



Estructura repetitiva: while

La estructura repetitiva while en Java es similar a la que se encuentra en Python y otros muchos lenguajes de programación. Permite ejecutar un bloque de código repetidamente mientras una condición especificada sea verdadera.

En Java, la estructura while tiene la siguiente sintaxis:

```
while (condición) {  
    // Código a ejecutar mientras la condición sea verdadera  
}
```

En esta estructura, condición es una expresión booleana que se evalúa antes de cada iteración. Si la condición es verdadera, el bloque de código dentro del while se ejecuta. Después de cada iteración, la condición se vuelve a evaluar. Si la condición es falsa, la ejecución del bucle while se detiene y el control pasa a la siguiente instrucción después del bucle.

Por ejemplo, el siguiente código en Java imprimirá los números del 1 al 5 utilizando un bucle while:

```
public class EjemploWhile {  
    public static void main(String[] args) {  
        int i = 1;  
        while (i <= 5) {  
            System.out.println(i);  
            i++;  
        }  
    }  
}
```

Este bucle while imprimirá los números del 1 al 5 porque la condición $i \leq 5$ se cumple inicialmente y se vuelve a evaluar en cada iteración hasta que i llega a 6, momento en el que la condición se vuelve falsa y el bucle se detiene.

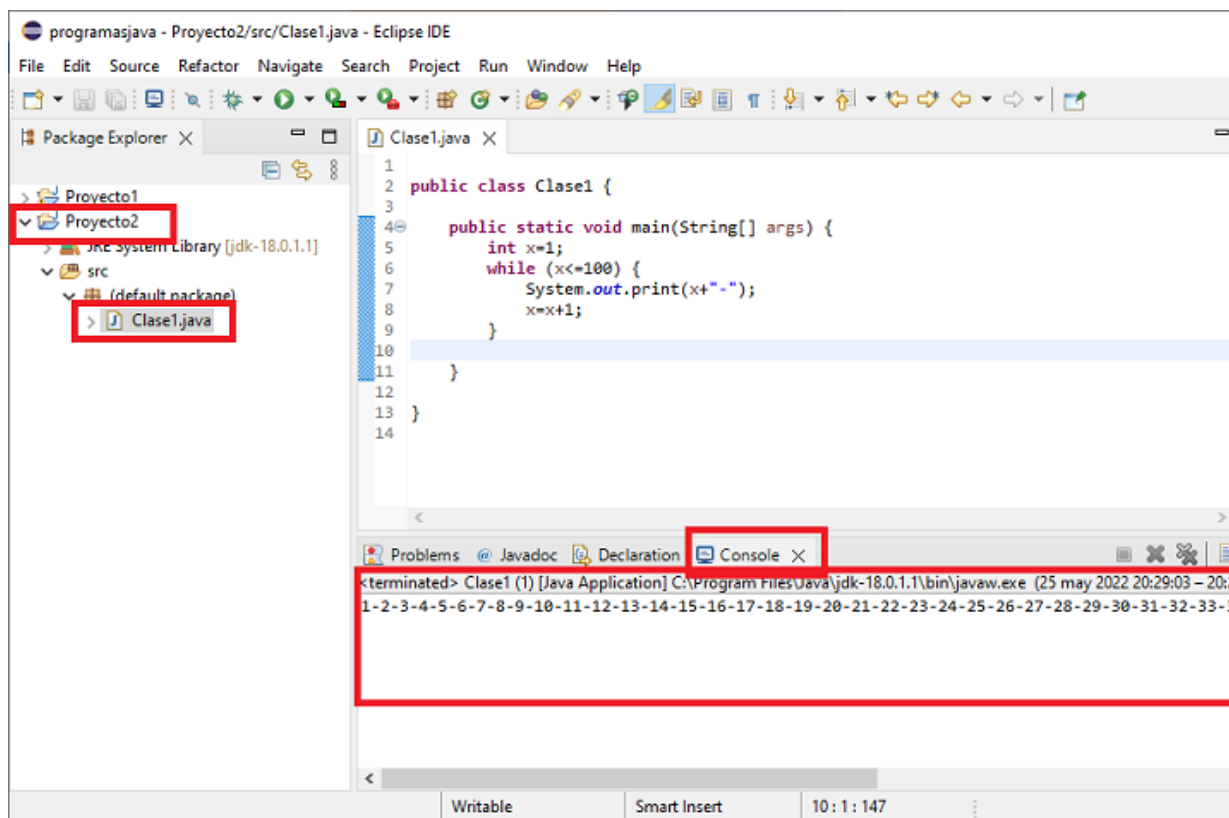
Problema 1:

