

Flujo (Parte II)

Manejo básico de flujo

Competencias

- 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.

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.java: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 son 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 = "HoIA"; //s1, objeto String que contiene la cadena HoIA
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 = "HoIA"; //s1, objeto String que contiene la cadena HoIA
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.

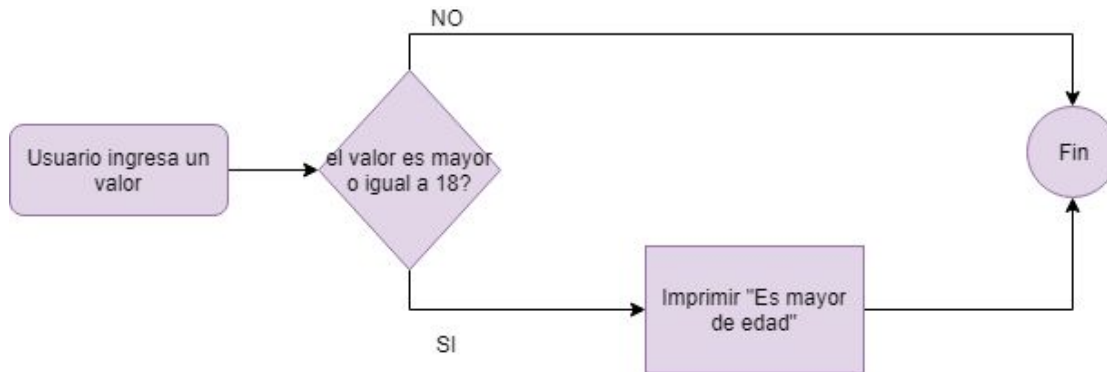


Imagen 2. Diagrama de flujo para una condición IF.
Fuente: Desafío Latam.

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 indentar 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");  
}
```

Ejercicio propuesto (1)

Requerimientos:

- Declarar dos cadenas de texto.
- Comparar las cadenas de texto con equals, y mostrar un mensaje en pantalla según sea el caso.
- Comparar las cadenas de texto con compareTo y ver cuál es mayor, menor o iguales.

Operadores de control

Competencias

- Hacer uso de condiciones para evaluar los datos contenidos en variables.
- Resolver algoritmos de una forma más eficaz mediante las estructura de control de flujo.

Introducción

Ya hemos visto cómo utilizar los operadores lógicos o de decisión para evaluar ciertos datos que contienen nuestras variables o bien objetos, como por ejemplo el objeto String.

Los operadores de control del flujo son aquellos que nos permitirán decidir si el flujo debe ir por un lado o por otro, y así hacer cálculos u otro tipo de lógica en base a cómo va avanzando nuestro algoritmo.

Estructura IF

“SI” o “IF” en inglés permite controlar la ejecución de una o varias instrucciones si y sólo si se cumple la condición **if**. Por ejemplo, el siguiente pseudocódigo permitirá imprimir por pantalla qué número es mayor a otro.

EjemploIF

```
leer numero1;
leer numero2;

si numero1 > numero2 entonces
    imprimir "Número 1 es mayor a número 2"
fin si
si numero1 < numero2 entonces
    imprimir "Número 2 es mayor a número 1"
fin si

si numero1 == numero2 entonces
    imprimir "Los números son iguales"
fin si
```

Fin EjemploIF

Estructura IF: Sintaxis

La sintaxis de una condición IF en Java es la siguiente:

```
if(condicion){
    //Instrucciones
}
```

Estructura IF-ELSE

IF-ELSE (Si o de lo contrario) permite crear una condición que ejecuta una serie de instrucciones en el caso de que se cumpla la condición y en el caso de que NO se cumpla la condición. El siguiente pseudocódigo evalúa si el usuario ingresado es "Pepito", si no es dará un mensaje "USUARIO NO PERMITIDO".

```
Inicio EjemploUsuario
  leer usuario;

