1. **Consider the following Java code snippet:**

```
public int divide (int a, int b){
    int c= -1;

    try{
        c = a/b;
    }
    catch(Exception e){
        System.err.print("Exception ");
    }
    finally{
        System.err.println("Finally ");
    }
    return c;
}
```

What will our code print when we call divide (4,0)?

Pick one of the choices:

- **Exception Finally**
- **Finally Exception**
- **Exception**

El método divide intenta dividir el valor de a por b. En este caso, b es 0, lo que provoca una excepción de división por cero (ArithmeticException). El bloque catch(Exception e) captura cualquier excepción y ejecuta la instrucción System.err.print("Exception "), que imprime "Exception". Luego, el bloque finally se ejecuta independientemente de si se lanzó una excepción o no, e imprime "Finally". Por lo tanto, el resultado final será "Exception Finally".

2. **The feature which allows different methods to have the same name and arguments type, but the different implementation is called?**

Pick one of the choices:

- **Overloading(SobreCarga)**
- **Overriding (SobreEscritura @Override)**
- **Java does not permit methods with same and type signature**
- **None of the above**

Si buscamos que dos métodos tengan el mismo nombre y argumentos, pero implementados diferentes estamos hablando de SobreEscritura.

3. **What does the following for loop output?**

```
for (int i=10, j=1, i>j; --i, ++j)
System.out.print(j %i);
```

Pick one of the choices:

- **12321**
- **12345**
- **11111**

- **00000**

Obviando el hecho de que no compila por que falta un ;, ya que no es una opcion en las respuestas, al ejecutar el loop, este nos da 1 2 3 4 5, ya que después de terminar el loop es cuando ejecuta la parte de la derecha de el for, ocasionando que no tengamos que tomar en cuenta realmente el que la resta se haga previo por el --i.

4. **We perform the following sequence of actions:**

    1. **Insert the following elements into a set: 1,2,9,1,2,3,1,4,1,5,7.**
    2. **Convert the set into a list and sort it in ascending order.**

Which option denotes the sorted list?

Pick one of the choices:
- **{1, 2, 3, 4, 5, 7, 9}**
- **{9, 7, 5, 4, 3, 2, 1}**
- **{1, 1, 1, 1, 2, 2, 3, 4, 5, 7, 9}**
- **None of the above**

Un set no permite tener elementos duplicados por lo que al pasarlo a una lista y ordenarla con sort, se mostrará simplemente una sola vez los números, aunque los agregaramos varias veces al set.

5. **What is the output for the below Java code?**

```
public class Test{
    public static void main (String[] args)
     {
        int i = 010;
        int j = 07;
        System.out.println(i);
        System.out.println(j);
    }
}
```

Pick one of the choices:
- **8 7**
- **10 7**
- **Compilation fails with an error at line 3**
- **Compilation fails with an error at line 5**

Cuando se crea un int con un 0 previo al numero, esta haciendo referencia a que pertenece al sistema octal, y por esto mismo, el valor 10 equivale al 8, y el 7 se conserva como tal simplemente quitando el 0.

6. **A public data member with the same name is provided in both base as well as derived classes. Which of the following is true?**

Pick one of the choices:

- **It is a compiler error to provide a field with the same name in both base and derived class**
- **The program will compile and this feature is called overloading**
- **The program will compile and this feature is called overriding**
- **The program will compile and this feature is called as hiding or shadowing**

Esto se puede realizar, y como dice la respuesta a esta feature se le llama hiding o shadowing, sin embargo es importante mencionar que si se declara una Superclase como variable de referencia y se le pasa una subClase como objeto, al momento de intentar acceder a este dato, nos regresara siempre el del padre, o dicho de otra forma, el de la variable de referencia, a diferencia de si declaramos un método lo mandamos a llamar con el objeto de una subClase.

7. **Given three clases A, B and C.**

B is a subclass of A

C is a subclass of B

Which one of these boolean expressions is true only when an object denoted by reference o has actually been instantiated from class B, as opposed to from A or C?