Mostrar por pantalla los números comprendidos del 1 al 100 de 1 en 1 empleando la estructura repetitiva while.

Respuesta:

```
public class Clase1 {  
    public static void main(String[] args) {  
        int x=1;  
        while (x<=100) {  
            System.out.print(x+"-");  
            x=x+1;  
        }  
    }  
}
```

Recordemos cómo crear un nuevo proyecto y verificar la consola:



Analicemos el código del problema 1.

La estructura “base” es simplemente una Clase y un método main donde ejecutaremos nuestro código:

```
public class Clase1 {  
    public static void main(String[] args) {  
        // código  
    }  
}
```

Luego, declaramos una variable entera “x” con un valor inicial de 1:

Posteriormente, escribiremos la instrucción **while** seguida de una condición **entre paréntesis**. Como todas las estructuras en Java, utilizamos llaves para determinar el bloque de código que se ejecutará:

```
while (x<=100) {  
    // Código a repetir  
}
```

```
public class Clase1 {  
    public static void main(String[] args) {  
        int x=1;  
        while (x<=100) {  
            System.out.print(x+"-");  
            x=x+1;  
        }  
    }  
}
```

Dentro del bloque while, imprimiremos el valor actual de la variable x, seguido de un caracter guión:

```
System.out.print(x+"-");
```

E incrementaremos la variable x en 1:

```
x=x+1;
```

Esta última instrucción es de **suma importancia** en el while, ya que determina su condición de fin. Siempre que programemos una estructura while, debemos corroborar que tenga condición de fin, de lo contrario nuestro programa se ejecutará en un ciclo sin fin!

Cuando x toma un valor de 101, la condición del while se vuelve falsa y por tanto el while se detiene. En Java, existen dos alternativas para incrementar en 1 una variable:

```
x++; y x+=1;
```

Podés usar la que más te guste!.

Problema 2:

Generar un valor aleatorio entre 2 y 100. Luego mostrar todos los valores pares que hay desde 2 hasta dicho valor.

```
public class Clase2 {  
  
    public static void main(String[] args) {  
        int aleatorio=2+(int)(Math.random()*99);  
        int x=2;  
        System.out.println("Valores pares entre 2 y "+aleatorio);  
        while (x<=aleatorio) {  
            System.out.print(x+"-");  
            x=x+2;  
        }  
    }  
}
```

Generamos un valor aleatorio comprendido entre 2 y 100:

```
int aleatorio=2+(int)(Math.random()*99);
```

Inicializamos el contador en 2:

```
int x=2;
```

Mientras el contador sea menor o igual al valor aleatorio mostramos el contador y lo incrementamos en 2:

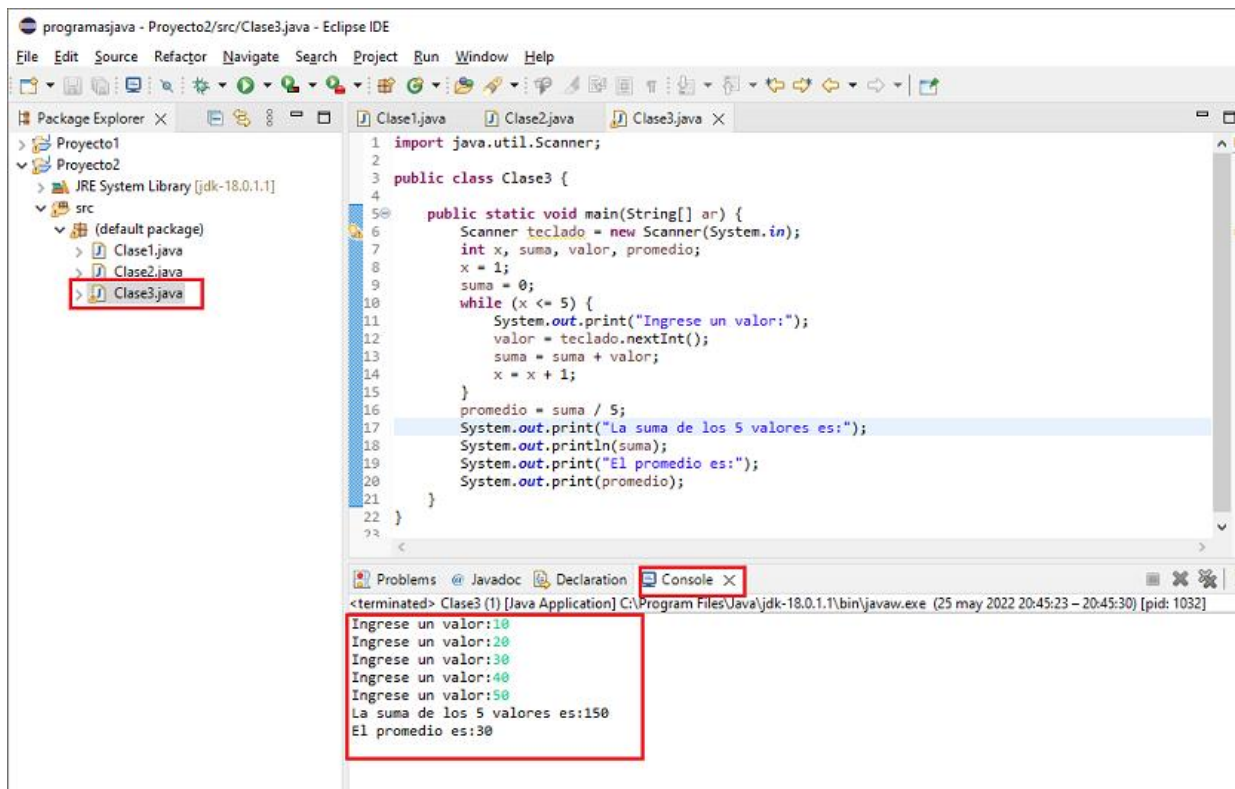
```
while (x<=aleatorio) {  
    System.out.print(x+"-");  
    x=x+2;  
}
```

En forma similar a Python podemos utilizar el operador matemático +=

```
while (x<=aleatorio) {  
    System.out.print(x+"-");  
    x+=2;  
}
```

Problema 3:

Desarrollar un programa que permita la carga de 5 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y su promedio.



El código fuente a implementar es:

```
import java.util.Scanner;
```

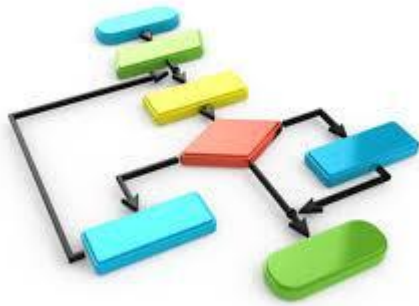
```
public class Clase3 {
```

```

    public static void main(String[] ar) {
        Scanner teclado = new Scanner(System.in);
        int x, suma, valor, promedio;
        x = 1;
        suma = 0;
        while (x <= 5) {
            System.out.print("Ingrese un valor:");
            valor = teclado.nextInt();
            suma = suma + valor;
            x = x + 1;
        }
        promedio = suma / 5;
        System.out.print("La suma de los 5 valores es:");
        System.out.println(suma);
        System.out.print("El promedio es:");
        System.out.print(promedio);
    }
}

```

Creamos un objeto de la clase Scanner como habíamos visto en la clase anterior, ésta nos permite ingresar por teclado los 5 valores.



