1. Which declaration initializes a boolean variable?
   a) boolean m = null – Es un primitivo, no puede ser inicializado con null
   [✓ b) Boolean j = (1<5)] – Expresión que evalua, aunque este declarado como Boolean (wrapper) funciona por el autoboxing de Java.
   c) boolean k = 0
   d) boolean h = 1
   Las dos opciones son incorrectas debido que boolean solo acepta true / false.

2. What is the DTO pattern used for?

   [✓ a) To Exchange data between processes] → Su función principal es servir como contenedor de datos que puede ser serializado y enviado através de procesos.
   b) To implement the data Access layer → la capa de acceso generalmente usa DAOs
   c) To implement the presentation layer → Incorrecto. la capa de presentación puede usar DTOs, pero su proposito no es implementarla.

3. What value should replace kk in line 18 to cause jj = 5 to be output?

```
public class MyFive {
     public static void main(String[] args) {
          //short kk = ?;
          short ii;
          short jj = 0;
          for (ii = kk; ii > 6; ii-=1) { //Decremento ii=ii-1
                 jj++; // El ciclo for debe ser iterado 5 veces.
          }
          System.out.println("jj = " + jj);
     }
}
```

a) -1
b) 1
c) 5
d) 8
[✓ e) 11] ← Respuesta correcta.

Prueba de escritorio.

| //kk | ii | jj |
|---|---|---|
| 11 | 11 | 0 |
| | | 1 |
| | 10 | 2 |
| | 9 | 3 |
| | 8 | 4 |
| | 7 | 5 |
| | 6 | |

4. ¿Cuál será el resultado?

```
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 int getHash() {
         return 44444444;
   }
}
class SampleClassB extends SampleClass {
   public int getHash() {
         return 999999999;
   }
}
```

El programa crea las instancias de las clases: SampleClass, SampleClassA, sampleClass B. y luego imprime los valores de retorno de sus respectivos métodos getHash().

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.] ← Respuesta correcta.

5. ¿Cuál sería el resultado?

```
public classDoCompare4 {
     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);
```

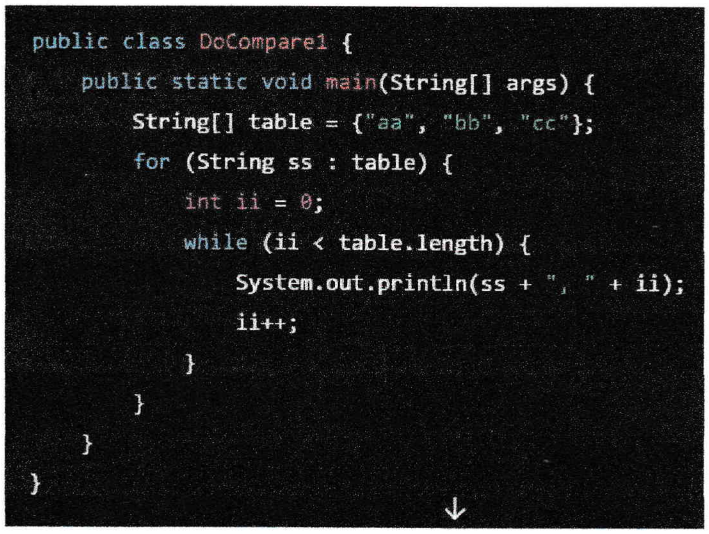
Comienza un bucle que se ejecutara mientras ii sea menor que la longitud de table.

imprime el valor actual de ii

```
        }
}
```

a) 0
[✓ b) 0 1 2] ← Respuesta correcta
c) 0 1 2 0 1 2 0 1 2
d) Compilation fails

6. ¿Cuál sería el resultado?



```
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++;
            }
        }
    }
}
```

