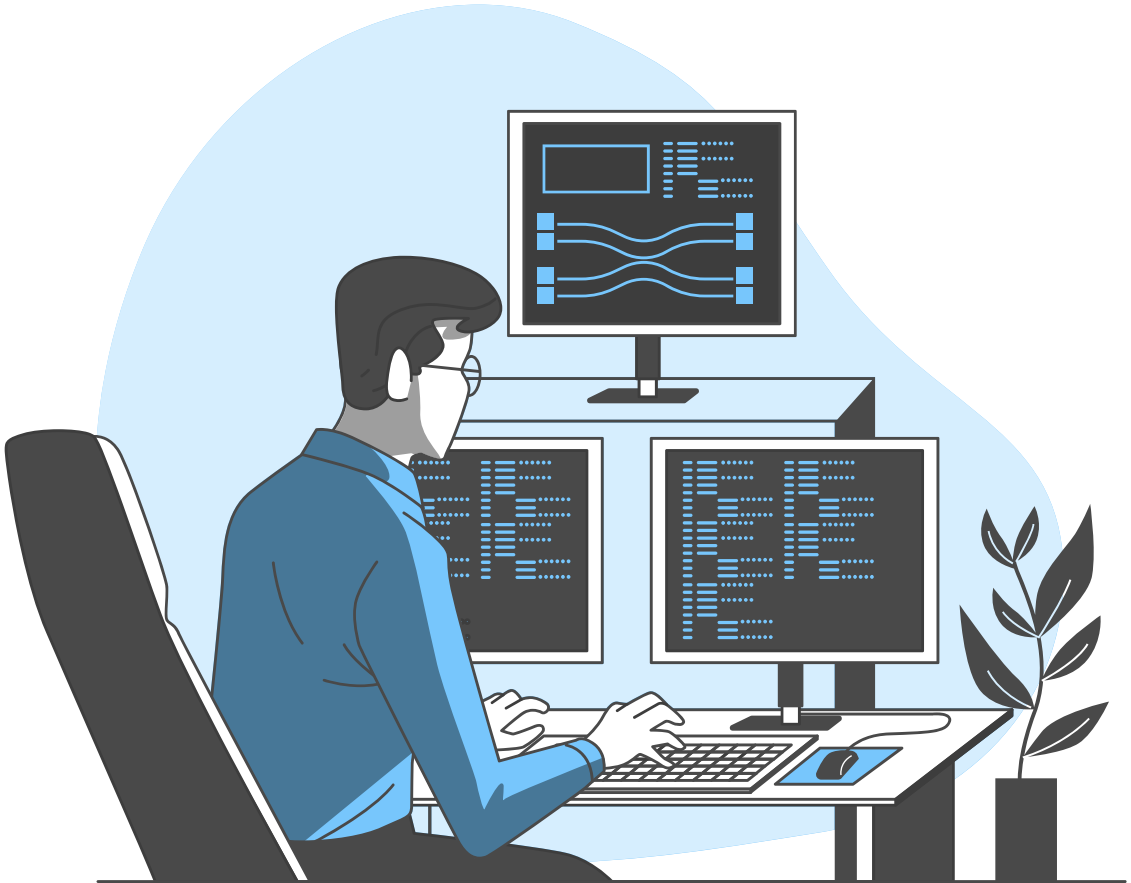


Introducción a la programación

Clase 5

Profesor – Martín Gómez Vega

Da Vinci
PRIMERA ESCUELA DE ARTE MULTIMEDIAL

Agenda

1. **Condicionales anidados**
2. Operadores en Java
3. Caracteres de escape
4. Scanner
5. JOptionPane
6. Práctica

Condicionales anidados

Son útiles cuando deseas verificar varias condiciones secuencialmente y ejecutar diferentes bloques de código según el resultado de estas condiciones.

La estructura típica de los condicionales anidados se ve así:

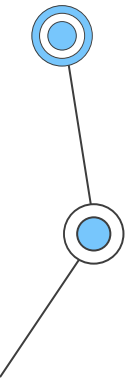
```
if (condicion1) {
  // Código a ejecutar si condicion1 es verdadera

  if (condicion2) {
    // Código a ejecutar si condicion2 también es verdadera
  } else {
    // Código a ejecutar si condicion1 es verdadera pero condicion2 es falsa
  }
} else if (condicion3) {
  // Código a ejecutar si condicion1 es falsa pero condicion3 es verdadera
} else {
  // Código a ejecutar si ninguna de las condiciones anteriores es verdadera
}
```

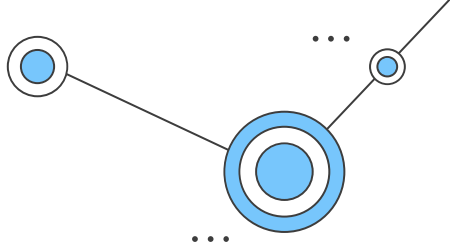
Aquí tienes una explicación paso a paso de cómo funcionan los condicionales anidados:

- Se evalúa la condicion1. Si es verdadera, se ejecuta el código dentro del primer bloque if.
- Dentro del bloque if que sigue a condicion1, se evalúa condicion2. Si también es verdadera, se ejecuta el código dentro de ese bloque. Si condicion2 es falsa, se ejecuta el código dentro del else correspondiente a condicion1.
- Si condicion1 es falsa, se pasa a la evaluación de condicion3 en el bloque else if. Si condicion3 es verdadera, se ejecuta el código dentro de ese bloque.
- Si ninguna de las condiciones anteriores es verdadera, se ejecuta el código dentro del bloque else, que maneja todos los demás casos posibles.

