



Vendor: Oracle

Exam Code: 1Z0-803

Exam Name: Java SE 7 Programmer I

Version: 16.032

QUESTION 1

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: CD**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 2

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 3

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 "// 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.

Incorrect answers:

asc = sc.clone();

Incompatible types.

asc =sc;

Incompatible types.

asc = (object) sc;

Syntax error

QUESTION 4

Given the code fragment:

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

What is the result?

- A. Result: 10
Result: 30

- B. Result: 10
Result: 25
- C. Result: 235**
Result: 215
- D. Result: 215
Result: 215
- E. Compilation fails

Answer: C

Explanation:

First line:

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

String concatenation is produced.

Second line:

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

3*5 is calculated to 15 and is appended to string 2. Result 215.

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 + 1 * 5));

run:

Result: 10

Result: 7

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 5

Which code fragment is illegal?

- A) class Basel {
 abstract class Abs1 { }
}
- B) abstract class Abs1 {
 void doit() { }
}
- C) class Basel { }
 abstract class Abs1 extends Basel { }
- D) abstract int var1 = 89;

A. Option A

B. Option B

- C. Option C
- D. Option D**

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 6

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 7

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 8

Given a java source file:

```
class X {
    X() { }
    private void one() { }
}

public class Y extends X {
    Y() { }
    private void two() { one(); }
    public static void main(String[] args) {
        new Y().two();
    }
}
```

What changes will make this code compile?

- A. adding the public modifier to the declaration of class x
- B. adding the protected modifier to the x() constructor
- C. changing the private modifier on the declaration of the one() method to protected
- D. removing the Y () constructor
- E. removing the private modifier from the two () method

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 9

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 10

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.*;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: CEF

Explanation:

Three separate solutions:

C: the full class path to the method must be stated (when we have not imported the package)

E: We can import the hold dandy class

F: we can import the specific method

QUESTION 11

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: CDF

QUESTION 12

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.

Incorrect answers:

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 compares values for equality.

There is no generic comparison method named same.

= = (with a space) is not a valid method.

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 13

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: AD

Explanation:

int[][] array2D; is the standard convention to declare a 2-dimensional integer array.

int[] array2D[]; works as well, but it is not recommended.

Incorrect answers:

int[2][2] array2D;

The size of the array cannot be defined this way.

int array2D[]; is good definition of a one-dimensional array.

int[] []array2D[]; is good definition of a three-dimensional array.

QUESTION 14

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: AD

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.

Incorrect answers:

B: The second println statement would not run.

C: The first println statement will always run.

QUESTION 15

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 16

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: AB

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 17

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

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

Answer: BC

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 18

Given the code fragment:
What is the result?

```
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");  
}
```

- 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 19

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 20

Which three are bad practices?

- A. Checking for `ArrayIndexOutOfBoundsException` when iterating through an array to determine when all elements have been visited
- B. Checking for `Error` and, if necessary, restarting the program to ensure that users are unaware of problems
- C. Checking for `FileNotFoundException` to inform a user that a filename entered is not valid
- D. Checking for `ArrayIndexOutOfBoundsException` and ensuring that the program can recover if one occurs
- E. Checking for `IOException` and ensuring that the program can recover if one occurs

Answer: ABD

QUESTION 21

Given:

Which two declarations will compile?

```
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;  
}
```

- 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: AD

Explanation:

Incorrect answers:

`int a, b, c;`

OK, but duplicate definition.

QUESTION 22

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 23

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 24

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:

```
(int capacity)
StringBuilder
```

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.

Incorrect answers:

```
StringBuilder sb = new String("128");
StringBuilder not String is required.
setCapacity or getInstance do not work.
```

QUESTION 25

Given:

What is the result?

```
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);
    }
}
```

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 26

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 27

Given the fragment:

What is the result?

```
int[] array = {1,2,3,4,5};  
System.arraycopy(array, 2, array, 1, 2);  
System.out.print(array[1]);  
System.out.print(array[4]);
```

- 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 copyfrom and the array to copyto. 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 28

Given the following code fragment:

What is the result if the integer value is 33?

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

- 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 29

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: ACE

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.

Incorrect answers:

B: The standard exceptions do not cover all possible errors.

D: Exceptions do not need to be handled in the method where they occurred.

They can be handled in the method which is calling the method that throws an exception.

QUESTION 30

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.

Incorrect answers:

the compareTo method produces and int, not a boolean.
i = 1 is an assignment, not a comparison.
i == 2? -1:0 would produce the integer 0. A Boolean value is needed.

QUESTION 31

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();
square.foo("bar");

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

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

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

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

Answer: D

QUESTION 32

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: AE

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.

Incorrect answers:

Java.awt: basic hierarchy of packages for native GUI components Javax.net: networking

operations, sockets, DNS lookups, etc.

QUESTION 33

Given:

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

```
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 { }
```

- A. Reference type is Z; object type is Z.
- B. Reference type is Y; object type is Y.
- C. Reference type is Z; object type is Y.
- D. Reference type is X; object type is Z.

Answer: C

QUESTION 34

Given:

What is the result?

```
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 { }
```

- 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 35

Given the code fragment:

How many times is 2 printed?

```
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(ii);  
            ii++;  
            break;  
        }  
    }  
}
```

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 elements in table.

The break statement breaks the inner loop immediately each time.

2 will be printed once only.

Note: If the line int ii = 0; is missing the program would not compile.

QUESTION 36

Given:

What is the result?

```
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;
    }
}
```

- 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 37

Given:

What is true about the class Wow?

```
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. }
```

- 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

Explanation:

An abstract class can contain abstract and non-abstract methods. However, an abstract method must be contained within an abstract class.

An abstract class can have private methods as long as they are not declared abstract.

An abstract class can have instance variables.

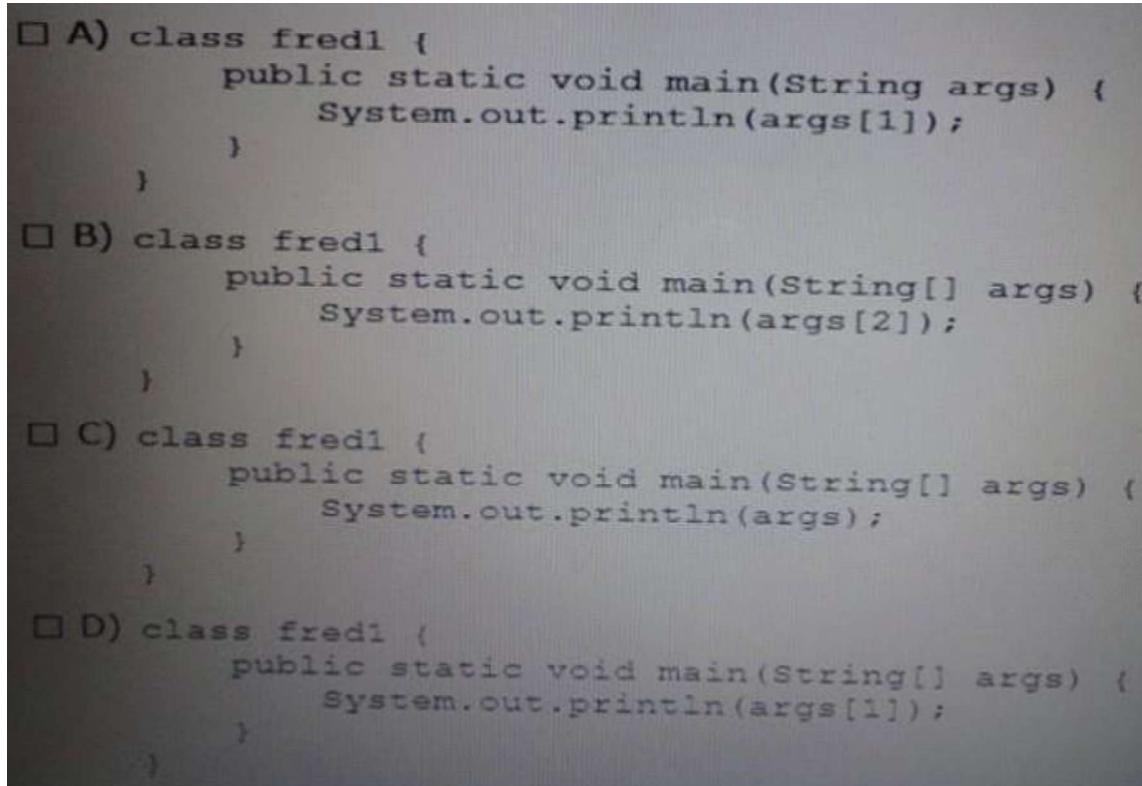
An abstract class does not have to have at least one abstract method.

An abstract class does not have to have a constructor without arguments.

QUESTION 38

Which two will compile, and can be run successfully using the command:

Java fred1 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]);
    }
}
```

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

Answer: CD**Explanation:**

Throws java.lang.ArrayIndexOutOfBoundsException: 2

at certquestions.Fred1.main(Fred1.java:3)

C. Prints out: [Ljava.lang.String;@39341183

D. Prints out: walls

QUESTION 39

Given:

What is the result?

```
class X {  
    static void m(int i) {  
        i += 7;  
    }  
    public static void main(String[] args) {  
        int j = 12;  
        m(j);  
        System.out.println(j);  
    }  
}
```

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

Answer: B

QUESTION 40

Given:

What is the result?

```
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);  
    }  
}
```

- 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 41

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.
- E. An abstract class can be extended by a concrete class.**
- F. An abstract class CANNOT be extended by an abstract class.

Answer: AE

Explanation:

<http://docs.oracle.com/javase/tutorial/java/landl/abstract.html>

QUESTION 42

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 43

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 `ArrayListOutOfBoundsException` is thrown at runtime
- E. A `NullPointerException` is thrown at runtime
- F. A `StringArrayIndexOutOfBoundsException` is thrown at runtime

Answer: C

Explanation:

The code compiles fine.

At runtime an `IndexOutOfBoundsException` is thrown when the second list item is added.

QUESTION 44

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 45

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 46

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 47

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 48

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: AD

QUESTION 49

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:

According to Oracle, the Java programming language does not require methods to catch or to specify unchecked exceptions (RuntimeException, Error, and their subclasses).

QUESTION 50

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 implements SampleCloseable {  
        public void close() throws java.io.FileNotFoundException {  
            // do something  
        }  
    }  
  
 D) public class Test extends SampleCloseable {  
        public void close() throws java.io.IOException {  
            // do something  
        }  
    }  
  
 E) public class Test implements SampleCloseable {  
        public void close() {  
            // do something  
        }  
    }
```

A. Option A

B. Option B

- C. Option C
- D. Option D
- E. Option E

Answer: ACE

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.