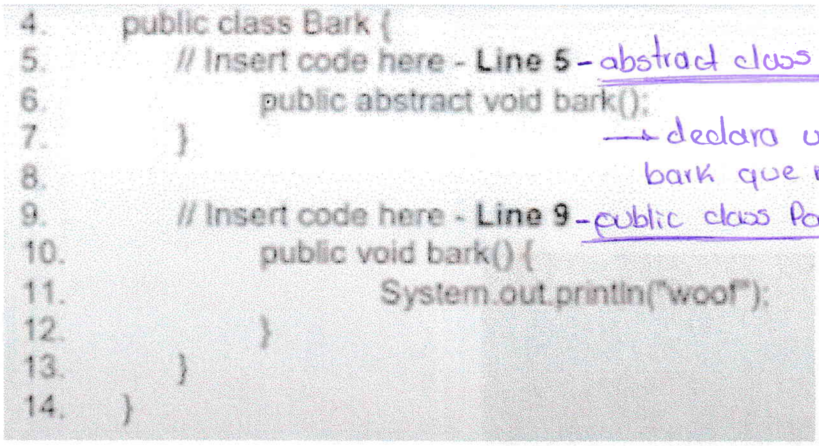
↓

```
public class DoCompare1 {
    public static void main(String[] args) {
        String[] table = {"aa", "bb", "cc"};
        for (String ss : table) {→ For each recorre cada elemento y lo asigna a ss
            int ii = 0;
            while (ii < table.length) {
                System.out.println(ss + ", " + ii);→ Imprime el valor actual
                ii++;                                de ss (,) y valor actual de ii
            }
        }
    }
}
```

[✓ a) Zero.] // Solo se imprimen los índices, no los valores de los elementos del array, por tanto esta es correcta.
b) Once.
c) Twice
d) Thrice

e) Compilation fails

7. What code should be inserted?



```
4.      public class Bark {
5.          // Insert code here - Line 5
6.              public abstract void bark();
7.          }
8.
9.          // Insert code here - Line 9
10.             public void bark() {
11.                 System.out.println("woof");
12.             }
13.         }
14.     }
```

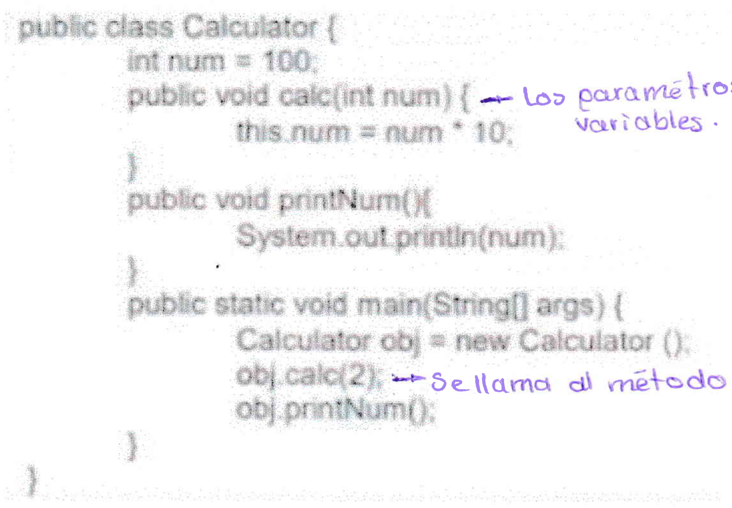
- abstract class Dog{ // Declara una clase abstracta llamada Dog.
→ declara un método abstracto llamado bark que no tiene implementación.
- public class Poodle extends Dog{ // clase que hereda de Dog.

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 {] ← Respuesta.
d) 5. abstract 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 {

8. Wich statement initializes a stringBuilder to a capacity of 128?

a) StringBuilder sb = new String("128"); - Incorrecto, trata de inicializar como un string.
b) StringBuilder sb = StringBuilder.setCapacity(128); C.- No existe setCapacity en stringBuilder.
c) StringBuilder sb = StringBuilder.getInstance(128); D.- No existe método getInstance en StringBuilder
[✓ d) StringBuilder sb = new StringBuilder (128);]
↳ Cuando se crea una instancia se puede especificar su capacidad inicial que define cuantos caracteres puede tener. el objeto.

9. What is the result?



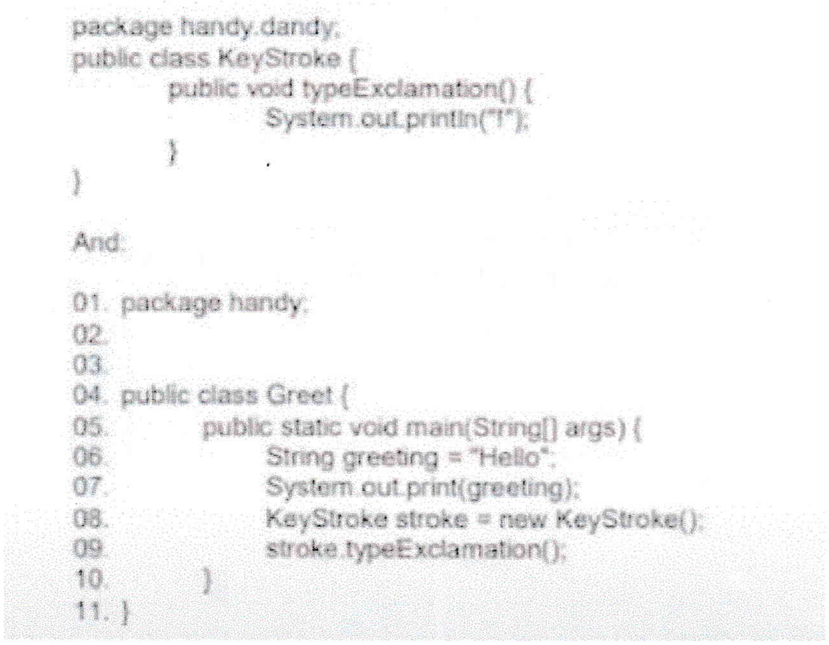
```
public class Calculator {
        int num = 100;
        public void calc(int num) {
                this.num = num * 10;
        }
        public void printNum(){
                System.out.println(num);
        }
        public static void main(String[] args) {
                Calculator obj = new Calculator ();
                obj.calc(2);
                obj.printNum();
        }
}
```