Pick one of the choices:

- **(o instanceof B) && (! (o instanceof A))**
- **(o instanceof B) && (! (o instanceof C))**
- **(o instanceof B)**
- **None of the above**

Ya que tanto B como C son instancias de A, al ser sus subClases, al realizar la primera, nunca entrara a la validación y siempre retornara false, sin embargo, si mencionamos que sea instancia de B, pero no de C que es su su clase hija, entonces resolvemos el problema.

8. **Which statement is true?**

Pick one of the choices:

- **Non-static member classes must have either default or public accessibility**
- **All nested classes can declare static member classes**
- **Methods in all nested classes can be declared static**
- **Static member classes can contain non-static methods**

Las clases miembro estáticas pueden contener métodos no estáticos. De hecho, pueden contener tanto métodos estáticos como no estáticos.

9. **A constructor is called whenever**

Pick one of the choices:

- **An object is declared**

- **An object is used**
- **A class is declared**
- **A class is used**

Al declarar un objeto por defecto lo hacemos mediante el constructor del mismo, por lo que decir que el constructor es llamado cuando el objeto es declarado es correcto.

10. **You are implementing a student registration and student information retrieval system for a school using a simple class roster in Java. When a stundent is registered, the system must assign an integer ID(enrollmentNumber), starting at 1 and adding 1 as each student is registered. The stundent's name is stored with the assigned enrollmentNumber. The retrieval request should return a student's registration information.**

The student class should implement:

- **The constructor. Student (String name)**
- **The method String toString() to return the string "(enrollmentNumber):(name)"**

The locked stub code in the editor validates the implementation of the Student class.

After each student is registered, the code stub requests and prints the student's information to test your code.

Constraints

- **1 <= numberOfStudents <=10^3**

Sample Input For Custom Testing

```
3
Erica
Bob
Maria
```

Sample Output

```
1: Erica
2: Bob
3: Maria
```

The three students are registered in the following order:

- The first student to be registered is Erica so she is assigned 1 as the enrollment number by the portal.
- Bob is second, so he is assigned the number2.
- Maria is third, so she es assigned the numer 3.

Now, the information of all the students is printed in the order in which they are registered.

11. **Which of the following data types in Java are primitive?**

Pick one of the choices:

- **String**
- **Struct**

- **Boolean**
- **char**

La forma más sencilla de saber si una variable es primitiva o no, es mediante la primera letra, ya que los primitivos comienzan en minúscula, y los Wrappers así como las clases comienzan con Mayúscula.

12. Which of the following are true for Java Classes?

Pick one of the choices:

- **The Void class extends the Class class**
- **The Float class extends the Double class**
- **The System class extends the Runtime class**
- **The Integer class extends the Number class**

La mayoría de Wrappers en Java (entiéndase por wrappers como la versión Clase de los primitivos en Java), extienden de la clase Number, entre ellos Integer.

13. The following code snippet is a demostration of a particular design pattern. Which desing pattern is it?

```
public class Mystery{
    private static Mystery instance = null;
    protected Mystery(){
        public static Mystery getInstance(){
            if(instance == null){
                instance = new Mystery();
            }
            return instance;
        }
    }
}
```

Pick one of the choices:

- **Factory Design Pattern**
- **Strategy Pattern**
- **Singleton**
- **Facade Design Pattern**

Si nos fijamos al realizar el código solamente se realizará una instancia de Mystery, y posterior a esto, siempre que lo solicitamos nos devolverá esta única instancia, por lo que encaja con lo que es el patrón Singleton.

14. Which of the following Java declaration of the String array is correct?

Pick one of the choices:

- **String temp [] = new String {“j”, “a”, “z”};**
- **String temp [] = {“j” “b” “c”};**
- **String temp = {“a”, “b”, “c”};**

- **String temp [] = {“a”, “b”, “c”};**

Si queremos crear un array de un String, necesitamos declararlo con comillas y entre brackets después del igual, así como separar por comas, además es necesario decirle a Java que será un array colocando [] en cualquier lugar entre frases después del String y antes del =.