  si usuario es igual a "Pepito" entonces
    imprimir "Acceso permitido";
  de lo contrario
    imprimir "ACCESO NO PERMITIDO";
  fin si
Fin EjemploUsuario
```

El ejemplo anterior en diagrama de flujo sería el siguiente:

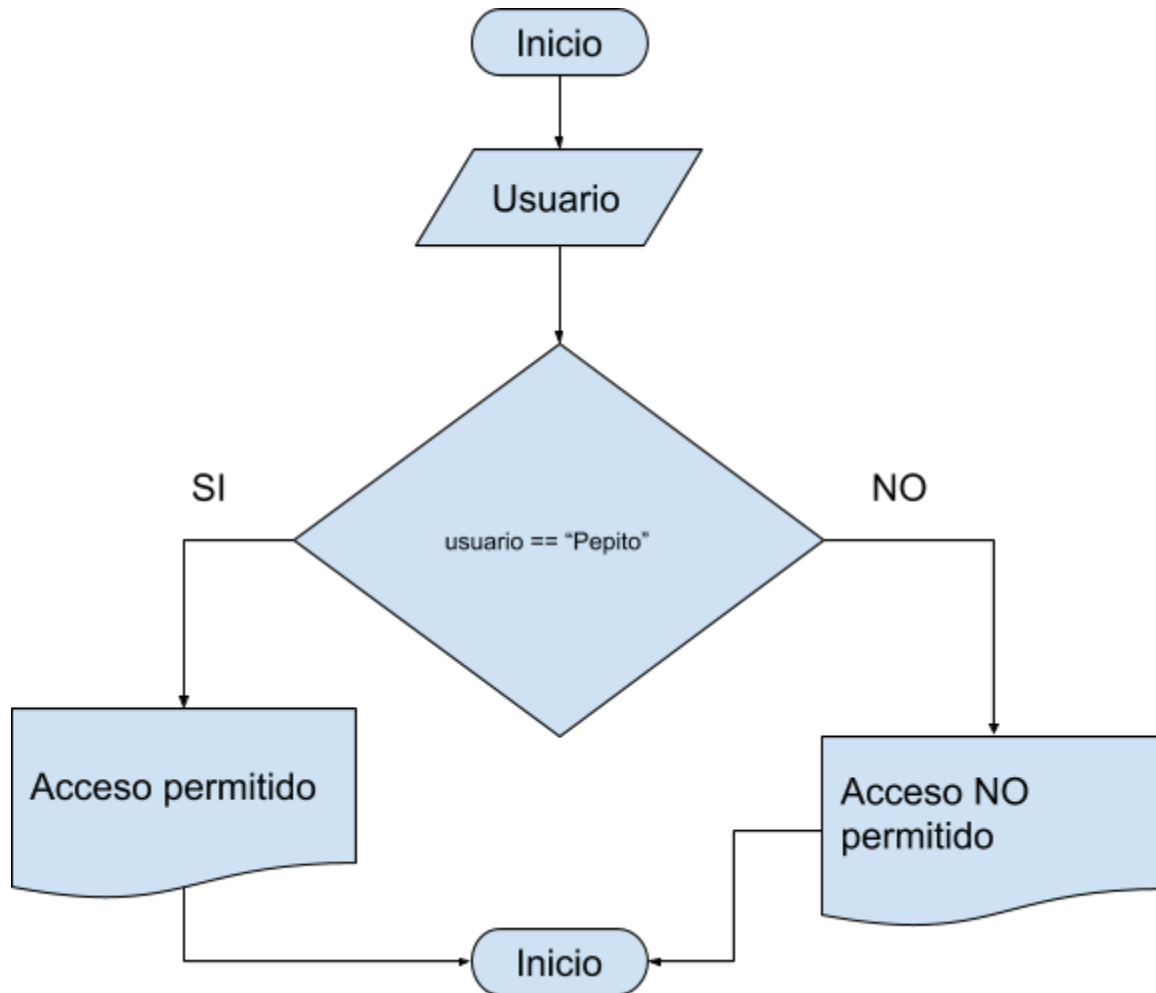


Imagen 3. Diagrama de flujo condicional IF.
Fuente: Desafío Latam.

IF-ELSE: Sintaxis

La sintaxis de una condición IF-ELSE en Java es la siguiente:

```
if(condicion){  
    //Instrucciones  
}else{  
    //Instrucciones  
}
```

Estructura IF-ELSE-IF

En ocasiones es necesario hacer varias condiciones para evaluar diferentes opciones a seguir dentro del flujo. Para esto, podemos utilizar condiciones anidadas como se muestra en el siguiente ejemplo escrito en pseudocódigo:

```
Inicio EjemploIfAnidado  
    leer opcion;  
    si opcion == 1 entonces  
        leer valor1;  
        leer valor2;  
        imprimir valor1 + valor2;  
    de lo contrario si opcion == 2{  
        leer valor1;  
        leer valor2;  
        imprimir valor1 - valor2;  
    de lo contrario  
        imprimir "Opción ingresada no es válida";  
    fin si;  
Fin EjemploIfAnidado
```

El ejemplo anterior, plantea un pseudocódigo que permite ingresar dos opciones. 1 para sumar dos valores o 2 para restarlos. Si la opción no es 1 ni 2 entonces se imprimirá por pantalla el mensaje "Opción ingresada no es válida".

Estructura IF-ELSE-IF: Sintaxis

La sintaxis de una condición IF-ELSE-IF en Java es la siguiente:

```
if(condicion){  
    //Instrucciones  
}else if(condicion){  
    //Instrucciones  
}else{  
    //Instrucciones  
}
```

Ejercicio guiado: Estado de avance de un proyecto

Crear un programa en Java que permita conocer el estado de avance de un proyecto en base al porcentaje de avance indicado.

Para esto vamos a crear una variable de tipo `int` con el porcentaje de avance del proyecto.