← Los paramétros se llaman igual pero son diferentes variables.

← Se llama al método calc pasando el 2 como parametro.

[✓ a) 20] → Respuesta correcta.
b) 100
c) 1000
d) 2

10. What three modifications, made independently, made to class Greet, enable the code to compile and run?



```
package handy.dandy;
public class KeyStroke {
        public void typeExclamation() {
                System.out.println("!");
        }
}

And:

01. package handy;
02.
03.
04. public class Greet {
05.         public static void main(String[] args) {
06.                 String greeting = "Hello";
07.                 System.out.print(greeting);
08.                 KeyStroke stroke = new KeyStroke();
09.                 stroke.typeExclamation();
10.         }
11. }
```

a) Line 8 replaced with handy.dandy.KeyStroke stroke = new KeyStroke();
b) Line 8 replaced with handy.*.KeyStroke stroke = new KeyStroke();
[✓ c) Line 8 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 after line 1.

- Importar todo el paquete
- Importar la clase especifica
- Pasar toda la ruta a la instancia.

Ⓐ

1. Consider the following Java code snippet: *Dividir 4/0 genera una excepción del tipo ArithmeticException.

```
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;
}
```

c = a/b; → lo que lanza una excepción. ArithmeticException.

catch(Exception e){ → Como ocurre una excepción aquí se imprime el mensaje " Exception"

System.err.println("Finally "); → Finally se ejecuta siempre. por lo que imprime el mensaje "Finally"

return c; ↳ El bloque no termina abruptamente por el bloque catch así que (c) conserva su valor inicial.

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

[✓ a) Exception Finally] → Respuesta correcta.
b) Finally Exception
c) Exception

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

a) Overloading(SobreCarga)
[✓ b) Overriding (SobreEscritura @Override)] → Misma firma (nombre y tipo de argumentos)
c) Java does not permit methods with same and type signature
d) None of the above

3. What does the following for loop output?

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

--i → decrementa
++j → incrementa

Prueba de escritorio:

| Iteración. | i | j | j%i | Salida. |
|---|---|---|---|---|
| 1 | 10 | 1 | 1 % 10 = 1 | 1 |
| 2 | 9 | 2 | 2 % 9 = 2 | 12 |
| 3 | 8 | 3 | 3 % 8 = 3 | 123 |
| 4 | 7 | 4 | 4 % 7 = 4 | 1234 |
| 5 | 6 | 5 | 5 % 6 = 5 | 12345 |

a) 12321
[✓ b) 12345]
c) 11111
d) 00000

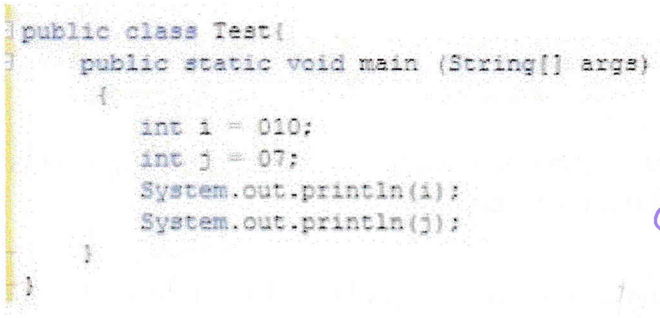
4. We perform the following sequence of actions:

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? – Un set no permite elementos duplicados, al insertar valores repetidos solo guarda una copia
– Al convertir un set a una lista, se mantiene el mismo conjunto de valores únicos y luego se ordenan los datos.

