



Java SE 7 Programmer I  
Exam: 1Z0-803  
*Edition: 4.0*

© 2014 - 2015 Troy Tec, LTD All Rights Reserved

**QUESTION: 1**

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

**Answer: E**

**QUESTION: 2**

Given the code fragment: List colors = new ArrayList(); colors.add("green"); colors.add("red"); colors.add("blue"); colors.add("yellow"); colors.remove(2); colors.add(3,"cyan"); System.out.print(colors); What is the result?

- A. [green, red, yellow, cyan]
- B. [green, blue, yellow, cyan]
- C. [green, red, cyan, yellow]
- D. An IndexOutOfBoundsException is thrown at runtime

**Answer: A**

**Explanation:**

First the list [green, red, blue, yellow] is build. The blue element is removed:

[green, red, yellow]

Finally the element cyan is added at then end of the list (index 3). [green, red, yellow, cyan]

**QUESTION: 3**

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:** 4

Given:

```

public 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

**Answer:** C

**QUESTION:** 5

Given:

```
5. // insert code here
6.     public abstract void bark();
7. }
8.
9. // insert code here
10.    public void bark() {
11.        System.out.println("woof");
12.    }
13. }
```

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. abstract class Dog {
      9. public class Poodle extends Dog {
Ⓓ D) 5. class Dog {
      9. public class Poodle implements Dog {
Ⓔ E) 5. abstract Dog {
      9. public class Poodle implements Dog {
Ⓕ F) 5. abstract class Dog {
      9. public class Poodle implements Dog {
```

- A. Option A
- B. Option B
- C. Option C
- D. Option D
- E. Option E
- F. Option F

**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:** 6

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.println( "..." );
```

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 is not 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: 7**

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 ArrayIndexOutOfBoundsException 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: 8**

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: 9**

Given the code fragment:

```
StringBuilder sb = new StringBuilder ( ) ; Sb.append ("world");
```

Which code fragment prints Hello World?

- A. `sb.insert(0,"Hello "); System.out.println(sb);`
- B. `sb.append(0,"Hello "); System.out.println(sb);`
- C. `sb.add(0,"Hello "); System.out.println(sb);`
- D. `sb.set(0,"Hello "); System.out.println(sb);`

**Answer:** A

**Explanation:**

The `java.lang.StringBuilder.insert(int offset, char c)` method inserts the string representation of the `char` argument into this sequence. The second argument is inserted into the contents of this sequence at the position indicated by `offset`. The length of this sequence increases by one. The `offset` argument must be greater than or equal to 0, and less than or equal to the length of this sequence.

**Reference:** `Java.lang.StringBuilder.insert()` Method

**QUESTION: 10**

Given the code fragment?

```
public class Test {
    public static void main(String[] args) { Test t = new Test();
    int[] arr = new int[10]; arr = t.subArray(arr,0,2);
    }
    // insert code here
}
```

Which method can be inserted at line `// insert code here` to enable the code to compile?

- A. `public int[] subArray(int[] src, int start, int end) { return src; }`
- B. `public int subArray(int src, int start, int end) { return src; }`
- C. `public int[] subArray(int src, int start, int end) { return src; }`

```

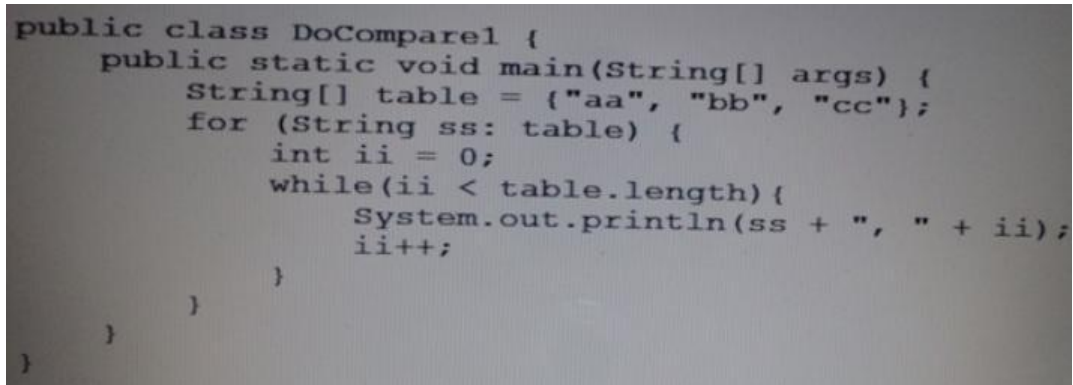
}
D. public int subArray(int[] src, int start, int end) { return src;
}

```

**Answer:** A

**QUESTION: 11**

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:** D

**QUESTION: 12**

Given the code fragment:

```

String[] cartoons = {"tom","jerry","micky","tom"}; int counter =0;
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

**Answer:** A

**Explanation:**

Counter++ will be executed only once because of the else if constructs.

**QUESTION:** 13

Given the code fragment:

```
Boolean b1 = true; Boolean b2 = false; int i = 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:** 14

Given: class Mid {

```
public int findMid(int n1, int n2) { return (n1 + n2) / 2;
```

```
}
```

```
}
```

```
public class Calc extends Mid { public static void main(String[] args) { int n1 = 22, n2 = 2;
```

```
// insert code here System.out.print(n3);
```

```
}
```

```
}
```

Which two code fragments, when inserted at // insert code here, enable the code to compile and print 12?

A. Calc c = new Calc();

```
int n3 = c.findMid(n1,n2);
```

B. int n3 = super.findMid(n1,n3);

C. Calc c = new Mid();

```
int n3 = c.findMid(n1, n2);
```

D. Mid m1 = new Calc();

```
int n3 = m1.findMid(n1, n2);
```

E. int n3 = Calc.findMid(n1, n2);

**Answer:** A, D

**Explanation:**

Incorrect:

Not B: circular definition of n3.

Not C: Compilation error. line Calc c = new Mid(); required: Calc  
found: Mid

Not E: Compilation error. line int n3 = Calc.findMid(n1, n2);  
non-static method findMid(int,int) cannot be referenced from a static context

**QUESTION: 15**

Given:

```
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

**Answer:** C

**Explanation:**

1200 Strawberry

1230 Chocolate

**QUESTION: 16**

Given:

```

1. public abstract class Wow {
2.     private int wow;
3.     public Wow(int wow) {
4.         this.wow = wow;
5.     }
6.     public void wow() { }
7.     private void wowza() { }
8. }

```

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:** A

**QUESTION:** 17

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 ";
    }
}

```

- A. X XX
- B. X Y X
- C. Y Y X
- D. Y YY

**Answer:** D

**QUESTION:** 18