```
int porcentaje = 50;
```

Y luego de esto, definiremos el mensaje de salida según sea el caso:

Definimos los 3 mensajes de salida:

- Si el porcentaje es 0, el estado será: "Pendiente".
- Si el porcentaje es mayor a 0 y menor o igual a 99, el estado será "En proceso".
- Si el porcentaje es 100, el estado será "Terminado".

Ahora lo definimos en código quedando de la siguiente manera:

```
int porcentaje = 50;

if (porcentaje == 0) {
    System.out.println("Pendiente");
} else if (porcentaje > 0 && porcentaje <= 99){
    System.out.println("En proceso");
} else {
    System.out.println("Terminado");
}
```


Ejercicio propuesto (2)

Siguiendo con los ejercicios propuestos de la primera parte debemos calcular el promedio final de un estudiante, con este promedio le indicaremos al estudiante si aprueba o reprueba.

Los requerimientos son:

- Si el promedio es menor a 4, se debe mostrar el mensaje: "Reprobado".
- Si el promedio es mayor a 4 y 5.5 se debe mostrar el mensaje: "Aprobado".

```
package calificaciones;
import Java.util.Scanner;
public class Calificaciones{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String estudiante = sc.nextLine();
        String asignatura = sc.nextLine();
        String docente = sc.nextLine();
        int nota1 = sc.nextInt();
        int nota2 = sc.nextInt();
        int nota3 = sc.nextInt();

        float promedio = (nota1 + nota2 + nota3) / 3;

        System.out.printf(
            "Estudiante:%s %s\n"
            + "Asignatura: %s\n"
            + "Docente: %s\n"
            + "Nota 1: %d\n",
            + "Nota 2: %d\n",
            + "Nota 3: %d\n",
            + "Promedio: %d\n",
            estudiante, asignatura, docente, nota1, nota2, nota3, promedio);
    }
}
```

Operadores lógicos

Competencias

- Hacer uso de los operadores lógicos para evaluar y simplificar expresiones.

Introducción

Las operaciones lógicas son todas aquellas que tienen como resultado dos caminos, es decir, si son verdaderas o falsas. Los operadores lógicos nos permitirán aprender cómo implementar una pregunta lógica que debiera cumplirse o no, es decir, ser verdadera o no.

Simbología y conceptos

Para resolver una operación lógica, es decir, que su resultado será verdadero o falso, debemos conocer cuáles son los operadores lógicos que podemos implementar. El concepto de cada uno se explica en la siguiente tabla:

Operador	Nombre	Ejemplo	Resultado
&&	Y (and)	false && true	Devuelve true si ambos operadores son true, o ambos false al mismo tiempo. En este ejemplo es false
	O (or)	false true	Devuelve true si al menos uno de los operadores es true, en este ejemplo es true.
!	NO (not)	!false	Devuelve lo opuesto al resultado de la evaluación, en este caso true.

Tabla 2. Simbología de operadores lógicos.

Fuente: Desafío Latam.

Observemos los siguientes ejemplos:

```
String nombre = "Carlos";  
String apellido = "Santana";  
boolean a;  
a = "Carlos".equals(nombre) && "Santana".equals(apellido);  
System.out.println(a); //true  
a = "Carlos".equals(nombre) && "Vives".equals(apellido);  
System.out.println(a); //false  
a = "Carlos".equals(nombre) || "Vives".equals(apellido);  
System.out.println(a); //true
```

Ejemplos operaciones lógicas

1. ¿Cuál es el resultado de las siguientes operaciones lógicas?
 - a. $(4 \leq 10) \ \&\& \ (4 > 3) \rightarrow \text{true}$
 - b. $!(4 \leq 10) \rightarrow \text{false}$
 - c. $!((4 \leq 10) \ \&\& \ (4 > 3)) \rightarrow \text{false}$
 - d. $!((4 \leq 10) \ || \ (4 > 3)) \rightarrow \text{false}$
 - e. $\text{true} \ || \ \text{false} \rightarrow \text{true}$
 - f. $1 == 1 = \text{true}$
 - g. $(1 == 1) \ \&\& \ (2 != 0) \ || \ (2 \leq 3) \rightarrow \text{true}$

Ejercicio guiado: Cálculo del IMC (Índice de Masa Corporal)

Crear un programa que permita calcular el IMC estándar de una persona adulta. La fórmula para el cálculo es el siguiente:

$$IMC = \frac{\text{peso (Kg)}}{\text{estatura (m)}^2}$$

Requerimiento 1: La persona que utilizará el programa debe ser mayor a 17 años. Por lo que el programa debe preguntar la edad del usuario.

Requerimiento 2: Si la persona es mayor de edad, se le debe preguntar al usuario cuál es su estatura en metros y su peso en kilogramos.

Requerimiento 3: El resultado debe ser en base a la siguiente tabla:

IMC	Nivel de peso
Por debajo de 18.5	Bajo peso
18.5 – 24.9	Normal
25.0 – 29.9	Sobrepeso
30.0 o más	Obeso

Tabla 3. Interpretación del IMC para adultos.
Fuente: Desafío Latam.

Solución ejercicio guiado: Cálculo del IMC (Índice de Masa Corporal)

Requerimiento 1: Después de crear el proyecto y la clase main, se agrega la lectura de datos.