[✓ a) {1, 2, 3, 4, 5, 7, 9}]
b) {9, 7, 5, 4, 3, 2, 1}
c) {1, 1, 1, 1, 2, 2, 3, 4, 5, 7, 9}
d) None of the above

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);
    }
}
```

* Un número octal se representa con un prefijo 0

Octal 010 = 8 decimal
Octal 07 = 7 decimal

[✓ a) 8 7]
b) 10 7
c) Compilation fails with an error at line 3
d) Compilation fails with an error at line 5

6. A public data member with the same name is provided in both base as well as derived clases. Which of the following is true?
//incorrecto, El compilador permite campos con el mismo nombre en ambas clases.
a) It is a compiler error to provide a field with the same name in both base and derived class

* Cuando un campo o atributo en una clase derivada tiene el mismo nombre que un campo en la clase base, el campo de la clase derivada oculta el campo de la clase base.

b) The program will compile and this feature is called overloading - Sobrecarga no aplica a los campos.
c) The program will compile and this feature is called overriding - Sobre-escritura no aplica en campos
[✓ d) The program will compile and this feature is called as hiding or shadowing]

7. Which statement is true?
a) Non-static member classes must have either default or public accessibility - pueden tener cualquier nivel de accesibilidad
b) All nested classes can declare static member classes - Solo las clases externas y las clases estáticas anidadas pueden declarar miembros no estáticos
c) Methods in all nested classes can be declared static
[✓ d) Static member classes can contain non-static methods]

c) los métodos estáticos solo pueden ser declaradas en clases estaticas o externas.

8. A constructor is called whenever

[✓ a) An object is declared] ← Es llamado automáticamente cuando un objeto de la clase es creado, cuando se usa la palabra new
b) An object is used - Usar un objeto no invoca al constructor
c) A class is declared
d) A class is used

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

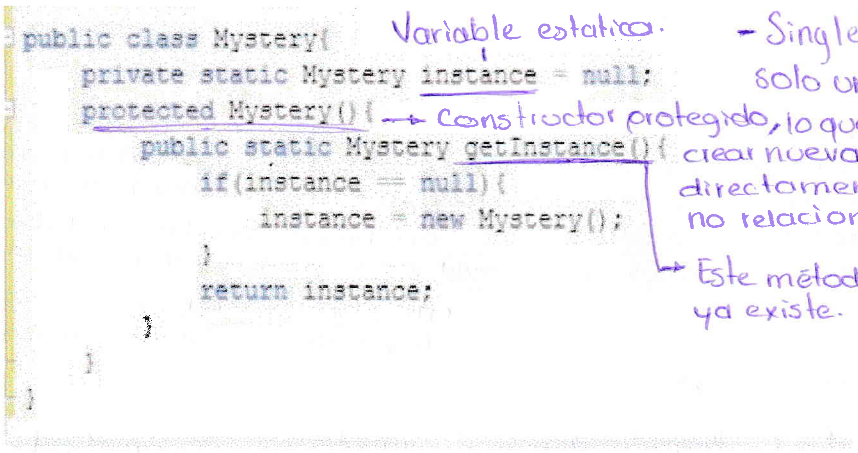
a) String
b) Struct
[✓ c) Boolean]
[✓ d) Char]

Boolean y Character son clases envolventes (wrappers) que permiten tratar los tipos primitivos como objetos.

10. Which of the following are true for Java Classes?
a) The Void class extends the Class class - Void no extiende a ninguna clase.
b) The Float class extends the Double class - Float y Double son wrappers independientes de java.lang
c) The System class extends the Runtime class — System proporciona métodos estáticos para el acceso a recursos del sistema y Runtime permite interactuar con el entorno de ejecución de Java.
[✓ d) The Integer class extends the Number class]

Integer es una clase wrapper para valores primitivos de tipo int y extiende la clase abstracta Number.

11. 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;
        }
    }
}
```

Variable estatica.

- Singleton asegura que tenga solo una instancia.

→ Constructor protegido, lo que significa que no puede crear nueva instancia de la clase directamente desde fuera de las clases no relacionadas.