```
public class ForTest {
public static void main(String[] args) { int[] arrar = {1,2,3};
for ( foo ) {
}
}
}
```

Which three are valid replacements for foo so that the program will compiled and run?

- A. int i: array
- B. int i = 0; i < 1; i++
- C. ;;
- D. ; i < 1; i++
- E. ; i < 1;

**Answer:** A, B, C

**QUESTION:** 19

Given:

```
class Test {
    int sum = 0;
    public void doCheck(int number) {
        if (number % 2 == 0) {
            break;
        } else {
            for (int i = 0; i < number; i++) {
                sum += i;
            }
        }
    }
}

public static void main(String[] args) {
    Test obj = new Test();
    System.out.println("Red " + obj.sum);
    obj.doCheck(2);
    System.out.println("Orange " + obj.sum);
    obj.doCheck(3);
    System.out.println("Green " + obj.sum);
}
```

What is the result?

- A. Red 0 Orange 0 Green 3
- B. Red 0 Orange 0 Green 6
- C. Red 0 Orange 1
- D. Green 4
- E. Compilation fails

**Answer:** E

**QUESTION:** 20

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  
at Main.doSomething(Main.java:8)  
at Main.main(Main.java:3)
- ☐ B) Before if clause  
Exception in 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

- A. Option A
- B. Option B
- C. Option C
- D. Option D

**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: 21**

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: 22**

Given:

```

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, 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

**Answer:** C, E, F

**Explanation:**

Three separate solutions:

C: the full 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:** 23

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(){}  
}

- A. Option A
- B. Option B
- C. Option C
- D. Option D
- E. Option E

**Answer:** B, C

**Explanation:**

When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class (C). However, if it does not, then the subclass must also be declared abstract (B). Note: An abstract class is a class that is declared abstract—it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed.

**QUESTION:** 24

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 convention to declare a 2-dimensional integer array. `int[] array2D[];` works as well, but it is not recommended.

**QUESTION:** 25

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

```

C A) Square square = new Square("bar");
    square.foo("bar");
    square.foo();

C B) Square square = new Square("bar");
    square.foo();
    square.foo("bar");

C C) Square square = new Square();
    square.foo();
    square.foo("bar");

C D) Square square = new Square();
    square.foo("bar");
    square.foo();

C E) Square square = new Square();
    square.foo();
    square.foo();

```

- A. Option A
- B. Option B
- C. Option C
- D. Option D
- E. Option E

**Answer:** D

**QUESTION:** 26

Given the code fragment:

```
String[] colors = {"red", "blue", "green", "yellow", "maroon", "cyan"};
```

Which code fragment prints blue, cyan, ?

```

C A) for (String c:colors){
        if (c.length() != 4) {
            continue;
        }
        System.out.print(c+", ");
    }

C B) for (String c:colors[]) {
        if (c.length() <= 4) {
            continue;
        }
        System.out.print(c+", ");
    }

C C) for (String c:String[] colors) {
        if (c.length() >= 3) {
            continue;
        }
        System.out.print(c+", ");
    }

C D) for (String c:colors){
        if (c.length() != 4) {
            system.out.print(c+", ");
            continue;
        }
    }

```

- A. Option A
- B. Option B
- C. Option C
- D. Option D

**Answer:** A

**QUESTION:** 27

Which three statements are true about the structure of a Java class?

- A. A class can have only one private constructor.
- B. A method can have the same name as a field.
- C. A class can have overloaded static methods.
- D. A public class must have a main method.
- E. The methods are mandatory components of a class.

F. The fields need not be initialized before use.

**Answer:** A, B, C

**Explanation:**

A: Private constructors prevent a class from being explicitly instantiated by its callers. If the programmer does not provide a constructor for a class, then the system will always provide a default, public no-argument constructor. To disable this default constructor, simply add a private no-argument constructor to the class. This private constructor may be empty.

B: The following works fine: `int cake() {  
int cake=0; return (1);  
}`

C: We can overload static method in Java. In terms of method overloading static method are just like normal methods and in order to overload static method you need to provide another static method with same name but different method signature.

Incorrect:

Not D: Only a public class in an application need to have a main method. Not E:

Example:

```
class A
{
public string something; public int a;
}
```

Q: What do you call classes without methods? Most of the time: An anti pattern.

Why? Because it facilitates procedural programming with "Operator" classes and data structures. You separate data and behaviour which isn't exactly good OOP.

Often times: A DTO (Data Transfer Object)

Read only datastructures meant to exchange data, derived from a business/domain object.

Sometimes: Just data structure.

Well sometimes, you just gotta have those structures to hold data that is just plain and simple and has no operations on it.

Not F: Fields need to be initialized. If not the code will not compile. Example:

Uncompilable source code - variable x might not have been initialized

**QUESTION:** 28

Which statement is true about the default constructor of a top-level class?

A. It can take arguments.

B. It has private access modifier in its declaration.

C. It can be overloaded.

D. The default constructor of a subclass always invokes the no-argument constructor of its superclass.

**Answer:** D

**Explanation:**

In both Java and C#, a "default constructor" refers to a nullary constructor that is automatically generated by the compiler if no constructors have been defined for the class. The default constructor is also empty, meaning that it does nothing. A programmer-defined constructor that takes no parameters is also called a default constructor.

**QUESTION: 29**

Given:

```
public class MyFor3 {
    public static void main(String[] args) {
        int[] xx = null;
        for (int ii : xx) {
            System.out.println(ii);
        }
    }
}
```

What is the result?

- A. Null
- B. Compilation fails
- C. An exception is thrown at runtime
- D. 0

**Answer: C**

**QUESTION: 30**

Given:

```
public class FieldInit { char c;
    boolean b; float f;
    void printAll() { System.out.println("c = " + c); System.out.println("c = " + b);
        System.out.println("c = " + f);
    }
    public static void main(String[] args) { FieldInit f = new FieldInit(); f.printAll();
    }
}
```

What is the result?

- A. c = null  
b = false f = 0.0F
- B. c = 0 b = false f = 0.0f
- C. c = null b = true  
f = 0.0
- D. c =  
b = false f = 0.0

**Answer: D**

**QUESTION: 31**

Given:

```
public class Test {
public static void main(String[] args) { int arr[] = new int[4];
arr[0] = 1;
arr[1] = 2;
arr[2] = 4;
arr[3] = 5; int sum = 0; try {
for (int pos = 0; pos <= 4; pos++) { sum = sum + arr[pos];
}
} catch (Exception e) { System.out.println("Invalid index");
}
System.out.println(sum);
}
}
```

What is the result?

- A. 12
- B. Invalid Index 12
- C. Invalid Index
- D. Compilation fails

**Answer: B**

**Explanation:**

The loop ( for (int pos = 0; pos <= 4; pos++) { }, it should be pos <= 3, causes an exception, which is caught. Then the correct sum is printed.

