Manejo básico de flujo

Manejo básico de flujo **1**
¿Qué aprenderás? 2
Introducción 2
Recordando diagrama de flujo 3
La instrucción IF 4
Ejemplo de IF 4
Operadores de comparación 5
Operadores de comparación en String 5
Método equals 7
Método equalsIgnoreCase 7
Método compareTo 8
Ejercicio guiado: Comparar dos cadenas de texto 10
Ejercicio propuesto (5) 10

¡Comencemos!

¿Qué aprenderás?

- Construir algoritmos que tomen decisiones en base al valor de una variable para controlar de mejor manera el flujo y la lógica del programa.
- Aplicar los operadores de comparación y lógicos para comprobar si se cumple una determinada condición.

Introducción

Una gran parte de los algoritmos más complejos, por no decir todos, necesitan tener la capacidad de decidir ante los valores de una o más variables de entrada, o bien en base al resultado de algún cálculo u otro parámetro a considerar dentro de un flujo. Necesitaremos decidir, por ejemplo, si el auto es de 1.4 cc o 1.6 cc y cuesta más de cierto valor, entonces compro uno u otro auto; si una persona es mayor de edad, entonces esta puede conducirlo, etc.

En este apartado, aprenderemos a utilizar los operadores de control del flujo para tomar decisiones en base a valores que necesitemos manejar dentro del programa.

¡Vamos con todo!

Recordando diagrama de flujo

Al inicio analizamos el siguiente diagrama de flujo. El rombo corresponde a una evaluación condicional. En este punto se toma una decisión y el programa continua su flujo en función de esa decisión. A continuación, aprenderemos a escribir programas que implementen estas decisiones.

Imagen 1. Diagrama de flujo para saber si la lámpara funciona.
Fuente: Wikipedia.

La instrucción IF

Java y muchos otros lenguajes de programación tienen instrucciones para implementar condiciones. La más utilizada es la instrucción IF.

```
if(condicion){
//código que se ejecutará solo si se cumple la condición
}
```

Lo anterior se lee como: ”Si se cumple la condición, entonces ejecuta el código”.

Ejemplo de IF

Analicemos el siguiente ejemplo: En muchos países de Latinoamérica, la mayoría de edad se cumple a los 18 años. Crearemos un programa que pregunte la edad al usuario. Si la edad es mayor o igual a 18, entonces le diremos que es mayor de edad.

```
System.out.println("¿Qué edad tienes?");
int edad = sc.nextInt();
if(edad >= 18) {
    System.out.println("Eres mayor de edad");
}
```

Si ejecutamos el programa e introducimos un valor mayor o igual a 18 veremos el mensaje ”Eres mayor de edad”, en caso contrario no veremos ningún mensaje. En este ejercicio comparamos utilizando el operador `>=`, pero existen varios operadores que nos permiten comparar, estos los estudiaremos a continuación.

Operadores de comparación

Los operadores de comparación son aquellos que comparan dos valores y obtienen como resultado `true` (verdadero) o `false` (falso). A este tipo de objeto, resultado de una comparación, se le conoce como booleano.

Operador	Nombre	Ejemplo	Resultado
==	Igual a	2 == 2	true
!=	Distinto a	2 != 2	false
>	Mayor a	3 > 4	false
>=	Mayor o Igual a	4 >= 3	true
<	Menor a	3 < 4	true
<=	Menor o Igual a	4 <= 4	true

Tabla 1. Operadores de comparación en números.
Fuente: Desafío Latam.

Realicemos una prueba donde el usuario ingrese los valores y veamos si el primero es mayor que el segundo.

```
int a = sc.nextInt(); //1
int b = sc.nextInt(); //6
System.out.println(a > b); //false
```

Operadores de comparación en String

Aunque en la tabla solo hayamos mostrado números, podemos comparar dos objetos utilizando un operador de comparación.

```
String a = "texto 1";
String b = "texto 2":
System.out.println(a == b); //false
```

¿Puede ser un texto mayor que otro?

La respuesta es No. Los objetos (como lo es los Strings) no tienen implementado el operador `<` o `>`. Si tratamos de compilar esto, la consola nos dirá:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The operator < is undefined for the argument type(s) java.lang.String,
java.lang.String
at
MiPrimerPrograma/miprimerprograma.MiPrimerPrograma.main(MiPrimerPrograma.ja
va:1
1)
```

Para comparar variables de tipo String, tenemos básicamente tres métodos de la clase String que nos permiten evaluar o hacer comparaciones entre dos String. Las más utilizadas son las siguientes:

- equals.
- equalsIgnoreCase.
- compareTo.

Método equals

El método equals permite evaluar si dos objetos String son o no iguales. De ser iguales, este devuelve `true` o de lo contrario devuelve `false`:

```
String s1 = "Hola"; //s1, objeto String que contiene la cadena Hola
String s2 = "Hola"; //s2, objeto String que contiene la cadena Hola

System.out.println(s1.equals(s2)); //true
```

Este método evalúa exactamente que los dos objetos, es decir, `s1` y `s2` sean iguales, por lo que si agregamos un mínimo cambio en una de los dos objetos, por ejemplo, una letra de minúscula a mayúscula, el resultado será `false`.

