TROYTEC TECHNOLOGIES

Troy Tec TM

Java SE 7 Programmer I

Exam: 1Z0-803

Edition: 2.0

QUESTION: 1
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

Answer: D

Explanation:
The length of the elementwith index 0, {0, 1, 2}, is 3. Output: 3
The elementwith index 1, {3, 4, 5, 6}, is of type array. Output: true
The elementwith index 0,{0, 1, 2} has the elementwith index 1:1. Output: 1

QUESTION: 2
View the exhibit:

```
public class Student {
 public String name = "";
 public int age = 0;
 public String major = "Undeclared";
 public boolean fulltime = true;
 public void display() {
 System.out.println("Name: " + name + " Major: " + major);
 }
public boolean isFullTime() {
 return fulltime;
}
}
```

Given:

```
Public class TestStudent {
Public static void main(String[] args) { Student bob = new Student (); Student jian = new
Student();
bob.name = "Bob";
bob.age = 19;
jian = bob; jian.name = "Jian";
System.out.println("Bob's Name: " + bob.name);
}
}
```

What is the result when this program is executed?

A. Bob's Name: Bob
B. Bob's Name: Jian
C. Nothing prints
D. Bob’s name

Answer: B

Explanation:
After the statement jian = bob; the jian will reference the same object as bob.

QUESTION: 3
Given the code fragment:
```
String valid = "true";
if (valid) System.out.println (“valid”);
else system.out.println ("not valid");
```
What is the result?

A. Valid
B. not valid
C. Compilation fails
D. An IllegalArgumentException is thrown at run time

Answer: C

Explanation:
In segment 'if (valid)' valid must be of type boolean, but it is a string. This makes the compilation fail.

QUESTION: 4
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

Answer: A

Explanation:
Within main z is assigned 6. z is printed. Output: 6 Within doStuff z is assigned 5.DoStuff2 locally sets z to 4(but MyScope.z is set to 4), but in Dostuff z is still 5. z is printed. Output: 5 Again z is printed within main (with local z set to 6). Output: 6 Finally MyScope.z is printed. MyScope.z has been set to 4 within doStuff2(). Output: 4

QUESTION: 5
Which two are valid instantiations and initializations of a multi dimensional array?

A. int [] [] array 2D ={ { 0, 1, 2, 4} {5, 6}}; B. int [] [] array2D = new int [2] [2]; array2D[0] [0] = 1;
array2D[0] [1] =2; array2D[1] [0] =3; array2D[1] [1] =4;
C. int [] [] []array3D = {{0, 1}, {2, 3}, {4, 5}};
D. int [] [] [] array3D = new int [2] [2] [2]; array3D [0] [0] = array;
array3D [0] [1] = array; array3D [1] [0] = array; array3D [0] [1] = array;
E. int [] [] array2D = {0, 1};

Answer: B, D

Explanation:
In the Java programming language, a multidimensional array is simply an array whose components are themselves arrays.

QUESTION: 6
An unchecked exception occurs in a method dosomething() Should other code be added in the dosomething() method for it to compile and execute?

A. The Exception must be caught
B. The Exception must be declared to be thrown.
C. The Exception must be caught or declared to be thrown.
D. No other code needs to be added.

Answer: C

Explanation:
Valid Java programming language code must honor the Catch or Specify Requirement. This means that code that might throw certain exceptions must be enclosed by either of the following:
* A try statement that catches the exception. The try must provide a handler for the exception, as described in Catching and Handling Exceptions.
* A method that specifies that it can throw the exception. The method must provide a throws clause that lists the exception, as described in Specifying the Exceptions Thrown by a Method. Code that fails to honor the Catch or Specify Requirement will not compile.

QUESTION: 7
Given the code fragment:
int b = 4;
b -- ;
System.out.println (-- b); System.out.println(b); What is the result?

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

Answer: A

Explanation:
Variable b is set to 4. Variable b is decreased to 3.
Variable b is decreased to 2 and then printed. Output: 2
Variable b is printed. Output: 2

QUESTION: 8
Given the code fragment:
interface SampleClosable {
public void close () throws java.io.IOException;
}
Which three implementations are valid?

A. public class Test implements SampleCloseable { Public void close () throws java.io.IOException { / /do something

}
}
B. public class Test implements SampleCloseable { Public void close () throws Exception {
/ / do something
}
}
C. public class Test implementations SampleCloseable { Public void close () throws Exception { / / do something
}
}
D. public classTest extends SampleCloseable { Public voidclose ()throws java.IO.IOException{ / / do something
}
}

Answer: D

Explanation:
To declare a class that implements an interface, you include an implements clause in the class declaration. One interface might extended another interface, but a class cannot extend an interface. Checked exceptions are subject to the Catch or Specify Requirement. All exceptions are checked exceptions, except for those indicated by Error, RuntimeException, and their subclasses.