**QUESTION: 32**

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:** 33

Given:

```

public class SampleClass {
    public static void main(String[] args){
        AnotherSampleClass asc = new AnotherSampleClass();
        SampleClass sc = new SampleClass();
        sc = asc;
        System.out.println("sc: " + sc.getClass());
        System.out.println("asc: " + asc.getClass());
    }
}
class AnotherSampleClass extends SampleClass {
}

```

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 method Returns 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:** 34

Given the code fragment:

```

String color = "teal";
switch (color) {
    case "Red":
        System.out.println("Found Red");
    case "Blue":
        System.out.println("Found Blue");
        break;
    case "Teal":
        System.out.println("Found Teal");
        break;
    default:
        System.out.println("Found Default");
}

```

What is the result?

- A. Found Red Found Default



- B. Found Teal
- C. Found Red Found Blue Found Teal
- D. Found Red Found Blue Found Teal Found Default
- E. Found Default

**Answer:** B

**QUESTION:** 35

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

**QUESTION:** 36

```
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

- A. Option A
- B. Option B
- C. Option C
- D. Option D

**Answer:** B

**QUESTION:** 37  
Give:

```

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.print(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 to runtime error

**Answer:** D

**QUESTION:** 38

Given:

```

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 obj1 = null; obj2.num = 60; graceMarks(obj2);
}
}

```

How many objects are created in the memory runtime?

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

**Answer: B**

**Explanation:**

obj1 and obj3.

when you do e2 = e1 you're copying object references - you're not making a copy of the object - and so the variables e1 and e2 will both point to the same object.

**QUESTION: 39**

```
boolean log3 = ( 5.0 != 6.0) && ( 4 != 5);
```

```
boolean log4 = (4 != 4) || (4 == 4); System.out.println("log3:" + log3 + "\nlog4" + log4);
```

What is the result?

- A. log3:false log4:true
- B. log3:true log4:true
- C. log3:true log4:false
- D. log3:false log4:false

**Answer: B**

**QUESTION: 40**

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.

**Answer: A**

**QUESTION: 41**

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 public static int getLetter() { }; .

**QUESTION: 42**

Given the code format:

```
class DBConfiguration {
    String user;
    String password;
}

And:

4. public class DBHandler {
5.     DBConfiguration configureDB(String uname, String password) {
6.         // insert code here
7.     }
8.     public static void main(String[] args) {
9.         DBHandler r = new DBHandler();
10.        DBConfiguration dbConf = r.configureDB("manager", "manager");
11.    }
12. }
```

Which code fragment must be inserted at line 6 to enable the code to compile?

- A. DBConfiguration f; return f;
- B. Return DBConfiguration;
- C. Return new DBConfiguration;
- D. Retutn 0;

**Answer: B**

**QUESTION: 43**

Given:

```
package p1; public class Test {  
    static double dvalue; static Test ref;  
    public static void main(String[] args) { System.out.println(ref); System.out.println(dvalue);  
    }  
}
```

What is the result?

- A. p1.Test.class 0.0
- B. <the summary address referenced by ref> 0.000000
- C. Null 0.0
- D. Compilation fails
- E. A NullPointerException is thrown at runtime

**Answer: C**

**QUESTION: 44**

Given the code fragment  
`int var1 = -5;  
int var2 = var1--; int var3 = 0;  
if (var2 < 0) { var3 = var2++;  
} else {  
var3 = --var2;  
}  
System.out.println(var3);`  
What is the result?

- A. - 6
- B. - 4
- C. - 5
- D. 5
- E. 4
- F. Compilation fails

**Answer: C**

**QUESTION: 45**

Given the code fragment:

```
System.out.println(2 + 4 * 9 - 3); //Line 21
```

```
System.out.println((2 + 4) * 9 - 3); // Line 22
System.out.println(2 + (4 * 9) - 3); // Line 23
System.out.println(2 + 4 * (9 - 3)); // Line 24
System.out.println((2 + 4 * 9) - 3); // Line 25
```

Which line of codes prints the highest number?

- A. Line 21
- B. Line 22
- C. Line 23
- D. Line 24
- E. Line 25

**Answer:** B

**Explanation:**

The following is printed: 35

```
51
35
26
35
```

**QUESTION:** 46

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 (); bob.name = "Bob";
bob.age = 18;
bob.year = 1982;
}
}
```

What is the result?

- A. year is set to 1982.
- B. bob.year is set to 1982
- C. A runtime error is generated.
- D. A compile time error is generated.

**Answer:** D

**QUESTION:** 47

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. AnArrayIndexOutOfBoundsException is thrown at runtime

**Answer:** A

**QUESTION:** 48

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:** 49

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:** 50

```
public class Two {
    public static void main(String[] args) { try {
        doStuff(); system.out.println("1");
    }
    catch { system.out.println("2");
    }}
    public static void do Stuff() {
        if (Math.random() > 0.5) throw new RuntimeException(); doMoreStuff();
        System.out.println("3 ");
    }
    public static void doMoreStuff() { System.out.println("4");
    }
    }
}
```

Which two are possible outputs?

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

**Answer:** A, B

**Explanation:**

A: Output is 2 if Math.random() is greater than 0.5. B: If Math.random() returns a value less equal to 0.5, the code won't throw an exception, it will continue with the doMore() method which will println "4" after which the program will continue with the doStuff() method and will println "3", after that we will be back in main() and the program will print "1".

**QUESTION:** 51

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:**

Because the Java programming language does not require methods to catch or to specify unchecked exceptions (RuntimeException, Error, and their subclasses), programmers may be tempted to write code that throws only unchecked exceptions or to make all their exception subclasses inherit from RuntimeException. Both of these shortcuts allow programmers to write code without bothering with compiler errors and without bothering to specify or to catch any exceptions. Although this may seem convenient to the programmer, it sidesteps the intent of the catch or specify requirement and can cause problems for others using your classes.

**QUESTION:** 52

Given:

```
package p1;
public interface DoInterface {
    void m1(int n);
    public void m2(int n);          // line n1
}

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.method1(100);
        doi.method2(200);
    }
}
```

What is the result?

- A. 100210
- 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

**Answer:** C

**QUESTION:** 53

Given the following four Java file definitions:

```
// Foo.java
package facades; public interface Foo { }
// Boo.java package facades;
public interface Boo extends Foo { }
// Woofy.java package org.domain
// line n1
public class Woofy implements Boo, Foo { }
// Test.java package.org; public class Test {
public static void main(String[] args) { Foo obj=new Woofy();
Which set modifications enable the code to compile and run?
```

- A. At line n1, Insert: import facades;At line n2, insert:import facades;import org.domain;
- B. At line n1, Insert: import facades.\*;At line n2, insert:import facades;import org.\*;
- C. At line n1, Insert: import facades.\*;At line n2, insert:import facades.Boo;import org.\*;
- D. At line n1, Insert: import facades.Foo, Boo;At line n2, insert:import org.domain.Woofy;
- E. At line n1, Insert: import facades.\*;At line n2, insert:import facades;import org.domain.Woofy;

**Answer:** E

**QUESTION:** 54

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 at test.Teagle.main(Teagle.java:14)

D. Invalid Divisor Exception in thread "main" java.lang.ArithmeticException: / by zero at test.Teagle.main(Teagle.java:14) Successful Division 1

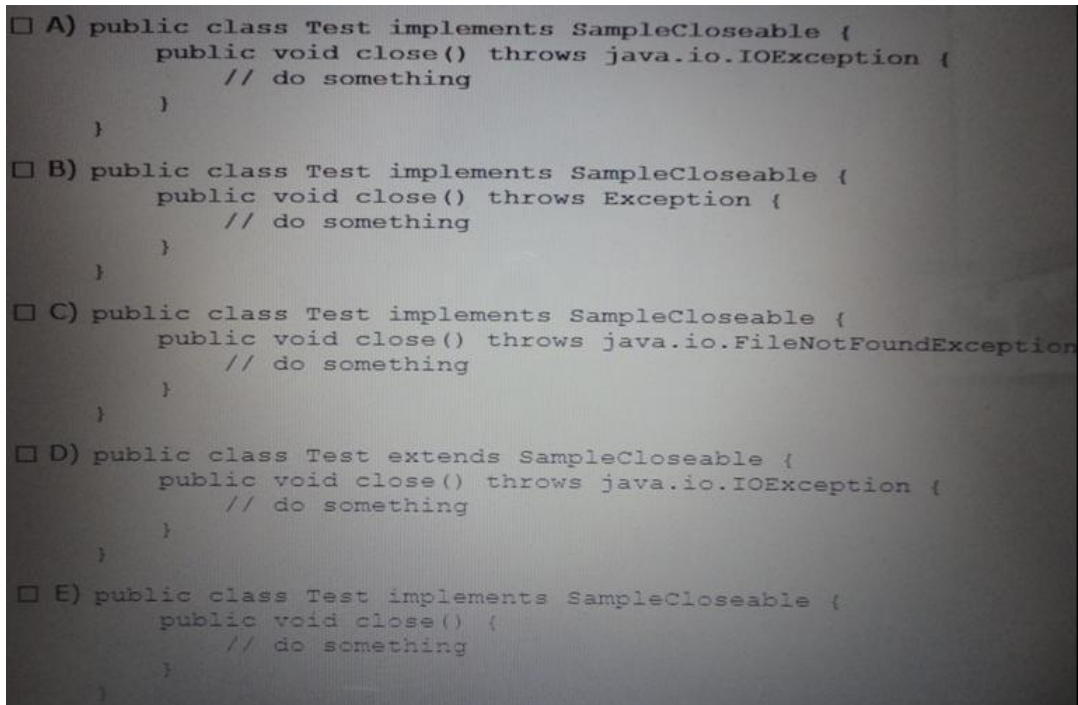
**Answer:** C

**QUESTION: 55**

Given the code fragment:

```
interface SampleClosable {
    public void close () throws java.io.IOException;
}
```

Which three implementations are valid?



A. Option A

B. Option B

C. Option C

D. Option D

E. Option E

**Answer:** A, C, E

**Explanation:**

A: Throwing the same exception is fine.

C: Using a subclass of java.io.IOException (here java.io.FileNotFoundException) is fine E:  
Not using a throw clause is fine.

**QUESTION: 56**

```
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

**Answer: A**

**QUESTION: 57**

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();
    }
}
```

- 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

**Answer:** B

**QUESTION:** 58

Given:

```
class Patient {
    String name;
    public Patient(String name) {
        this.name = name;
    }
}

And the code fragment:

8. public class Test {
9.     public static void main(String[] args) {
10.         List ps = new ArrayList();
11.         Patient p2 = new Patient("Mike");
12.         ps.add(p2);
13.
14.         // insert code here
15.
16.         if (f >= 0 ) {
17.             System.out.print("Mike Found");
18.         }
19.     }
20. }
```

Which code fragment, when inserted at line 14, enables the code to print Mike Found?

- A. int f = ps.indexOf {new patient ("Mike")};  
 B. int f = ps.indexOf (patient("Mike"));  
 C. patient p = new Patient ("Mike"); int f = pas.indexOf(P)  
 D. int f = ps.indexOf(p2);

**Answer:** C

**QUESTION:** 59

Given:



```

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();
    }
}

```

- A. Ym Xm2
- B. Ym Xm1
- C. Compilation fails
- D. A ClassCastException is thrown at runtime

**Answer:** B

**QUESTION:** 60

Given the following code:

```

1. public class Simple {
2.     public float price;
3.     public static void main(String[] args) {
4.         Simple price = new Simple();
5.         price = 4;
6.     }
7. }

```

What will make this code compile and run?

- A. Change line 2 to the following: Public int price
- B. Change line 4 to the following: int price = new simple ();
- C. Change line 4 to the following: Float price = 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:** 61

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:** A

**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: 62**

Which two are Java Exception classes?

- A. SercurityException
- B. DuplicatePathException
- C. IllegalArgumentException
- D. TooManyArgumentsException

**Answer:** A, C

**QUESTION: 63**

Given:

```
public class ScopeTest {
    int j, int k;
    public static void main(String[] args) { ew ScopeTest().doStuff(); }
    void doStuff() { nt x = 5; oStuff2();
    System.out.println("x");
    }
    void doStuff2() { nt y = 7;
    ystem.out.println("y");
    or (int z = 0; z < 5; z++) { ystem.out.println("z");
    ystem.out.println("y");
    }
}
```

Which two items are fields?

- A. j
- B. k
- C. x
- D. y
- E. z

**Answer:** A, B

**QUESTION: 64**

Give:

```

public class MyFive {
    public 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 loop 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:** 65

Given:

```

public class Test {
    public static void main(String[] args) { int day = 1;
    switch (day) {
    case "7": System.out.print("Uranus");
    case "6": System.out.print("Saturn");
    case "1": System.out.print("Mercury");
    case "2": System.out.print("Venus");
    case "3": System.out.print("Earth");
    case "4": System.out.print("Mars");
    case "5": System.out.print("Jupiter");
    }
    }
}

```

Which two modifications, made independently, enable the code to compile and run?