```
//Se agrega la última letra en mayúscula
String s1 = "HolA"; //s1, objeto String que contiene la cadena HolA
String s2 = "Hola"; //s2, objeto String que contiene la cadena Hola

System.out.println(s1.equals(s2)); //false
```

Método equalsIgnoreCase

El método equalsIgnoreCase es similar a equals, con la diferencia de que éste ignora las letras mayúsculas y minúsculas, por lo que el ejemplo anterior, utilizando este método, dará como resultado `true`.

```
//Se agrega la última letra en mayúscula
String s1 = "HolA"; //s1, objeto String que contiene la cadena HolA
String s2 = "Hola"; //s2, objeto String que contiene la cadena Hola

System.out.println(s1.equalsIgnoreCase(s2)); //true
```

Método compareTo

Este método permite comparar dos objetos de tipo String y dar como resultado si son iguales, si uno es menor que otro o bien cuál de los dos es mayor.

```
String uno = "Hola";
String dos = "Chao";
if(uno.compareTo(dos) < 0) {
    System.out.println("uno es menor");
}
```

compareTo retorna:

- 0 si los String son iguales
- -1 si el primero es menor
- 1 si el primero es mayor

No podemos decir si un string es mayor en realidad que otro, sino que lo que se compara es carácter a carácter, en los caracteres ASCII.

Como referencia, el código ASCII es un estándar que asocia un carácter a un valor numérico. Si deseas conocer más información acerca del código ASCII, revisa el documento **Código ASCII** ubicado en “Material Complementario”.

Ahora que ya conocemos los operadores de comparación, volvamos a nuestro código anterior y analicemos el diagrama de flujo.

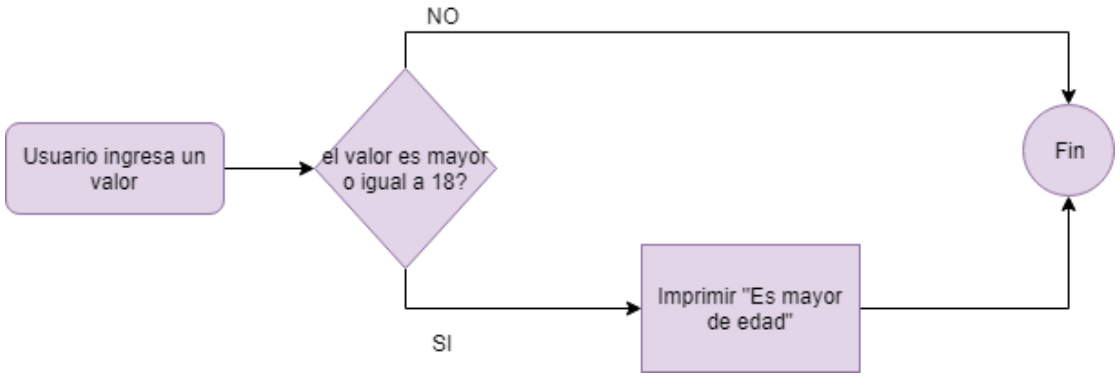

Imagen 2. Diagrama de flujo para una condición IF.
Fuente: Desafío Latam.

Importante:

- Las llaves después de la condición deben estar siempre. Solo se pueden omitir si las líneas a ejecutar dentro de condición es solo una.
- Todo el código que esté dentro de esas llaves sucederá solo si se cumple la condición.
- Es recomendable identar el código para reconocer de forma sencilla y visual dentro del código: donde empieza, termina y que código se ejecuta dentro de la condición.

Ejercicio guiado: Comparar dos cadenas de texto

Requerimientos:

- Comparar dos cadenas de texto.
- Si son iguales imprimimos en pantalla que ambas cadenas de texto son iguales.
- Si no, imprimimos que son distintas.

Lo primero que debemos hacer es declarar las variables que almacenarán estas cadenas de texto:

```
String cadenaUno = "Bienvenido a Desafío Latam";
String cadenaDos = "Bienvenidos a Desafío Latam";
```

Luego tenemos que comparar ambas cadenas e imprimir el texto según se cumpla la condición.

```
if(cadenaUno.equals(cadenaDos)){
  System.out.println("Las cadenas son iguales");
}else{
  System.out.println("Las cadenas no son iguales");
}
```