«Talento Tech»

Back-End

Java

Clase 02

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase 2: Fundamentos de Programación en Java

Índice

Objetivos de la Clase

- Comprender la importancia de las **variables** y practicar su declaración y uso.
- Aplicar **operadores aritméticos, lógicos y relacionales** para realizar cálculos y comparaciones.
- Utilizar **entrada y salida de datos** para interactuar con el usuario (o con otros componentes del sistema).
- Emplear **estructuras condicionales (if, else if, else)** para tomar decisiones en función de determinadas condiciones.
- Implementar **bucles (for, while)** para automatizar tareas repetitivas.

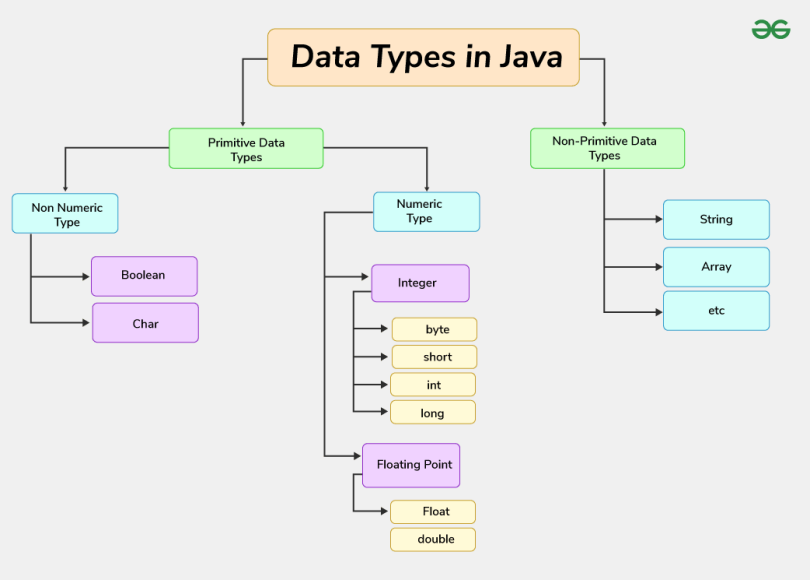
Buenos Aires aprende
Agencia de Habilidades para el Futuro

Introducción.

¡Les damos la bienvenida nuevamente! Después de haber configurado nuestro entorno y creado nuestro primer programa en Java, hoy avanzaremos hacia los fundamentos esenciales de la programación en este lenguaje. Aprenderemos a manejar variables, operadores, entrada/salida de datos y estructuras de control, herramientas indispensables para resolver problemas del día a día en nuestro proyecto.

Recordá que en TechLab buscamos soluciones digitales eficientes. La idea es que cada concepto técnico que veamos responda a un desafío concreto de nuestra empresa, haciendo el aprendizaje más relevante y aplicable.

Tipos de Datos y Variables.



Las variables nos permiten almacenar información en la memoria y usarla a lo largo del programa. Podemos guardar números, texto, valores booleanos, etc.

Tipos de Datos Primitivos:

- `int`: Números enteros (ej. 0, 10, -5). Ideal para cantidades: stock, demanda, etc.
- `double`: Números con decimales (ej. 3.14, 250.75). Perfecto para precios o totales monetarios.
- `boolean`: Valores lógicos (true o false). Muy útil para condiciones: ¿stock suficiente? ¿aplica descuento?
- `char`: Un solo carácter (ej. 'A') - menos común en estos ejemplos.

Existen otros tipos primitivos (`long`, `short`, `byte`, `float`), pero `int`, `double` y `boolean` serán los más usados en este proyecto.

Tipo String:
Aunque no es primitivo, `String` se usa muchísimo para texto: nombres de productos, mensajes al usuario, etc. Por ejemplo:

```
String nombreProducto = "Café Premium Molido";
```

La elección del tipo es importante. Para stock, un `int` es lógico; para precios, `double` es excelente. Esto facilita cálculos y evita errores.