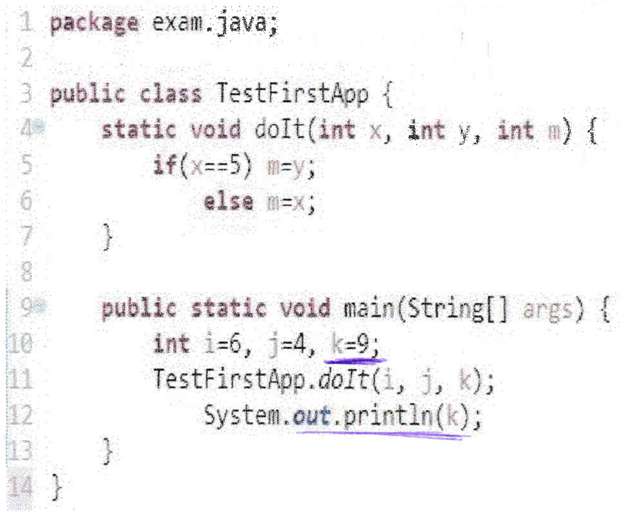
→ Este método verifica si la instancia ya existe.

a) Factory Design Pattern
b) Strategy Pattern
[✓ c) Singleton]
d) Facade Design Pattern

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

a) String temp [] = new String {"j", "a", "z"}; – Incorrecto, la sintaxis para inicializar un arreglo con valores no puede usar new String.
Faltan comas → b) String temp [] = {"j" "b" "c"};
c) String temp = {"a", "b", "c"}; → Aqui solo se declara una variable temp como objeto string.
[✓ d) S tring te mp [] = { " a" , "b", "c"};] → Sintaxis válida

13. Which is true of the following program?



```
package exam.java;

public class TestFirstApp {
    static void doIt(int x, int y, int m) {
        if(x==5) m=y;
            else m=x;
    }

    public static void main(String[] args) {
        int i=6, j=4, k=9;
        TestFirstApp.doIt(i, j, k);
            System.out.println(k);
    }
}
```

- En el método main, K se inicializa con 9
- En el método doIt no puede modificar el valor de K debido a que m es solo una copia del valor K.

- Por lo tanto cuando se imprime K en main, su valor sigue siendo el mismo.

a) Doesn't matter what the values of *i* and *j* are, the output will always be 5.
b) Doesn't matter what the values of *k* and *j* are, the output will always be 5.
c) Doesn't matter what the values of *i* and *j* are, the output will always be 9.
[✓ d) Doesn' t matte r what the v alue s of *k* and *j* are, the output will always be 9.]

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

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

15. 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);
      }

   }

}
```

- Genera un error porque el método compare no tiene la firma correcta para la interfaz Comparator.

Para que compile:
1. Usar generics en Comparator.
// Comparator <String>

2- Cambiar los argumentos de compare a Object.
// compare (Object a, Object b)

a) -
1 b)
1 c)
2
[✓ d) Compilation fails]

16. To delete all pairs of keys and values in a given HashMap, which of the following methods should be used?

a) clearAll() - No existe para HashMap
b) empty() - Este método no existe en HashMap isEmpty() verifica si está vacío.
c) remove() - Elimina solo un par clave-valor, identificado por la key proporcionada
[✓ d) clear()] - Elimina todos los pares de claves y valores.

17. Which pattern do you see in the code below:

java.util.Calendar.getInstance();

-Factory pattern es un patrón de diseño que proporciona un método para crear objetos sin especificar la clase exacta, dónde el objeto se creó.

a) Singleton Pattern
[✓ b) Factory Pattern]
c) Facade Pattern
d) Adaptor Pattern

Oculta los detalles de la creación del objeto y proporciona una instancia de la subclase.

18. What is the output of the following program:

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

a) Null
b) Complicatio fails
[✓ c) Inside BaseC::method] → Salida de ejecución.
d) None of the above

19. Consider the following three classes:

-instanceof verifica si un objeto es una instancia de una clase especifica o de superclase

```
class A {}
class B extends A {}
class C extends B {}
```

Consider n object of class B is instantiated, i.e.,

```
B b = new B();
```

Which of the following Boolean expressions evaluates to true:

[✓ a) (b instanceof B)] - Verdadero
b) (b instanceof B) && (!(b instanceof A)) - Falso, b tambien es un instancia de A
[✓ c) (b instanceof B) && (!(b instanceof C))] - Verdadero - b no es instancia de C
d) None of the above

20. What is the output of the following program:

```
class Constructor
        {
        static String str;
        public void Constructor(){
                System.out.println("I n  constructor");
                str = " He llo  Wor ld";
        }
        public static void main (String [] args){
                Constructor C = new Constructor ();
                System.out.println(str);
        }
}
```

→ Este no es un constructor, es un método llamado constructor porque retorna.

← Cuando se crea el objeto el compilador llama al constructor predeterminado. con el mismo nombre de la clase.

a) In Constructor
[✓ b) Null]
c) Compilation fails
d) None of the above