- A. Adding a break statement after each print statement
- B. Adding a default section within the switch code-block
- C. Changing the string literals in each case label to integer
- D. Changing the type of the variable day to String

E. Arranging the case labels in ascending order

**Answer:** A, C

**Explanation:**

The following will work fine:

```
public class Test {
    public static void main(String[] args) { int day = 1;
    switch (day) {
    case 7: System.out.print("Uranus"); break; case 6: System.out.print("Saturn"); break; case
    1: System.out.print("Mercury"); break; case 2: System.out.print("Venus"); break; case 3:
    System.out.print("Earth"); break; case 4: System.out.print("Mars"); break; case 5:
    System.out.print("Jupiter"); break;
    }
    }
}
```

**QUESTION:** 66

Given the code fragment:

```
float x = 22.00f % 3.00f; int y = 22 % 3;
System.out.print(x + ", " + y);
```

What is the result?

- A. 1.0, 1
- B. 1.0f, 1
- C. 7.33, 7
- D. Compilation fails
- E. An exception is thrown at runtime

**Answer:** A

**QUESTION:** 67

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. 3 false 1
- B. 2 true 3

- C. 2 false 3
- D. 3 true 1
- E. 3 false 3
- F. 2 true 1
- G. 2 false 1

**Answer:** D

**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:** 68

Which two statements correctly describe checked exception?

- A. These are exceptional conditions that a well-written application should anticipate and recover from.
- B. These are exceptional conditions that are external to the application, and that the application usually cannot anticipate or recover from.
- C. These are exceptional conditions that are internal to the application, and that the application usually cannot anticipate or recover from.
- D. Every class that is a subclass of RuntimeException and Error is categorized as checked exception.
- E. Every class that is a subclass of Exception, excluding RuntimeException and its subclasses, is categorized as checked exception.

**Answer:** B, D

**Explanation:**

Checked exceptions:

\* (B) represent invalid conditions in areas outside the immediate control of the program (invalid user input, database problems, network outages, absent files)

\* are subclasses of Exception

It's somewhat confusing, but note as well that RuntimeException (unchecked) is itself a subclass of Exception (checked).

\* a method is obliged to establish a policy for all checked exceptions thrown by its implementation (either pass the checked exception further up the stack, or handle it somehow)

**Reference:** Checked versus unchecked exceptions

**QUESTION:** 69

Given:

```

1. public class SampleClass {
2.     public static void main(String[] args){
3.         AnotherSampleClass asc = new AnotherSampleClass();
4.         SampleClass sc = new SampleClass();
5.         //insert code here
6.     }
7. }
8. class AnotherSampleClass extends SampleClass {
9. }

```

Which statement, when inserted into line 5, is valid change?

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

**Answer:** B

**Explanation:**

Works fine.

**QUESTION:** 70

Given:

```

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

**Answer:** B

**QUESTION:** 71

Given the code fragment:

```
if (aVar++ < 10) {  
    System.out.println(aVar + " Hello World!");  
} else {  
    System.out.println(aVar + " Hello Universe!");  
}
```

What is the result if the integer aVar is 9?

- A. 10 Hello world!
- B. 10 Hello universe!
- C. 9 Hello world!
- D. Compilation fails.

**Answer:** A

**QUESTION:** 72

Given:

```
public class CharToStr {  
    public static void main(String[] args) {  
        String str1 = "Java";  
        char str2[] = { 'J', 'a', 'v', 'a' };  
        String str3 = null;  
        for (char c : str2) {  
            str3 = str3 + c;  
        }  
        if (str1.equals(str3))  
            System.out.print("Successful");  
        else  
            System.out.print("Unsuccessful");  
    }  
}
```

What is result?

- A. Successful
- B. Unsuccessful
- C. Compilation fails
- D. An exception is thrown at runtime

**Answer:** C

**QUESTION: 73**

Given:

```

1. import java.io.Error;
2.     public class TestApp {
3.     public static void main(String[] args) {
4.         TestApp t = new TestApp();
5.         try {
6.             t.doPrint();
7.             t.doList();
8.
9.         } catch (Exception e2) {
10.            System.out.println("Caught " + e2);
11.        }
12.    }
13.    public void doList() throws Exception {
14.        throw new Error("Error");
15.    }
16.    public void doPrint() throws Exception {
17.        throw new RuntimeException("Exception");
18.    }
19. }

```

What is the result?

```

C 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)

C B) Exception in thread "main" java.lang.Error: Error
    at TestApp.doList(TestApp.java: 14)
    at TestApp.main(TestApp.java: 6)

C C) Caught java.lang.RuntimeException: Exception
    Caught java.lang.Error: Error

C D) Caught java.lang.RuntimeException: Exception

```

- A. Option A
- B. Option B
- C. Option C
- D. Option D

**Answer: C****QUESTION: 74**

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 doSomething() 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:** 75

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: 76**

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

**Answer: D**

**QUESTION: 77**

Given:

```
public class Natural { private int i;
void disp() { while (i <= 5) {
for (int i=1; i <=5;) { System.out.print(i + " "); i++;
} i++;
}
}
public static void main(String[] args) { new Natural().disp();
}
}
```

What is the result?

- A. Prints 1 2 3 4 5 once
- B. Prints 1 3 5 once
- C. Prints 1 2 3 4 5 five times
- D. Prints 1 2 3 4 5 six times
- E. Compilation fails

**Answer:** D

**Explanation:**

1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

**QUESTION:** 78

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

**Answer:** A, D

**Reference:**

[http://www.tutorialspoint.com/java/java\\_access\\_modifiers.htm](http://www.tutorialspoint.com/java/java_access_modifiers.htm)

**QUESTION:** 79

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: 80**

Given the code fragment:

```
for (int ii = 0; ii < 3; ii++) { int count = 0;
for (int jj = 3; jj > 0; jj--) { if (ii == jj) {
++count; break;
}
}
System.out.print(count); continue;
}
```

What is the result?

- A. 011
- B. 012
- C. 123
- D. 000

**Answer: A**

**QUESTION: 81**

Which two statements are true for a two-dimensional array?

- A. It is implemented as an array of the specified element type.
- B. Using a row by column convention, each row of a two-dimensional array must be of the same size.
- C. At declaration time, the number of elements of the array in each dimension must be specified.
- D. All methods of the class Object may be invoked on the two-dimensional array.

**Answer: A, D**

**QUESTION: 82**

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

```

☐ A) int[][] array2D = { {0,1,2,4}, {5,6} };
☐ B) int[][] array2D = new int[][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[] array = {0,1};
    int[][][] array3D = new int[2][2][2];
    array3D[0][0] = array;
    array3D[0][1] = array;
    array3D[1][0] = array;
    array3D[1][1] = array;
☐ E) int[][] array2D = { 0,1 };

```

- A. Option A
- B. Option B
- C. Option C
- D. Option D
- E. Option E

**Answer:** B, D

**Explanation:**

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

**QUESTION:** 83

Given:

```

public class Vowel {
    private char var;
    public static void main(String[] args) {
        char var1 = 'a';
        char var2 = var1;
        var2 = 'e';

        Vowel obj1 = new Vowel();
        Vowel obj2 = obj1;
        obj1.var = 'i';
        obj2.var = 'o';

        System.out.println(var1 + ", " + var2);
        System.out.print(obj1.var + ", " + obj2.var);
    }
}

```

- A. a, e i, o
- B. a, e o, o
- C. e, e I, o
- D. e, e o, o

**Answer:** B

**QUESTION:** 84

Which three statements are benefits of encapsulation?

- A. Allows a class implementation to change without changing the clients
- B. Protects confidential data from leaking out of the objects
- C. Prevents code from causing exceptions
- D. Enables the class implementation to protect its invariants
- E. Permits classes to be combined into the same package
- F. Enables multiple instances of the same class to be created safely

**Answer:** A, B, D

**QUESTION:** 85

1. class StaticMethods {
2. static void one() {
3. two();
4. StaticMethods.two();
5. three();
6. StaticMethods.four(); 7. }
8. static void two() { }
9. void three() {
10. one();
11. StaticMethods.two();
12. four();
13. StaticMethods.four();
14. }
15. void four() { }
16. }

Which three lines are illegal?

- A. line 3
- B. line 4
- C. line 5
- D. line 6
- E. line 10

- F. line 11
- G. line 12
- H. line 13

**Answer:** C, D, H

**QUESTION:** 86

Which two statements are true for a two-dimensional array of primitive data type?

- A. It cannot contain elements of different types.
- B. The length of each dimension must be the same.
- C. At the declaration time, the number of elements of the array in each dimension must be specified.
- D. All methods of the class object may be invoked on the two-dimensional array.

**Answer:** C, D

**Explanation:**

<http://stackoverflow.com/questions/12806739/is-an-array-a-primitive-type-or-an-object-or-something-else-entirely>

**QUESTION:** 87

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 of `java.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.

### QUESTION: 88

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: A**

### QUESTION: 89

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. `Java.util`
- D. `Javax.net`
- E. `Java.*`
- F. The package with no name



**Answer:** A, F

**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:** 90

Which is a valid abstract class?

- A. `public abstract class Car { protected void accelerate(); }`
- B. `public interface Car { protected abstract void accelerate(); }`
- C. `public abstract class Car { protected final void accelerate(); }`
- D. `public abstract class Car { protected abstract void accelerate(); }`
- E. `public abstract class Car { protected abstract void accelerate() { //more car can do } }`

**Answer:** D

**QUESTION:** 91

Given:

```
public class App {
// Insert code here
System.out.print("Welcome to the world of Java");
}
}
```

Which two code fragments, when inserted independently at line // Insert code here, enable the program to execute and print the welcome message on the screen?

- A. `static public void main (String [] args) {`
- B. `static void main (String [] args) {`
- C. `public static void Main (String [] args) {`
- D. `public static void main (String [] args) {`
- E. `public void main (String [] args) {`

**Answer:** A, D

**Explanation:**

Incorrect:

Not B: No main class found.

Not C: Main method not found not E: Main method is not static.

**QUESTION: 92**

View the Exhibit.

```
public class Hat { public int ID =0;
public String name = "hat";
public String size = "One Size Fit All"; public String color="";
public String getName() { return name; } public void setName(String name) { this.name =
name;
}
}
```

Given

```
public class TestHat {
public static void main(String[] args) { Hat blackCowboyHat = new Hat();
}
}
```

Which statement sets the name of the Hat instance?

- A. blackCowboyHat.setName = "Cowboy Hat";
- B. setName("Cowboy Hat");
- C. Hat.setName("Cowboy Hat");
- D. blackCowboyHat.setName("Cowboy Hat");

**Answer:** D

**QUESTION: 93**

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: 94**

Given:

```
public class Test {

    static void dispResult(int[] num) {
        try {
            System.out.println(num[1] / (num[1] - num[2]));
        } catch (ArithmeticException e) {
            System.err.println("first exception");
        }
        System.out.println("Done");
    }

    public static void main(String[] args) {
        try {
            int[] arr = {100, 100};
            dispResult(arr);
        } catch (IllegalArgumentException e) {
            System.err.println("second exception");
        } catch (Exception e) {
            System.err.println("third exception");
        }
    }
}
```

What is the result?

- A. 0 Done
- B. First Exception Done
- C. Second Exception
- D. Done Third Exception
- E. Third Exception

**Answer: B**

**QUESTION: 95**

Given:

```

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
- B. 3 y: 4
- C. x: 0 y: 0
- D. 3 y: 4
- E. x: 1 y: 2
- F. 0 y: 0
- G. x: 0 y: 0
- H. 0 y: 0

**Answer:** C

**QUESTION:** 96

Given:

```

abstract class A1 {
    public abstract void m1();
    public void m2() { System.out.println("Green"); }
}

abstract class A2 extends A1 { public abstract void m3();
    public void m1() { System.out.println("Cyan"); } public void m2() {
        System.out.println("Blue"); }
}

public class A3 extends A2 {
    public void m1() { System.out.println("Yellow"); } public void m2() {
        System.out.println("Pink"); } public void m3() { System.out.println("Red"); } public static
    void main(String[] args) {
        A2 tp = new A3(); tp.m1();
        tp.m2();
    }
}

```

```
tp.m3();
}
}
```

What is the result?

- A. Yellow Pink Red
- B. Cyan Blue Red
- C. Cyan Green Red
- D. Compilation Fails

**Answer:** A

**QUESTION:** 97

Given:

```
public class ColorTest {
    public static void main(String[] args) {
        String[] colors = {"red", "blue", "green", "yellow", "maroon", "cyan"}; int count = 0;
        for (String c : colors) { if (count >= 4) { break;
        }
        else { continue;
        }
        if (c.length() >= 4) { colors[count] = c.substring(0,3);
        }
        count++;
        }
        System.out.println(colors[count]);
        }
        }
```

What is the result?

- A. Yellow
- B. Maroon
- C. Compilation fails
- D. A StringIndexOutOfBoundsException is thrown at runtime.

**Answer:** C

**Explanation:**

The line, if (c.length() >= 4) {, is never reached. This causes a compilation error.

Note: The continue statement skips the current iteration of a for, while, or do-while loop.

An unlabeled break statement terminates the innermost switch, for, while, or do-while statement, but a labeled break terminates an outer statement.