Problemas propuestos

- 1 - Generar un valor aleatorio entre 0 y 1000. Mostrar la cantidad de dígitos que tiene dicho número.
- 2 – Se ingresan un conjunto de n alturas de personas por teclado, ingresar n por teclado. Mostrar la altura promedio de las personas.
- 3 – Mostrar los múltiplos de 8 hasta el valor 500. Debe aparecer en pantalla 8 - 16 - 24, etc.
- 4 - Desarrollar un programa que permita cargar n números enteros y luego nos informe cuántos valores fueron pares y cuántos impares.

Emplear el operador % en la condición de la estructura condicional:

```
if (valor%2==0)    //Si el if da verdadero luego es par.
```

Estructura repetitiva: do/while

La estructura repetitiva do/while no existe en Python.

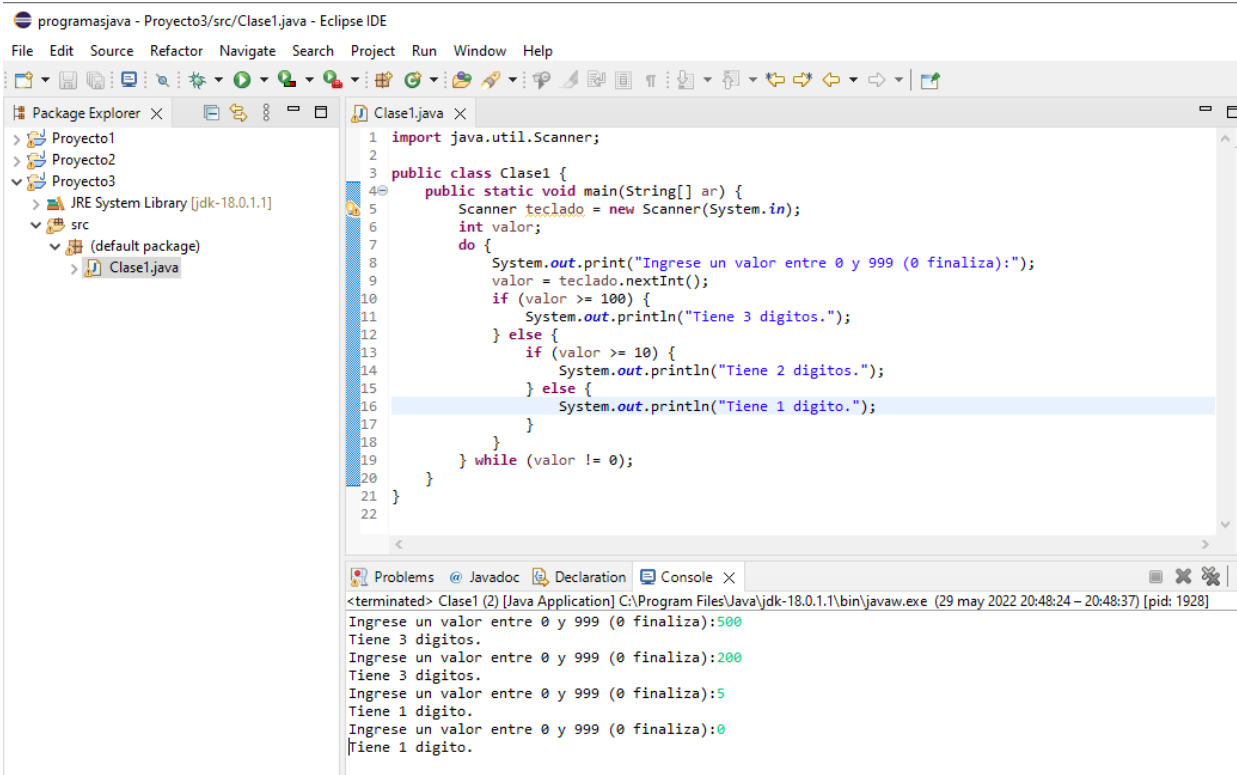
La estructura do while se ejecuta su bloque al menos una vez, a diferencia del while que podía no ejecutar el bloque.

