }
      System.out.println(x);
   }
}
```

A. It will not compile.
B. It will go in infinite loop when run.
C. The program will write 10 to the standard output.
D. The program will write 0 to the standard output.
E. None of the above.

**Correct Answer:** D
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**
This is just a simple code that is meant to confuse you.
Notice the for statement: for(int i=10; i<0; i--). i is being initialized to 10

and the test is i<0, which is false. Therefore, the control will never get inside the for loop, none of the weird code will be executed, and x will remain 0, which is what is printed.

**QUESTION 581**
**What will the following code print?**

```
void crazyLoop(){
   int c = 0;
   JACK: while (c < 8){
       JILL: System.out.println(c);
       if (c > 3) break JILL; else c++;
   }
}
```

A. It will not compile
B. It will throw an exception at runtime.
C. It will print numbers from 0 to 8
D. It will print numbers from 0 to 3
E. It will print numbers from 0 to 4

**Correct Answer:** A
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**
Because break JILL; would be valid only when it is within the block of code under the scope of the label JILL.
In this case, the scope of JILL extends only up till System.out.println(c); and break JILL; is out of the scope of the label.

**QUESTION 582**
**What will be the result of attempting to compile and run the following program?**

```
class TestClass{
   public static void main(String args[]){
      int i = 0;
      loop :         // 1
      {
         System.out.println("Loop Lable line");
         try{
            for (  ;  true ;  i++ ){
               if( i >5) break loop;       // 2
            }
         }
         catch(Exception e){
            System.out.println("Exception in loop.");
         }
         finally{
            System.out.println("In Finally");      // 3
         }
      }
   }
}
```

A. Compilation error at line 1 as this is an invalid syntax for defining a label.
B. Compilation error at line 2 as 'loop' is not visible here.
C. No compilation error and line 3 will be executed.
D. No compilation error and line 3 will NOT be executed.
E. Only the line with the label loop will be printed.

**Correct Answer:** C
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**
A break without a label breaks the current loop (i.e. no iterations any more) and a break with a label tries to pass the control to the given label. 'Tries to' means that if the break is in a try block and the try block has a finally clause associated with it then it will be executed.

**QUESTION 583**
**What will the following program print?**

```
public class TestClass{
  public static void main(String[] args){
     for : for(int i = 0; i< 10; i++){
        for (int j = 0; j< 10; j++){
             if ( i+ j > 10 )  break for;
        }
        System.out.println( "hello");
     }
  }
}
```

A. It will print hello 6 times.
B. It will not compile.
C. It will print hello 2 times.
D. It will print hello 5 times.
E. It will print hello 4 times.

**Correct Answer:** B
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**
Note that for is a keyword and so cannot be used as a label. But you can use any other identifier as a label. For example, The following code is valid even though String is a class name and String is also used as an identifier!

```
     String String = "";   //This is valid.
     String : for(int i = 0; i< 10; i++) //This is valid too!
     {
        for (int j = 0; j< 10; j++){
             if ( i+ j > 10 )  break String;
        }
       System.out.println( "hello");
      }
```

It will print hello 2 times.

**QUESTION 584**
**What will the following code print when compiled and run?**

```
public class DaysTest {
   static String[] days = {"monday", "tuesday", "wednesday", "thursday",
"friday", "saturday", "sunday" };
   public static void main(String[] args) {
      int index = 0;
      for(String day : days){
          if(index == 3){
              break;
          } else {
              continue;
          }
          index++;
          if(days[index].length()>3) {
```

```
                days[index] = day.substring(0,3);
            }
        }
        System.out.println(days[index]);
    }
}
```

A. wed
B. thu
C. fri
D. It will not compile.
E. It will throw an exception at run time.

**Correct Answer:** D
**Section: Using Loops Constructs**
**Explanation**

**Explanation/Reference:**

```
Main.java:12: error: unreachable statement
          index++;
```

Notice the statement :

```
if(index == 3){
    break;
}
else {
    continue;
}
```

In no situation can the control go beyond this statement in the for loop. Therefore, rest of the statements in the for loop are unreachable and so the code will not compile.