Exam : 1Z0-803

Title : Oracle Java SE7 Programmer I Exam

Version : Demo

http://www.test4actual.com

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/Reference:

Explanation:

The length of the element with index 0, {0, 1, 2}, is 3. Output: 3 The element with index 1, {3, 4, 5, 6}, is of type array. Output: true The element with index 0, {0, 1, 2} has the element with 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/Reference:
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/Reference:
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
   4
   5
   4

C. 6
   4
   6
   4

D. 6
   4
   4
   4

Answer: A

Explanation/Reference:
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: BD

Explanation/Reference:
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: D

Explanation/Reference:
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.
Thus, Java compilers do not require that you declare or catch **unchecked** exceptions in your program code.

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. 2
   1
C. 3
   2
D. 3
   3

Answer: A

Explanation/Reference:
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 implementation is 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 class Test extends SampleCloseable {
    Public void close () throws java.IO.IOException {
     // do something
    }
  }
```

Answer: A

Explanation/Reference:

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.