Incorrect answers:

- B: Exception is not a subclass of java.io.IOException and cannot be used here.
- D: Not extends. SampleCloseable cannot be the superclass of Test, a superclass must be a class.

(An interface extends other interfaces.)

QUESTION 51

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 52

Given the following code:

What will make this code compile and run?

```
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. }
```

- 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 53

Given:

What is the result?

```
public class DoWhile1 {  
    public static void main(String[] args) {  
        int ii = 2;  
        do {  
            System.out.println(ii);  
        } while (--ii);  
    }  
}
```

- 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 54

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: AD

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 55

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: AD

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 56

Which three are valid types for switch?

- A. int
- B. float
- C. double
- D. integer

- E. String
- F. Float

Answer: ADE

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 57

Give:

What value should replace KK in line x to cause jj = 5 to be output?

```
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);
    }
}
```

- 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 58

Given the following code fragment:

What is the result if the integer value is 33?

```
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( .... );
```

- A. The fox jump lazy...
- B. The fox lazy...**
- C. Quick fox over lazy ...
- D. Quick fox the

Answer: B

QUESTION 59

Given the code fragment:

```
Int [] [] array = {{0}, {0, 1}, {0, 2, 4}, {0, 3, 6, 9}, {0, 4, 8, 12,
16}};
System.out.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 `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:

Exception in thread "main"

java.lang.ArrayIndexOutOfBoundsException: 4

QUESTION 60

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**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 61

Given:

Which line causes a compilation error?

```
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;  
}
```

- 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 62

Given:

What is the result?

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

- A. null
- B. compilation fails
- C. Java.lang.NullPointerException**
- D. 0

Answer: C**QUESTION 63**

Given:

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

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

- 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 to declare the exception in doSomeThingElse and catch it in the doSomeThing method.

QUESTION 64

Given:

What is the result?

```
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);
    }
}
```

- 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 65

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 66

Given:

Why will the code not compile?

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

- 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 67

Given:

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?

```
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();  
    }  
}
```

- 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: ABC**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 recompute the area.

QUESTION 68

Given a code fragment:

What is the result?

```
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");
}
```

- 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 class does not override the equals method so it uses the equals method of Object.

StringBuffer

If a and b are two objects from a class which doesn't override equals, then a.equals(b) is the same as a == b

QUESTION 69

Which two are possible outputs?

```
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");
}
}
```

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

Answer: AB

Explanation:

A: Output is 2 if Math.random() is greater than 0.5.
B: If Math.random() returns a value less than or 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 70

Given:

```
public class MyFor {
public static void main(String[] args) {
for (int ii = 0; ii < 4; ii++) {
System.out.println("ii = "+ ii);
ii = ii +1;
}
}
}
```

What is the result?

- A. ii = 0
ii = 2
- B. ii = 0
ii = 1
ii = 2
ii = 3
- C. ii =
- D. Compilation fails.

Answer: A

QUESTION 71

Given the code fragment:

```
int [][] array2d = new int[2][3];
System.out.println("Loading the data.");
for ( int x = 0; x < array2d.length; x++) {
    for ( int y = 0; y < array2d[0].length; y++) {
        System.out.println(" x = " + x);
        System.out.println(" y = " + y);
        // insert load statement here.
    }
}
System.out.println("Modify the data. ");
for ( int x = 0; x < array2d.length; x++) {
    for ( int y = 0; y < array2d[0].length; y++) {
        System.out.println(" x = " + x);
        System.out.println(" y = " + y);
        // insert modify statement here.
    }
}
```

Which pair of load and modify statement should be inserted in the code? The load statement should set the array's x row and y column value to the sum of x and y
The modify statement should modify the array's x row and y column value by multiplying it by 2

- A. Load statement: $\text{array2d}(x,y) = x + y;$
Modify statement: $\text{array2d}(x,y) = \text{array2d}(x,y) * 2$
- B. Load statement: $\text{array2d}[x][y] = x + y;$
Modify statement: $\text{array2d}[x][y] = \text{array2d}[x][y] * 2$
- C. Load statement: $\text{array2d}[x,y] = x + y;$
Modify statement: $\text{array2d}[x,y] = \text{array2d}[x,y] * 2$
- D. Load statement: $\text{array2d}[x][y] = x + y;$
Modify statement: $\text{array2d}[x][y] = \text{array2d}[x][y] * 2$
- E. Load statement: $\text{array2d}[[x][y]] = x + y;$
Modify statement: $\text{array2d}[[x][y]] = \text{array2d}[[x][y]] * 2$

Answer: D

QUESTION 72

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 73

Which three lines are illegal?

```
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. }
```

- 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: CDH

QUESTION 74

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 75

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 76

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: AD

Explanation:

If there is sinle choice question in the real exam, only D is right.

QUESTION 77

Given:

```
public class Main {
public static void main(String[] args) {
try {
doSomething();
}
catch (SpecialException e) {
System.out.println(e);
}
static void doSomething() {
int [] ages = new int[4];
ages[4] = 17;
doSomethingElse();
}
static void doSomethingElse() {
throw new SpecialException("Thrown at end of doSomething() method");
}
```

What is the output?

- A. SpecialException: Thrown at end of doSomething() method
- B. Error in thread "main" java.lang.
ArrayIndexOutOfBoundsException
- C. Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 4 at
Main.doSomething(Main.java:12)
at Main.main(Main.java:4)
- D. SpecialException: Thrown at end of doSomething() method at

```
Main.doSomethingElse(Main.java:16)
at Main.doSomething(Main.java:13)
at Main.main(Main.java:4)
```

Answer: C

Explanation:

The following line causes a runtime exception (as the index is out of bounds):

```
ages[4] = 17;
```

A runtime exception is thrown as an `ArrayIndexOutOfBoundsException`.

Note: The third kind of exception (compared to checked exceptions and errors) is the runtime exception. These are exceptional conditions that are internal to the application, and that the application usually cannot anticipate or recover from. These usually indicate programming bugs, such as logic errors or improper use of an API.

Runtime exceptions are not subject to the Catch or Specify Requirement. Runtime exceptions are those indicated by `RuntimeException` and its subclasses.

QUESTION 78

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;
    }
}
```

Which line of code initializes a student instance?

- A. `Student student1;`
- B. `Student student1 = Student.new();`
- C. `Student student1 = new Student();`
- D. `Student student1 = Student();`

Answer: C

QUESTION 79

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

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

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

Answer: D

QUESTION 80

What is the result?

```
int i, j=0;
i = (3* 2 +4 +5 ) ;
j = (3 * ((2+4) + 5));
System.out.println("i:"+ i + "\nj":+j);
```

- 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 A
- D. Option D

Answer: B

QUESTION 81

What is the result?

```
boolean log3 = ( 5.0 != 6.0) && ( 4 != 5);
boolean log4 = (4 != 4) || (4 == 4);
System.out.println("log3:"+ log3 + "\nlog4" + log4);
```

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

Answer: B

QUESTION 82

Which statement will empty the contents of a StringBuilder variable named sb?

- A. sb.deleteAll();
- B. sb.delete(0, sb.size());
- C. **sb.delete(0, sb.length());**
- D. sb.removeAll();

Answer: C

QUESTION 83

What is the result?

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

- A. **10 10**
- B. 8 9
- C. 9 8
- D. 7 10

Answer: A

QUESTION 84

Which two are valid array declaration?

- A. Object array[];
- B. Boolean array[3];
- C. **int[] array;**
- D. Float[2] array;

Answer: AC

QUESTION 85

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 86

Given:

```
public class MainMethod {
void main() {
System.out.println("one");
}
static void main(String args) {
System.out.println("two");
}
public static void main(String[] args) {
System.out.println("three");
}
void mina(Object[] args) {
System.out.println("four");
}
}
```

What is printed out when the program is excuted?

- A. one
- B. two
- C. three**
- D. four

Answer: C

QUESTION 87

Given:

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

Which two items are fields?

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

Answer: AB

QUESTION 88

A method is declared to take three arguments.
A program calls this method and passes only two arguments.
What is the results?

- 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 falsy value for its declared type.
- F. An exception occurs when the method attempts to access the third argument.

Answer: A

QUESTION 89

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

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

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

Answer: ABC

QUESTION 90

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

QUESTION 91

Given the code fragment:

```
int b = 3;  
if ( !(b > 3)) {  
    System.out.println("square");  
}  
System.out.println("circle");  
System.out.println("...");
```

What is the result?

- A. square...
- B. circle...
- C. squarecircle...**

- D. Compilation fails.

Answer: C

QUESTION 92

What is the proper way to define a method that takes two int values and returns their sum as an int value?

- A. int sum(int first, int second) { first + second; }
- B. int sum(int first, second) { return first + second; }
- C. sum(int first, int second) { return first + second; }
- D. int sum(int first, int second) { return first + second; }**
- E. void sum (int first, int second) { return first + second; }

Answer: D

Explanation:

Incorrect answers:

A: no return statement

QUESTION 93

Which two are Java Exception classes?

- A. SecurityException**
- B. DuplicatePathException
- C. IllegalArgumentException**
- D. TooManyArgumentsException

Answer: AC

QUESTION 94

Given the for loop construct:

```
for ( expr1 ; expr2 ; expr3 ) {  
    statement;  
}
```

Which two statements are true?

- A. This is not the only valid for loop construct; there exists another form of for loop constructor.
- B. The expression expr1 is optional.**
it initializes the loop and is evaluated once, as the loop begins.
- C. When expr2 evaluates to false, the loop terminates.**
It is evaluated only after each iteration through the loop.
- D. The expression expr3 must be present.
It is evaluated after each iteration through the loop.

Answer: BC

Explanation:

The for statement has this forms:

```
for ( init-stmt; condition; next-stmt ) {
```

```
body  
{}
```

There are three clauses in the for statement.

The init-stmt statement is done before the loop is started, usually to initialize an iteration variable.

The condition expression is tested before each time the loop is done.

The loop isn't executed if the boolean expression is false (the same as the while loop).

The next-stmt statement is done after the body is executed. It typically increments an iteration variable.