Esta estructura repetitiva se utiliza cuando conocemos de antemano que por lo menos una vez se ejecutará el bloque repetitivo.

La condición de la estructura está abajo del bloque a repetir, a diferencia del while que está en la parte superior.

Problema 4:

Escribir un programa que solicite la carga de un número entre 0 y 999, y nos muestre un mensaje de cuántos dígitos tiene el mismo. Finalizar el programa cuando se cargue el valor 0.



```
programasjava - Proyecto3/src/Clase1.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
> Proyecto1
> Proyecto2
> Proyecto3
  > JRE System Library [jdk-18.0.1.1]
    > src
      > (default package)
        > Clase1.java

Clase1.java
1 import java.util.Scanner;
2
3 public class Clase1 {
4     public static void main(String[] ar) {
5         Scanner teclado = new Scanner(System.in);
6         int valor;
7         do {
8             System.out.print("Ingrese un valor entre 0 y 999 (0 finaliza):");
9             valor = teclado.nextInt();
10            if (valor >= 100) {
11                System.out.println("Tiene 3 digitos.");
12            } else {
13                if (valor >= 10) {
14                    System.out.println("Tiene 2 digitos.");
15                } else {
16                    System.out.println("Tiene 1 digito.");
17                }
18            }
19        } while (valor != 0);
20    }
21 }
22

Problems Javadoc Declaration Console
<terminated> Clase1 (2) [Java Application] C:\Program Files\Java\jdk-18.0.1.1\bin\javaw.exe (29 may 2022 20:48:24 - 20:48:37) [pid: 1928]
Ingrese un valor entre 0 y 999 (0 finaliza):500
Tiene 3 digitos.
Ingrese un valor entre 0 y 999 (0 finaliza):200
Tiene 3 digitos.
Ingrese un valor entre 0 y 999 (0 finaliza):5
Tiene 1 digito.
Ingrese un valor entre 0 y 999 (0 finaliza):0
Tiene 1 digito.
```

El código fuente del problema es:

```
import java.util.Scanner;
```

```
public class Clase1 {
    public static void main(String[] ar) {
        Scanner teclado = new Scanner(System.in);
```

```

        int valor;
        do {
            System.out.print("Ingrese un valor entre 0 y 999 (0
finaliza:");
            valor = teclado.nextInt();
            if (valor >= 100) {
                System.out.println("Tiene 3 digitos.");
            } else {
                if (valor >= 10) {
                    System.out.println("Tiene 2 digitos.");
                } else {
                    System.out.println("Tiene 1 digito.");
                }
            }
        } while (valor != 0);
    }
}

```

Como podemos comprobar la estructura repetitiva do/while comienza con la palabra clave ‘do’ seguida de la llave de comienzo de bloque:

```
do {
```

Luego disponemos el bloque de instrucciones que queremos repetir, y finalmente la llave de cerrado del bloque con la palabra clave while y entre paréntesis la condición. Lleva punto y como el final de do/while:

```
} while (valor != 0);
```

Problema 5:

Escribir un programa que solicite la carga de números por teclado, obtener su promedio. Finalizar la carga de valores cuando se cargue el valor 0.

```
import java.util.Scanner;
```

