Apunte actividad 3

Scanner: Lectura de datos del usuario

Scanner es una clase de Java que te permite leer datos que el usuario ingresa por la consola, como números, texto o cualquier otro tipo de información. Es la forma más común de interactuar con el usuario en programas de consola.

Para usar Scanner, debes seguir estos pasos:

1. **Importar la clase Scanner**: Debes agregar la línea `import java.util.Scanner;` al inicio de tu archivo Java, antes de la declaración de la clase.
2. **Crear un objeto Scanner**: Dentro del método `main`, debes crear una instancia de la clase Scanner. Se recomienda usar `System.in` como argumento para indicar que la entrada de datos provendrá del teclado.

```
import java.util.Scanner;

public class MiPrimerProyecto {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
```

3. **Leer los datos**: Utiliza los métodos del objeto Scanner para leer el tipo de dato que esperas. Algunos de los métodos más comunes son:
    - nextInt(): Para leer un número entero (int).
    - nextDouble(): Para leer un número decimal (double).
    - nextLine(): Para leer una línea de texto completa (String).
    - next(): Para leer una sola palabra (String).

Ejemplo:

```
Scanner sc = new Scanner(System.in);

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

System.out.print("Ingresa tu edad: ");
int edad = sc.nextInt();

System.out.println("Hola, " + nombre + ". Tienes " + edad + " años.");

sc.close(); // Siempre se debe cerrar el objeto Scanner
```

Importante: Cuando usas nextInt() o nextDouble() seguido de nextLine(), puede ocurrir un problema. El método nextInt() solo lee el número y deja un salto de línea (\n) en el búfer de entrada. La siguiente llamada a nextLine() lee este salto de línea vacío, por lo que parece que se "salteó" la entrada. Para evitarlo, puedes agregar un teclado.nextLine() extra después de leer el número para "limpiar" el búfer.

Concatenación de Datos

La **concatenación** es el proceso de unir o combinar dos o más cadenas de texto (String). En Java, se utiliza el operador **+** para concatenar. Si uno de los operandos es un String, el otro operando (sin importar su tipo de dato) se convertirá automáticamente a String y se unirá.

Esto es muy útil para construir mensajes dinámicos que incluyan variables.

Ejemplo:

```
String saludo = "Hola, ";
String nombre = "Ana";
String mensajeCompleto = saludo + nombre; // El resultado es "Hola, Ana"

int edad = 28;
String mensajeEdad = "Mi edad es " + edad; // El resultado es "Mi edad es 28"
```

Caracteres de Escape

Los **caracteres de escape** son combinaciones especiales de caracteres que empiezan con una barra invertida (\). Se usan dentro de una cadena de texto para representar caracteres que son difíciles de escribir o que tienen un significado especial.

Los más comunes son:

Carácter de Escape	Descripción	Ejemplo de Uso
\n	**Salto de línea**	System.out.print("Línea 1\nLínea 2");
\t	**Tabulación**	System.out.print("Nombre:\tJuan");
\\	**Barra invertida**	System.out.print("C:\\Programas");
\"	**Comillas dobles**	System.out.print("Dijo: \"Hola\"");
\'	**Comillas simples**	System.out.print("Caracter: \'A\'");

Ejemplo:

```
System.out.println("Este es un texto con un salto de línea.\nEl resto del texto está en otra línea.");
```

Operadores Aritméticos

Los **operadores aritméticos** se utilizan para realizar operaciones matemáticas. En Java, los operadores básicos son:

Operador	Nombre	Ejemplo
+	**Suma**	5 + 3
-	**Resta**	10 - 4
*	**Multiplicación**	2 * 6
/	**División**	10 / 2
%	**Módulo**	7 % 3 (El resto de la división, que es 1)

Ejemplo:

```
int a = 15;
int b = 4;

int suma = a + b;         // 19
int resta = a - b;        // 11
int multiplicacion = a * b; // 60
double division = (double) a / b; // 3.75.
int modulo = a % b;       // 3
```