códigofacilito

Java Profesional

Bloque 2: Fundamentos del lenguaje

Tatiana Borda
https://tatianaborda.com/
https://www.linkedin.com/in/tatiana-borda/
https://www.youtube.com/@AlienExplorer

códigofacilito

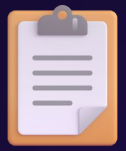

Tabla de contenido

códigofacilito

/01 Variables y tipos de datos en Java

En Java, una variable es un espacio reservado en memoria para guardar un valor.
Al ser un lenguaje fuertemente tipado para poder usar una variable, necesitamos declarar su tipo de dato.
Tenemos datos del tipo primitivo y no primitivo.

códigofacilito

Qué son los tipos de datos primitivos?

En Java, los datos primitivos son tipos de datos básicos que representan valores simples y se utilizan para representar valores lógicos, números enteros, caracteres, y números decimales. Sus principales características son:

*Tamaño fijo:
Cada tipo de dato primitivo tiene un tamaño fijo en bytes, lo que define el rango de valores que puede almacenar.

*No son objetos:
Los datos primitivos no son objetos, lo que significa que no tienen métodos asociados.

*Inmutabilidad:
Los valores de los tipos primitivos son inmutables, es decir, no se pueden modificar una vez que se han creado.

*Almacenamiento directo:
Los valores de los datos primitivos se almacenan directamente en la memoria

códigofacilito

TIPOS PRIMITIVOS EN JAVA

TIPO	DESCRIPCIÓN	EJEMPLO
byte	Entero muy pequeño	byte edad = 25;
short	Entero corto	short x = 1234;
int	Entero estándar	int a = 42;
long	Entero largo	long n = 100L;
float	Decimal de precisión simple	float f = 3,14f;
double	Decimal de doble precisión	double pi = 3,1416;
char	Un solo carácter	char letra = "A";
boolean	Valor verdadero/falso	boolean activo = true;

códigofacilito

Y qué NO es primitivo?

Los datos no primitivos son también conocidos como objetos o tipos de referencia.Sus principales características son:

*No almacenan valores directamente:
En lugar de contener el valor en sí, contienen una referencia a la ubicación donde se guarda el objeto en memoria.

*Representan objetos:
Son instancias de clases, incluyendo clases predefinidas como String y clases personalizadas.

*Pueden tener métodos:
Los objetos no primitivos tienen métodos asociados que permiten realizar operaciones sobre ellos.

*Comienzan con mayúscula:
A diferencia de los tipos primitivos, los nombres de los tipos de referencia empiezan con mayúscula y su valor por defecto es null

*Clases personalizadas: El programador puede definir sus propios tipos de datos mediante clases.

códigofacilito

Ejemplos de datos NO primitivos:

String (Cadena de texto)

Es una clase (no un tipo primitivo), pero tiene sintaxis simplificada con comillas dobles ("texto").

Inmutable: No puede modificarse después de su creación (cada operación crea un nuevo objeto).

Tiene métodos útiles como

length()
substring()
equals()

```
String nombre = "Java";  // Forma literal (recomendada)
String lenguaje = new String("Python");  // Usando constructor
System.out.println(nombre.length());  // 4
```

códigofacilito

Ejemplos de datos NO primitivos:

Array (Arreglo)

Estructura que almacena múltiples valores del mismo tipo.