QUESTION: 9
Given the code fragment:
Int [] [] array = {{0}, {0, 1}, {0, 2, 4}, {0, 3, 6, 9}, {0, 4, 8, 12, 16}};
Systemout.printIn(array [4] [1]);
System.out.printIn (array) [1][4]);
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

Answer: D

Explanation:
The first println statement, System.out.println(array [4][1]);, works fine. It selects the element/array with index 4, {0, 4, 8, 12, 16}, and from this array it selects the element with index 1, 4 Output: 4 The second println statement, System.out.println(array) [1][4]);, fails. It

selects the array/element with index 1, {0, 1}, and from this array it try to select the element with index 4. This causes an exception.
Output:
4
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4

QUESTION: 10
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.

Answer: C

Explanation:
The for statement, for (String ss: table), is executed one time for each of the three elements in table. The while loop will print a 2 once for each element.
Output: aa, 0 aa, 1 aa, 2 bb, 0 bb, 1 bb, 2 cc, 0 cc, 1 cc, 2

QUESTION: 11
Given:
```
import java.io.IOException;
public class Y {
 public static void main(String[] args) {
 try {
 doSomething();
 }
 catch (RuntimeException e) {
 System.out.println(e);
 }
 }
 static void doSomething() {
```

```
 if (Math.random() > 0.5) throw new IOException();
 throw new RuntimeException();
 }
 }
```

Which two actions, used independently, will permit this class to compile?

A. Adding throws IOException to the main() method signature
B. Adding throws IOException to the doSoomething() method signature
C. Adding throws IOException to the main() method signature and to the dosomething() method
D. Adding throws IOException to the dosomething() method signature and changing the catch argument to IOException
E. Adding throws IOException to the main() method signature and changing the catch argument to IOException

Answer: C, D

Explanation:
The IOException must be caught or be declared to be thrown. We must add a throws exception to the doSomething () method signature (static void doSomething() throws IOException). Then we can either add the same throws IOException to the main method (public static void main(String[] args) throws IOException), or change the catch statement in main to IOException.

QUESTION: 12
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. Compilation fails
D. The program prints nothing
E. An exception is thrown at run time

Answer: A

Explanation:

The program compiles fine. The program runs fine. The output is: hello

QUESTION: 13
Given:
public class SampleClass {
public static void main(String[] args) { AnotherSampleClass asc = new AnotherSampleClass(); SampleClass sc = new SampleClass();
// TODO code application logic here
}
}
class AnotherSampleClass extends SampleClass {
}
Which statement, when inserted into line "// TODO code application logic here ", is valid change?

A. asc = sc;
B. sc = asc;
C. asc = (object) sc;
D. asc= sc.clone ()

Answer: B

Explanation:
Works fine.

QUESTION: 14
Given the code fragment:
System.out.println("Result: " + 2 + 3 + 5);
System.out.println("Result: " + 2 + 1 * 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

Answer: C

Explanation:

String concatenatiois produced.
The output is: Result: 235
Result: 215
Note #1:
To produce an arithmetic result, the following code would have to be used: System.out.println("Result: " + (2 + 3 + 5));
System.out.println("Result: " + (2 + 3 * 5));
run: Result: 10
Result: 17
Note #2:
If the code was as follows: System.out.println("Result: " + 2 + 3 + 5"); System.out.println("Result: " + 2 + 1 * 5");
The compilation would fail. There is an unclosed string literal, 5", on each line.

QUESTION: 15
Which code fragment is illegal?

A. class Base1{ abstract class Abs1{ }}
B. abstract class Abs1{ void doit () { }
}
C. class Basel {
abstract class Abs1extends Basel {
D. abstract int var1= 89;

Answer: D

Explanation:
The abstract keyword cannot be used to declare an int variable. The abstract keyword is used to declare a class or method to be abstract[3]. An abstract method has no implementation; all classes containing abstract methods must themselves be abstract, although not all abstract classes have abstract methods.

QUESTION: 16
Given the code fragment:
int a = 0;
a++; System.out.println(a++); System.out.println(a);
What is the result?

A.
1
2
B.
0
1
C.

1
1
D.
2
2

Answer: A

Explanation:
The first println prints variable a with value 1 and then increases the variable to 2.

QUESTION: 17
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. There is no output
B. d is output
C. AStringIndexOutOfBoundsException is thrown at runtime
D. AnArrayIndexOutOfBoundsException is thrown at runtime
E. A NullPointException is thrown at runtime
F. A StringArrayIndexOutOfBoundsException is thrown at runtime

Answer: C

Explanation:
There are only 11 characters in the string "Hello World". The code theString.charAt(11) retrieves the 12th character, which does not exist. A StringIndexOutOfBoundsException is thrown. Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range:11

QUESTION: 18
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 ofclass x
B. addingthe protected modifier to the x()constructor
C. changingthe private modifier on the declarationof the one() method to protected
D. removing the Y () constructor
E. removing the private modifier from the two () method

Answer: C

Explanation:
Using the private protected, instead of the private modifier, for the declaration of the one() method, would enable the two() method to access the one() method.