```
//Se pregunta al usuario cuál es su edad
Scanner sc = new Scanner(System.in);
System.out.println("Ingrese su edad: ");
//Se guarda la edad en una variable entera
int edad = sc.nextInt();
//Se evalúa si la edad ingresada es mayor a 17 años
if(edad >= 18) {

}else {
    System.out.println("Usted es menor de edad");
}
```

Requerimiento 2: Se agregan las siguientes preguntas y lectura de datos en el caso de que la persona o usuario que está operando sea mayor de edad.

```
//Se pregunta al usuario cuál es su edad
Scanner sc = new Scanner(System.in);
System.out.println("Ingrese su edad: ");
//Se guarda la edad en una variable entera
int edad = sc.nextInt();
    //Se evalúa si la edad ingresada es mayor a 17 años
if(edad >= 18) {
    //Se consultan los datos para el cálculo
    System.out.println("Cuál es su estatura (metros): ");
    float estatura = sc.nextFloat();
    System.out.println("Cuál es su peso (kilogramos): ");
    float peso = sc.nextFloat();
}else {
    System.out.println("Usted es menor de edad");
}
```

Requerimiento 3: Se agrega el cálculo y se evalúa el estado de la persona. Para esto, al inicio además se declara la variable resultado.

```
//Se pregunta al usuario cuál es su edad
Scanner sc = new Scanner(System.in);
//Variable resultado
float resultado = 0f;
System.out.println("Ingrese su edad: ");
//Se guarda la edad en una variable entera
int edad = sc.nextInt();
//Se evalúa si la edad ingresada es mayor a 17 años
if(edad >= 18) {
    //Se consultan los datos para el cálculo
    System.out.println("Cuál es su estatura (metros): ");
    float estatura = sc.nextFloat();
    System.out.println("Cuál es su peso (kilogramos): ");
    float peso = sc.nextFloat();
    /*
     * Se hace el cálculo del IMC. Se utiliza la clase Math y su método
     * pow para elevar la estatura al cuadrado.
     */
    resultado = (float) (peso / Math.pow(estatura, 2));
}else {
    System.out.println("Usted es menor de edad");
}
//Se agregan las condiciones en base a la tabla
/*Resultado debe ser distinto de cero, para validar que el
 * cálculo si se realizó
 */
if(resultado < 18.5 && resultado != 0f) { //Bajo peso
    System.out.println("Usted esta bajo peso");
}else if(resultado >= 18.5 && resultado <= 24.9) { //Normal
    System.out.println("Usted esta Normal");
}else if(resultado >= 25.0 && resultado <= 29.9) { //Sobrepeso
    System.out.println("Usted esta sobrepeso");
}else if(resultado >= 30.0) { //Obeso
    System.out.println("Usted Obeso");
}
```

Cálculo del IMC (Índice de Masa Corporal) - Solución completa y resultados

Código completo.

```
package cl.desafiolatam;

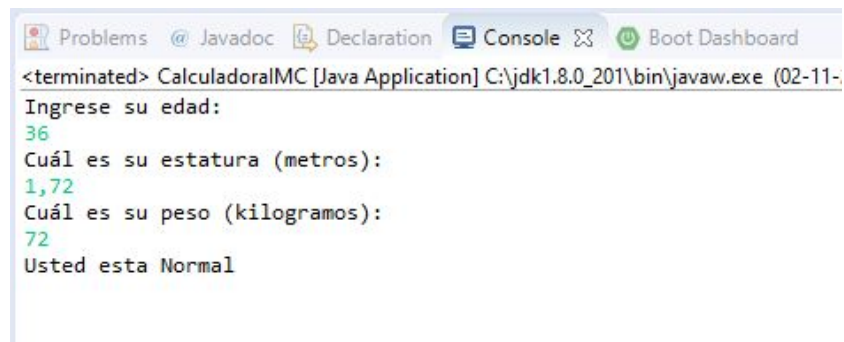
import java.util.Scanner;

public class CalculadoraIMC {

    public static void main(String[] args) {
        //Se pregunta al usuario cuál es su edad
        Scanner sc = new Scanner(System.in);
        //Variable resultado
        float resultado = 0f;
        System.out.println("Ingresa su edad: ");
        //Se guarda la edad en una variable entera
        int edad = sc.nextInt();
        //Se evalúa si la edad ingresada es mayor a 17 años
        if(edad >= 18) {
            //Se consultan los datos para el cálculo
            System.out.println("Cuál es su estatura (metros): ");
            float estatura = sc.nextFloat();
            System.out.println("Cuál es su peso (kilogramos): ");
            float peso = sc.nextFloat();
            /*
             * Se hace el cálculo del IMC. Se utiliza la clase Math y
             * su método
             * pow para elevar la estatura al cuadrado.
             */
            resultado = (float) (peso / Math.pow(estatura, 2));
        }else {
            System.out.println("Usted es menor de edad");
        }
        //Se agregan las condiciones en base a la tabla
        /*Resultado debe ser distinto de cero, para validar que el
        cálculo si se realizó*/
        if(resultado < 18.5 && resultado != 0f) { //Bajo peso
```