Tamaño fijo después de su creación

Índices comienzan en 0

```java
int[] numeros = {10, 20, 30};  // Forma simplificada
String[] palabras = new String[3];  // Forma con tamaño definido
palabras[0] = "Hola";
System.out.println(numeros[1]);  // 20
```

códigofacilito

Ejemplos de datos NO primitivos:

Clases Personalizadas:

Plantillas para crear objetos con atributos (variables) y métodos (funciones)

Se instancian con `new`

```java
// Definición de la clase
public class Persona {
    String nombre;
    int edad;

    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    void saludar() {
        System.out.println("Hola, soy " + nombre);
    }
}

// Uso
Persona persona1 = new Persona("Ana", 25);
persona1.saludar();  // "Hola, soy Ana"
```

códigofacilito

/02 Operadores en Java

Los operadores en Java son símbolos que indican a la computadora qué operación realizar en los datos de una expresión.
Se utilizan para realizar cálculos, evaluar condiciones, asignar valores a variables, y mucho más.
Son esenciales para manipular datos y variables dentro de nuestro programa

códigofacilito

Tipos de operadores:

Operadores aritméticos: Permiten realizar cálculos matemáticos como

Suma +

Resta –

Multiplicación *

División /

Resto %

```
int suma = 5 + 3;          // 8
int resto = 10 % 3;        // 1 (módulo)
double division = 10.0 / 3; // 3.333...
```

Tipos de operadores:

Operadores de comparación: Permiten comparar dos valores y devolver un resultado booleano (true o false)

igualdad ==

desigualdad !=

mayor que >

menor que <

mayor o igual que >=

menor o igual que <=

```
boolean esIgual = (5 == 5);     // true
boolean esMayor = (10 > 20);    // false
boolean esDiferente = (7 != 3); // true
```

Tipos de operadores:

Operadores lógicos: Permiten combinar o modificar valores booleanos.

conjunción (AND) &&

disyunción (OR) ||

negación (NOT) !

```
boolean andLogico = (true && false); // false
boolean orLogico = (true || false);  // true
boolean notLogico = !true;           // false
```

códigofacilito

Tipos de operadores:

Operadores de asignación: Permiten asignar valores a variables y sus variantes que modifican la variable al mismo tiempo

Asignación básica `=`

Asignación y suma `+=`

Asignación y resta `-=`

Asignación y multiplicación `*=`

Asignación y división `/=`

```
int x = 10;           // Asignación básica
x += 5;               // Equivale a x = x + 5 (x = 15)
String nombre = "Ana"; // Asignación a String
```

códigofacilito

Tipos de operadores:

Operadores de incremento/decremento: Permiten aumentar o disminuir el valor de una variable en 1

Incremental `++`

Decremental `- -`

```
int contador = 0;
contador++;          // Incrementa a 1 (post-incremento)
int y = --contador;  // Decrementa a 0 (pre-decremento)
```

Operadores ternarios: Permiten realizar operaciones condicionales en una sola línea

Condicional `? :`

```
int edad = 18;
String mensaje = (edad >= 18) ? "Mayor" : "Menor"; // "Mayor"
```

códigofacilito

/03

Casting : Conversión de tipos

Es el proceso de convertir un valor de un tipo de datos a otro tipo de datos. Esta conversión si es entre primitivos puede ser explícita (realizada por el programador) o implícita (realizada por el compilador). Pero también hay otros casos de conversión de datos no primitivos, veamos cada uno con un ejemplo!

códigofacilito

Casos comunes de casting:

Entre tipos numéricos primitivos

*Implícito (widening): Ocurre automáticamente cuando pasás de un tipo más pequeño a uno más grande, no hay pérdida de información, no sería estrictamente casting

```
int x = 10;
double y = x; // ok
```

*Explícito (narrowing): Ocurre cuando querés pasar de un tipo más grande a uno más pequeño y puede haber pérdida de datos, por eso tenés que indicarlo vos.

```
double z = 10.5;
int n = (int) z; // n = 10
```

códigofacilito

Casos comunes de casting:

Entre clases (objetos)

*Upcasting (automático): Es cuando un objeto de una subclase pasa a una super clase, no sería estrictamente casting java lo hace automaticamente. Por ejemplo: Yo soy programadora, pero también soy persona. El upcasting es convertir un objeto específico (programadora) a un tipo más general (persona)

*Downcasting (explícito):Es cuando un objeto de una superclase pasa a una sub clase, tenés que indicarlo vos y puede presentar errores. Por ejemplo: si hacemos el caso anterior pero al revés agarramos a una persona cualquiera Java te pide confirmarlo con (Programadora), porque no todas las personas son programadoras. Podría ser una doctora, un chef o un dj.En el downcasting necesitás asegurarte de que la persona realmente sea programadora, o Java te tira un error en tiempo de ejecución.