Agenda

1. Condicionales anidados
2. **Operadores en Java**
3. Caracteres de escape
4. Scanner
5. JOptionPane
6. Práctica





Operadores en Java

Java ofrece una variedad de operadores que se utilizan para realizar diversas operaciones en el lenguaje. Estos operadores se dividen en varias categorías, incluyendo operadores aritméticos, operadores de comparación, operadores lógicos, operadores de asignación y operadores especiales.

Los operadores más comunes en Java:

Aritméticos

+: Suma.
-: Resta.
*: Multiplicación.
/: División.
%: Módulo (obtiene el residuo de una división).

Comparación

==: Igual a.
!=: Diferente de.
<: Menor que.
>: Mayor que.
<=: Menor o igual que.
>=: Mayor o igual que.

Lógicos

&&: AND lógico (conjunción).
||: OR lógico (disyunción).
!: NOT lógico (negación).

Asignación

=: Asignación simple.
+=: Asignación con suma.
-=: Asignación con resta.
*=: Asignación con multiplicación.
/=: Asignación con división.
%=: Asignación con módulo.

Operadores en Java

Los operadores especiales en Java:

Incremento Decremento

++: Incremento (aumenta el valor de la variable en 1).
--: Decremento (reduce el valor de la variable en 1).

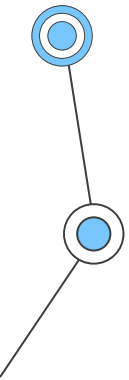
Condicional (Ternario)

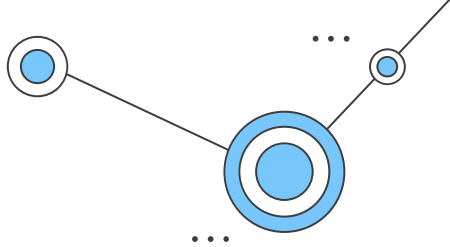
condición ? valor1 : valor2

Descripción: Se utiliza para asignar un valor a una variable en función de una condición. Si la condición es verdadera, se asigna valor1; de lo contrario, se asigna valor2.

Agenda

1. Condicionales anidados
2. Operadores en Java
3. **Caracteres de escape**
4. Scanner
5. JOptionPane
6. Práctica



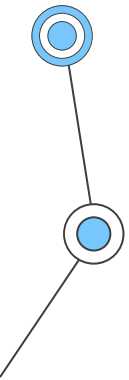


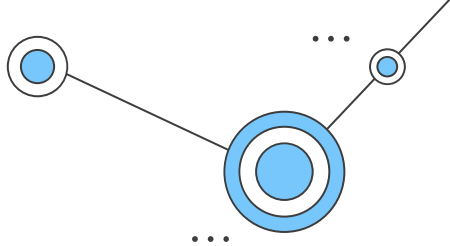
Caracteres de escape

Caracteres de escape	Descripción
\n	Salto de línea
\t	Tabulador
\r	Retorno de carro
\b	Borrado a la izquierda
\\	Contrabarra \
\»	Comillas dobles
\'	Comillas simples

Agenda

1. Condicionales anidados
2. Operadores en Java
3. Caracteres de escape
4. **Scanner**
5. JOptionPane
6. Práctica





Scanner

El principal uso de Scanner es la entrada de datos desde el usuario a través del teclado y la manipulación de estos datos dentro de un programa.

Lectura de entrada del usuario: Scanner permite leer datos que un usuario ingresa a través del teclado. Esto es útil para crear programas interactivos que solicitan información al usuario.

```java
import java.util.Scanner;

public class App {
    Run | Debug
    public static void main(String[] args) throws Exception {
        Scanner scanner = new Scanner(System.in);

        System.out.print(s:"Ingrese su nombre: ");
        String nombre = scanner.nextLine();

        System.out.println("Hola, " + nombre + "!");

        scanner.close(); // Es importante cerrar el Scanner cuando ya no se necesita.
    }
}
```

```java
import java.util.Scanner;

public class App {
    Run | Debug
    public static void main(String[] args) throws Exception {
        Scanner scanner = new Scanner(System.in);

        System.out.print(s:"Ingrese su nombre: ");
        String nombre = scanner.next(); // Utilizamos next() para leer la primera palabra

        System.out.println("Hola, " + nombre + "!");

        // Leer cualquier texto adicional en el búfer de entrada (puede ser un apellido)
        String restoDelTexto = scanner.nextLine().trim();

        if (!restoDelTexto.isEmpty()) {
            System.out.println("También ingresaste: " + restoDelTexto);
        }

        scanner.close();
    }
}
```