```
        System.out.println("Usted esta bajo peso");  
    }else if(resultado >= 18.5 && resultado <= 24.9) { //Normal  
        System.out.println("Usted esta Normal");  
    }else if(resultado >= 25.0 && resultado <= 29.9) { //Sobrepeso  
        System.out.println("Usted esta sobrepeso");  
    }else if(resultado >= 30.0) { //Obeso  
        System.out.println("Usted Obeso");  
    }  
    }  
}
```

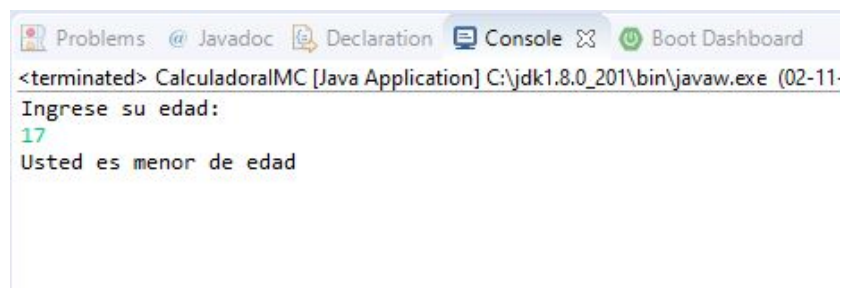
Resultado 1:



```
<terminated> CalculadoraIMC [Java Application] C:\jdk1.8.0_201\bin\javaw.exe (02-11-  
Ingrese su edad:  
36  
Cuál es su estatura (metros):  
1,72  
Cuál es su peso (kilogramos):  
72  
Usted esta Normal
```

Imagen 4: Resultado para una edad válida.
Fuente: Desafío Latam.

Resultado 2:



```
<terminated> CalculadoraIMC [Java Application] C:\jdk1.8.0_201\bin\javaw.exe (02-11-  
Ingrese su edad:  
17  
Usted es menor de edad
```

Imagen 5: Resultado para una edad inválida.
Fuente: Desafío Latam.

Ejercicio propuesto (3)

Siguiendo con los ejercicios propuestos de la primera parte debemos calcular el promedio final de un estudiante, con este promedio le indicaremos al estudiante si debe rendir una prueba recuperatoria, si debe rendir el examen final o si está eximido.

Requerimientos:

- Si el promedio es menor a 4, se debe mostrar el mensaje: "Debes dar una prueba de recuperación para poder rendir el examen final".
- Si el promedio es entre 4 y 5.5 se debe mostrar el mensaje: "Debes rendir el examen final para aprobar".
- Si el promedio es mayor a 5.5 debe mostrar: "Felicitaciones! Te eximiste del examen y aprobaste".

Ciclos y sentencias repetitivas

Competencias

- Comprender la lógica de las sentencias repetitivas

Introducción

Las sentencias repetitivas nos permiten, en resumen, ejecutar una cantidad de líneas de códigos tantas veces hasta que se cumpla una condición. A esto le llamamos **bucles** en la jerga de la programación de software. A continuación, introduciremos la lógica de estas sentencias repetitivas mediante algoritmos de diagrama de flujo y ejemplificando mediante códigos sencillos en java.

Control de flujo y condiciones

En un programa computacional la mayoría de las veces, por no decir siempre, es necesario controlar el flujo del programa y evaluar condiciones, las cuales si se cumplen nos permitirán realizar ciertas acciones o procesos, tales como cálculos u operaciones varias. Por ejemplo, necesitamos crear un programa que evalúe identificar si un número es par. El programa debiese terminar si o solo si el usuario ingresa un número par, si no debiese volver a preguntar. La solución, diagrama de flujo y algoritmo en Java sería el siguiente:

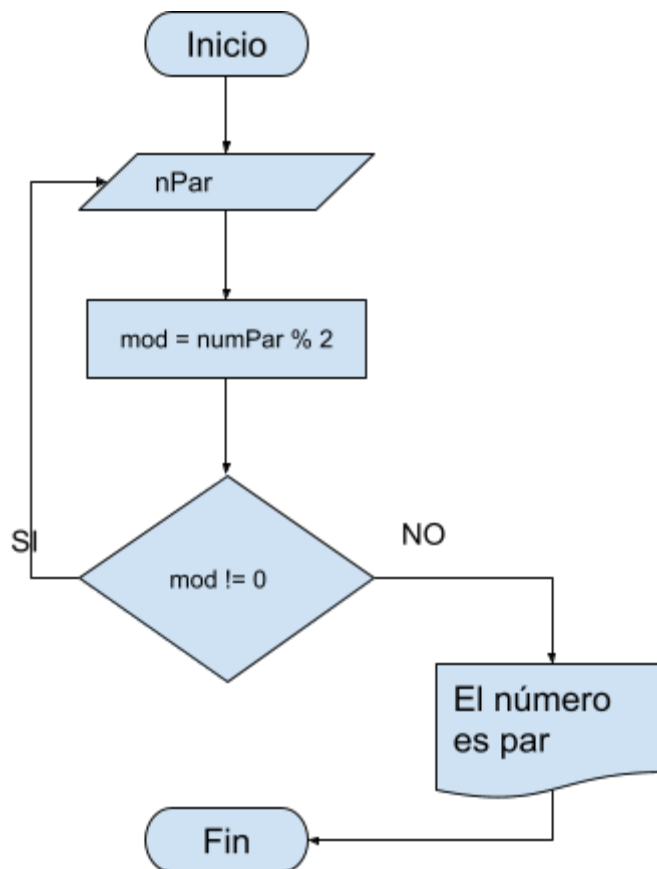


Imagen 6: Diagrama de flujos - Solución para ingresar un número par.
Fuente: Desafío Latam.

El diagrama de la imagen representa la solución para ingresar un número par. Nótese que en el caso que la condición se cumpla, el flujo del programa se devuelve al inicio, y este solo terminará si la condición no se cumple. Este bucle o sentencia repetitiva se llama DO-WHILE..

Por último, la solución en lenguaje Java del flujo anterior, sería como se muestra a continuación:

```
package cl.desafiolatam;
import java.util.Scanner;
public class MiPrimerPrograma {
    public static void main(String[] args) {
        //Se declaran las variables de entrada de tipo double
        double nPar = 0;
        double mod = 0;
        /*Ciclo DO-WHILE, bucle que no terminará hasta que el usuario
ingrese un número par */
        do{
            System.out.printf("Ingresa un número: ");
            Scanner sc = new Scanner(System.in);
            nPar = sc.nextLong();
            mod = nPar % 2;
        }while(mod != 0);

        System.out.println("El número es par");
    }
}
```

Bucles y representación en diagramas de flujos

Como se mencionaba antes, un bucle es una sentencia que permite repetir una serie de instrucciones hasta que se cumpla una condición. En este ámbito, tenemos tres tipos de sentencias que vamos a estudiar:

- Bucle For
- Bucle While
- Bucle Do While

Bucle FOR

El bucle FOR es aquel que nos permitirá repetir las sentencias desde y hasta un número determinado. En este caso, este bucle siempre se apoya de un contador.

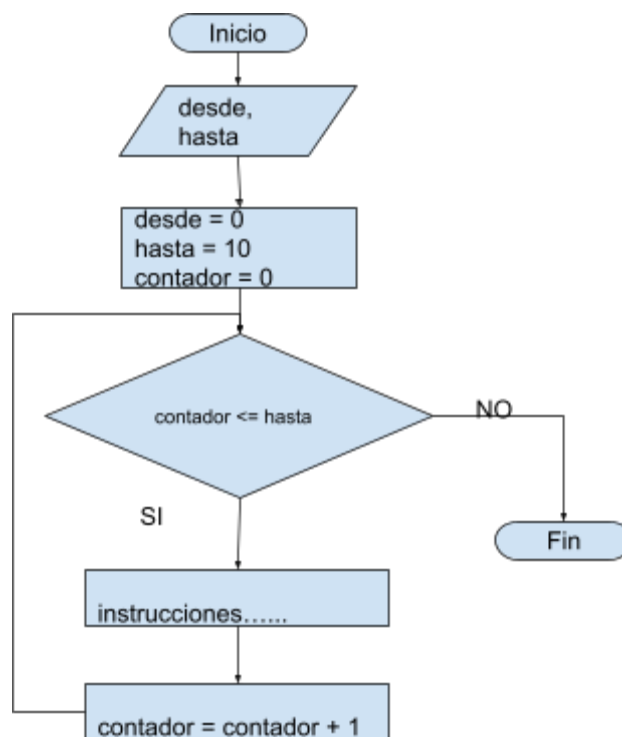


Imagen 7: Diagrama de flujo bucle FOR.

Fuente: Desafío Latam.

En el flujo anterior, se declara una variable que indica desde dónde vamos a contar, otra que indica hasta donde o cuantas veces se repetirán las instrucciones y por último, un contador que cuenta cada una de las repeticiones. Para efecto del flujo, este repetirá desde 0 hasta 10 veces las instrucciones.

Bucle While