Problema: Seguimiento del Stock de Productos

Contexto: Silvia informa que el cliente quiere saber cuántas unidades de un producto específico tenemos en stock para ajustar su estrategia de venta.

Solución con Variables:

- Creamos una variable entera (`int`) para almacenar la cantidad de productos en stock.
- Suponé que tenemos un producto "Café Premium" y actualmente hay 150 unidades.

Ejemplo de Código:

```
int stockCafePremium = 150;
System.out.println("Stock actual de Café Premium: " +
stockCafePremium);
```

Esto nos permitirá mostrar el stock por pantalla y utilizar ese valor más adelante.

Operadores Aritméticos, Lógicos y Relacionales

Operadores Aritméticos:

- + Suma
- - Resta
- * Multiplicación
- / División (si ambos operandos son int, el resultado es entero; si uno es double, el resultado es decimal)
- % Módulo (resto de la división)

Símbolo	Significado	Ejemplo
=	Asignación	x = 4
+	Suma	x + 3
-	Resta	x - 3
/	División	x / 3
*	Multiplicación	x * 4 / 2
%	Módulo (resto entero)	20 % 3

Ejemplo (Cálculo del costo total de un pedido):

```
int precioUnitario = 200;

int cantidad = 10;

int costoTotal = precioUnitario * cantidad; // 2000

System.out.println("El costo total del pedido es: $" + costoTotal);
```

Operadores Relacionales:

- `>` Mayor que
- `<` Menor que
- `>=` Mayor o igual que
- `<=` Menor o igual que
- `==` Igual
- `!=` Distinto

Con estos operadores, podemos verificar, por ejemplo, si el stock es mayor o igual a la demanda (`stock >= demanda`).

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
==	igual que	`7 == 38`	`false`
!=	distinto que	`'a' != 'k'`	`true`
<	menor que	`'G' < 'B'`	`false`
>	mayor que	`'b' > 'a'`	`true`
<=	menor o igual que	`7.5 <= 7.38`	`false`
>=	mayor o igual que	`38 >= 7`	`true`

Operadores Lógicos:

- && (AND) verdadero si ambas condiciones son verdaderas.
- | | (OR) verdadero si al menos una condición es verdadera.
- ! (NOT) invierte el valor booleano.

OPERADOR	NOMBRE	EJEMPLO	DEVUELVE VERDADERO CUANDO...
&&	y	(7 > 2) && (2 < 4)	las dos condiciones son verdaderas
||	o	(7 > 2) || (2 < 4)	al menos una de las condiciones es verdadera
!	no	!(7 > 2)	la condición es falsa

Estos se combinan con operadores relacionales. Ej: `if (cantidad > 100 && aplicaDescuento)`

verificará las dos condiciones antes de imprimir un mensaje.

Problema: Cálculo del Costo Total de un Pedido

Contexto: Ahora, el cliente “Sibelius” pregunta: si compran 10 unidades de Café Premium a un precio de $200 cada una, ¿cuál es el costo total?

Solución con Operadores Aritméticos:

TIP: Está bueno siempre entender el problema y detectar las palabras claves, para con ellas, crear las variables. En este caso las palabras clave son: Precio Unitario, Unidades/Cantidad, Costo total.

- Precio unitario: 200
- Cantidad: 10
- Costo total = precio unitario * cantidad

Ejemplo de Código:

```
int precioUnitario = 200;
int cantidad = 10;
int costoTotal = precioUnitario * cantidad;
System.out.println("El costo total del pedido es: $" + costoTotal);
```

Entrada y Salida de Datos

Para hacer más dinámica la aplicación, podemos solicitar datos al usuario. Esto se logra usando, por ejemplo, `Scanner` para leer datos desde la consola.

Para personalizar la experiencia, necesitamos pedir datos al usuario. La clase `Scanner` nos lo permite. Esto es clave para, por ejemplo, preguntar el nombre del cliente o la cantidad que desea comprar.