Incorrect answers:

A: A try for a counter-example:

```
for (int i : a) {  
}
```

This is a correct statement. However, Expr2 and Expr3 are here empty expressions, so it is still in the format defined above.

QUESTION 95

What is the result?

```
public class StringReplace {  
    public static void main(String[] args) {  
        String message = "Hi everyone!";  
        System.out.println("message = " + message.replace("e", "X"));  
    }  
}
```

- A. message = Hi everyone!
- B. message = Hi XvXryonX!**
- C. A compile time error is produced.
- D. A runtime error is produced.
- E. message =
- F. message = Hi Xeveryone!

Answer: B

QUESTION 96

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: AD

QUESTION 97

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: ABD

QUESTION 98

Given the code fragment:

```
1. ArrayList<Integer> list = new ArrayList<>(1);  
2. list.add(1001);  
3. list.add(1002);  
4. System.out.println(list.get(list.size()));
```

What is the result?

- A. Compilation fails due to an error on line 1.
- B. An exception is thrown at run time due to error on line 3
- C. An exception is thrown at run time due to error on line 4
- D. 1002

Answer: C

Explanation:

The code compiles fine.

At runtime an `IndexOutOfBoundsException` is thrown when the second list item is added.

QUESTION 99

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

Explanation:

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Type mismatch: cannot convert from double to float

QUESTION 100

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 101

Which code fragment cause a compilation error?

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

Answer: D

Explanation:

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Type mismatch: cannot convert from double to float

QUESTION 102

Given:

```
class X {  
    static void m (int[] i) {  
        i[0] += 7;  
    }  
    public static void main (String[] args) {  
        int[] j = new int[1];  
        j[0] = 12;  
        m(j);  
        System.out.println(j[0]);  
    }  
}
```

What is the result?

- A. 7
- B. 12
- C. 19**
- D. Compilation fails.
- E. An exception is thrown at runtime.

Answer: C

QUESTION 103

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, enables the code to compile?

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

Answer: B

QUESTION 104

Which statement will empty the contents of a StringBuilder variable named sb?

- A. sb.deleteAll();
- B. sb.delete(0, sb.size());
- C. sb.delete(0, sb.length());**
- D. sb.removeAll();

Answer: C

QUESTION 105

Given:

```
class MarksOutOfBoundsException extends IndexOutOfBoundsException {}  
public class GradingProcess {  
    void verify(int marks) throws IndexOutOfBoundsException { if (marks >  
        100) {  
        throw new MarksOutOfBoundsException();  
    }  
    if (marks > 50) {
```

```
System.out.print("Pass");
} else {
System.out.print("Fail");
}
}

public static void main(String[] args) {
int marks = Integer.parseInt(args[2]);
try {
new GradingProcess().verify(marks);
} catch (Exception e) {
System.out.print(e.getClass());
}
}
}
```

And the command line invocation:

```
java GradingProcess 89 50 104
```

What is the result?

- A. Pass
- B. Fail
- C. class MarksOutOfBoundsException
- D. class IndexOutOfBoundsException
- E. class Exception

Answer: C

QUESTION 106

Given:

1. interface Pet { }
2. class Dog implements Pet { }
3. class Beagle extends Dog { }

Which three are valid?

- A. Pet a = new Dog();
- B. Pet b = new Pet();
- C. Dog f = new Pet();
- D. Dog d = new Beagle();
- E. Pet e = new Beagle();
- F. Beagle c = new Dog();

Answer: ADE

QUESTION 107

Given the code fragment:

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

```
sb.append("World");
```

Which 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

QUESTION 108

Given:

```
package pkg1;  
class Bb { }  
public class Ee {  
private Ee() { }  
}  
package pkg2;  
final class Ww;  
package pkg3;  
public abstract class Dd { void m() { } }
```

And,

1. package pkg4;
2. import pkg1.*;
3. import pkg2.*;
4. import pkg3.*;
5. // insert a class definition here

Which two class definitions, when inserted independently at line 5, enable the code to compile?

- A. class Cc extends Bb {}
- B. class Cc extends Ww {}
- C. class Cc extends Ee {}
- D. class Cc extends Dd {}

Answer: AD

QUESTION 109

Given:

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. }

Which will make this code compile and run?

- A. Change line 5 to:
price = 4f;
- B. Change line 5 to:
price.price = 4;**
- C. Change line 5 to:
price = (float) 4;
- D. Change line 5 to:
price = (Simple) 4;
- E. The code compiles and runs properly; no changes are necessary.

Answer: B

QUESTION 110

Given the code fragment:
Which statement is true?

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

And,

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

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

Answer: C

QUESTION 111

Given:

```
public class Test {
```

```
public static void main (String[] args) {  
    char[] arr = {97, '\t', 'e', '\n', 'i', '\t', 'o'};  
  
    for (char var: arr) {  
        System.out.print(var);  
    }  
    System.out.print("\nThe length is :" + arr.length);  
}  
}
```

What is the result?

- A. a e
The length is : 2
- B. a e
i o
The length is : 4
- C. aeio
The length is : 4
- D. a e
i o
The length is : 7
- E. Compilation fails.

Answer: D

QUESTION 112

Given the class definitions:

```
class Shape {}  
class Square extends Shape {}
```

Given the variable declarations:

```
Shape shape1 = null;  
Square square1 = null;
```

Which four compile?

- A. shape1 = (Square) new Square();
- B. shape1 = new Square();
- C. square1 = (Square) new Shape();
- D. square1 = new Shape();
- E. square1 = new Square();
shape1 = square1;
- F. shape1 = new Shape();
square1 = shape1;

Answer: ABCE

QUESTION 113

Given the code fragments:

```
9. class Student {  
10. int rollnumber;  
11. String name;  
12. List courses = new ArrayList();  
13. // insert code fragment here  
14. public String toString() {  
15. return rollnumber + " : " + name + " : " + courses;  
16. }  
17. }
```

And,

```
public class Test {  
public static void main (String[] args) {  
List cs = new ArrayList();  
cs.add("Java");  
cs.add("C");  
Student s = new Student(123,"Fred",cs);  
System.out.println(s);  
}  
}
```

Which code fragment, when inserted at line 13, enables class Test to print 123 : Fred : [Java, C] ?

- A. private Student(int i, String name, List cs) {
/* initialization code goes here */
}
- B. public void Student(int i, String name, List cs) {
/* initialization code goes here */
}
- C. **Student(int i, String name, List cs) {
/* initialization code goes here */
}**
- D. Student(int i, String name, ArrayList cs) {
/* initialization code goes here */
}

Answer: C

QUESTION 114

Given:

What is the result?

```
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);
        }
    }
}
```

- A. 2 4 6 8
- B. 2 4 6 8 9
- C. 1 3 5 7
- D. 1 3 5 7 9
- E. Compilation fails
- F. An exception is thrown at runtime

Answer: A

QUESTION 115



Given:

What is the result?

```
public class Test2 {
    public static void doChange(int[] arr) {
        for(int pos = 0; pos < arr.length; pos++) {
            arr[pos] = arr[pos] + 1;
        }
    }
    public static void main(String[] args) {
        int[] arr = {10, 20, 30};
        doChange(arr);
        for(int x: arr) {
            System.out.print(x + ", ");
        }
        doChange(arr[0], arr[1], arr[2]);
        System.out.print(arr[0] + ", " + arr[1] + ", " + arr[2]);
    }
}
```

- A. 11, 21, 31, 11, 21, 31
- B. 11, 21, 31, 12, 22, 32
- C. 12, 22, 32, 12, 22, 32
- D. 10, 20, 30, 10, 20, 30

Answer: B

QUESTION 116

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: AD

QUESTION 117

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();  
    }  
}
```

- 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

QUESTION 118

Given:

```
public class CheckIt {  
    public static void main (String[] args) {  
        if (doCheck()) {  
            System.out.print("square ");  
        }  
        System.out.print("...");  
    }  
    public static int doCheck() {  
        return 0;  
    }  
}
```

- A. square ...
- B. ...
- C. Compilation fails.**
- D. An exception is thrown at runtime.

Answer: C

QUESTION 119

Given:

```
public class Test {  
}
```

From which class does the Java compiler implicitly derive Test?

- A. Object
- B. Class
- C. an anonymous class
- D. Objects

Answer: A

QUESTION 120

Given:

```
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: CD

QUESTION 121

Given:

```
class App {  
    private boolean flag;  
    public void displayResult() {  
        int result = flag ? 5 : 10;  
        System.out.print("Result = " + result++);  
    }  
    public static void main (String[] args) {  
        new App().displayResult();  
    }  
}
```

What is the result?

- A. Compilation fails.
- B. Result = 5
- C. Result = 6
- D. Result = 10**
- E. Result = 11

Answer: D

QUESTION 122

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 123

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. Xm1
Ym

Answer: A

QUESTION 124

Given:

```
public abstract class Wow {  
    private int wow;  
    public Wow (int wow) {  
        this.wow = wow;  
    }  
    public void wow() { }  
    private void wowza() { }  
}
```

Which 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 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

Explanation:

An abstract class can contain abstract and non-abstract methods. However, an abstract method must be contained within an abstract class.

An abstract class can have private methods as long as they are not declared abstract.

An abstract class can have instance variables.

An abstract class does not have to have at least one abstract method.

An abstract class does not have to have a constructor without arguments.

QUESTION 125

Given:

```
class Prime {  
    int num;  
  
    Prime(int num) {  
        this.num = num;  
    }  
}  
public class Test {  
    public static void main (String[] args) {  
        Prime obj1 = new Prime(13);  
        Prime obj2 = new Prime(13);  
  
        if (obj1 == obj2) {  
            System.out.println("Equal");  
        } else {  
            System.out.println("Not equal");  
        }  
  
        if (obj1.equals(obj2)) {
```

```
System.out.println("Equal");
} else {
System.out.println("Not equal");
}
}
```

What is the result?

- A. Equal
 Not equal
- B. Not equal
 Equal
- C. Equal
 Equal
- D. Not equal
 Not equal

Answer: D

QUESTION 126

Given the code fragment:

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

And,

```
if ("tom".equals(cartoons[0])) {
counter++;
} else if ("tom".equals(cartoons[1])) {
counter++;
} else if ("tom".equals(cartoons[2])) {
counter++;
} else if ("tom".equals(cartoons[3])) {
counter++;
}
System.out.print(counter);
```

What is the result?