El bucle While es otra sentencia repetitiva que ejecuta una serie de instrucciones mientras se cumple una condición. En este caso, puede que este Bucle nunca ejecute las instrucciones ya que la condición puede que nunca se cumpla.

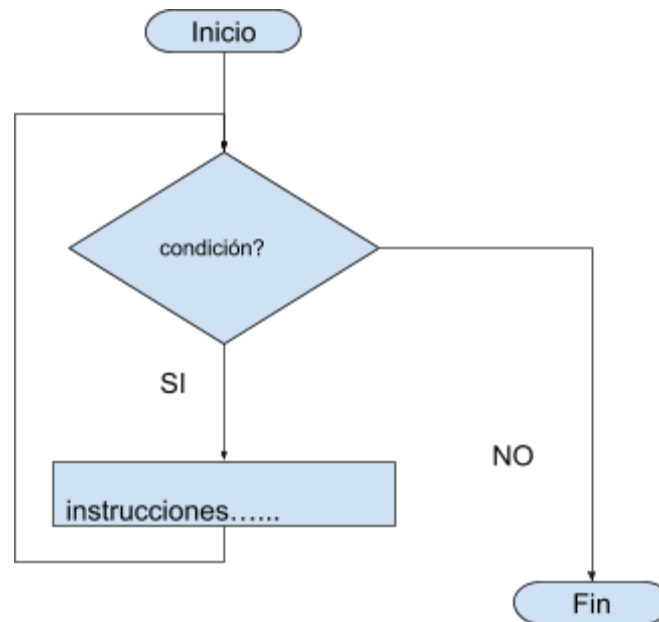


Imagen 8. Diagrama de flujo bucle While.
Fuente: Desafío Latam.

Bucle DO-While

El Bucle DO-While es muy similar al Bucle While, no obstante este siempre va a ejecutar las instrucciones al menos una vez antes de salir del bucle.

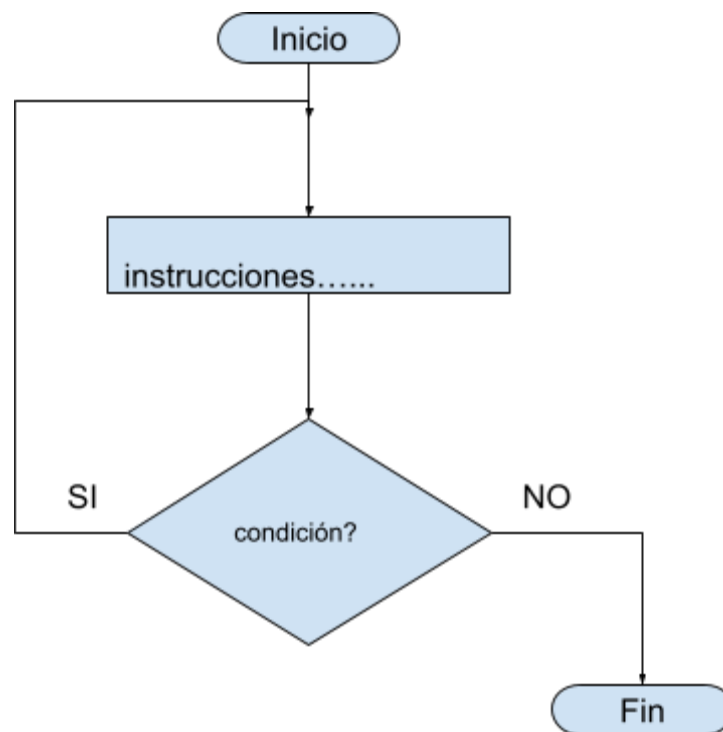


Figura 9. Diagrama de flujos Bucle DO-WHILE.
Fuente: Desafío Latam.

Soluciones ejercicios propuestos

Ejercicio propuesto (1)

Requerimientos:

- Declarar dos cadenas de texto, se pueden asignar sus valores desde el comienzo, o se pueden crear mediante entrada de datos.

```
// Variables asignadas
String CadenaUno = "Aprendiendo en Desafío Latam";
String CadenaDos = "Bienvenidos a Desafío Latam";

// Recibir por entrada de datos

Scanner sc = new Scanner(System.in);

//Ingresamos la primera cadena de texto
System.out.println("Ingrese un texto: ");
String texto = sc.nextLine();

//Ingresamos la segunda cadena de texto
System.out.println("Ingrese otro texto: ");
String texto2 = sc.nextLine();
```

- Comparar las cadenas de texto con equals, y mostrar un mensaje en pantalla según sea el caso.

```
if(texto.equals(texto2)){
    System.out.println("Las cadenas son iguales");
}else{
    System.out.println("Las cadenas no son iguales");
}
```


- Comparar las cadenas de texto con `compareTo` y ver cuál es mayor, menor o iguales.