Uso de Scanner:

Importar la clase: `import java.util.Scanner;`

Crear una instancia:
`Scanner sc = new Scanner(System.in);`

Métodos comunes:

- `nextLine()`: Lee una línea completa de texto (ideal para nombres).
- `nextInt()`: Lee un número entero.
- `nextDouble()`: Lee un número decimal.

Ejemplo (Obtener el nombre del cliente):

```
System.out.print("Por favor, ingresá tu nombre: ");

String nombreCliente = sc.nextLine();

System.out.println("¡Bienvenide a nuestra tienda, " + nombreCliente +
"!");
```

Ejemplo (Obtener la cantidad de productos que quiere el cliente):

```
System.out.print("¿Cuántos productos querés comprar? ");

int cantidadCliente = sc.nextInt();

System.out.println("Vas a comprar " + cantidadCliente + " productos.");
```

Esto hace la aplicación más interactiva, adaptándose a diferentes escenarios sin cambiar el código fuente.

Problema: Obtener Datos del Cliente

Contexto: Queremos personalizar la experiencia. Le pedimos al cliente que ingrese su nombre para saludarle en pantalla.

Solución con Entrada/Salida de Datos:

- Usamos `Scanner` para leer el nombre por consola.
- Mostramos un saludo personalizado.

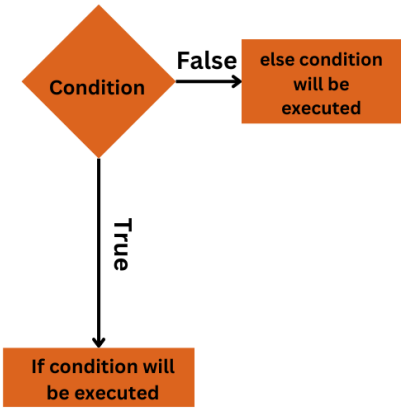
Ejemplo de Código:

```
import java.util.Scanner;

Scanner sc = new Scanner(System.in);
System.out.print("Por favor, ingresá tu nombre: ");
String nombreCliente = sc.nextLine();
System.out.println("¡Bienvenide a nuestra tienda, " + nombreCliente +
"!");
```

Estructuras de Control: Condicionales (if, else if, else)

Estas estructuras nos permiten tomar decisiones dependiendo de si se cumplen ciertas condiciones.



If-Else Statement

Problema: Verificación de Stock vs. Demanda

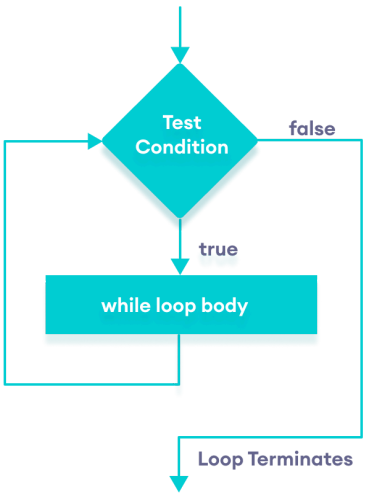
Contexto: Silvia pide verificar si el stock actual del Café Premium alcanza para cubrir una demanda de 120 unidades. Si el stock es suficiente, avisamos que podemos procesar el pedido; si no, informamos que no alcanza.

Solución con Condicionales:

```
int demanda = 120;
if (stockCafePremium >= demanda) {
   System.out.println("Podemos cubrir el pedido de " + demanda + "
unidades.");
} else {
   System.out.println("No hay stock suficiente para cubrir el
pedido.");
}
```

Estructuras de Control: Bucles (for, while)

Los bucles nos permiten repetir acciones sin escribir el mismo código muchas veces.



Buenos Aires aprende
Agencia de Habilidades para el Futuro

Problema: Listado de Productos Pendientes (while vs. for)

Contexto: Matías y Sabrina (Los devs de TechLab) deben revisar una lista de productos pendientes. Primero lo intentan con un `while`, y luego se dan cuenta de que un `for` sería más adecuado ya que conocen la cantidad exacta de productos a revisar.