**QUESTION: 98**

Given the code fragment:

```
public class ForTest {
    public static void main(String[] args) { int[] array = {1, 2, 3};
    for ( foo ) {
    }
    }
```

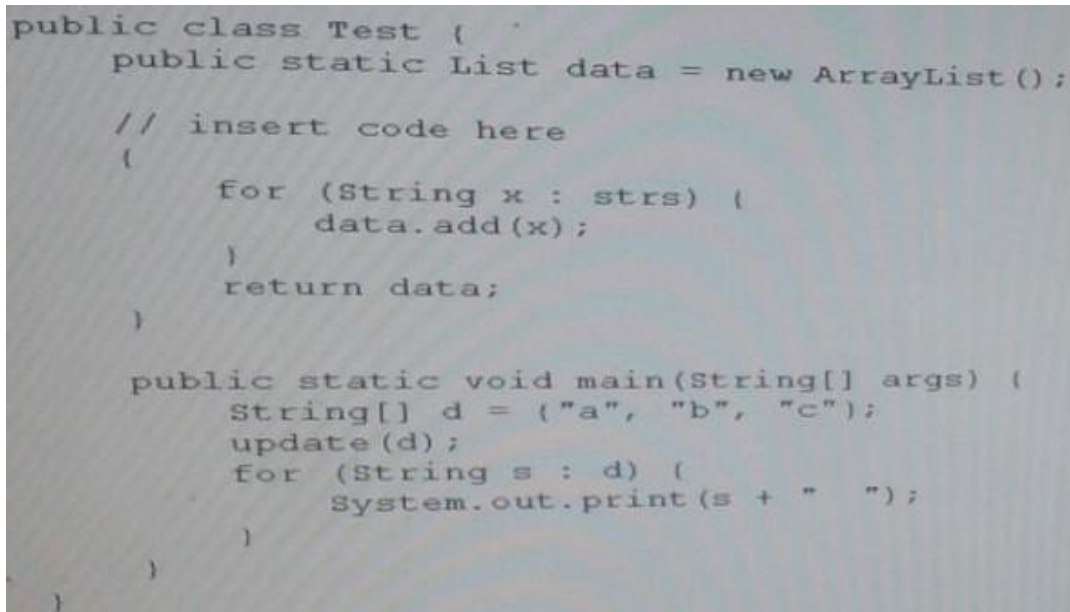
Which three code fragments, when replaced individually for foo, enables the program to compile?

- A. `int i : array`
- B. `int i = 0; i < 1;`
- C. `;;`
- D. `; i < 1; i++`
- E. `i = 0; i < 1;`

**Answer:** A, B, C

**QUESTION: 99**

Given the code fragment:



```
public class Test {
    public static List data = new ArrayList();

    // insert code here
    {
        for (String x : strs) {
            data.add(x);
        }
        return data;
    }

    public static void main(String[] args) {
        String[] d = {"a", "b", "c"};
        update(d);
        for (String s : d) {
            System.out.print(s + " ");
        }
    }
}
```

Which code fragment, when inserted at // insert code here, enables the code to compile and print a b c?

- A. `List update (String[] strs)`
- B. `Static ArrayListupdate(String [] strs)`
- C. `Static List update (String [] strs)`

- D. Static void update (String[] str)
- E. ArrayList static update(String [] str)

**Answer:** E

**QUESTION:** 100

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

**Answer:** D

**Reference:**

<http://beginnersbook.com/2013/05/java-access-modifiers/>

**QUESTION:** 101

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. A StringIndexOutOfBoundsException is thrown at runtime
- D. An ArrayIndexOutOfBoundsException is thrown at runtime
- E. A NullPointerException 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: 102**

Given:

```
import java.util.*; public class Ref {
public static void main(String[] args) {
StringBuilder s1 = new StringBuilder("Hello Java!"); String s2 = s1.toString();
List<String> lst = new ArrayList<String>(); lst.add(s2); System.out.println(s1.getClass());
System.out.println(s2.getClass()); System.out.println(lst.getClass());
}
}
```

What is the result?

- A. class java.lang.String class java.lang.String class java.util.ArrayList
- B. class java.lang.Object class java.lang. Object class java.util.Collection
- C. class java.lang.StringBuilder class java.lang.String class java.util.ArrayList
- D. class java.lang.StringBuilder class java.lang.String class java.util.List

**Answer: C**

**Explanation:**

class java.lang.StringBuilder class java.lang.String  
class java.util.ArrayList

**QUESTION: 103**

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;
```

Which code fragment can be inserted at line //insert code here to enable the code to compile?

- A. Cricket() {  
super(11, "Cricket", "Condition 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, ns) { this.num_umpires =nu; this.num_substitutes=ns;}
D. Cricket() { this.num_umpires =3; this.num_substitutes=2;
super(11, "Cricket", "Condidtion OK");}

```

**Answer:** A

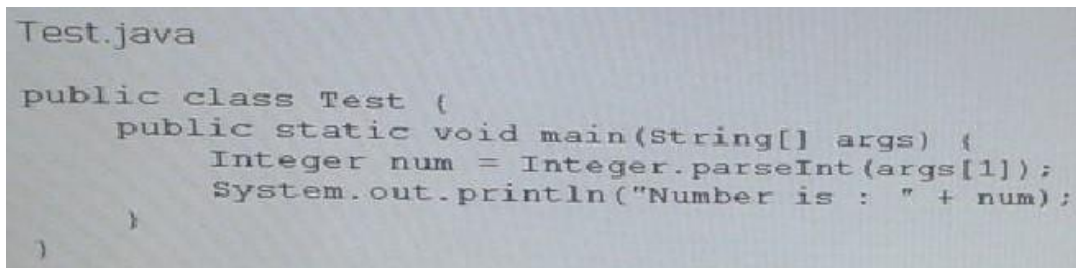
**Explanation:**

Incorrect:

not C, not D: call to super must be the first statement in constructor.

**QUESTION:** 104

Given:



```

Test.java

public class Test {
    public static void main(String[] args) {
        Integer num = Integer.parseInt(args[1]);
        System.out.println("Number is : " + num);
    }
}

```

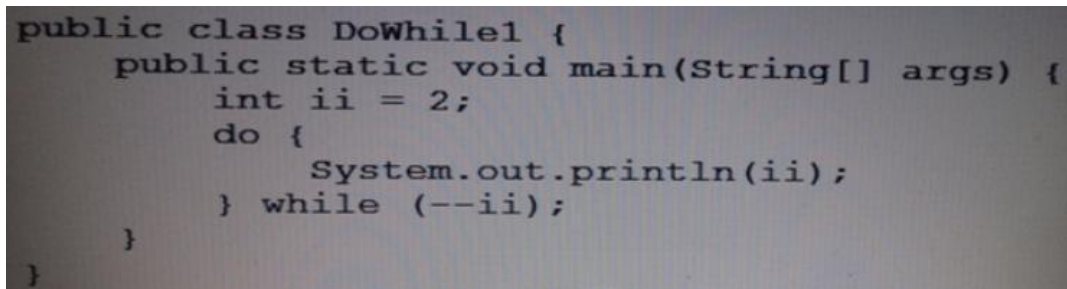
And the commands: Javac Test.java Java Test 12345 What is the result?

