Práctica 02: Fundamentos y sintaxis del lenguaje.

I. Introducción

La presente práctica tiene como finalidad reforzar los fundamentos de la programación orientada a objetos mediante el uso de variables, constantes, expresiones y estructuras de control de flujo.

Estos elementos constituyen la base para el desarrollo de programas más complejos, ya que permiten representar datos, realizar operaciones y establecer la lógica necesaria para la ejecución ordenada de instrucciones.

Asimismo, se retoman aspectos vistos en prácticas anteriores, como la importación de proyectos en el entorno de NetBeans, junto con el uso de herramientas de apoyo como las abreviaturas y los *snippets*, que facilitan la escritura del código al automatizar fragmentos comunes de programación.

Con ello, se busca que se adquiera mayor fluidez en el uso del lenguaje, fortaleciendo la comprensión de los principios que guían el diseño y la implementación de programas orientados a objetos.

II. Objetivo

Crear programas que implementen variables y constantes de diferentes tipos de datos, expresiones y estructuras de control de flujo.

III. Actividades

- Crear variables y constantes de diferentes tipos de datos.
- Crear diversas expresiones (operadores, declaraciones, etc.).
- Implementar estructuras de control de flujo.

IV. Desarrollo de Actividades

Proyecto P01 - Programación Orientada a Objetos en Java

Teoría Básica

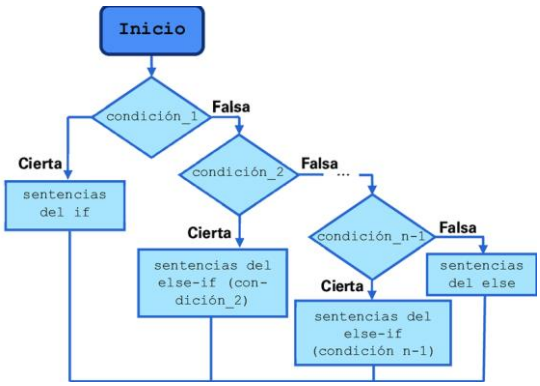
En Java, los tipos de datos primitivos (como int, double, char, boolean) permiten almacenar valores básicos. A partir de ellos se construyen expresiones y se implementan estructuras que controlan la ejecución del programa.
Algunos conceptos aplicados fueron:

Declaración e inicialización de variables: permiten almacenar valores que se usan en cálculos o condiciones.

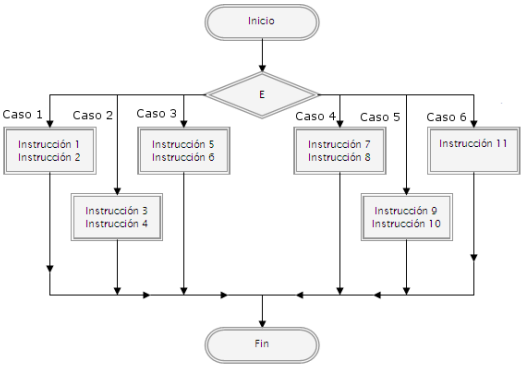
Constantes: valores fijos que no cambian durante la ejecución del programa (aunque en este caso no se usaron explícitamente, se podrían haber definido con final).

Estructuras de control de flujo: Son instrucciones que permiten decidir y repetir acciones:

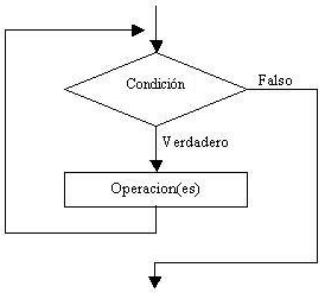
if, else if, else



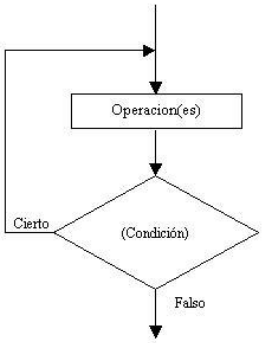
switch



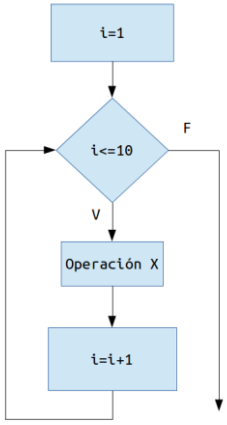
ciclos: while



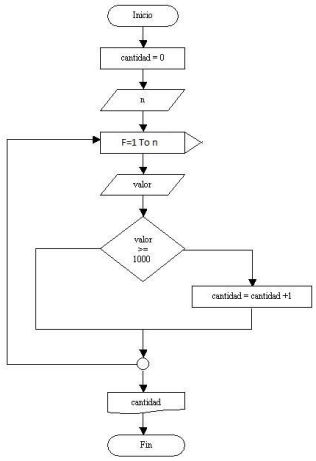
do-while



for



For – each



Métodos: bloques de código que realizan una tarea específica, como el método menor(int a, int b) que compara dos valores.

Arreglos: estructuras que permiten almacenar múltiples elementos bajo un mismo nombre.