- A. 1
- B. 2
- C. 4
- D. 0

Answer: A

QUESTION 127

Given:

```
class X {
static void m(StringBuilder sb1) {
sb1.append("er");
```

```
}

public static void main (String[] args) {
    StringBuiler sb2 = new StringBuiler("moth");
    m(sb2);
    System.out.println(sb2);
}
}
```

What is the result?

- A. moth
- B. er
- C. mother**
- D. Compilation fails
- E. An exception is thrown at run time

Answer: C

QUESTION 128

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 code here
 }
}

Answer: D

QUESTION 129

Given:

```
public class DoWhile1 {
public static void main (String[] args) {
int i = 2;
do {
System.out.println(i);
} while (--i);
}
}
```

What is the result?

- A. 2
- B. 2
- C. An exception is thrown at runtime
- D. The loop executes infinite times
- E. Compilation fails

Answer: E

Explanation:

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Type mismatch: cannot convert from int to boolean

QUESTION 130

Given the code fragment:

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

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

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

Answer: AD

QUESTION 131

Given:

```
class X {  
    private X() {}  
    void one() {}  
}  
  
public class Y extends X {  
    private Y() {}  
    public static void main (String[] args) {  
        new Y().one();  
    }  
}
```

}

}

Which change will make this code compile?

- A. Add the public modifier to the declaration of class X
- B. Remove the private modifier from the X() constructor**
- C. Add the protected modifier to the declaration of the one() method
- D. Remove the Y() constructor
- E. Remove the private modifier from Y() constructor

Answer: B

QUESTION 132

Given:

```
public class App {  
    public static void main(String[] args) {  
        char[] arr = {'A', 'e', 'I', 'O', 'u'};  
        int count = 0;  
  
        for (int i = 0; i < arr.length; i++) {  
  
            switch (arr[i]) {  
                case 'A':  
                    continue;  
                case 'a':  
                    count++;  
                    break;  
                case 'E':  
                    count++;  
                    break;  
                case 'I':  
                    count++;  
                    continue;  
                case 'O':  
                    count++;  
                    break;  
                case 'U':  
                    count++;  
            }  
        }  
        System.out.print("Total match found: " + count);  
    }  
}
```

What is the result?

- A. Total match found: 1
- B. Total match found: 2**
- C. Total match found: 3
- D. Total match found: 5

Answer: B

QUESTION 133

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 obj3 = null;  
        obj2.num = 60;  
        graceMarks(obj2);  
    }  
}
```

How many objects are created in the memory at runtime?

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

Answer: A

QUESTION 134

Given:

What is the result?

```
public class Series {  
    private boolean flag;  
  
    public void displaySeries() {  
        int num = 2;  
        while (flag) {  
            if (num % 7 == 0)  
                flag = false;  
            System.out.print(num);  
            num += 2;  
        }  
    }  
    public static void main(String[] args) {  
        new Series().displaySeries();  
    }  
}
```

- A. 2 4 6 8 10 12
- B. 2 4 6 8 10 12 14

- C. Compilation fails
- D. The program prints multiple of 2 infinite times
- E. The program prints nothing**

Answer: E

QUESTION 135

Given:

```
class X {  
    String str = "default";  
    int ivalue = 17;  
    X(String s) {  
        str = s;  
    }  
    X(int i) {  
        ivalue = i;  
    }  
    void print() {  
        System.out.println(str + " " + ivalue);  
    }  
    public static void main(String[] args) {  
        new X("hello").print();  
        new X(92).print();  
    }  
}
```

What is the result?

- A. default 17
 hello 92
- B. hello 92
 default 17
- C. hello 17
 default 92**
- D. default 92
 hello 17

Answer: C

QUESTION 136

Which four statements are true regarding exception handling in Java?

- A. In multicatch blocks, the subclass catch handler must be caught after the superclass catch handler.
- B. A try block must be followed by either a catch or finally block**
- C. The Exception class is the superclass of all errors and exception in the Java language
- D. A single catch block can handle more than one type of exception**
- E. A checked exception must be caught explicitly**
- F. In a catch block that can handle more than one exception, the subclass exception handler must be caught before the superclass exception handler**

Answer: BDEF

QUESTION 137

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: BD

QUESTION 138

Given the code fragment:

```
class MySearch {
public static void main(String[] args) {
String url = "http://www.domain.com/index.html";
if (XXXX) {
```

```
System.out.println("Found");
}
}
}
```

Which code fragment, replace with XXXX, enables the code to print Found?

- A. url.indexOf("com") != -1
- B. url.indexOf("com")
- C. url.indexOf("com") == 19
- D. url.indexOf("com") != false

Answer: A

QUESTION 139

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: BCF

QUESTION 140

Given:

```
class Lang {
    private String category = "procedural";

    public static void main (String[] args) {
        Lang obj1 = new Lang();
        Lang obj2 = new Lang();

        if (obj1.category == obj2.category) {
            System.out.println("Equal");
        } else {
            System.out.println("Not equal");
        }
        if (obj1.category.equals(obj2.category)) {
            System.out.println("Equal");
        } else {
            System.out.println("Not equal");
        }
    }
}
```

What is the result?

- A. Equal

- Not equal
- B. Not equal
- Equal
- C. Equal
- Equal
- D. Not equal
- Not equal

Answer: C

QUESTION 141

Given the code fragment:

```
public static void main(Stirng[] args) {  
    int x = 353;  
    int j = x++;  
  
    switch (j) {  
        case 317:  
        case 353:  
        case 367:  
            System.out.println("Is a prime number.");  
        case 353:  
        case 363:  
            System.out.println("Is a palindrome.");  
        break;  
    default:  
        System.out.println("Invalid value.");  
    }  
}
```

What is the result?

- A. Compilation fails
- B. Is a prime number.
- C. Is a palindrome.
- D. Is a prime number.
Is a palindrome.
- E. Invalid value.

Answer: A

Explanation:

Exception in thread "main" java.lang.Error: Unresolved compilation problems:

Duplicate case

Duplicate case

QUESTION 142

Given the code fragment:

```
boolean log1 = (1 < 9) && (1 > 5);  
boolean log2 = (3 == 4) || (3 == 3);  
System.out.println("log1:" + log1 + "\nlog2:" + log2);
```

What is the result?

- A. log1: false
log2: true
- B. log1: true
log2: true
- C. log1: false
log2: false
- D. log1: true
log2: false

Answer: A

QUESTION 143

Given:

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

What is the result?

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

Answer: E

Explanation:

Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: 6 at java.lang.String.substring(Unknown Source)

QUESTION 144

Given:

```
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. the package with no name

Answer: AE

QUESTION 145

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

QUESTION 146

Given:

```
class X {  
    static void m(String s1) {  
        s1 = "acting";  
    }  
    public static void main(String[] args) {
```

```
String s2 = "action";
m(s2);
System.out.println(s2);
}
```

What is the result?

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

Answer: B

QUESTION 147

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()); }
```

- A. : Fresco
- B. null : Fresco**
- C. Fresco : Fresco
- D. A NullPointerException is thrown at runtime

Answer: B

QUESTION 148

Which two statement 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: AE

QUESTION 149

Given the code fragment:
What is the result?

```
System.out.println( 28 + 5 <= 4 + 29 );
System.out.println( ( 28 + 5 ) <= ( 4 + 29 ) );
```

- A. 28false29
true
- B. 285 < 429
true
- C. true
true
- D. Compilation fails

Answer: C

QUESTION 150

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 0;
- D. return;

Answer: AD

QUESTION 151

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
x:3 y:4
- B. x:0 y:0
x:3 y:4**
- C. x:3 y:4
x:0 y:0
- D. x:3 y:4
x:1 y:2

Answer: B

QUESTION 152

Given:

What is the result?

```
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 + ".");  
    }  
}
```

- 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

Explanation:

Exception in thread "main" java.lang.Error: Unresolved compilation problems:

The local variable sum may not have been initialized

The local variable sum may not have been initialized

QUESTION 153

Given the code fragment:

```
5. // insert code here
6.
7. arr[0] = new int[3];
8. arr[0][0] = 1;
9. arr[0][1] = 2;
10. arr[0][2] = 3;
11.
12. arr[1] = new int[4];
13. arr[1][0] = 10;
14. arr[1][1] = 20;
15. arr[1][2] = 30;
16. arr[1][3] = 40;
```

Which two statements, when inserted independently at line 5, enable the code to compile?

- A. int [] arr = null;
- B. int [] arr = new int[2];
- C. int [][] arr = new int[2][];
- D. int [] arr = new int[]{4};
- E. int [][] arr = new int[2][0];
- F. int [] arr = new int[0][4];

Answer: CE

QUESTION 154

Given the code fragment:

```
12. for (int row = 4; row > 0; row--) {
13.     int col = row;
14.     while (col <= 4) {
15.         System.out.print(col);
16.         col++;
17.     }
18.     System.out.println();
19. }
```

What is the result?

- A. 4
34
234
1234
- B. 4
43
432
4321
- C. 4321
432
43
4

- D. 4567
- 345
- 23
- 1

Answer: A

QUESTION 155

Given:

```
public class Test3 {  
    public static void main(String[] args) {  
        double[] array = {10, 20.23, 'c', 300.00f};  
        for (double d : array) {  
            d = d + 10;  
            System.out.print(d + " ");  
        }  
    }  
}
```

What is the result?

- A. 20.0 30.23 109.0 310.0
- B. 20.0 30.23 c10 310.0
- C. Compilation fails.
- D. An exception is thrown at runtime.

Answer: A

QUESTION 156

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);
    }
}
```

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

Answer: D

QUESTION 157

Given this code in a file Traveler.java:
And the commands:

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

What is the result?

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

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

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

Answer: A

QUESTION 158

Given:

What is the result?

```
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. XXX
- B. XYX
- C. YYX
- D. YYY

Answer: D

QUESTION 159

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

QUESTION 160

Given:

What is the result?

```
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 161

Given:

What is the result?

```

class Star {
    public void doStuff() {
        System.out.println("Twinkling Star");
    }
}

interface Universe {
    public void doStuff();
}

class Sun extends Star implements Universe {
    public void doStuff() {
        System.out.println("Shining Sun");
    }
}

public class Bob {
    public static void main(String[] args) {
        Sun obj2 = new Sun();
        Star obj3 = obj2;
        ((Sun) obj3).doStuff();
        ((Star) obj2).doStuff();
        ((Universe) obj2).doStuff();
    }
}

```

- A. Shining Sun
 Shining Sun
 Shining Sun
- B. Shining Sun
 Twinkling Star
 Shining Sun
- C. Compilation fails.
- D. A ClassCastException is thrown at runtime.

