JAVA

JVM (Maquina virtual de java)

maven para trabajar todas las dependencias.

DEFINICION DE UNA CLASE: Una clase es como un plano que define las características de los objetos, mientras que un método es una acción que estos objetos pueden realizar. son plantillas para la creación de objetos

SEGUNDA DEFINICION DE CLASE: es una plantilla que define la estructura y el comportamiento de los objetos que se pueden crear a partir de ella

DEFINICION DE METODO: Los métodos son bloques de código que se utilizan para realizar tareas específicas

¿Por qué es necesario compilar?:

Recuerda que la compilación traduce tu código a bytecode que la JVM puede ejecutar, permitiendo que la aplicación sea independiente de la plataforma.

Podemos usar bloques try-catch para capturar y manejar errores

DEFINICION VARIABLE: Una variable es un espacio reservado en memoria que puede almacenar datos. Estos datos se pueden modificar durante la ejecución del programa. Cada variable se declara con un tipo de dato específico y un nombre.

TIPOS DE DATOS

DATOS PRIMITIVOS: int, double y boolean. Todos estos ocupan un espacio fijo en la memoria.

DATOS NO PRIMITIVOS: objetos como String, Arrays y Classes, que permiten realizar operaciones más complejas. Los espacios de memoria para tipos no primitivos se gestionan de manera dinámica. Esto significa que el tamaño de estos espacios de memoria no está fijado en tiempo de compilación, sino que se determina durante el

tiempo de ejecución.

TIPOS DE DATOS

MUTABLES: permiten cambiar su estado después de su creación, lo cual es útil para modificar cadenas de texto frecuentemente sin generar desperdicio de memoria.

INMMUTABLE: no cambian su estado después de ser creados, lo que ayuda a mantener la seguridad y la integridad de los datos.

Java maneja la memoria automáticamente a través del recolector de basura, que se encarga de liberar la memoria ocupada por objetos que ya no se utilizan.

OPERADORES

Estos se utilizan para realizar operaciones sobre variables y constantes.

Tenemos operadores aritméticos, de asignación, lógicos y relacionales, unarios

y ternarios.

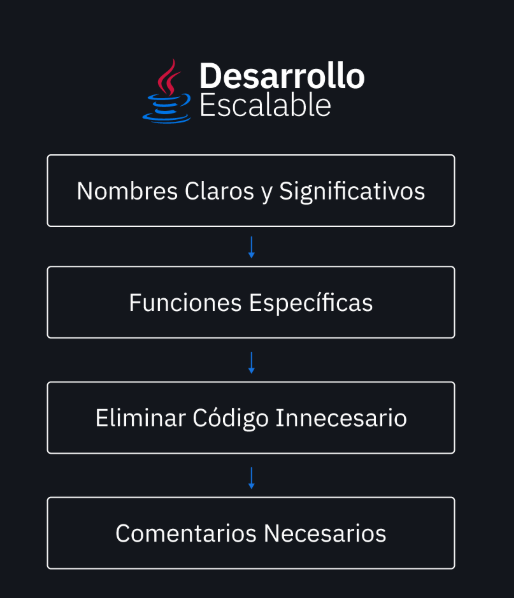
OPERADORES ARITMETICOS: Los operadores aritméticos incluyen operaciones como suma, resta, multiplicación y división.

OPERADORES DE ASIGNACION: Los operadores de asignación se utilizan para asignar valores a las variables.

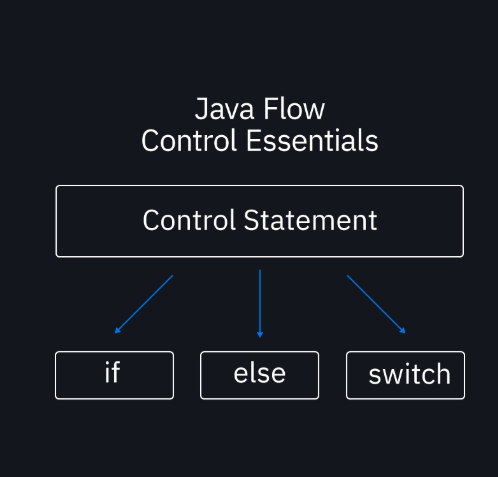
OPERADORES LOGICOS Y RELACIONALES: Los operadores lógicos y relacionales son cruciales para la toma de decisiones en el código. (son los operadores que normalmente se manejan en las condiciones de las funciones o metodos (&& || and or = == === <= >= !)

