Alba Maydeu Ariño
TA5

Ejercicio 1

¿Por qué incluir esMayor en la declaración System.out.println?

Aprendizaje: La variable esMayor se incluye para imprimir el resultado de la comparación variable2 > variable1, que es un valor booleano (true o false).

¿Es esMayor lo mismo que >?

Aprendizaje: No, > es un operador de comparación, mientras que esMayor es una variable booleana que almacena el resultado de la comparación usando el operador >.

Ejemplos de otras declaraciones System.out.println:

Aprendizaje: Se proporcionaron varios ejemplos para imprimir los valores de las variables y los resultados de diferentes comparaciones.

¿Siempre necesito decir "es mayor que" para esMayor?

Aprendizaje: No, puedes formatear la declaración System.out.println de diferentes maneras, como usar solo los números y el resultado para una salida más concisa.

Ejercicio 2.

Para que se imprima syso “Bienvenida (importante añadir un espacio para que se imprima bien i no quede todo junto) ” + nombre

Ejercicio 3

El ejercicio no pide que modifiquemos la aplicación anterior, para que nos pida el nombre que queremos introducir. También nos pide que usemos para ello JOptionPane, es decir mostrar ventanas de diálogo/pop up.
Para ello planteamos el ejercicio siguiendo estos pasos-

1. **Importar JOptionPane**:
    - Usamos import javax.swing.JOptionPane; para poder usar la clase JOptionPane.
2. **Solicitar el nombre**:
    - JOptionPane.showInputDialog muestra un cuadro de diálogo que solicita al usuario que ingrese su nombre. El valor ingresado se almacena en la variable nombre.
3. **Mostrar mensaje de bienvenida**:
    - JOptionPane.showMessageDialog muestra un cuadro de diálogo con el mensaje de bienvenida, incluyendo el nombre ingresado.

Recordemos -

¿Qué es javax.swing?
javax.swing es un **paquete de Java** que proporciona una amplia gama de componentes gráficos para crear interfaces de usuario (GUI, por sus siglas en inglés). El nombre javax.swing se desglosa de la siguiente manera:

1. **javax**: Indica que es una extensión de Java (la "x" viene de "extension"). Originalmente, javax se usó para paquetes que no formaban parte del núcleo de Java, pero con el tiempo, muchos de estos paquetes se han vuelto estándar.
2. **swing**: Es el nombre de la biblioteca gráfica que proporciona componentes más modernos y flexibles que los de la antigua biblioteca AWT (Abstract Window Toolkit).

¿Qué es JOptionPane?

JOptionPane es una **clase** dentro del paquete javax.swing que se utiliza para mostrar cuadros de diálogo simples, como:

- Cuadros de diálogo para ingresar datos (showInputDialog).
- Cuadros de diálogo para mostrar mensajes (showMessageDialog).
- Cuadros de diálogo de confirmación (showConfirmDialog).

Bueno despues de todo esto no he podido ejecutar sin errores el ejercicio ya que me sale - do no find or load to main class. Lo he resueto volviendo a repetir el ejercicio desde cero creando un nuevo proyecto y al final me ha salido.

4) Ejercicio 4

El objetivo de este programa es calcular el área de un círculo a partir del radio proporcionado por el usuario. Para lograrlo, se siguen los siguientes pasos:

Paso 1- Solicitar el radio al usuario:

- El programa utiliza la clase JOptionPane de la biblioteca javax.swing para mostrar un cuadro de diálogo gráfico que solicita al usuario que ingrese el radio del círculo. Este valor se almacena en una variable de tipo String llamada radioStr.

Paso 2- Convertir el radio a un número:

- Dado que el valor ingresado por el usuario es un String, es necesario convertirlo a un número de tipo double para poder realizar operaciones matemáticas. Esto se hace utilizando el método Double.parseDouble, que convierte una cadena de texto en un número decimal.

Paso 3- Calcular el área del círculo:

- Una vez que se tiene el radio como un número, se procede a calcular el área del círculo utilizando la fórmula matemática: $\text{Area}=\pi \times R^2$

Paso 4-Mostrar el resultado al usuario:

- Finalmente, el programa muestra el área calculada en otro cuadro de diálogo gráfico utilizando JOptionPane.showMessageDialog. Esto permite que el usuario vea el resultado de manera clara y direct

Como fun fact-Muy interesante para mirar el método pow para el interés compuesto que por ejemplo se utiliza en finanzas.

Ejercicio 5

La mejor manera de saber si algo es par o impar es dividendo por 2. SI la division da 0 es par. Por ejemplo 100%2=0.

Aternativamente num/2 = “int” - es decir, sin decimales

Ejercicio 6

También es importante recordar que import -java.util.Scanner- siempre se ha de poner antes que el nombre del ejercicio para que asi funcione. porque si no, no funcionará porque no reconocerá la clase Scanner.

Final doble- final significa que el valor es constante, no se puede cambiar
Final doble IVA= 0.21; //21%

Ejercicio 7

-while- es una estructura de control que se utiliza para repetir un bloque de código mientras se cumple una condición. La condición se evalúa antes de ejecutar el bloque de código. Si la condición es verdadera, el bloque de código se ejecuta. Si la condición es falsa, el bloque de código no se ejecuta y el programa continúa con la instrucción siguiente después del bucle while.

Es decir, el bucle while se ejecutará mientras que la condición sea verdadera.
La teoria i el esquema para yo entenderlo es este—

¿Para qué sirven los bucles?

Imagina que tienes que hacer algo muchas veces, como contar del 1 al 10 o lavar varios platos. En lugar de hacerlo una y otra vez manualmente, los bucles son como una herramienta que te permite repetir una tarea automáticamente. ¡Son como un "asistente" que hace el trabajo repetitivo por ti!

