

21/03/2025

UT1-PDI

1) Public static void zoop () {

baffle();

System.out.print("Key greater");

baffle();

}

public static void main (String [] args) {

System.out.print("No, yo");

zoop();

System.out.print("yo");

baffle();

}

public static void baffle () {

System.out.print("pae");

long();

}

public static void pao () {

System.out.println("O");

}

La relación:

No, Yo soy.

"Por favor pa.

Yo soy.

2) Public class Zumbido {

```
    public static void desconectar (String dirigible) {  
        System.out.println (dirigible) 4  
        upo ("ping", -5); 5  
    }
```

```
    public static void upo (String membrillo, int flag) {  
        if (flag < 0) {  
            System.out.println (membrillo + "up") 6  
        } else {  
            System.out.println ("uf"); 2  
            desconectar (membrillo); 3  
            System.out.println ("rubor-ja-ja-ja"); 7  
        }  
    }  
}
```

```
    public static void main (String [] args) {  
        upo ("troquelos", 13) 1  
    }  
}
```

1) La primera sentencia que se ejecuta es
 `upo ("troquelos", 13)` en `main()`.

```
public class Alumno {  
    private String nombre;  
    public Alumno () {  
        nombre = null;  
    }
```

```
    public String getNombreAdmision () {  
        return nombre.repeat ("1");  
    }
```

```
    public static void main (String [] args) {  
        Alumno alumno = new Alumno ();  
        System.out.println (alumno.getNombreAdmision ());  
    }  
}
```

```
    public static int recorrer (String cadena) {  
        int ref = 0;  
        for (int i = 1; i <= cadena.length (); i++) {  
            if (cadena.charAt (i) != '.') {  
                ref++;  
            }  
        }  
        return ref;  
    }
```

```
    public static int getValor () {  
        int vector [] = {6, 16, 26, 36, 46, 56, 66, 76};  
        int rdz = 8;  
        return vector [rdz];  
    }
```

```
public static char getFirstCharacter(String palabra) {  
    String string[] = new String[5];  
    return (String[1].charAt(1));  
}
```

```
public static String printString(int a) {  
    Object x1 = new Integer(a);  
    return (String)(x1);  
}
```

a) No funciona porque en el constructor de la clase Alumno el nombre es null. Es decir "nombre" se inicializa con null. Para corregirlo se debe inicializar con un valor predeterminado, por ejemplo, una cadena vacía "".

b) El bucle va ($i=1$ en lugar de $i=0$, por lo que no accede a la primera posición en la iteración número uno y además no termina en cadena.length() - 1 por lo tanto queda fuera de rango. Se corrige cambiando a $i=0$ y cadena.length() - 1

c) El error se trata de el índice, al parecer se quiere obtener el último elemento del vector pero los posiciones comienzan del 0 a n porciones. La solución es cambiar idx a 7 que sería la última posición de la lista.

d) El código se encarga de obtener el primer carácter de un String, intenta acceder a string[1], pero el array string se ha declarado sin inicializar sus elementos, por lo que string[1] es null. Para corregirlo se debe inicializar el array antes de acceder a sus elementos y cambiar charAt(1) a charAt(0).

e) Se intenta devolver un objeto de Integer a String con (String) x1, lo que genera una ~~OverflowException~~ porque Integer no es un String. Para la corrección se usa String.valueOf(a).

UT1 - PDT

Ejercicio 1

```
public class IdentifyMyParty {  
    public static int x = 7;  
    public int y = 3  
}
```

a) ¿Cuáles son las variables de clase?

b) ¿Cuáles son las variables de instancia?

¿Cuál es la salida que produce el siguiente código?

```
IdentifyMyParty a = new IdentifyMyParty();
```

```
IdentifyMyParty b = new IdentifyMyParty();
```

```
a.y = 5;
```

```
b.y = 6;
```

```
a.x = 1;
```

```
b.x = 2;
```

```
System.out.println("a.y = " + a.y);
```

```
System.out.println("b.y = " + b.y);
```

```
System.out.println("a.x = " + a.x);
```

```
System.out.println("b.x = " + b.x);
```

```
System.out.println("IdentifyMyParty.x = " + IdentifyMyParty.x);
```

a) La variable de clase es x porque tiene la palabra clave el static.

b) La variable de instancia es y porque no tiene static y se trata de una copia de y en esta instancia.

$$\text{La tabla es: } \begin{array}{l} 2.y = 5 \\ b.y = 6 \end{array}$$

$$2.x = 2$$

$$b.z = 2$$

Identify My Parts. $x = 2$

H11 - ITU

Ejercicio 2

public class SomethingWrong {

 public static void main (String [2 args]) {

 Rectangle myRect;

 myRect.width = 40;

 myRect.height = 50;

 System.out.println ("myRect's area is " + myRect.area());

}

}

1) No se ha creado el objeto Rectangle y por lo tanto no podremos acceder a los atributos width y height. Se habrá la variable del tipo de este último. No se proporciona la definición de la clase Rectangle, por lo que asumiremos que debe tener atributos width, height y un método llamado area().