Agenda

1. Condicionales anidados
2. Operadores en Java
3. Caracteres de escape
4. Scanner
5. **JOptionPane**
6. Práctica

JOptionPane

JOptionPane es una clase en la biblioteca estándar de Java Swing que se utiliza para crear cuadros de diálogo de forma fácily rápida en aplicaciones de escritorio. Estos cuadros de diálogo son ventanas emergentes que permiten la interacción del usuario con la aplicación en un contexto específico.
JOptionPane es particularmente útil para la entrada de datos, la confirmación de acciones y la presentación de mensajes informativos.

Utilidades de JOptionPane

1. ***Mensajes Informativos***: Puedes utilizar JOptionPane para mostrar mensajes informativos al usuario. Esto es útil para proporcionar información sobre el estado de la aplicación o para dar retroalimentación al usuario.
   JOptionPane.showMessageDialog(null, "Operación completada con éxito", "Éxito", JOptionPane.INFORMATION_MESSAGE);

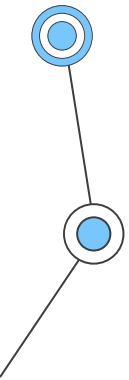
2. ***Solicitar Entrada del Usuario***: JOptionPane permite mostrar un cuadro de diálogo que solicita al usuario que ingrese datos. Esto es útil para recibir entrada de texto, números u otros tipos de datos.
   String nombre = JOptionPane.showInputDialog("Ingrese su nombre:");

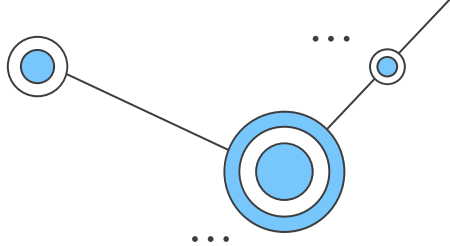
3. ***Confirmación de Acciones***: Puedes utilizar JOptionPane para solicitar confirmación del usuario antes de realizar una acción crítica, como eliminar un archivo o salir de la aplicación.
   ```
   int confirmacion = JOptionPane.showConfirmDialog(null, "¿Está seguro de eliminar este archivo?", "Confirmar", JOptionPane.YES_NO_OPTION);
   if (confirmacion == JOptionPane.YES_OPTION) {
       // Eliminar el archivo
   }
   ```

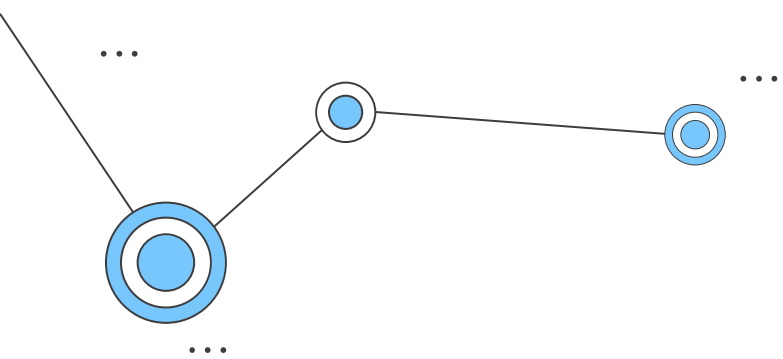
4. ***Selección de Opciones***: En situaciones donde el usuario debe elegir entre varias opciones, JOptionPane puede presentar un cuadro de diálogo con botones de opción.
   ```
   String[ ] opciones = {"Opción 1", "Opción 2", "Opción 3"};
   int seleccion = JOptionPane.showOptionDialog(null, "Seleccione una opción:", "Selección", JOptionPane.DEFAULT_OPTION,
               JOptionPane.QUESTION_MESSAGE, null, opciones, opciones[0]);
   ```

Agenda

1. Condicionales anidados
2. Operadores en Java
3. Caracteres de escape
4. Scanner
5. JOptionPane
6. **Práctica**

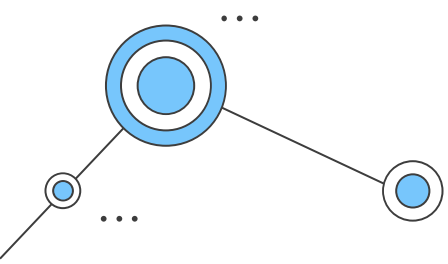






Práctica

Crear un sistema que permita de manera segura gestionar el retiro de dinero de una cuenta, indicando el saldo disponible en la extracción.



¿Dudas?