- A. Number us : 12345
- B. A NullPointerException is thrown at runtime
- C. A NumberFormatException is thrown at runtime
- D. AnArrayIndexOutOfBoundsException is thrown at runtime.

**Answer:** A

**QUESTION:** 105

Given:



```

public class DoWhile1 {
    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:** 106

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:** 107

Given:

```
public class TestLoop {
    public static void main(String[] args) { int array[] = {0, 1, 2, 3, 4};
    int key = 3;
    for (int pos = 0; pos < array.length; ++pos) { if (array[pos] == key) { break;}
    }
    System.out.print("Found " + key + "at " + pos);
    }
}
```

What is the result?

- A. Found 3 at 2

- B. Found 3 at 3
- C. Compilation fails
- D. An exception is thrown at runtime

**Answer:** C

**Explanation:**

The following line does not compile: `System.out.print("Found " + key + "at " + pos);` The variable `pos` is undefined at this line, as its scope is only valid in the for loop. Any variables created inside of a loop are LOCAL TO THE LOOP.

**QUESTION:** 108

```
int [] array = {1,2,3,4,5}; for (int i: array) {
if ( i < 2) { keyword1 ;
}
System.out.println(i); if ( i == 3) {
keyword2 ;
}}
```

What should `keyword1` and `keyword2` be respectively, in order to produce output 2345?

- A. continue, break
- B. break, break
- C. break, continue
- D. continue, continue

**Answer:** D

**QUESTION:** 109

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 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 equals method compares 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:** 110

Given the code fragment:

```
String name = "Spot"; int age = 4;
```

```
String str = "My dog " + name + " is " + age; System.out.println(str);
```

And

```
StringBuilder sb = new StringBuilder();
```

Using StringBuilder, which code fragment is the best potion to build and print the following string My dog Spot is 4

- A. sb.append("My dog " + name + " is " + age); System.out.println(sb);
- B. sb.insert("My dog ").append( name + " is " + age); System.out.println(sb);
- C. sb.insert("My dog ").insert( name ).insert(" is ").insert(age); System.out.println(sb);
- D. sb.append("My dog ").append( name ).append(" is ").append(age); System.out.println(sb);

**Answer:** A, D

**QUESTION:** 111

Given the code fragment:

```

12. int row = 10;
13. for ( ; row > 0 ; ) {
14.     int col = row;
15.     while (col >= 0) {
16.         System.out.print(col + " ");
17.         col -= 2;
18.     }
19.     row = row / col;
20. }

```

What is the result?

- A. 10 8 6 4 2 0
- B. 10 8 6 4 2
- C. AnArithmeticException is thrown at runtime
- D. The program goes into an infinite loop outputting: 10 8 6 4 2 0. . .
- E. Compilation fails

**Answer:** B

**QUESTION:** 112

Given the code fragment:

```

public class Test {
    static String[][] arr = new String[3][]; private static void doPrint() {
//insert code here
    }
    public static void main(String[] args) { String[] class1 = {"A","B","C"};
    String[] class2 = {"L","M","N","O"}; String[] class3 = {"I","J"};
    arr[0] = class1; arr[1] = class2; arr[2] = class3; Test.doPrint();
    }
}

```

Which code fragment, when inserted at line //insert code here, enables the code to print COJ?

- A. `int i = 0;`  
`for (String[] sub: arr) { int j = sub.length - 1; for (String str: sub) {`  
`System.out.println(str[j]); i++;`  
`}`  
`}`
- B. `private static void doPrint() { for (int i = 0; i < arr.length; i++) { int j = arr[i].length - 1;`  
`System.out.print(arr[i][j]);`  
`}`  
`}`
- C. `int i = 0;`

```

for (String[] sub: arr[i]) { int j = sub.length; System.out.print(arr[i][j]); i++;
}
D. for (int i = 0; i < arr.length-1; i++) { int j = arr[i].length-1; System.out.print(arr[i][j]);
i++;
}

```

**Answer:** B

**Explanation:**

Incorrect:

not A: The following line causes a compile error: `System.out.println(str[j]);`

Not C: Compile error line: `for (String[] sub: arr[i])` not D: Output: C

**QUESTION:** 113

Given:

```

public class Equal {
public static void main(String[] args) { String str1 = "Java";
String[] str2 = {"J","a","v","a"}; String str3 = "";
for (String str : str2) { str3 = str3+str;
}
boolean b1 = (str1 == str3); boolean b2 = (str1.equals(str3)); System.out.print(b1+" "+b2);
}

```

What is the result?

- A. true, false
- B. false, true
- C. true, true
- D. false, false

**Answer:** B

**Explanation:**

`==` strict equality. `equals` compare state, not identity.

**QUESTION:** 114

Given:

```

1. public class Speak {
2.     public static void main(String[] args){
3.         Speak speakIt = new Tell();
4.         Tell tellIt = new Tell();
5.         speakIt.tellItLikeItIs();
6.         (Truth)speakIt.tellItLikeItIs();
7.         ((Truth)speakIt).tellItLikeItIs();
8.         tellIt.tellItLikeItIs();
9.         (Truth)tellIt.tellItLikeItIs();
10.        ((Truth)tellIt).tellItLikeItIs();
11.    }
12. }
13. class Tell extends Speak implements Truth {
14.     public void tellItLikeItIs() {
15.         System.out.println("Right on!");
16.     }
17. }
18. interface Truth { public void tellItLikeItIs(); }

```

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

- A. Line 5
- B. Line 6
- C. Line 7
- D. Line 8
- E. Line 9
- F. Line 10

**Answer:** C, D, F

**QUESTION:** 115

Given:

```

public class MyFor3 {
    public static void main(String[] args) {
        int[] xx = null;
        for (int ii: xx) {
            System.out.println(ii);
        }
    }
}

```

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:** 116

Given:

```
class Base {  
    public static void main(String[] args) { System.out.println("Base " + args[2]);  
    }  
}  
public class Sub extends Base{  
    public static void main(String[] args) { System.out.println("Overriden " + args[1]);  
    }  
}
```

And the commands: javac Sub.java java Sub 10 20 30 What is the result?

- A. Base 30
- B. Overriden 20
- C. Overriden 20 Base 30
- D. Base 30 Overriden 20

**Answer:** B