Answer: A

QUESTION 162

Given:

```

public class VarScope {
public static void main (String[] args) {
String color = "red";
int qty = 10;
if (color.equals("red")) { // line n1
int amount = qty * 10;
} else if (color.equals("green")) {
int amount = qty * 15; // line n2
} else if (color.equals("blue")) {
int amount = qty * 5; // line n3
}
System.out.print(amount); // line n4
}
}

```

Which line causes a compilation error?

- A. line n1
- B. line n2
- C. line n3
- D. line n4

Answer: D

QUESTION 163

Given:

Which line causes a compilation error?

```
public class TestA extends Root {
    public static void main(String[] args) {
        Root r = new TestA();
        System.out.println(r.method1());      // line n1
        System.out.println(r.method2());      // line n2
    }
}
class Root {
    private static final int MAX = 20000;
    private int method1() {
        int a = 100 + MAX;                  // line n3
        return a;
    }
    protected int method2() {              // line n4
        int a = 200 + MAX;
        return a;
    }
}
```

- A. line n1
- B. line n2
- C. line n3
- D. line n4

Answer: A

QUESTION 164

Given:

```
public class Test {
public static void main(String[] args) {
int i = 1;
do {
if (i % 2 == 0)
continue;
if (i == 5)
break;
System.out.print(i + "\t");
i++;
}
```

```
} while (true);  
}  
}
```

What is the result?

- A. 1 3
- B. 2 4
- C. The program prints nothing.
- D. Prints 1 infinite times.
- E. Prints 1 once and the loop executes infinite times.

Answer: E

QUESTION 165

Which three statements are true regarding exception handling in Java?

- A. A try block can be followed by multiple finally blocks.
- B. A try block can be followed by a catch or finally block.
- C. In multiple catch blocks, the superclass catch handler must be caught after the subclass catch handler.
- D. An unchecked exception must be caught explicitly.
- E. A finally block can be written before the catch block.
- F. Any Exception subclass can be used as the parent class of a user-defined exception.

Answer: BCF

QUESTION 166

Given:

```
public class Test {  
    public static void main (String[] args) {  
        int i = 25;  
        int j = i++ + 1;  
        if (j % 5 == 0) {  
            System.out.println(j + " is divisible by 5");  
        } else {  
            System.out.println(j + " is not divisible by 5");  
        }  
        System.out.println("Done");  
    }  
}
```

What is the result?

- A. Compilation fails.
- B. 25 is divisible by 5
Done
- C. 26 is not divisible by 5
Done
- D. 27 is not divisible by 5
Done

Answer: C

QUESTION 167

Given the code fragment:

```
1. class Test {  
2.     public static void main(String[] args) {  
3.         Test t = new Test();  
4.         int[] arr = new int[10];  
5.         arr = t.subArray(arr, 0, 2);  
6.     }  
7.     // insert code fragment here  
8. }
```

Which method definition can be inserted at line 7 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 168

An unchecked exception occurs in a method doSomething().

Must other code be added in the doSomething() method for it to compile and run successfully?

- 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:

According to Oracle, the Java programming language does not require methods to catch or to specify unchecked exceptions (RuntimeException, Error, and their subclasses).

QUESTION 169

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, enable the code to compile and run?

- A. line 6 replaced with handy.dandy.Keystroke stroke = new Keystroke();
- B. line 6 replaced with handy.*.Keystroke stroke = new Keystroke();
- C. line 6 replaced with handy.dandy.Keystroke stroke = new handy.dandy.Keystroke();
- D. import handy.*; added before line 1
- E. import handy.dandy.*; added after line 1
- F. import handy.dandy.Keystroke; added after line 1
- G. import handy.dandy.Keystroke.typeExclamation(); added before line 1

Answer: CEF

QUESTION 170

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

Explanation:

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

QUESTION 171

Given:

What is the result?

Given:

```
class Caller {  
    private void init() {  
        System.out.println("Initialized");  
    }  
  
    public void start() {  
        init();  
        System.out.println("Started");  
    }  
}  
  
public class TestCall {  
    public static void main(String[] args) {  
        Caller c = new Caller();  
        c.start();  
        c.init();  
    }  
}
```

- A. Initialized
 Started
- B. Initialized
 Started
 Initialized
- C. Compilation fails
- D. An exception is thrown at runtime

Answer: C

QUESTION 172

Given:

```
class Alpha {
    int ns;
    static int s;
    Alpha(int ns) {
        if (s < ns) {
            s = ns;
            this.ns = ns;
        }
    }
    void doPrint() {
        System.out.println("ns = " + ns + " s = " + s);
    }
}
```

And,

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

What is the result?

- A. ns = 50 S = 125
ns = 125 S = 125
ns = 100 S = 125
- B. ns = 50 S = 125
ns = 125 S = 125
ns = 0 S = 125
- C. ns = 50 S = 50
ns = 125 S = 125
ns = 100 S = 100
- D. ns = 50 S = 50
ns = 125 S = 125
ns = 0 S = 125

Answer: B

QUESTION 173

Given:

```
Class A { }
Class B { }
Interface X { }
Interface Y { }
```

Which two definitions of class C are valid?

- A. Class C extends A implements X {}

- B. Class C implements Y extends B {}
- C. Class C extends A, B {}
- D. Class C implements X, Y extends B {}
- E. Class C extends B implements X, Y {}**

Answer: AE

Explanation:

extends is for extending a class.

implements is for implementing an interface.

Java allows for a class to implement many interfaces.

QUESTION 174

View the exhibit.

```
class MissingInfoException extends Exception {}  
class AgeOutOfRangeException extends Exception {}  
  
class Candidate {  
    String name;  
    int age;  
    Candidate(String name, int age) throws Exception {  
        if (name == null) {  
            throw new MissingInfoException();  
        } else if (age <= 10 || age >= 150) {  
            throw new AgeOutOfRangeException();  
        } else {  
            this.name = name;  
            this.age = age;  
        }  
    }  
    public String toString() {  
        return name + " age: " + age;  
    }  
}
```

Given the code fragment:

```
4. public class Test {  
5.     public static void main(String[] args) {  
6.         Candidate c = new Candidate("James", 20);  
7.         Candidate c1 = new Candidate("Williams", 32);  
8.         System.out.println(c);  
9.         System.out.println(c1);  
10.    }  
11. }
```

Which change enables the code to print the following?

James age: 20
Williams age: 32

- A. Replacing line 5 with public static void main (String [] args) throws MissingInfoException, AgeOutOfRangeExcepton {
- B. Replacing line 5 with public static void main (String [] args) throws Exception {**
- C. Enclosing line 6 and line 7 within a try block and adding:
catch(Exception e1) { //code goes here}
catch (missingInfoException e2) { //code goes here}
catch (AgeOutOfRangeException e3) { //code goes here}
- D. Enclosing line 6 and line 7 within a try block and adding:
catch (missingInfoException e2) { //code goes here}
catch (AgeOutOfRangeException e3) { //code goes here}

Answer: B

QUESTION 175

Given:

What is the result?

```
public class Test {

    static void dispResult(int[] num) {
        try {
            System.out.println(num[1] / (num[1] - num[2]));
        } catch(ArithmetricException 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");
        }
    }
}
```

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

Answer: B

QUESTION 176

Given the code format:

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

```
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. }
```

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

Answer: C

QUESTION 177

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: A

QUESTION 178

Given:

What is the result?

```
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);  
        }  
    }  
}
```

- A. 2 4 6 8
- B. 2 4 6 8 9
- C. 1 3 5 7
- D. 1 3 5 7 9

Answer: A

QUESTION 179

Given:

What is the result?

```
public class MyFor1 {  
    public static void main(String[] args) {  
        int[] x = {6, 7, 8};  
        for (int i : x) {  
            System.out.print(i + " ");  
            i++;  
        }  
    }  
}
```

- A. 6 7 8
- B. 7 8 9
- C. 0 1 2

- D. 6 8 10
- E. Compilation fails

Answer: A

QUESTION 180

Given:

What is the result?

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

package p3;
import p1.DoInterface;
public class DoClass implements DoInterface{
    int x1,x2;
    DoClass(){
        this.x1 = 0;
        this.x2 = 10;
    }
    public void m1(int p1) { x1+=p1; System.out.println(x1); } // line n2
    public void m2(int p1) { x2+=p1; System.out.println(x2); }
}

package p2;
import p1.*;
import p3.*;
class Test {
    public static void main(String[] args){
        DoInterface doi= new DoClass();           // line n3
        doi.method1(100);
        doi.method2(200);
    }
}
```

A. 100

- 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: A

QUESTION 181

Given:

What is the result?

```
public class TestTry {
    public static void main(String[] args) {
        StringBuilder message = new StringBuilder("hello java!");
        int pos = 0;
        try {
            for (pos = 0; pos < 12; pos++) {
                switch (message.charAt(pos)) {
                    case 'a':
                    case 'e':
                    case 'o':
                        String uc=Character.toString(message.charAt(pos)).toUpperCase();
                        message.replace(pos, pos+1, uc);
                }
            }
        } catch (Exception e) {
            System.out.println("Out of limits");
        }
        System.out.println(message);
    }
}
```

- A. hEllojAvA!
- B. Hello java!
- C. Out of limits
hEllojAvA!
- D. Out of limits

Answer: C

QUESTION 182

Given:

What is the result?

```
public class App {
    public static void main(String[] args) {
        int i = 10;
        int j = 20;
        int k = j += i / 5;
        System.out.print(i + " : " + j + " : " + k);
    }
}
```

- A. 10 : 22 : 20
- B. 10 : 22 : 22**
- C. 10 : 22 : 6
- D. 10 : 30 : 6

Answer: B

QUESTION 183

Given the code fragment:

What is the result?

```
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);
```

- A. 20
- B. 25
- C. 29
- D. Compilation fails
- E. AnArrayIndexOutOfBoundsException is thrown at runtime

Answer: A

QUESTION 184

Given:

What is the result?

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

- A. The program prints nothing
- B. d
- C. A StringIndexOutOfBoundsException is thrown at runtime.
- D. AnArrayIndexOutOfBoundsException is thrown at runtime.
- E. A NullPointerException is thrown at runtime.

Answer: C

QUESTION 185

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: AD

Explanation:

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

QUESTION 186

Given:

What is the result?

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

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

Answer: C

QUESTION 187

Given:

What is the result?