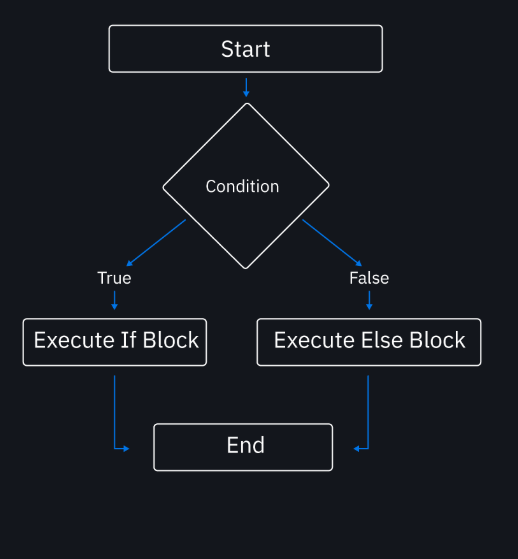
OPERADORES UNARIOS: Los operadores unarios, como el incremento y decremento, modifican el valor de una variable en uno.

OPERADORES TERNARIOS: los operadores ternarios permiten simplificar el código al realizar asignaciones condicionales en una sola línea.



CONTROL DE FLUJO: En programación llamamos control de flujo a la toma de decisiones y ejecutar acciones basadas en condiciones específicas.



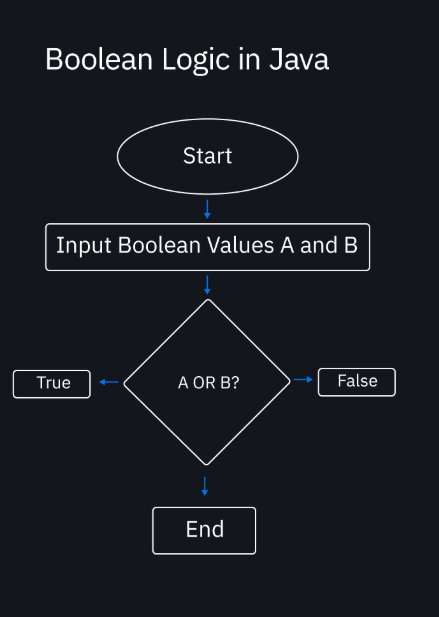


IF

ELSE

Si cumple con las condiciones, mostrara un resultado que ya hemos definido, sino, mostrara el otro resultado que nosotros definamos en el campo.

BOOLEAN



La lógica booleana utiliza operadores como && (AND), || (OR), y ! (NOT) para combinar y manipular valores booleanos. El tipo de dato booleano solo recibe dos tipos de valores “true” and “false”

SWITCH

es una declaración de control de flujo que permite ejecutar diferentes bloques de código basados en el valor de una expresión.

BUCLES

son estructuras que permiten ejecutar un bloque de código repetidamente hasta que una condición específica deje de ser verdadera.

Los bucles son fundamentales para automatizar tareas repetitivas, como:

* **Procesar listas de datos:** Recorrer arrays, listas u otras colecciones para realizar operaciones sobre cada elemento.
* **Realizar cálculos iterativos:** Repetir cálculos hasta alcanzar un resultado deseado.
* **Validar entradas de usuario:** Solicitar al usuario que ingrese datos hasta que se cumplan ciertos criterios.
* **Crear animaciones y juegos:** Actualizar elementos gráficos o estados del juego en cada iteración.