QUESTION: 19
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 withhandy.dandy.KeyStroke Stroke = new handy.dandy.KeyStroke();
D. import handy.*;addedbeforeline 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

Answer: C, D, F

Explanation:
Three separate solutions:
C: thefull class path to the method must be stated (when we have not imported the package)
D: We can import the hold dandy class
F: we can import the specific method

QUESTION: 20
Given:

```
String message1 = "Wham bam!";
String message2 = new String("Wham bam!");
if (message1 == message2)
 System.out.println("They match");
if (message1.equals(message2))
 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 Prints
E. They really match

Answer: B

Explanation:
The strings are not the same objects so the == comparison fails. See note #1 below. As the value of the strings are the same equals is true. The equalsmethodcompares values for equality.
Note: #1 ==
Compares references, not values. The use of == with object references is generally limited to the following:
Comparing to see if a reference is null.
Comparing two enum values. This works because there is only one object for each enum constant. You want to know if two references are to the same object.

QUESTION: 21
Given:

```
public class Speak {  /* Line 1 */
 public static void main(String[] args) {  /* Line 2 */
Speak speakIT = new Tell();  /* Line 3 */
Tell tellIt = new Tell();  /* Line 4 */
speakIT.tellItLikeItIs();  /* Line 5 */
(Truth)speakIt.tellItLikeItIs();  /* Line 6 */
((Truth)speakIt).tellItLikeItIs();  /* Line 7 */
tellIt.tellItLikeItIs();  /* Line 8 */
(Truth)tellIt.tellItLikeItIs();  /* Line 9 */
((Truth)tellIt).tellItLikeItIs();  /* Line 10 */
 }
}
class Tell extends Speak implements Truth {
 public void tellItLikeItIs() {
 System.out.println("Right on!");
 }
}
interface Truth { public void tellItLikeItIs()};
```

Which three lines will compile and output “right on!”?

A. Line 5
B. Line6
C. Line 7
D. Line 8
E. Line 9
F. Line 10

Answer: A, D, F

QUESTION: 22
Given the code fragment:

```
String h1 = "Bob";
String h2 = new String ("Bob");
```

What is the best way to test that the values of h1 and h2 are the same?

A. if(h1 == h2)
B. if (h1.equals(h2))
C. if (h1 = = h2)
D. if (h1.same(h2))

Answer: B

Explanation:
The equals method compares values for equality.

QUESTION: 23
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[];

Answer: A, D

Explanation:
int[][] array2D; is the standard conventionto declare a 2-dimensional integer array.
int[]array2D[]; works as well, but it is not recommended.

QUESTION: 24
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

Answer: B

Explanation:
In the first statement 3 and 5 are treated as strings and are simply concatenated. In the first statement 3 and 5 are treated as integers and their sum is calculated.

QUESTION: 25
Given:
```
public class Main {
 public static void main(String[] args) throws Exception {
 doSomething();
 }
 private static void doSomething() throws Exception {
 System.out.println("Before if clause");
 if (Math.random() > 0.5) {
 throw new Exception();
 }
 System.out.println ("After if clause");
}
}
```

Which two are possible outputs?

A. Before if clause
Exception in thread “main” java.lang.Exception
AtMain.doSomething (Main.java:8) At Main.main (Main.java:3)
B. Before if clause
Exceptionin thread “main” java.lang.Exception
At Main.doSomething (Main.java:8) At Main.main (Main.java:3)
After if clause
C. Exception in thread “main” java.lang.Exception
At Main.doSomething (Main.java:8) At Main.main (Main.java:3)
D. Before if clause
After if clause

Answer: A, D

Explanation:
The first println statement, System.out.println("Before if clause");, will always run. If Math.Random() > 0.5 then there is an exception. The exception message is displayed and the program terminates. If Math.Random() > 0.5 is false, then the second println statement runs as well.

QUESTION: 26
A method doSomething () that has no exception handling code is modified to trail a method that throws a checked exception. Which two modifications, made independently, will allow the program to compile?

A. Catch the exception in the method doSomething().
B. Declare the exception to be thrown in the doSomething() method signature.
C. Cast the exception to a RunTimeException in the doSomething() method.
D. Catch the exception in the method that calls doSomething().

Answer: A, B

Explanation:
Valid Java programming language code must honor the Catch or Specify Requirement. This means that code that might throw certain exceptions must be enclosed by either of the following:
* A try statement that catches the exception. The try must provide a handler for the exception, as described in Catching and Handling Exceptions.

* A method that specifies that it can throw the exception. The method must provide a throws clause that lists the exception, as described in Specifying the Exceptions Thrown by a Method. Code that fails to honor the Catch or Specify Requirement will not compile.

QUESTION: 27
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

Answer: B

Explanation:
As there is no break statement after the case "Red" statement the case Blue statement will run as well.
Note: The body of a switch statement is known as a switch block. A statement in the switch block can be labeled with one or more case or default labels. The switch statement evaluates its expression, then executes all statements that follow the matching case label. Each break statement terminates the enclosing switch statement. Control flow continues with the first statement following the switch block. The break statements are necessary because without them, statements in switch blocks fall through: All statements after the matching case label are executed in sequence, regardless of the expression of subsequent case labels, until a break statement is encountered.

QUESTION: 28
Which two may precede the word "class" in a class declaration?

A. local
B. public
C. static
D. volatile
E. synchronized

Answer: B, C

Explanation:
B:A class can be declared as public or private.
C: You can declare two kinds of classes: top-level classes and inner classes. You define an inner class within a top-level class. Depending on how it is defined, an inner class can be one of the following four types:Anonymous, Local, Member and Nested top-level. A nested top-level class is a member classes with a static modifier. A nested top-level class is just like any other top-level class except that it is declared within another class or interface. Nested top-level classes are typically used as a convenient way to group related classes without creating a new package. The following is an example:

```
public class Main {
static class Killer {
```

QUESTION: 29
Which three are bad practices?

A. Checking for ArrayindexoutofBoundsException when iterating through an array to determinewhen all elements have been visited
B. Checking for Error and. If necessary, restartingthe program to ensure that users are unaware problems
C. Checking for FileNotFoundException to inform a user that a filename entered is not valid
D. Checking for ArrayIndexoutofBoundsExcepcion and ensuring that the program can recover if one occur
E. Checking for an IOException and ensuring that the program can recover if one occurs

Answer: A, B, E

Explanation:
A, E: Better to check if the index is within bounds.
B: Restarting the program would not be a good practice. It should be done as a last possibility only.

QUESTION: 30
Given:

```
public class Bark {
 // Insert code here - Line 5
 public abstract void bark(); // Line 6
}  // Line 7
 // Line 8
 // Insert code here - Line 9
 public  void bark() {
System.out.println("woof");
 }
}
```

What code should be inserted?

A. 5.class Dog {
9. public class Poodle extends Dog {
B. 5.abstract Dog {
9. public class poodle extends Dog {
C. 5.abstractclassDog {
9. public class Poodle extends Dog {
D. 5.abstract Dog {
9.public class Poodle implements Dog {
E. 5. abstractDog {
9. public class Poodle implements Dog {
F. 5.abstract class Dog {
9.public class Poodle implements Dog {

Answer: C

Explanation:
Dog should be an abstract class. The correct syntax for this is: abstract class Dog { Poodle should extend Dog (not implement).

QUESTION: 31
Given:
class X {}
class Y {Y () {}}
class Z {z(int i) {} }
Which class has a default constructor?

A. X only
B. Y only
C. Z only
D. X and Y
E. Y and Z
F. X and Z
G. X, Y and Z

Answer: G

Explanation:
No constructors are defined for the X, Y, or Z classes. All three have empty declarations. Java will create default constructors for all three classes. Note: A java constructor has the same name as the name of the class to which it belongs. Constructor's syntax does not include a return type, since constructors never return a value. Constructors may include parameters of various types. When the constructor is invoked using the new operator, the types must match those that are specified in the constructor definition. Java provides a default constructor which takes no arguments and performs no special actions or

initializations, when no explicit constructors are provided. The only action taken by the implicit default constructor is to call the superclass constructor using the super() call. Constructor arguments provide you with a way to provide parameters for the initialization of an object. Below is an example of a cube class containing 2 constructors. (one default and one parameterized constr

QUESTION: 32
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;

Answer: A, D

QUESTION: 33
Given the code fragment:
int j=0, k =0;
for (int i=0; i < x; i++) {
do {
k=0;
while (k < z) {
k++;
System.out.print(k + " ");
}
System.out.println(" ");
j++;
} while (j< y);
System.out.println("----");
}
What values of x, y, z will produce the following result?
1 2 3 4
1 2 3 4
1 2 3 4
------
1 2 3 4

A. X = 4, Y = 3, Z = 2
B. X = 3, Y = 2, Z = 3
C. X = 2, Y = 3, Z = 3
D. X = 4, Y = 2, Z= 3
E. X = 2, Y = 3, Z = 4

Answer: E

Explanation:
Z is for the innermost loop. Should print 1 2 3 4. So Z must be 4. Y is for the middle loop. Should print three lines of 1 2 3 4. So Y must be set 3. X is for the outmost loop. Should print 2 lines of ----. So X should be 2.

QUESTION: 34
Which statement initializes a stringBuilder to a capacity of 128?

A. StringBuildersb = new String("128");
B. StringBuildersb = StringBuilder.setCapacity(128);
C. StringBuildersb = StringBuilder.getInstance(128);
D. StringBuildersb = new StringBuilder(128);

Answer: D

Explanation:
StringBuilder(int capacity)   Constructs a string builder with no characters in it and an initial capacity specified by the capacity argument. Note: An instance of a StringBuilder is a mutable sequence of characters. The principal operations on a StringBuilder are the append and insert methods, which are overloaded so as to accept data of any type. Each effectively converts a given datum to a string and then appends or inserts the characters of that string to the string builder. The append method always adds these characters at the end of the builder; the insert method adds the characters at a specified point.

QUESTION: 35
Given:

```
public class DoCompare4 {
public static void main(String[] args) {
String[]  table = {"aa", "bb", "cc"};
int ii =0;
do
while (ii < table.length) System.out.println(ii++); while (ii < table.length);
}
}
```

What is the result?

A. 0
B. 0
1
2
C. 0
1
2
0
1
2
0
1
2
D. Compilation fails

Answer: B

Explanation:
table.length is 3. So the do-while loop will run 3 times with ii=0, ii=1 and ii=2. The second while statement will break the do-loop when ii = 3. Note:The Java programming language provides a do-while statement, which can be expressed as follows:
do {
statement(s)
} while (expression);

QUESTION: 36
A method is declared to take three arguments. A program calls this method and passes only two arguments. What is the result?

A. Compilation fails.
B. The third argument is given the value null. C. The third argument is given the value void.
D. The third argument is given the value zero.
E. The third argument is given the appropriate false value for its declared type.
F. An exception occurs when the method attempts to access the third argument.

Answer: A

Explanation:
The problem is noticed at build/compile time. At build you would receive an error message like:
required: int,int,int found: int,int

QUESTION: 37

Given the fragment:
int [] array = {1, 2, 3, 4, 5}; System.arraycopy (array, 2, array, 1, 2); System.out.print (array [1]); System.out.print (array[4]);
What is the result?

A. 14
B. 15
C. 24
D. 25
E. 34
F. 35

Answer: F

Explanation:
The two elements 3 and 4 (starting from position with index 2) are copied into position index 1 and 2 in the same array.
After the arraycopy command the array looks like:
{1, 3, 4, 4, 5};
Then element with index 1 is printed: 3
Then element with index 4 is printed: 5
Note:The System class has an arraycopy method that you can use to efficiently copy data from one array into another:
public static void arraycopy(Object src, int srcPos, Object dest, int destPos, int length) The two Object arguments specify the array to copy from and the array to copy to. The three int arguments specify the starting position in the source array, the starting position in the destination array, and the number of array elements to copy.

QUESTION: 38
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 ….

Answer: B

Explanation:
33 is greater than 0.
33 isnot equal to 0. the is printed.
33 is greater than 30
fox is printed
33 is greater then 10 (the two else if are skipped)
lazy is printed finally … is printed.

QUESTION: 39
Which three are advantages of the 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

Answer: A, C, E

Explanation:
A: The error handling is separated from the normal program logic.
C: You have some choice where to handle the exceptions.
E: You can create your own exceptions.

QUESTION: 40
Given:
```
public class MyFor3 {
public static void main(String [] args) {
int [] xx = null;
System.out.println(xx);
}
}
```
What is the result?

A. null
B. compilation fails

C. Java.lang.NullPointerException
D. 0

Answer: A

Explanation:
An array variable (here xx) can very well have the null value.
Note:
Null is the reserved constant used in Java to represent a void reference i.e a pointer to nothing. Internally it is just a binary 0, but in the high level Java language, it is a magic constant, quite distinct from zero, that internally could have any representation.

QUESTION: 41
Given:

```
public class Main {
public static void main (String[] args) {
doSomething();
}
private static void doSomething() {
doSomeThingElse();
}
private static void doSomeThingElse() {
throw new Exception();
}
}
```

Which approach ensures that the class can be compiled and run?

A. Put the throw new Exception() statement in the try block of try – catch
B. Put the doSomethingElse() method in the try block of a try – catch
C. Put the doSomething() method in the try block of a try – catch
D. Put the doSomething() method and the doSomethingElse() method in the try block of a try – catch

Answer: B

Explanation:
We need to catch the exception in the doSomethingElse() method. Such as:

```
private static void doSomeThingElse() {
try {
throw new Exception();}
catch (Exception e)
{}
}
```

Note: One alternative, but not an option here, is the declare the exception in doSomeThingElse and catch it in the doSomeThing method.

QUESTION: 42
Given:
```
public class ScopeTest1 {
public static void main(String[] args) {
doStuff();  // line x1 int x1 = x2;  // line x2 int x2 = j;  // line x3
}
static void doStuff() {
System.out.println(j);  // line x4
}
static int j;
}
```
Which line causes a compilation error?

A. line x1
B. line x2
C. line x3
D. line x4

Answer: B

Explanation:
The variable x2 is used before it has been declared.

QUESTION: 43
Given:
```
class Overloading {
void x (int i) {
System.out.println("one");
}
void x (String s) {
System.out.println("two");
}
void x (double d) {
System.out.println("three");
}
public static void main(String[] args) {
new Overloading().x (4.0);
}
}
```
What is the result?

A. One
B. Two
C. Three
D. Compilation fails

Answer: C

Explanation:
In this scenario the overloading method is called with a double/float value, 4.0. This makes the third overload method to run. Note: The Java programming language supports overloading methods, and Java can distinguish between methods with different method signatures. This means that methods within a class can have the same name if they have different parameter lists. Overloaded methods are differentiated by the number and the type of the arguments passed into the method.

QUESTION: 44
Which declaration initializes a boolean variable?

A. boolean h = 1;
B. boolean k = 0;
C. boolean m = null;
D. boolean j = (1 < 5) ;

Answer: D

Explanation:
The primitive type boolean has only two possible values: true and false. Here j is set to (1 <5), which evaluates to true.

QUESTION: 45
Given:

```
public class Basic {
private static int letter;
public static int getLetter();
public static void Main(String[] args)  {
System.out.println(getLetter());
}
}
```

Why will the code not compile?

A. A static field cannot be private.
B. The getLetter method has no body.
C. There is no setletter method.
D. The letter field is uninitialized.
E. It contains a method named Main instead of ma

Answer: B

Explanation:
The getLetter() method needs a body.

QUESTION: 46
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. What three modifications are necessary to ensure that the class is being properly encapsulated?

A. Change the access modifier of the setradius () method to private
B. Change the getArea () method public double getArea () { return area; }
C. When the radius is set in the Circle constructor and the setRadius () method, recomputed the area and store it into the area field
D. Change the getRadius () method:
public double getRadius () { area = Math.PI * radius * radius; return radius;
}

Answer: A, B, C

Explanation:
A: There is no need to have SetRadius as public as the radius can be set through the Circle method.
B: We need to return the area in the GetArea method.
C: When the radius changes the Area must change as well.

Incorrect answer:
D: the GetRadius() method does not change the radius, so there is no need to recomputed the area.

QUESTION: 47
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

Answer: D

Explanation:
Strings can not be compared with the usual <, <=, >, or >= operators, and the ==
and != operators don't compare the characters in the strings. So the first if statement fails. Equals works fine on strings. But it does not work here.The second if-statement also fails.The StringBuffer class does not override the equals method so it uses the equals method of Object. If a and b are two objects from a class which doesn't override equals, thena.equals(b) is the same as a == b

QUESTION: 48
Given the following code:
public class Simple { /* Line 1 */
public float price; /* Line 2 */
public static void main (String[] args) { /* Line 3 */
Simple price = new Simple (); /* Line 4 */
price = 4; /* Line 5 */
} /* Line 6 */
} /* Line 7 */
What will make this code compile and run?

A. Change line 2 to the following: Publicint price
B. Change line 4 to the following:
int price = new simple ();
C. Change line 4 to the following: Floatprice = new simple ();
D. Change line 5 to the following: Price = 4f;
E. Change line 5 to the following:
price.price = 4;
F. Change line 5 to the following: Price= (float) 4:
G. Change line 5 to the following: Price= (Simple) 4;
H. The code compiles and runs properly; no changes are necessary

Answer: E

Explanation:
price.price =4; is correct, not price=4;
The attribute price of the instance must be set, not the instance itself.

QUESTION: 49
Given:
```
public class DoWhile {
public static void main (String [] args) {
int ii = 2;
do {
System.out.println (ii);
} while (--ii);
}
}
```
What is the result?

A. 2
1
B. 2
1
0
C. null
D. an infinite loop
E. compilation fails

Answer: E

Explanation:
The line while (--ii); will cause the compilation to fail.
--ii is not a boolean value.
A correct line would be while (--ii>0);

QUESTION: 50
You are writing a method that is declared not to return a value. Which two are permitted in the method body?

A. omission of the return statement
B. return null;
C. return void;
D. return;

Answer: A, D

Explanation:
Any method declared void doesn't return a value. It does not need to contain a return statement, but it may do so. In such a case, a return statement can be used to branch out of a control flow block and exit the method and is simply used like this:
return;

QUESTION: 51
Identify two benefits of using ArrayList over array in software development.

A. reduces memory footprint
B. implements the Collection API
C. is multi.thread safe
D. dynamically resizes based on the number of elements in the list

Answer: A, D

Explanation:
ArrayList supports dynamic arrays that can grow as needed. In Java, standard arrays are of a fixed length. After arrays are created, they cannot grow or shrink, which means that you must know in advance how many elements an array will hold. But, sometimes, you may not know until run time precisely how large of an array you need. To handle this situation, the collections framework defines ArrayList. In essence, an ArrayList is a variable-length array of object references. That is, an ArrayList can dynamically increase or decrease in size. Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array may be shrunk.

QUESTION: 52
Which three are valid types for switch?

A. int
B. float
C. double
D. integer
E. String
F. Float

Answer: A, D, E

Explanation:
A switch works with the byte, short, char, and int primitive data types. It also works with enumerated types the String class, and a few special classes that wrap certain primitive types: Character, Byte, Short, and Integer.

QUESTION: 53
Give:
```
public class MyFive {
 static void main(String[] args) {
short ii;
short jj = 0;
for (ii = kk;ii > 6; ii -= 1) {  // line x  //
 jj++;
}
System.out.println("jj = " + jj);
 }
}
```
What value should replace KK in line x to cause jj = 5 to be output?

A. -1
B. 1
C. 5
D. 8
E. 11

Answer: E

Explanation:
We need to get jj to 5. It is initially set to 0. So we need to go through the for loop 5 times. The for loops ends when ii > 6 and ii decreases for every loop. So we need to initially set ii to 11. We set kk to 11.

QUESTION: 54
Given the code fragment:
```
Boolean b1 = true; Boolean b2 = false; int 1 = 0;
while (foo) {}
```
Which one is valid as a replacement for foo?

A. b1.compareTo(b2)
B. i = 1
C. i == 2? -1:0
D. "foo".equals("bar")

Answer: D

Explanation:
equals works fine on strings. equals produces a Boolean value.

QUESTION: 55

Given:

```
public class SuperTest {
 public static void main(String[] args) {
statement1 statement2 statement3
 }
}
class Shape {
 public Shape() {
 System.out.println("Shape: constructor");
 }
 public void foo() {
 System.out.println("Shape: foo");
 }
}
class Square extends Shape {
 public Square() {
 super();
 }
 public Square(String label) {
 System.out.println("Square: constructor");
 }
 public void foo() {
 super.foo();
 }
 public void foo(String label) {
 System.out.println("Square: foo");
 }
}
```

What should statement1, statement2, and statement3, be respectively, in order to produce the result?

```
Shape: constructor
Square: foo
Shape: foo
```

A. Square square = new Square ("bar");
square.foo ("bar");
square.foo();
B. Square square = new Square ("bar");
square.foo ("bar");
square.foo ("bar");
C. Square square = new Square ();
square.foo ();
square.foo("bar");
D. Square square = new Square ();
square.foo ();
square.foo("bar");
E. Square square = newSquare ();
square.foo ();
square.foo ();

Answer: D

QUESTION: 56
Give:
Public Class Test {
}
Which two 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

Answer: A, E

Explanation:
For convenience, the Java compiler automatically imports three entire packages for each source file: (1) the package with no name, (2) the java.lang package, and (3) the current package (the package for the current file). Note:Packages in the Java language itself begin with java. or javax.

QUESTION: 57
Given:

```
public class X implements Z {
public String toString() { return "I am X"; }
 public static void main(String[] args) {
 Y myY = new Y();
 X myX = myY;
 Z myZ = myX;
 System.out.println(myZ);
 }
}
class Y extends X {
 public String toString() { return "I am Y"; }
}
interface Z {}
```

What is the reference type of myZ and what is the type of the object it references?

A. Reference type isZ; object type isZ.
B. Reference type isY;object type isY.
C. Reference type isZ;object type is Y.

D. Reference type is X; object type is Z.

Answer: B

Explanation:
Note: Because Java handles objects and arrays by reference, classes and array types are known as reference types.

QUESTION: 58
Given:
What is the result?

A. sc: class.Object
asc: class.AnotherSampleClass
B. sc: class.SampleClass asc:class.AnotherSampleClass
C. sc: class.AnotherSampleClass asc:class.SampleClass
D. sc:class.AnotherSampleClass asc:class.AnotherSampleClass

Answer: D

Explanation:
Note: The getClass methodReturns the runtime class of an object. That Class object is the object that is locked by static synchronized methods of the represented class. Note: Because Java handles objects and arrays by reference, classes and array types are known as reference types.

QUESTION: 59
Given the code fragment:

```
public static void main(String[] args) {
String [] table = {"aa", "bb", "cc"};
int ii = 0;
for (String ss:table) { while (ii < table.length) { System.out.println (ii); ii++;
break;
 }
}
}
```

How many times is 2 printed?

A. zero
B. once
C. twice
D. thrice
E. it is not printed because compilation fails

Answer: B

Explanation:
The outer loop will run three times, one time each for the elementsin table. The break statement breaks the inner loop immediatelyeach time.
2 will be printed once only.
Note: If the line int ii = 0; is missing the program would not compile.

QUESTION: 60
Given:
```
public class SampleClass {
 public static void main(String[] args) {
SampleClass sc, scA, scB; sc = new SampleClass(); scA = new SampleClassA();
 scB = new SampleClassB();
 System.out.println("Hash is : " +
 sc.getHash() + ", " + scA.getHash() + ", " + scB.getHash());
 }
 public int getHash() {
 return 111111;
 }
}
class SampleClassA extends SampleClass {
 public long getHash() {
 return 44444444;
 }
}
class SampleClassB extends SampleClass {
 public long getHash() {
 return 999999999;
 }
}
```
What is the result?

A. Compilation fails
B. An exception is thrown at runtime
C. There is no result because this is not correct way to determine the hash code
D. Hash is: 111111, 44444444, 999999999

Answer: A

Explanation:
The compilation fails as SampleClassA and SampleClassB cannot override SampleClass because the return type of SampleClass is int, while the return type of SampleClassA and SampleClassB is long.
Note: If all three classes had the same return type the output would be: Hash is : 111111, 44444444, 999999999

QUESTION: 61
Which two will compile, and can be run successfully using the command:
Java fred hello walls

A. class Fred1{
public static void main (String args) { System.out.println(args[1]);
}
}
B. class Fred1{
public static void main (String [] args) { System.out.println(args[2]);
}
}
C. class Fred1 {
public static void main (String [] args) { System.out.println (args);
}
}
D. class Fred1 {
public static void main (String [] args){ System.out.println (args [1]);
}
}

Answer: B, C

QUESTION: 62
Given:
public abstract class Wow {
private int wow;
public wow (int wow) {
this.wow = wow;
}
public void wow () {}
private void wowza () {}
}
What is true about the class Wow?

A. It compiles without error.
B. It does not compile because an abstract class cannot have private methods.
C. It does not compile because an abstract class cannot have instance variables.
D. It does not compile because an abstract class must have at least one abstract method.
E. It does not compile because an abstract class must have a constructor with no arguments.

Answer: C

Explanation:
An abstract class is a class that is declared abstract—it may or may not include abstract methods (not B, not D). Abstract classes cannot be instantiated, but they can be subclassed. The code compiles with a failure for line'public wow (int wow){'

QUESTION: 63
Given:

```
class X {
static void m(int i) {
}
public static void main (String [] args) {
int j = 12;
m (j);
System.out.println(j);
}
}
```

What is the result?

A. 7
B. 12
C. 19
D. Compilation fails
E. An exception is thrown at run time

Answer: B

QUESTION: 64
Which two statements are true?

A. An abstract class can implement an interface.
B. An abstract class can be extended by an interface.
C. An interface CANNOT be extended by another interface. D An interface can be extended by an abstract class.
D. An abstract class can be extended by a concrete class.
E. An abstract class CANNOT be extended by an abstract class.

Answer: C, E

Explanation:
E:When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class. Reference:http://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

QUESTION: 65
Given:

```
class Overloading {
 int x(double d) {
 System.out.println("one");
 return 0;
 }
 String x(double d) {
 System.out.println("two");
 return null;
 }
 double x(double d) {
 System.out.println("three");
 return 0.0;
 }
 public static void main(String[] args) {
 new Overloading().x(4.0)
 }
}
```

What is the result?

A. One
B. Two
C. Three
D. Compilation fails

Answer: D

Explanation:
overloading of the x method fails as the input argument in all three cases are double. To use overloading of methods the argument types must be different. Note: The Java programming language supports overloading methods, and Java can distinguish between methods with different method signatures. This means that methods within a class can have the same name if they have different parameter lists

QUESTION: 66
The catch clause argument is always of type____________.

A. Exception
B. Exception but NOT including RuntimeException
C. Throwable
D. RuntimeException
E. CheckedException F. Error

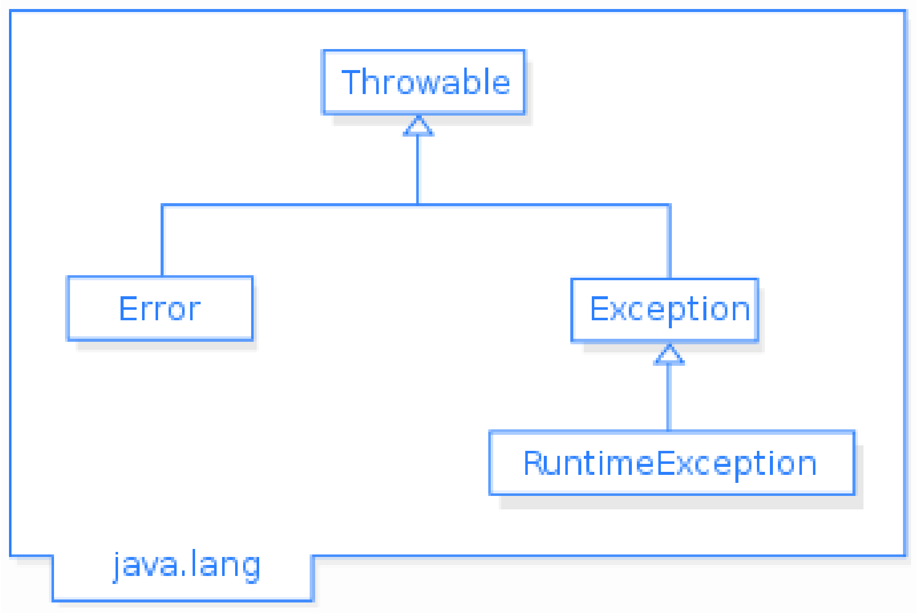
Answer: C

Explanation:
Because all exceptions in Java are the sub-class ofjava.lang.Exception class, you can have a single catch block that catches an exception of type Exception only. Hence the compiler is fooled into thinking that this block can handle any exception. See the following example:

```
try
{
// ...
}
catch(Exception ex)
{

// Exception handling code for ANY exception
}
```

You can also use the java.lang.Throwable class here, since Throwable is the parent class for the application-specific Exception classes. However, this is discouraged in Java programming circles. This is because Throwable happens to also be the parent class for the non-application specific Error classes which are not meant to be handled explicitly as they are catered for by the JVM itself.
Note: The Throwable class is the superclass of all errors and exceptions in the Java language. Only objects that are instances of this class (or one of its subclasses) are thrown by the Java Virtual Machine or can be thrown by the Java throw statement.
A throwable contains a snapshot of the execution stack of its thread at the time it was created. It can also contain a message string that gives more information about the error.


Throwable
Error
Exception
RuntimeException
java.lang