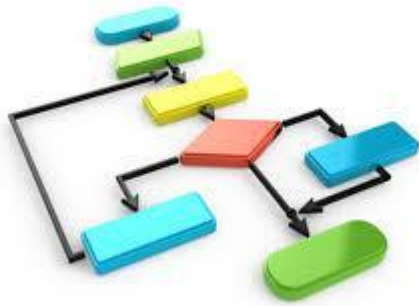
```

public class Clase2 {
    public static void main(String[] ar) {
        Scanner teclado = new Scanner(System.in);
        int suma, cant, valor, promedio;
        suma = 0;
        cant = 0;
        do {
            System.out.print("Ingrese un valor (0 para finalizar:");
            valor = teclado.nextInt();
            if (valor != 0) {
                suma = suma + valor;
                cant++;
            }
        } while (valor != 0);
        if (cant != 0) {
            promedio = suma / cant;
        }
    }
}

```



```
        System.out.print("El promedio de los valores ingresados es:");  
        System.out.print(promedio);  
    } else {  
        System.out.print("No se ingresaron valores.");  
    }  
}  
}
```



Problemas propuestos

1 - Realizar un programa que acumule (sume) valores ingresados por teclado hasta ingresar el 9999 (no sumar dicho valor, indica que ha finalizado la carga). Imprimir el valor acumulado e informar si dicho valor es cero, mayor a cero o menor a cero.

2 – En un banco se procesan datos de las cuentas corrientes de sus clientes. De cada cuenta corriente se conoce: número de cuenta y saldo actual. El ingreso de datos debe finalizar al ingresar un valor negativo en el número de cuenta.

Se pide confeccionar un programa que lea los datos de las cuentas corrientes e informe:

a) De cada cuenta: número de cuenta y estado de la cuenta según su saldo, sabiendo que:

Estado de la cuenta 'Acreedor' si el saldo es >0 .

 'Deudor' si el saldo es <0 .

 'Nulo' si el saldo es $=0$.

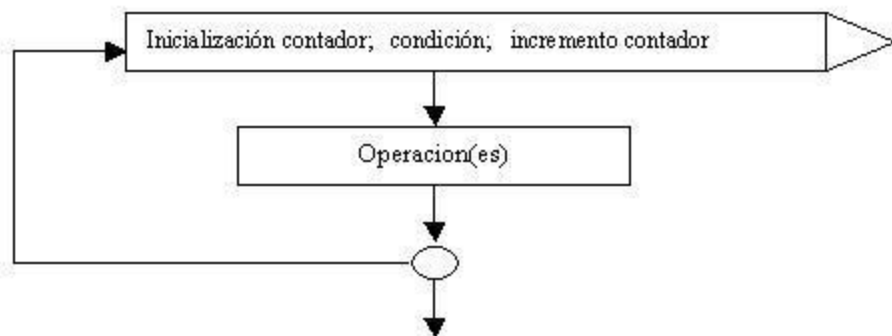
b) La suma total de los saldos acreedores.

Estructura repetitiva: for

La estructura repetitiva for es bastante distinta a la que existe en Python.

Cualquier problema que requiera una estructura repetitiva se puede resolver empleando la estructura while. Pero hay otra estructura repetitiva cuyo planteo es más sencillo en ciertas situaciones.

En general, la estructura for se usa en aquellas situaciones en las cuales CONOCEMOS la cantidad de veces que queremos que se ejecute el bloque de instrucciones. Ejemplo: cargar 10 números, ingresar 5 notas de alumnos, etc. Conocemos de antemano la cantidad de veces que queremos que el bloque se repita. Veremos, sin embargo, que en el lenguaje Java la estructura for puede usarse en cualquier situación repetitiva, porque en última instancia no es otra cosa que una estructura while generalizada.