```
public class Test3 {
    public static void main(String[] args) {
        String names[] = new String[3];
        names[0] = "Mary Brown";
        names[1] = "Nancy Red";
        names[2] = "Jessy Orange";
        try {
            for(String n: names) {
                try {
                    String pwd = n.substring(0, 3)+n.substring(6, 10);
                    System.out.println(pwd);
                }
                catch(StringIndexOutOfBoundsException sie) {
                    System.out.println("string out of limits");
                }
            }
            catch(ArrayIndexOutOfBoundsException e) {
                System.out.println("array out of limits");
            }
        }
    }
}
```

- A. Marrown
String out of limits
JesOran
- B. Marrown
String out of limits
Array out of limits
- C. Marrown
String out of limits

D. Marrown
NanRed
JesOran

Answer: A

QUESTION 188

Given:

```
public abstract class Shape {  
    private int x;  
    private int y;  
    public abstract void draw();  
    public void setAnchor(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
}
```

Which two classes use the shape class correctly?

- A)

```
public class Circle implements Shape {  
    private int radius;  
}
```
- B)

```
public abstract class Circle extends Shape {  
    private int radius;  
}
```
- C)

```
public class Circle extends Shape {  
    private int radius;  
    public void draw();  
}
```
- D)

```
public abstract class Circle implements Shape {  
    private int radius;  
    public void draw();  
}
```
- E)

```
public class Circle extends Shape {  
    private int radius;  
    public void draw() /* code here */  
}
```
- F)

```
public abstract class Circle implements Shape {  
    private int radius;  
    public void draw() /* code here */  
}
```

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

E. Option E

F. Option F

Answer: BE**Explanation:**

When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class (E). 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 189

Given the class definitions:

```
class Alpha {
    public String doStuff(String msg) {
        return msg;
    }
}
class Beta extends Alpha {
    public String doStuff(String msg) {
        return msg.replace('a', 'e');
    }
}
class Gamma extends Beta {
    public String doStuff(String msg) {
        return msg.substring(2);
    }
}
```

And the code fragment of the main() method,

```
12. List<Alpha> strs = new ArrayList<Alpha>();
13. strs.add(new Alpha());
14. strs.add(new Beta());
15. strs.add(new Gamma());
16. for (Alpha t : strs) {
17.     System.out.println(t.doStuff("Java"));
18. }
```

What is the result?

A. Java

Java

Java

B. Java

Jeve

va

C. Java

Jeve

ve

D. Compilation fails

Answer: B

QUESTION 190

Given:

What is the result?

```
public class Msg { . .
    public static String doMsg(char x) {
        return "Good Day!";
    }
    public static String doMsg(int y) {
        return "Good Luck!";
    }
    public static void main(String[] args) {
        char x = 8;
        int z = '8';
        System.out.println(doMsg(x));
        System.out.print(doMsg(z));
    }
}
```

- A. Good Day!
Good Luck!
- B. Good Day!

Good Day!
- C. Good Luck!

Good Day!
- D. Good Luck!

Good Luck!
- E. Compilation fails

Answer: A

QUESTION 191

Which two items can legally be contained within a java class declaration?

- A. An import statement
- B. A field declaration
- C. A package declaration
- D. A method declaration

Answer: BD

Explanation:

<http://docs.oracle.com/javase/tutorial/java/javaOO/methods.html>

QUESTION 192

Given:

What is the result?

```

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

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

Answer: E

Explanation:

Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: 6 at java.lang.String.substring(Unknown Source)

QUESTION 193

Given the code fragments:
What is the result?

```

interface Contract{ }
class Super implements Contract{ }
class Sub extends Super {}

public class Ref {
    public static void main(String[] args) {
        List objs = new ArrayList();

        Contract c1 = new Super();                                // line n1
        Contract c2 = new Sub();
        Super s1 = new Sub();

        objs.add(c1);
        objs.add(c2);                                         // line n2
        objs.add(s1);

        for(Object itm: objs) {
            System.out.println(itm.getClass().getName());
        }
    }
}
    
```

- A. Super
Sub
Sub
- B. Contract
Contract
Super
- C. Compilation fails at line n1
- D. Compilation fails at line n2

Answer: A

QUESTION 194

Given:

```
public class MyClass {  
    public static void main(String[] args) {  
        String s = " Java Duke ";  
        int len = s.trim().length();  
        System.out.print(len);  
    }  
}
```

What is the result?

- A. 8
- B. 9**
- C. 11
- D. 10
- E. Compilation fails

Answer: B

Explanation:

Java-String trim() Method

This method returns a copy of the string, with leading and trailing whitespace omitted.

QUESTION 195

Given the code fragment:

What is the result?

```
public static void main(String[] args) {  
    ArrayList<String> list = new ArrayList<>();  
  
    list.add("SE");  
    list.add("EE");  
    list.add("ME");  
    list.add("SE");  
    list.add("EE");  
  
    list.remove("SE");  
  
    System.out.print("Values are : " + list);  
}
```

- A. Values are : [EE, ME]
- B. Values are : [EE, EE, ME]
- C. Values are : [EE, ME, EE]
- D. Values are : [SE, EE, ME, EE]
- E. Values are : [EE, ME, SE, EE]**

Answer: E

QUESTION 196

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: AD

Explanation:

http://www.tutorialspoint.com/java/java_access_modifiers.htm

QUESTION 197

Given:

The class is poorly encapsulated.

You need to change the circle class to compute and return the area instead.

Which two modifications are necessary to ensure that the class is being properly encapsulated?

```
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();  
    }  
}
```

- A. Remove the area field.
- B. Change the getArea() method as follows:
public double getArea () { return Match.PI * radius * radius; }
- C. Add the following method:
public double getArea () {area = Match.PI * radius * radius; }
- D. Change the access modifier of the SerRadius () method to be protected.

Answer: BD

QUESTION 198

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?

- A) Caught java.lang.RuntimeException: Exception
Exception in thread "main" java.lang.Error: Error
at TestApp.doList(TestApp.java: 14)
at TestApp.main(TestApp.java: 6)
- B) Exception in thread "main" java.lang.Error: Error
at TestApp.doList(TestApp.java: 14)
at TestApp.main(TestApp.java: 6)
- C) Caught java.lang.RuntimeException: Exception
Caught java.lang.Error: Error
- D) Caught java.lang.RuntimeException: Exception

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

Answer: D

QUESTION 199

Given:

```
1. public class TestLoop {
2.     public static void main(String[] args) {
3.         float myarray[] = {10.20f, 20.30f, 30.40f, 50.60f};
4.         int index = 0;
5.         boolean isFound = false;
6.         float key = 30.40f;
7.         // insert code here
8.         System.out.println(isFound);
9.     }
10. }
```

Which code fragment, when inserted at line 7, enables the code print true?

```
C A) while (key == myarray[index++]) {
        isFound = true;
    }

C B) while (index <= 4) {
    if (key == myarray[index]) {
        index++;
        isFound = true;
        break;
    }
}

C C) while (index++ < 5) {
    if (key == myarray[index]) {
        isFound = true;
    }
}

C D) while (index < 5) {
    if (key == myarray[index]) {
        isFound = true;
        break;
    }
    index++;
}
```

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

Answer: D

QUESTION 200

Given the code fragment:

What is the result if the integer aVar is 9?

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

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

Answer: A

QUESTION 201

Given:

And the commands:

Javac Test.java
Java Test 12345

What is the result?

```
Test.java

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

- 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: D

QUESTION 202

Given the code fragment:

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

```
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 + " ");
        }
    }
}
```

- A. List update (String[] strs)
- B. Static ArrayListupdate(String [] strs)
- C. Static List update (String [] strs)**
- D. Static void update (String[] strs)

E. `ArrayList static update(String [] strs)`

Answer: C

QUESTION 203

Given:

What is the result?

```
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);
    }
}
```

- A. x:1 y:2
x:3 y:4
- B. x:0 y:0
x:3 y:4**
- C. x:3 y:4
x:0 y:0
- D. x:3 y:4
x:1 y:2

Answer: B

QUESTION 204

Given:

```
class SpecialException extends Exception {
    public SpecialException(String message) {
        super(message);
        System.out.println(message);
    }
}

public class ExceptionTest {
    public static void main(String[] args) {
        try {
            doSomething();
        } catch (SpecialException e) {
            System.out.println(e);
        }
    }
    static void doSomething() throws SpecialException {
        int[] ages = new int[4];
        ages[4] = 17;
        doSomethingElse();
    }
    static void doSomethingElse() throws SpecialException {
        throw new SpecialException("Thrown at end of doSomething() method");
    }
}
```

What will be the output?

- A) SpecialException: Thrown at end of doSomething() method
- B) Error in thread "main" java.lang.ArrayIndexOutOfBoundsException
- C) Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4
at ExceptionTest.doSomething(ExceptionTest.java:13)
at ExceptionTest.main(ExceptionTest.java:4)
- D) SpecialException: Thrown at end of doSomething() method
at ExceptionTest.doSomethingElse(ExceptionTest.java:16)
at ExceptionTest.doSomething(ExceptionTest.java:13)
at ExceptionTest.main(ExceptionTest.java:4)

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

Answer: C

Explanation:

ages[4] goes in IndexOutOfBoundsException

QUESTION 205

Given the code fragment:

What could expression1 and expression2 be, respectively, in order to produce output -8, 16?

```
9. int a = -10;
10. int b = 17;
11. int c = expression1;
12. int d = expression2;
13. c++;
14. d--;
15. System.out.print(c + " , " + d);
```

- A. + +a, - -b
- B. + +a, b--**
- C. A+ +, -- b
- D. A + +, b --

Answer: B

QUESTION 206

Given:

What is the result?

```
public class Palindrome {
    public static int main(String[] args) {
        System.out.print(args[1]);
        return 0;
    }
}
```

And the commands:
javac Palindrome.java
java Palindrome Wow Mom

- A. Compilation fails
- B. The code compiles, but does not execute.**
- C. Paildrome
- D. Wow
- E. Mom

Answer: B

QUESTION 207

Given:

And the commands:

Javac Jump.java
Java Jump crazy elephant is always

What is the result?

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

- A. Lazy lion is jumping
- B. Lion is always jumping
- C. Crazy elephant is jumping
- D. Elephant is always jumping**
- E. Compilation fails

Answer: D

QUESTION 208

Given:

What is the result?