Notas sobre el desarrollo de la práctica

- Se comenzó con un programa sencillo que imprime mensajes en consola.
- Se implementó una comparación entre dos enteros (a y b) usando la estructura condicional if-else.
- Se creó un método externo (menor) para ejemplificar cómo encapsular lógica en funciones.
- Se aplicó la instrucción switch para imprimir días de la semana según el valor de una variable.
- Se trabajaron diferentes tipos de ciclos (while, do-while, for, for-each) para recorrer números y un arreglo.
- Finalmente, se realizó una pequeña actividad de práctica simulando la validación de edad para ingresar a un lugar.

Estructura del archivo

- El programa se encuentra dentro del paquete POOP2 y consta de:
- Clase principal POOP2: contiene el método main, punto de entrada del programa.
- Método menor(int a, int b): retorna true si a es menor que b.
- Uso de instrucciones de salida (System.out.println) para mostrar resultados.
- Declaración de variables locales (a, b, n, dia, edad, arreglo).
- Bloques de control de flujo (if-else, switch, ciclos).

V. CÓDIGO FUENTE

CÓDIGO *POOP02.java*

Hola mundo

```
public class POOP2 {
   public static void main(String[] args) {
      System.out.println("Hola mundo");
      System.out.println("Hola mundo abreviado");
      System.out.println("############################");
```

If – else if

```
      int a;
      a=5;
      int b;
      b=3;
      if (a<b) {
         System.out.println("a es menor que b");
      } else if (a==b){
         System.out.println("a es igual que b");
      } else{
         System.out.println("a es mayor que b");
      }
```

if en function /metodo

```
      System.out.println("########################## if en function
/metodo##################");
      if(menor(a,b)){
         System.out.println("metodo retorna true");
      } else {
         System.out.println("metodo retorna false");
      }
```

switch

```
System.out.println("####switch####");
int dia = 1;
switch(dia) {
case 1:
System.out.println("Lunes");
break;
case 2:
System.out.println("Martes");
break;
case 3:
System.out.println("Miercoles");
break;
case 4:
System.out.println("Jueves");
break;
case 5:
System.out.println("Viernes");
break;
case 6:
System.out.println("Sabado");
break;
case 7:
System.out.println("Domingo");
break;
default:
System.out.println("No es un dia de la semana");
}
```

While

```
System.out.println("### While###");
int n= 0;
while (n<10){
System.out.println("Contando hacia rriba n=" +n);
```

```
n++; // n=n+1
}
do {
System.out.println("Contando hacia abajo n=" +n);
n--; //n=n-1
} while (n>0);
System.out.println("n=" +n);
```

for

```
System.out.println("for");
for (int i = 10; i < 0; i++) {
System.out.println("Contando hacia arriba i=" +i);
}
for (int i = 10; i > 0; i--) {
System.out.println("Contando hacia abajo i=" +i);
}

System.out.println("####for####");

int[] arreglo = {1,2,3,4,5};
int tamanio =arreglo.length;
System.out.println("tamanio");
for (int i = 0; i < arreglo.length; i++) {
System.out.println("arreglo["+i+"]="+arreglo[i]);

}
```

for-each

```
System.out.println("### for-each ###");
for(int temp:arreglo){
System.out.println("Elemento de arreglo=" +temp);
}
```

Actividad

```
System.out.println("### Actividad ###");
System.out.println("Que edad tienes?");
int edad = 18;
System.out.println("edad=" +edad);
if (edad<18) {
System.out.println("No Puedes ingresar a Sambuca");
} else {
System.out.println("Puedes ingresar a Sambuca");
}
}
private static boolean menor(int a, int b){
return a<b; }
}
```

VI. URL

GitHub: https://github.com/EdLuna237/POOP02
GitHub Pages: https://edluna237.github.io/POOP02/

VII. Conclusiones

La práctica desarrollada representó una oportunidad significativa para afianzar los fundamentos del lenguaje Java y comprender de manera más profunda cómo se integran variables, expresiones y estructuras de control de flujo dentro de un programa. A lo largo de la implementación se pudo observar que, aunque los conceptos básicos parecen sencillos, su correcta aplicación constituye la base para construir programas más sólidos y escalables.

El ejercicio con estructuras condicionales y ciclos permitió visualizar de forma clara cómo las decisiones y repeticiones influyen en el comportamiento del programa, generando diferentes resultados según las condiciones establecidas. De igual manera, el uso de métodos independientes, como el que compara dos valores, evidenció la importancia de organizar el código en bloques reutilizables que faciliten su comprensión y mantenimiento.

El trabajo con arreglos resultó particularmente útil para comprender cómo almacenar y manipular conjuntos de datos de manera eficiente, mostrando que las estructuras de datos no solo representan un recurso técnico, sino también una forma de pensar la programación de manera más ordenada y sistemática.

Finalmente, el empleo de NetBeans como entorno de desarrollo ofreció un apoyo esencial, no solo por su capacidad para compilar y ejecutar los programas, sino también por las herramientas que optimizan el trabajo del estudiante, como las abreviaturas y los snippets. En lo personal, esta práctica me permitió darme cuenta de que el aprendizaje de la programación requiere paciencia, constancia y una disposición a ensayar diferentes soluciones, ya que cada error se convierte en una oportunidad de reforzar la lógica y ganar seguridad en el propio proceso de aprendizaje.