```
if(texto.compareTo(texto2) < 0) {  
    System.out.println("texto es menor que texto2");  
} else if(texto.compareTo(texto2) == 0) {  
    System.out.println("Los textos son iguales");  
} else {  
    System.out.println("Texto2 es menor que texto");  
}
```

Finalmente el ejercicio completo quedaría de la siguiente manera:

```
package cl.desafiolatam;  
import java.util.Scanner;  
public class CompararTextos {  
    public static void main(String[] args) {  
  
        // Variables asignadas  
        String CadenaUno = "Aprendiendo en Desafío Latam";  
        String CadenaDos = "Bienvenidos a Desafío Latam";  
  
        // Recibir por entrada de datos  
        Scanner sc = new Scanner(System.in);  
  
        //Ingresamos la primera cadena de texto  
        System.out.println("Ingrese un texto: ");  
        String texto = sc.nextLine();  
  
        //Ingresamos la segunda cadena de texto  
        System.out.println("Ingrese otro texto: ");  
        String texto2 = sc.nextLine();  
  
        if(texto.equals(texto2)) {  
            System.out.println("Las cadenas son iguales");  
        } else {  
            System.out.println("Las cadenas no son iguales");  
        }  
    }  
}
```

```
if(texto.compareTo(texto2) < 0) {  
    System.out.println("texto es menor que texto2");  
} else if(texto.compareTo(texto2) == 0) {  
    System.out.println("Los textos son iguales");  
} else {  
    System.out.println("Texto2 es menor que texto");  
}  
}  
}
```

Ejercicio propuesto (2)

Siguiendo con los ejercicios propuestos de la primera parte debemos calcular el promedio final de un estudiante, con este promedio le indicaremos al estudiante si aprueba o reprueba.

Los requerimientos son:

- Si el promedio es menor a 4, se debe mostrar el mensaje: "Reprobado".
- Si el promedio es mayor a 4 y 5.5 se debe mostrar el mensaje: "Aprobado".

```
package calificaciones;
import Java.util.Scanner;
public class Calificaciones{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Ingrese nombre del estudiante: ");
        String estudiante = sc.nextLine();
        System.out.println("Ingrese asignatura: ");
        String asignatura = sc.nextLine();
        System.out.println("Ingrese nombre del docente: ");
        String docente = sc.nextLine();
        System.out.println("Ingrese nota 1: ");
        int nota1 = sc.nextInt();
        System.out.println("Ingrese nota 2: ");
        int nota2 = sc.nextInt();
        System.out.println("Ingrese nota 3: ");
        int nota3 = sc.nextInt();

        float promedio = (nota1 + nota2 + nota3) / 3;

        if (promedio < 4) {
            estadoEstudiante = "Reprobado";
        } else {
            estadoEstudiante = "Aprobado";
        }

        System.out.printf(
            "Estudiante:%s %s\n"
```

```
+ "Asignatura: %s\n"  
+ "Docente: %s\n"  
+ "Nota 1: %d\n",  
+ "Nota 2: %d\n",  
+ "Nota 3: %d\n",  
+ "Promedio: %d\n",  
+ "Estado: %s\n",  
  
    estudiante, asignatura, docente, nota1, nota2, nota3, promedio,  
    estadoEstudiante);  
    }  
}
```

Ejercicio propuesto (3)

Siguiendo con los ejercicios propuestos de la primera parte debemos calcular el promedio final de un estudiante, con este promedio le indicaremos al estudiante si debe rendir una prueba recuperatoria, si debe rendir el examen final o si está eximido.

Requerimientos:

- Si el promedio es menor a 4, se debe mostrar el mensaje: "Debes dar una prueba de recuperación para poder rendir el examen final".
- Si el promedio es entre 4 y 5.5 se debe mostrar el mensaje: "Debes rendir el examen final para aprobar".
- Si el promedio es mayor a 5.5 debe mostrar: "Felicitaciones! Te eximiste del examen y aprobaste".

Utilizando el código del ejercicio anterior cambiaremos las sentencias de modo que podamos utilizar los operadores lógicos para mostrar el mensaje adecuado en base a los requerimientos.

```
package calificaciones;
import Java.util.Scanner;
public class Calificaciones{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Ingrese nombre del estudiante: ");
        String estudiante = sc.nextLine();
        System.out.println("Ingrese asignatura: ");
        String asignatura = sc.nextLine();
        System.out.println("Ingrese nombre del docente: ");
        String docente = sc.nextLine();
        System.out.println("Ingrese nota 1: ");
        int nota1 = sc.nextInt();
        System.out.println("Ingrese nota 2: ");
        int nota2 = sc.nextInt();
        System.out.println("Ingrese nota 3: ");
        int nota3 = sc.nextInt();

        float promedio = (nota1 + nota2 + nota3) / 3;
```

```
        if (promedio < 4) {
            estadoEstudiante = "Debes dar una prueba de recuperación para
poder rendir el examen final";
        } else if (promedio >= 4 && promedio <=55) {
            estadoEstudiante = "Debes rendir el examen final para aprobar";
        } else {
            estadoEstudiante = "Felicitaciones! Te eximiste del examen y
aprobaste";
        }

        System.out.printf(
            "Estudiante:%s %s\n"
            + "Asignatura: %s\n"
            + "Docente: %s\n"
            + "Nota 1: %d\n",
            + "Nota 2: %d\n",
            + "Nota 3: %d\n",
            + "Promedio: %d\n",
            + "Estado: %s\n",

            estudiante, asignatura, docente, nota1, nota2, nota3, promedio,
            estadoEstudiante);
    }
}
```