```
class Test {
    public static void main(String[] args) {
        int numbers[];
        numbers = new int[2];
        numbers[0] = 10;
        numbers[1] = 20;

        numbers = new int[4];
        numbers[2] = 30;
        numbers[3] = 40;
        for (int x : numbers) {
            System.out.print(" "+x);
        }
    }
}
```

- A. 10 20 30 40
- B. 0 0 30 40**
- C. Compilation fails
- D. An exception is thrown at runtime

Answer: B

Explanation:

Creating a new object (also an array) does initialize it as default.

QUESTION 209

Given:

What is the result?

```
public class Series {
    public static void main(String[] args) {
        int arr[] = {1, 2, 3};

        for (int var : arr) {
            int i = 1;
            while (i <= var);
                System.out.println(i++);
        }
    }
}
```

- A. 1
- B. 1
- C. 2
- D. Compilation fails
- E. The loop executes infinite times

Answer: E

QUESTION 210

Given:

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

```
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. }
```

- 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: D

QUESTION 211

Given:

What is the result?

```
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
```

- 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: A

QUESTION 212

Given:

What is the result?

```

public class Test {
    public static void main(String[] args) {
        Test ts = new Test();
        System.out.print(isAvailable + " ");
        isAvailable= ts.doStuff();
        System.out.println(isAvailable);
    }
    public static boolean doStuff() {
        return !isAvailable;
    }
    static boolean isAvailable = false;
}

```

- A. true true
- B. true false
- C. false true**
- D. false false
- E. Compilation fails

Answer: C

QUESTION 213

Given:

Which constructor initializes the variable x3?

```

class X {
    int x1, x2, x3;
}
class Y extends X {
    int y1;
    Y() {
        x1 = 1;
        x2 = 2;
        y1 = 10;
    }
}

class Z extends Y {
    int z1;
    Z() {
        x1 = 3;
        y1 = 20;
        z1 = 100;
    }
}

And,
public class Test3 {
    public static void main(String[] args) {
        Z obj = new Z();
        System.out.println(obj.x3 + ", " + obj.y1 + ", " + obj.z1);
    }
}

```

- A. Only the default constructor of class X

- B. Only the no-argument constructor of class Y
- C. Only the no-argument constructor of class Z
- D. Only the default constructor of object class

Answer: C

QUESTION 214

Given:

How many objects have been created when the line // do complex stuff is reached?

Given:

```
class Dog {  
    Dog() {  
        try {  
            throw new Exception();  
        } catch (Exception e) {}  
    }  
  
    class Test {  
        public static void main(String[] args) {  
            Dog d1 = new Dog();  
            Dog d2 = new Dog();  
            Dog d3 = d2;  
            // do complex stuff  
        }  
    }  
}
```

- A. Two
- B. Three
- C. Four
- D. Six

Answer: C

QUESTION 215

Given:

What is result?

```
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");
    }
}
```

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

Answer: B

Explanation:

Null is transformed into “null” while concat Strings

QUESTION 216

Given the fragment:

What is the result?

```
24. float var1 = (12_345.01 >= 123_45.00) ? 12_456 : 124_56.02f;
25. float var2 = var1 + 1024;
26. System.out.print(var2);
```

- A. 13480.0**
- B. 13480.02
- C. Compilation fails
- D. An exception is thrown at runtime

Answer: A

QUESTION 217

Given:

Which code fragment, when inserted at line 9, enables the code to print true?

```
7. StringBuilder sb1 = new StringBuilder("Duke");
8. String str1 = sb1.toString();
9. // insert code here
10. System.out.print(str1 == str2);
```

- A. String str2 = str1;**

- B. String str2 = new string (str1);
- C. String str2 = sb1.toString();
- D. String str2 = "Duke";

Answer: A

QUESTION 218

Given the code fragment:
What is the result?

```
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. }
```

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

Answer: A

QUESTION 219

Given the classes:

- AssertionError
- ArithmeticException
- ArrayIndexOutOfBoundsException
- FileNotFoundException
- IllegalArgumentException
- IOException
- IOException
- NumberFormatException
- SQLException

Which option lists only those classes that belong to the unchecked exception category?

- A. **AssertionError, ArrayIndexOutOfBoundsException, ArithmeticException**
- B. **AssertionError, IOException, IOException**
- C. **ArithmeticException, FileNotFoundException, NumberFormatException**
- D. **FileNotFoundException, IOException, SQLException**
- E. **ArrayIndexOutOfBoundsException, IllegalArgumentException, FileNotFoundException**

Answer: A

Explanation:

Not B: IOError and IOException are both checked errors.

Not C, not D, not E: FileNotFoundException is a checked error.

Note:

Checked exceptions:

- 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
- 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)

Note:

Unchecked exceptions:

- represent defects in the program (bugs) - often invalid arguments passed to a non-private method. To quote from The Java Programming Language, by Gosling, Arnold, and Holmes: "Unchecked runtime exceptions represent conditions that, generally speaking, reflect errors in your program's logic and cannot be reasonably recovered from at run time."
- are subclasses of RuntimeException, and are usually implemented using IllegalArgumentException, NullPointerException, or IllegalStateException
- method is not obliged to establish a policy for the unchecked exceptions thrown by its implementation (and they almost always do not do so)

QUESTION 220

Given:

```
public class Test1 {  
    static void doubling (Integer ref, int pv) {  
        ref =20;  
        pv = 20;  
    }  
    public static void main(String[] args) {  
        Integer iObj = new Integer(10);  
        int iVar = 10;  
        doubling(iObj++, iVar++);  
        System.out.println(iObj+ " , "+iVar);  
    }  
}
```

What is the result?

- A. 11, 11
- B. 10, 10
- C. 21, 11
- D. 20, 20
- E. 11, 12

Answer: A

Explanation:

The codedoubling(iObj++, iVar++);increases both variables from to 10 to 11.

QUESTION 221

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: AD

Explanation:

Incorrect:

Not B: circular definition of n3.

Not C: Compilation error. lineCalc c = new Mid();

required: Calc

found: Mid

Not E: Compilation error. lineint n3 = Calc.findMid(n1, n2);

non-static method findMid(int,int) cannot be referenced from a static context

QUESTION 222

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 223

Given:

```
public class ComputeSum {  
    public int x;  
    public int y;  
    public int sum;  
    public ComputeSum (int nx, int ny) {  
        x = nx; y = ny;  
        updateSum();  
    }  
    public void setX(int nx) { x = nx; updateSum();}  
    public void setY(int ny) { y = ny; updateSum();}  
    void updateSum() { sum = x + y; }  
}
```

This class needs to protect an invariant on the sum field.

Which three members must have the private access modifier to ensure that this invariant is maintained?

- A. The x field
- B. The y field**
- C. The sum field**
- D. The ComputerSum () constructor
- E. The setX () method
- F. The setY () method

Answer: ABC

QUESTION 224

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 225

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 = new Square ();
square.foo ();
square.foo ();
- F. Square square = new Square();
square.foo("bar");
square.foo();

Answer: F

QUESTION 226

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: A

QUESTION 227

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 228

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 229

Given:

```
class Base {  
    // insert code here  
}  
public class Derived extends Base{  
    public static void main(String[] args) {  
        Derived obj = new Derived();  
        obj.setNum(3);  
        System.out.println("Square = " + obj.getNum() * obj.getNum());  
    }  
}
```

Which two options, when inserted independently inside class Base, ensure that the class is being properly encapsulated and allow the program to execute and print the square of the number?

- A.** private int num;public int getNum() {return num;}
 public void setNum(int num) {this.num = num;}
- B. public int num;protected public int getNum() {return num;}
 protected public void setNum(int num) {this.num = num;}
- C. private int num;public int getNum() {return num;}private

- void setNum(int num) {this.num = num;}
- D. protected int num;public int getNum() {return num;}public void setNum(int num) {this.num = num;}
- E. protected int num;private int getNum() {return num;}public void setNum(int num) {this.num = num;}

Answer: AD

Explanation:

Incorrect:

Not B: illegal combination of modifiers: protected and public

not C: setNum method cannot be private.

not E: getNum method cannot be private.

QUESTION 230

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 231

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 232

Given the code fragment:

```
public static void main(String[] args) {
int iArray[] = {65, 68, 69};
iArray[2] = iArray[0];
iArray[0] = iArray[1];
iArray[1] = iArray[2];
for (int element : iArray) {
System.out.print(element + " ");
}
```

- A. 68, 65, 69
- B. 68, 65, 65**
- C. 65, 68, 65
- D. 65, 68, 69
- E. Compilation fails

Answer: B

Explanation:

68 65 65

QUESTION 233

Given:

```
public class TestLoop1 {
public static void main(String[] args) {
int a = 0, z=10;
while (a < z) {
a++;
--z;
}}
```

```
}
System.out.print(a + " : " + z);
}
}
```

What is the result?

- A. 5 : 5
- B. 6 : 4
- C. 6 : 5
- D. 5 : 4

Answer: A

Explanation:

5 : 5

QUESTION 234

Given:

```
public class MyClass {
public static void main(String[] args) {
while (int ii = 0; ii < 2) {
ii++;
System.out.println("ii = " + ii);
}
}
}
```

What is the result?

- A. ii = 1
ii = 2
- B. Compilation fails**
- C. The program prints nothing
- D. The program goes into an infinite loop with no output
- E. The program goes to an infinite loop outputting:
ii = 1
ii = 1

Answer: B

Explanation:

The while statement is incorrect. It has the syntax of a for statement.

The while statement continually executes a block of statements while a particular condition is true. Its syntax can be expressed as:

```
while (expression) {
statement(s)
}
```

The while statement evaluates expression, which must return a boolean value. If the expression evaluates to true, the while statement executes the statement(s) in the while block. The while statement continues testing the expression and executing its block until the expression evaluates to false.

QUESTION 235

Given:

```
public class String1 {  
    public static void main(String[] args) {  
        String s = "123";  
        if (s.length() > 2)  
            s.concat("456");  
        for (int x = 0; x < 3; x++)  
            s += "x";  
        System.out.println(s);  
    }  
}
```

What is the result?

- A. 123
- B. 123xxx**
- C. 123456
- D. 123456xxx
- E. Compilation fails

Answer: B

Explanation:

123xxx

The if clause is not applied.

Note: Syntax of if-statement:

```
if ( Statement ) {  
}
```

QUESTION 236

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

Explanation:

1.0, 1

QUESTION 237

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: BCF

QUESTION 238

Given:

```
class MarksOutOfBoundsException extends IndexOutOfBoundsException { }
public class GradingProcess {
void verify(int marks) throws IndexOutOfBoundsException {
if (marks > 100) {
throw new MarksOutOfBoundsException();
}
if (marks > 50) {
System.out.print("Pass");
} else {
System.out.print("Fail");
}
}
public static void main(String[] args) {
int marks = Integer.parseInt(args[2]);
try {
new GradingProcess().verify(marks));
} catch(Exception e) {
System.out.print(e.getClass());
}
}
}
```

And the command line invocation:

```
Java grading process 89 50 104
```

What is the result?

- A. Pass
- B. Fail
- C. Class MarketOutOfBoundsException
- D. Class IndexOutOfBoundsException
- E. Class Exception

Answer: C

Explanation:

The value 104 will cause aMarketOutOfBoundsException

QUESTION 239

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.

QUESTION 240

Given:

```
package p1;  
public interface DoInterface {  
void method1(int n1); // line n1  
}  
package p3;  
import p1.DoInterface;  
public class DoClass implements DoInterface {  
public DoClass(int p1) { }  
public void method1(int p1) { } // line n2  
private void method2(int p1) { } // line n3  
}  
public class Test {  
public static void main(String[] args) {  
DoInterface doi= new DoClass(100); // line n4  
doi.method1(100);  
doi.method2(100);  
}  
}
```

Which change will enable the code to compile?

- A. Adding the `public` modifier to the declaration of `method1` at line n1
- B. Removing the `public` modifier from the definition of `method1` at line n2
- C. Changing the `private` modifier on the declaration of method 2 `public` at line n3
- D. Changing the line n4 `DoClass doi = new DoClass ();`

Answer: C**Explanation:**

Private members (both fields and methods) are only accessible inside the class they are declared or inside inner classes. private keyword is one of four access modifier provided by Java and its a most restrictive among all four e.g. public, default(package), protected and private.

Read more: <http://javarevisited.blogspot.com/2012/03/private-in-java-why-should-you-always.html#ixzz3Sh3mOc4D>

QUESTION 241

Given the fragment:

```
String[][] arra = new String[3][];  
arra[0] = new String[]{"rose", "lily"};  
arra[1] = new String[]{"apple", "berry", "cherry", "grapes"};  
arra[0] = new String[]{"beans", "carrot", "potato"};  
// insert code fragment here
```

Which code fragment when inserted at line '// insert code fragment here', enables the code to successfully change arra elements to uppercase?

- A.

```
String[][] arra = new String[3][];  
arra[0] = new String[]{"rose", "lily"};  
arra[1] = new String[]{"apple", "berry", "cherry", "grapes"};  
arra[0] = new String[]{"beans", "carrot", "potato"};  
for (int i = 0; i < arra.length; i++) {  
    for (int j=0; j < arra[i].length; j++) {  
        arra[i][j] = arra[i][j].toUpperCase();  
    }  
}
```
- B.

```
for (int i = 0; i < 3; i++) {  
    for (int j=0; j < 4; j++) {  
        arra[i][j] = arra[i][j].toUpperCase();  
    }  
}
```
- C.

```
for (String a[]:arra[][]) {  
    for (String x:a[]) {  
        toUpperCase();  
    }  
}
```
- D.

```
for (int i:arra.length) {  
    for (String x:arra) {  
        arra[i].toUpperCase();  
    }  
}
```

Answer: A**QUESTION 242**

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[][]) {
 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 erro line:

for (String[] sub: arr[][])

not D: Output: C

QUESTION 243

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 244

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 245

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: AC

Explanation:

The following will work fine:

```
public classTest{  
    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 246

Given:

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

```
String[] arr = new String[4];
arr[1] = "Unix";
arr[2] = "Linux";
arr[3] = "Solarios";
for (String var : arr) {
    System.out.print(var + " ");
}
} catch(Exception e) {
    System.out.print(e.getClass());
}
}
```

What is the result?

- A. Unix Linux Solaris
- B. Null Unix Linux Solaris**
- C. Class java.lang.Exception
- D. Class java.lang.NullPointerException

Answer: B

Explanation:

null Unix Linux Solarios

The first element, arr[0], has not been defined.

QUESTION 247

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 248

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 the end of the list (index 3).

[green, red, yellow, cyan]

QUESTION 249

Given:

```
public class TestOperator {
public static void main(String[] args) {
int result = 30 -12 / (2*5)+1;
System.out.print("Result = " + result);
}
}
```

What is the result?

- A. Result = 2
- B. Result = 3
- C. Result = 28
- D. Result = 29
- E. Result = 30

Answer: E

QUESTION 250

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", "Condidtion OK");
 num_umpires =3;
 num_substitutes=2;
 }
- B. Cricket() {
 super.ground_condition = "Condition OK";
 super.name="Cricket";
 super.num_players = 11;
 num_umpires =3;
 num_substitutes=2;
 }
- C. Cricket() {
 this(3,2);
 super(11, "Cricket", "Condidtion OK");
 }
 Cricket(int nu, 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 251

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 252

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 253

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 254

Given:

```
public class TestField {
int x;
int y;
public void doStuff(int x, int y) {
this.x = x;
y =this.y;
}
public void display() {
System.out.print(x + " " + y + " : ");
}
public static void main(String[] args) {
TestField m1 = new TestField();
m1.x = 100;
m1.y = 200;
TestField m2 = new TestField();
m2.doStuff(m1.x, m1.y);
m1.display();
m2.display();
}
}
```

What is the result?

- A. 100 200 : 100 200
- B. 100 0 : 100 0 :
- C. 100 200 : 100 0 :**
- D. 100 0 : 100 200 :

Answer: C

QUESTION 255

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

Explanation:

null
0.0

QUESTION 256

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

QUESTION 257

Given:

```
public class Test {  
    static boolean bVar;
```

```
public static void main(String[] args) {  
    boolean bVar1 = true;  
    int count = 8;  
    do {  
        System.out.println("Hello Java! " + count);  
        if (count >= 7) {  
            bVar1 = false;  
        }  
    } while (bVar != bVar1 && count > 4);  
    count -= 2;  
}
```

What is the result?

- A. Hello Java! 8
Hello Java! 6
Hello Java! 4
- B. Hello Java! 8
Hello Java! 6
- C. Hello Java! 8
- D. Compilation fails

Answer: C

Explanation:

Hello Java! 8

QUESTION 258

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 259

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("Overridden " + args[1]);  
    }  
}
```

And the commands:

```
javac Sub.java  
java Sub 10 20 30
```

What is the result?

- A. Base 30
- B. Overridden 20**
- C. Overridden 20
 Base 30
- D. Base 30
 Overridden 20

Answer: B

QUESTION 260

Given:

```
interface Pet { }  
class Dog implements Pet { }  
public class Beagle extends Dog{ }
```

Which three are valid?

- A.** Pet a = new Dog();
- B. Pet b = new Pet();
- C. Dog f = new Pet();
- D.** Dog d = new Beagle();
- E.** Pet e = new Beagle();
- F. Beagle c = new Dog();

Answer: ADE

Explanation:

Incorrect:

Not B, not C: Pet is abstract, cannot be instantiated.

Not F: incompatible type. Required Beagle, found Dog.

QUESTION 261

Given the code fragment:

```
// insert code here  
arr[0] = new int[3];  
arr[0][0] = 1;  
arr[0][1] = 2;  
arr[0][2] = 3;  
arr[1] = new int[4];  
arr[1][0] = 10;  
arr[1][1] = 20;  
arr[1][2] = 30;  
arr[1][3] = 40;
```

Which two statements, when inserted independently at line // insert code here, enable the code to compile?

- A. int [] [] arr = null;
- B. int [] [] arr=new int [2];
- C. int [] [] arr=new int [2] [];
- D. int [] [] arr = new int [] [4];
- E. int [] [] arr=new int [2] [0];
- F. int [] [] arr=new int [0] [4];

Answer: CE

QUESTION 262

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 (ArithmetricException 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.ArithmetricException: /
by zero at test.Teagle.main(Teagle.java:14)

D. Invalid Divisor

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

Answer: C

QUESTION 263

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 264

Given the code fragment:

```
class Student {  
int rollnumber;  
String name;  
List courses = new ArrayList();  
// insert code here  
public String toString() {  
return rollnumber + " : " + name + " : " + courses;  
}  
}  
And,  
public class Test {  
public static void main(String[] args) {  
List cs = newArrayList();  
cs.add("Java");  
cs.add("C");  
Student s = new Student(123,"Fred", cs);
```

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

Which code fragment, when inserted at line // insert code here, enables class Test to print 123 :
Fred : [Java, C]?

- A. private Student(int i, String name, List cs) {
/* initialization code goes here */
}
- B. public voidStudent(int i, String name, List cs) {
/* initialization code goes here */
}
- C. Student(int i, String name, List cs) {
/* initialization code goes here */
}
- D. Student(int i, String name,ArrayList cs) {
/* initialization code goes here */
}

Answer: C

Explanation:

Incorrect:

Not A: Student has private access line:Student s = new Student(123,"Fred", cs);

Not D: Cannot be applied to given types. Line:Student s = new Student(123,"Fred", cs);

QUESTION 265

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: ABC

QUESTION 266

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

Explanation:

Yellow
Pink
Red

QUESTION 267

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: AE

QUESTION 268

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 269

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: AD

Explanation:

Incorrect:

Not B: No main class found.

Not C: Main method not found

not E: Main method is not static.

QUESTION 270

Given the code fragment:

```
public class Test {  
    public static void main(String[] args) {  
        boolean isChecked = false;  
        int arry[] = {1,3,5,7,8,9};  
        int index = arry.length;  
        while ( <code1> ) {  
            if (arry[index-1] % 2 ==0) {  
                isChecked = true;  
            }  
            <code2>  
            System.out.print(arry(index]+", "+isChecked));  
        }  
    }  
}
```

Which set of changes enable the code to print 1, true?

- A. Replacing <code1> with index > 0 and replacing <code2> with index--;
- B. Replacing <code1> with index > 0 and replacing <code2> with --index;
- C. Replacing <code1> with index > 5 and replacing <code2> with --index ;
- D. Replacing <code1> with index and replacing <code2> with --index ;

Answer: A

Explanation:

Note: Code in B (code2 is --index;). also works fine.

QUESTION 271

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.