**Tipos de bucles comunes:**

* **Bucle "for":**
  + Se utiliza cuando se conoce de antemano el número de repeticiones.
  + Consta de una inicialización, una condición y un incremento/decremento.
* **Bucle "while":**
  + Se utiliza cuando el número de repeticiones depende de una condición.
  + El bloque de código se ejecuta mientras la condición sea verdadera.
* **Bucle "do-while":**
  + Similar al "while", pero garantiza que el bloque de código se ejecute al menos una vez.
  + La evaluación de la condición se realiza después de la primera ejecución.
* **Bucle "foreach" o "for-each":**
  + Este bucle es utilizado para iterar a través de colecciones u objetos iterables, permitiendo que el programador itere sin preocuparse por los index o contadores que otros loops requieren.

ARRAYS: un **array** (o arreglo) es una estructura de datos que almacena una colección de elementos del mismo tipo bajo un solo nombre. Imagina un array como una fila de casillas, donde cada casilla guarda un valor.

Declaración y Uso de ArrayList:

Lo primero que necesitas es importar su clase.

Y luego usar métodos para añadir elementos con add, eliminar con remove, obtener con get y modificar con set.

Por ejemplo, HashMap.

Esta almacena pares clave-valor, lo que es ideal para asociaciones rápidas y eficientes, muy útil para búsquedas y actualizaciones rápidas.

HashMap no mantiene el orden de sus elementos, pero permite una búsqueda de datos extremadamente rápida.

**Qué es una LinkedList?**

* Una LinkedList (lista enlazada) es una estructura de datos lineal en la que los elementos no se almacenan en ubicaciones de memoria contiguas.
* En lugar de ello, cada elemento, llamado "nodo", contiene dos partes:
  + Los datos almacenados.
  + Una referencia (o puntero) al siguiente nodo en la secuencia.
* El modificador **private** limita el acceso solo dentro de la misma clase.
* También tenemos a **protected** que permite el acceso dentro del mismo paquete y por subclases.
* Por su parte **public** hace que los miembros sean accesibles desde cualquier clase.

**Herencia**: Permite que una clase hija herede atributos y métodos de una clase padre. Facilita la reutilización del código y la especialización de clases.

**Polimorfismo**: Permite que un objeto adopte muchas formas. En tiempo de ejecución, permite que los métodos sobrescritos se comporten de manera diferente según la subclase que los implemente.

**Encapsulación**: Permite ocultar los detalles internos de una clase y controlar el acceso a sus atributos mediante métodos públicos. Mejora la seguridad y la integridad del código.

**NullPointerException**:

* Esta excepción se lanza cuando intentas acceder a un método o atributo de un objeto que es null. En otras palabras, estás intentando usar una referencia que no apunta a ningún objeto en la memoria.

En Java, cuando intentas acceder a un método o a una propiedad de una variable que tiene el valor null, se lanza una **NullPointerException**. Esto ocurre porque no puedes invocar métodos ni acceder a miembros de un objeto que no ha sido inicializado (es decir, que es null).

Por ejemplo:

java

Copiar

String nombre = null;

System.out.println(nombre.length()); // Esto lanzará una NullPointerException

LAMBDA

Las **expresiones lambda** en Java son una característica introducida en **Java 8** que permite escribir **funciones anónimas** de manera más concisa y funcional. Las expresiones lambda representan una forma de pasar comportamientos (funciones) como parámetros a métodos, lo cual es útil especialmente en el contexto de la programación funcional.

En Java, los **Streams** son una característica introducida en **Java 8** como parte de la API de **Streams**, la cual permite trabajar con colecciones de manera más declarativa y funcional. Los Streams facilitan operaciones como filtrado, mapeo, reducción y otras transformaciones sobre secuencias de datos de manera concisa y fluida.

En términos simples, un **Stream** es una secuencia de elementos de una colección que permite realizar operaciones de manera funcional y perezosa (lazy evaluation), sin necesidad de modificar la colección original. Estas operaciones se pueden encadenar, lo que permite componer soluciones complejas de manera más declarativa.

### **¿Qué es un Stream?**

Un **Stream** en Java es una secuencia de elementos que puede ser procesada de manera secuencial o paralela. Los Streams pueden provenir de varias fuentes, como colecciones (listas, conjuntos, etc.), arreglos, archivos o generadores de datos.