Solución con `while`:

Supongamos que hay 5 productos por revisar. Usaremos un `while` para procesarlos uno a uno.

```
int productosPendientes = 5;

int contador = 1;

while (contador <= productosPendientes) {
   System.out.println("Revisando producto número: " + contador);
   contador++;
}
```

Solución con `for`:

Como sabemos exactamente cuántos son, un `for` es más claro:

```
for (int i = 1; i <= productosPendientes; i++) {
   System.out.println("Revisando producto número: " + i);
}
```

Ambas soluciones funcionan, pero el `for` se adapta mejor cuando conocemos de antemano la cantidad de iteraciones necesarias.

Materiales y Recursos Adicionales

- Documentación Oficial de Java
- Git - Libro Oficial
- Videos Recomendados:
    - "Fundamentos de Programación en Java" en YouTube.

Preguntas para Reflexionar

- ¿Cómo ayudan las variables a adaptarnos a cambios en el escenario de la empresa (cantidad de productos, precios, etc.)?
- ¿En qué situaciones preferirías un `while` sobre un `for` y viceversa?
- ¿De qué manera la entrada de datos del usuario puede mejorar la interacción con el sistema desarrollado?

Próximos Pasos

En la próxima clase profundizaremos en la organización del código, el uso de métodos y la modularización, para que nuestras soluciones sean más limpias y fáciles de mantener. Mientras tanto, seguí practicando los fundamentos aprendidos hoy.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Situación Inicial en Talento Lab.

Talento Lab

En **Talento Lab**, Silvia (Product Owner) ha recibido nuevas solicitudes del cliente “Sibelius”. El equipo de desarrollo (Matías y Sabrina) se enfrenta a varios pequeños problemas cotidianos del proyecto de e-commerce. Cada uno de estos problemas se resolverá aplicando un concepto de programación:

Los problemas que enfrentaremos en TechLab esta semana son los siguientes:

- Necesitamos llevar un registro del stock de productos.
- El cliente quiere saber el costo total de un pedido sumando el precio unitario por la cantidad de productos.
- Requerimos solicitar datos al cliente para personalizar la experiencia.
- Debemos verificar si tenemos suficiente stock para cubrir una demanda específica.
- Hay que procesar una lista de productos pendientes de revisión. Primero intentaremos una solución con `while` y luego notaremos que `for` podría ser más elegante.

Ejercicios prácticos.

Silvia

Product Owner

Silvia (Product Owner) ha recibido nuevas solicitudes del cliente:

1. Necesita saber el costo total de un pedido sumando el precio unitario por la cantidad de productos.
2. Requerimos solicitar datos al cliente para personalizar la experiencia.

3. Debemos verificar si tenemos suficiente stock para cubrir una demanda específica.
4. Hay que procesar una lista de productos pendientes de revisión. Primero intentaremos una solución con `while` y luego notaremos que `for` podría ser más elegante.

Para cumplir con necesidades el equipo de desarrollo (Matías, Sabrina) y vos deberán:

1) **Variables y Operadores:**
   - Crear variables para representar el precio de un producto y la cantidad deseada por el cliente. Calculá y mostrale en pantalla el costo total.
   - Modificá el precio o la cantidad y verificá el resultado.
2) **Entrada y Salida de Datos:**
   - Pedile al usuario que ingrese su nombre y la cantidad de productos que quiere comprar.
   - Mostrá un mensaje personalizado con el monto total (asignando un precio fijo por unidad).
3) **Condicionales:**
   - Suponé que si el cliente quiere más de 100 unidades, le ofrecemos un descuento.
   - Implementá un `if` que verifique si `cantidad > 100`. Si es así, mostrá un mensaje indicando que aplica un descuento especial.
4) **Bucles:**
   - Pedir al usuario o usuaria que ingrese un número, y luego usá un bucle `for` para imprimir desde 1 hasta ese número.
   - Repetí lo mismo con un `while` y compará cuál te resulta más intuitivo.

Buenos Aires Ciudad