Pero no te preocupes Lo veremos más adelante en POO!

códigofacilito

Conversión:

Entre String y tipos numéricos

*De String a int, double u otros

datos númericos primitivos

Debés hacerlo con clases envoltorio

```
String edadTexto = "25";
int edad = Integer.parseInt(edadTexto);
```

*De número a String

con el método valueOf()

```
int numero = 123;
String texto = String.valueOf(numero);
System.out.println(texto); // "123"
```

códigofacilito

/04 Entrada de datos

Para interactuar con el usuario por consola, usamos la clase Scanner.
`Scanner` es una clase del paquete java.util que permite leer datos ingresados por el usuario desde la consola (o también desde archivos, pero eso es más avanzado)
Veamos cómo usarla!

códigofacilito

Cómo se usa?

1. Importar la clase

2. Crear una instancia con la palabra `new`
`System.in` indica que queremos
leer desde la entrada estándar
que es el teclado

3. Leer diferentes tipos de datos
Según el método usado, en este
Caso lee la línea completa

4. Es una buena práctica cerrar
el Scanner al final del programa

```java
import java.util.Scanner;

Scanner sc = new Scanner(System.in);
System.out.print("Ingresa tu nombre: ");
String nombre = sc.nextLine();
```

```java
sc.close();
```

LECTURA DE DIFERENTES TIPOS DE DATOS

Método	Qué lee	Ejemplo
nextLine()	Línea completa	String nombre = sc.nextLine();
next()	Una sola palabra	String palabra = sc.next();
nextInt()	Entero (ínt)	int edad = sc.nextInt();
nextDouble()	Decimal (double)	double precio = sc.nextDouble();

códigofacilito

/05

Buenas prácticas de código

Son convenciones y técnicas que mejoran la legibilidad, mantenibilidad(código fácil de actualizar), el trabajo en equipo porque todos siguen las mismas reglas y la eficiencia del software ya que reduce bugs puesto que a través de una estructura clara hay menos errores. Algunas de ellas son:
*Usar nombres descriptivos: edad, nombreUsuario, etc. Que el nombre de variables, métodos y clases sea autoexplicativo

*Evitar comentarios innecesarios, que repitan lo que el código ya dice.Deberían explicar el "por qué" y no el "qué"

*Indentar correctamente: usar 4 espacios (o 1 tab) por nivel. Llaves ({}) en la misma línea o nueva línea de codigo (pero ser consistente)

códigofacilito

Comentarios:

Los comentarios nos ayudan a documentar el código, son una referencia y ayuda para otras personas que trabajan con nosotros y para nuestro yo del futuro también

```
// Esto es un comentario de una línea
/* Esto es
un comentario
de varias líneas */
```

códigofacilito

/06 HANDS ON

A codear, vamos al editor de código!

REPOSITORIO CON EL CÓDIGO:
https://github.com/tatianaborda/java-course

códigofacilito

Objetivo:
Aplicar todo lo aprendido en el bloque 2 creando un menú básico que:

Use variables y tipos de datos

Aplique operadores aritméticos y lógicos

Lea datos con Scanner

Realice validaciones simples

🧱 Lo que vamos a hacer:
Mostrar un menú con tres opciones:
Agregar tarea
Ver tareas (placeholder)
Salir
Leer la opción del usuario con Scanner
Verificar que la opción ingresada sea válida (uso de operadores lógicos)
Solicitar el nombre de la tarea y su prioridad (número)
Hacer una operación con esa prioridad (por ejemplo, sumarla a una variable total de prioridades)