15. Which is true of the following program?

```
1  package exam.java;
2
3  public class TestFirstApp {
4      static void doIt(int x, int y, int m) {
5          if(x==5) m=y;
6              else m=x;
7      }
8
9      public static void main(String[] args) {
10         int i=6, j=4, k=9;
11         TestFirstApp.doIt(i, j, k);
12             System.out.println(k);
13     }
14 }
```

Pick one of the choices

- **Doesn’t matter what the values of *i* and *j* are, the output will always be *5*.**
- **Doesn’t matter what the values of *k* and *j* are, the output will always be *5*.**
- **Doesn’t matter what the values of *i* and *j* are, the output will always be *9*.**
- **Doesn’t matter what the values of *k* and *j* are, the output will always be *9*.**

A pesar de que el código funciona correctamente, y en efecto esta mandando a llamar a una función, esta no regresa nada, y no está cambiando el valor de las variables que le pasamos en el scope del main, por lo que k nunca fue modificada e imprimirá el valor que le pasamos originalmente, el cual es 9.

16. Which of the following statements are correct. Select the correct answer.

- **Each Java file must have exactly one package statement to specify where the class is stored.**
- **If a java file has both import and package statement, the import statement must come before package statement.**
- **A java file has at least one class defined**
- **If a java file has a package statement, it must be the first statement (except comments).**

Aunque tendría ninguna función, es posible contar con un programa Java que no tenga la declaración de package, sin embargo, tampoco tendría otra cosa, por lo que no sería útil, sin embargo si lo tuviera debe ir en efecto como la primera línea de código que aparezca, sin contar los comentarios.

17. What is the output for the following program:

```
class Constructor
{
    static String str;
    public void Constructor(){
        System.out.println("In constructor");
        str="Hello World";
    }

    public static void main(String[] args) {

        Constructor c= new Constructor();
        System.out.println(str);
    }

}
```

Pick one of the choices

- **In Constructor**
- **Null**
- **Compilation Fails**
- **None of the above**

En ningún momento estamos agregando valor a str, ya que “Constructor” es un método y no un constructor como tal, por lo que al mandarlo a llamar desde el método main, nos regresará un null.

Given the following code, what is the most likely result.

```
import java.util.*;

public class Compares {

    public static void main(String[] args) {

    String [] cities= {"Bangalore","Pune","San Francisco","New York City"};
    MySort ms= new MySort();
    Arrays.sort(cities,ms);
    System.out.println(Arrays.binarySearch(cities,"New York City" ));
}
    static class MySort implements Comparator{
        public int compare(String a, String b){

            return b.compareTo(a);
        }

    }


}
```

Pick one of the choices

- **-1**
- **1**
- **2**
- **Compilation fails**

18. Which pattern do you see in the code below:
java.util.Calendar.getInstance();
Pick one of the choices

a) Singleton Pattern
b) Factory Pattern
c) Facade Pattern
d) Adaptor Pattern

El método getInstance, es un estándar cuando se habla de Singleton Pattern, por lo cual se puede deducir que se esta hablando de este cuando se ejecuta el codigo proporcionado.

19. What is the output of the following program:

```
interface BaseI { void method (); }
        class BaseC
        {
                public void method()
                {
                        System.out.println("Inside BaseC: :method");
                }
        }
        class ImplC extends BaseC implements BaseI
        {
                public static void main (String []s)
                {
                        (new ImplC()).method();
                }
        }
```

Pick one of the choices

a) **Null**
b) **Complication fails**
c) **Inside BaseC::method**
d) **None of the above**

El código es correcto, esa es una forma de llamar a un método de un objeto, sin necesidad de asignarle una variable de referencia al objeto que utilizamos.