Bucle for:

- **¿Cuándo lo usas?** Cuando sabes **exactamente cuántas veces** necesitas repetir algo. Por ejemplo, si tienes que contar del 1 al 10, sabes que lo harás 10 veces.

- **¿Cómo funciona?** Piensa en un bucle for como una receta con 3 pasos:
    1. **Inicio:** Decides desde dónde empiezas (por ejemplo, empezar a contar desde el número 1).
    2. **Condición:** Decides hasta dónde quieres llegar (por ejemplo, contar hasta el número 10).
    3. **Paso:** Decides cómo avanzas (por ejemplo, aumentar de 1 en 1).

Ejemplo de la vida real:
Si tienes que lavar 5 platos, sabes que harás exactamente 5 repeticiones. Usas un bucle for porque sabes cuántas veces lo harás.

Bucle while:

- **¿Cuándo lo usas?** Cuando **no sabes cuántas veces** tienes que repetir algo, pero sabes que debes hacerlo **mientras se cumpla una condición**. Por ejemplo, seguir intentando abrir una puerta hasta que encuentres la llave correcta.
- **¿Cómo funciona?** Solo necesitas una **condición**. El bucle se repite una y otra vez mientras esa condición sea verdadera.

Ejemplo de la vida real:
Imagina que estás esperando un autobús. No sabes cuánto tiempo tendrás que esperar, pero sabes que debes seguir esperando **mientras** el autobús no haya llegado. Aquí usarías un bucle while.

Diferencias entre for y while:

1. **Cuándo usarlos:**
    - Usas for cuando **sabes cuántas veces** necesitas repetir algo.
    - Usas while cuando **no sabes cuántas veces** lo harás, pero sabes que debes repetirlo **mientras se cumpla una condición**.
2. **Estructura:**
    - El for tiene una estructura más organizada, con un inicio, una condición y un paso.
    - El while es más simple, solo necesita una condición para funcionar.
3. **Ejemplos de la vida real:**
    - **for:** Como contar los días de la semana (sabes que son 7 días).
    - **while:** Como seguir intentando abrir una puerta hasta que encuentres la llave correcta (no sabes cuántas veces lo intentarás).

Ejercicio 8-

La teoría está en el ejercicio. Como observación se dice interador y no iterador.

Ejercicio 9

== significa condicion de pregunta
&& significa true and true; ambas condiciones han de ser verdaderas

¿Qué Aprendí?

Uso del bucle for:

- Aprendí que el bucle for es una estructura de control que permite repetir un bloque de código un número específico de veces. En este caso, se utilizó para recorrer los números del 1 al 100.
- La estructura del for incluye:
  - **Inicialización:** Donde se declara e inicializa una variable (por ejemplo, int iterador = 1).
  - **Condición:** Que define cuándo el bucle debe continuar (por ejemplo, iterador <= 100).
  - **Incremento:** Que actualiza la variable en cada iteración (por ejemplo, iterador++).

Uso de la condición if:

- Aprendí que la condición if permite ejecutar un bloque de código solo si se cumple una condición específica.
- En este ejercicio, se utilizó para verificar si un número era divisible entre 2 y 3 al mismo tiempo, usando el operador módulo %.

Operador módulo %:

- El operador módulo (%) devuelve el resto de una división. Si el resto es 0, significa que el número es divisible.
- Por ejemplo:
  - 6 % 2 == 0 (6 es divisible entre 2).
  - 6 % 3 == 0 (6 es divisible entre 3).

Operador lógico &&:

- El operador && (AND lógico) se utiliza para combinar dos condiciones. Ambas deben ser verdaderas para que la expresión completa sea verdadera.
- En este ejercicio, se usó para verificar si un número era divisible entre 2 **y** 3 al mismo tiempo.

Importancia de las llaves {}:

- Aprendí que las llaves {} se utilizan para definir bloques de código. Cada bloque que se abre con { debe cerrarse con }.
- Si no se cierran correctamente, el programa no funciona porque el compilador no entiende dónde termina cada bloque.

Problemas que tuve con el ejercicio que me hacía que no se ejecutara bien.

1. **Falta de llaves de cierre:**
    - El principal problema fue que no cerraba correctamente todos los bloques de código con llaves }. En Java, cada bloque que se abre con { debe cerrarse con }.
    - En mi código original, solo tenía 2 o 3 llaves de cierre, pero necesitaba **4 llaves** para cerrar

Ejercicio 10

Realiza una aplicación que nos pida un número de ventas a introducir, después nos pedirá tantas ventas por teclado como número de ventas se hayan indicado. Al final mostrará la suma de todas las ventas. Piensa que es lo que se repite y lo que no.

Ejercicio 11

Cosas que he aprendido en este ejercicio pero que sem usan bastante en java-
En Java, **Scanner** es una clase que se utiliza para leer datos de entrada. En este caso, se usa para leer lo que el usuario escribe en la consola. Cuando creas un objeto Scanner, estás preparando el programa para recibir información del usuario a través del teclado.
Ej- Scanner scanner = new Scanner(System.in);

- **System.in**: Representa la entrada estándar, que es el teclado. Le dice al Scanner que debe leer lo que el usuario escriba en la consola.
- **scanner**: Es el objeto que usaremos para leer la entrada del usuario que escribe en la consola

¿Cómo funciona en la consola?

Cuando ejecutas el programa, este se detiene en la línea donde se llama a scanner.nextLine() y espera a que el usuario escriba algo y presione **Enter**. Lo que el usuario escriba se almacenará en la variable dia.
Por ejemplo:
System.out.println("Introduce un día de la semana:");
String dia = scanner.nextLine();