2) Para reparar el error primero debemos definir la clase Rectangle con sus atributos width y length, y el método area(). En main se debe crear una instancia de Rectangle usando new Rectangle();

Class Rectangle {

 int width;

 int height;

 int area() {

 return width * height;

}

- 1) Quedan 2 referencias a estos objetos: luego de ejecutar el código, 1 referencia al array `students` y otra a "Peter Parker" en `students[0]`.

Ninguno de los dos va a ser eliminado por el garbage collector ya que el garbage collector se encarga de liberar memoria eliminando objetos que jamás tienen referencias activas en el programa. Pero en este caso el array `students` tiene referencias a `student` y "Peter Parker" tiene esta referencia por `student`.

- 2) Una de las formas es eliminar referencias a un objeto, si este ya no es necesario, podemos eliminarlo y así las referencias para que el garbage collector lo recicle. Siendo `ognos[0].name` el objeto `ognos[0]` ya no tiene referencias activas, por lo que el garbage collector puede eliminarlo en el futuro.

ejercicios 4

- No funcionan porque el programa no recibe ni solicita los argumentos pero la condición `if (args.length >= 2)` ejecuta true, pero esto no es cierto. La función de `args` (`args.length == 2`).
- Existen errores de conversión de String a Float, `Float.parseFloat(args[1])` tiene un espacio y está mal la sintaxis. `Float.parseFloat(args[1])` `float()` está mal escrito, `float()` no existe en `Float`, debió ser `parseFloat()`.

Ejercicio 5

```
public class ToStringDemo {  
    public static void main (String [] args) {  
        double d = 888.51;  
        String s = Double.toString (d);  
  
        int dot = s.indexOf ('.');//  
  
        System.out.println (dot + " digits " + " before decimal point.");  
        System.out.println ((s.length () - dot - 1) + " digits after  
        decimal point.");  
    }  
}
```

1) Se introduce al ejecutorlo ej: 3 digits before decimal point.
2 digits after decimal point.

2) El programa convierte el número 888.51 en una cadena
y usa `indexof (".")` para encontrar la posición del punto
decimal. Luego, calcula los dígitos antes y después del punto:
Antes del punto: La posición dd. indica que hay 3
dígitos.
Después del punto: Se restan la posición del punto y 1
de la longitud ($6 - 3 - 1 = 2$), por lo que hay 2 dígitos.

Ejercicio 8

String Builder St = new StringBuilder("Able was I ere I saw Elba.");

$$\hookrightarrow \text{Longitud} = 25$$

La capacidad inicial del StringBuilder es la longitud de la cadena más 16.

$$\text{Entonces ej: } 25 + 16 = 41.$$

Ejercicio 9

String hannah = "zel Hannah see jeep? Hannah did.";

a) El total de caracteres ej 32, por lo que hannah.length() es 32.

b) El valor retornado con el método hannah.charAt(12) es e.

c) La expresión que referencia las letras en la frase referida por "hannah" es "hannah.charAt(15)"

Ejercicio 10

¿Cuán larga es la string devuelta por la siguiente expresión? ¿Cuál es la string?

"Was it a car or a cat I saw?".substring(9, 12)

La longitud de la string devuelta por la expresión es:
La string es "car".

Ejercicio 3

Se define la string "Dot now I was Iod" que es un palíndromo, calcula la longitud de la cadena, luego copia los caracteres de la cadena en un array temporal. Invierte el orden de los caracteres y los almacena. Luego a ésta se la convierte en un string y se imprime.

Se puede notar que la cadena está invertida, pero el programa solo la invertirá sin procesar mayúsculas, espacios o puntuación, no verifica si es palíndromo.

ES

(P) Problema: "Saca I das la cosa de dentro de Iod"

UT1 - PD7

2) J1 Abreva el siguiente código:

```
String s = "1";
while(s != "1000") {
    s += "0";
}
```

En Java, el operador `==` compara referencias de objetos, no su contenido. Cada vez que se ejecuta `s += "0"`, se crea un nuevo objeto String en memoria. En Java, String es inmutable, lo que significa que no se modifica el objeto original, sino que se crea uno nuevo.

La condición `s != "1000"` no se evalúa con el contenido de la cadena, sino con los direcciones de memoria. "1000" es un literal de cadena almacenado en el pool de String, mientras que `s` apunta a un objeto diferente en el heap. El resultado es un bucle infinito ya que `s` nunca tiene la misma referencia que "1000".

En C#, el operador `==` para String compara el contenido, no las referencias de memoria. C# compara el contenido de los caracteres, no hay referencias de memoria, ya que `s == "1000"` verifica si los caracteres coinciden, el bucle terminará cuando `s` sea "1000".

Ejercicio 2

1) En Java, los objetos "holo" se almacenan en un pool de strings, cuando se asigna `String s1 = "holo";` el compilador coloca "holo" en el pool de strings. Cuando se asigna `String s2 = "holo";` en lugar de crear un nuevo objeto, se reutiliza la misma referencia del pool. Por lo tanto, `s1 == s2` devolverá true.

1 / 1

2) String s1 = new String ("Hola"); crea un nuevo objeto String en el heap, fuera del pool de String.
String s2 = "Hola"; toma la referencia del pool de String.
Cabeza s1 y s2 no son el mismo objeto en memoria,
s1 == s2 devuelve falso.