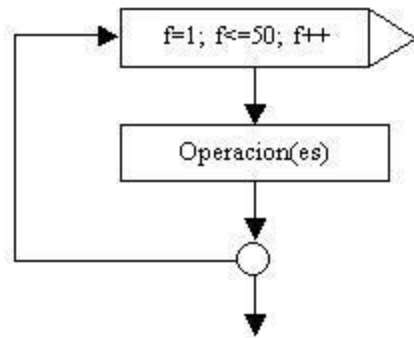


En su forma más típica y básica, esta estructura requiere una variable entera que cumple la función de un CONTADOR de vueltas. En la sección indicada como "iniciación contador", se suele colocar el nombre de la variable que hará de contador, asignándole a dicha variable un valor inicial. En la sección de "condición" se coloca la condición que deberá ser verdadera para que el ciclo continúe (en caso de un falso, el ciclo se detendrá). Y finalmente, en la sección de "incremento contador" se coloca una instrucción que permite modificar el valor de la variable que hace de contador (para permitir que alguna vez la condición sea falsa)

Cuando el ciclo comienza, antes de dar la primera vuelta, la variable del for toma el valor indicado en la sección de "iniciación contador". Inmediatamente se verifica, en forma automática, si la condición es verdadera. En caso de serlo se ejecuta el bloque de operaciones del ciclo, y al finalizar el mismo se ejecuta la instrucción que se haya colocado en la tercer sección.

Seguidamente, se vuelve a controlar el valor de la condición, y así prosigue hasta que dicha condición entregue un falso.

Si conocemos la cantidad de veces que se repite el bloque es muy sencillo emplear un for, por ejemplo si queremos que se repita 50 veces el bloque de instrucciones puede hacerse así:



La variable del for puede tener cualquier nombre. En este ejemplo se la ha definido con el nombre f.

Analicemos el ejemplo:

- La variable f toma inicialmente el valor 1.
- Se controla automáticamente el valor de la condición: como f vale 1 y esto es menor que 50, la condición da verdadero.
- Como la condición fue verdadera, se ejecutan la/s operación/es.
- Al finalizar de ejecutarlas, se retorna a la instrucción f++, por lo que la variable f se incrementa en uno.
- Se vuelve a controlar (automáticamente) si f es menor o igual a 50.

Como ahora su valor es 2, se ejecuta nuevamente el bloque de instrucciones e incrementa nuevamente la variable del for al terminar el mismo.

- El proceso se repetirá hasta que la variable f sea incrementada al valor 51.

En este momento la condición será falsa, y el ciclo se detendrá.

La variable f “PUEDE” ser modificada dentro del bloque de operaciones del for, aunque esto podría causar problemas de lógica si el programador es inexperto.

La variable f puede ser inicializada en cualquier valor y finalizar en cualquier valor. Además, no es obligatorio que la instrucción de modificación sea un incremento del tipo contador (f++).

Cualquier instrucción que modifique el valor de la variable es válida. Si por ejemplo se escribe f=f+2 en lugar de f++, el valor de f será incrementado de 2 en cada vuelta, y no de 1. En este caso, esto significará que el ciclo no efectuará las 50 vueltas sino sólo 25.

Problema 6:

Realizar un programa que imprima en pantalla los números del 1 al 100.

```
public class Clase1 {  
    public static void main(String[] ar) {  
        for (int f = 1; f <= 100; f++) {  
            System.out.print(f);  
            System.out.print("-");  
        }  
    }  
}
```

Problema 7:

Desarrollar un programa que permita la carga de 10 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y su promedio.

```
import java.util.Scanner;  
  
public class Clase2 {  
    public static void main(String[] ar) {  
        Scanner teclado = new Scanner(System.in);  
        int suma, f, valor, promedio;  
        suma = 0;  
        for (f = 1; f <= 10; f++) {  
            System.out.print("Ingrese valor:");  
            valor = teclado.nextInt();  
            suma = suma + valor;  
        }  
        System.out.print("La suma es:");  
        System.out.println(suma);  
        promedio = suma / 10;  
        System.out.print("El promedio es:");  
        System.out.print(promedio);  
    }  
}
```

Problema 8:

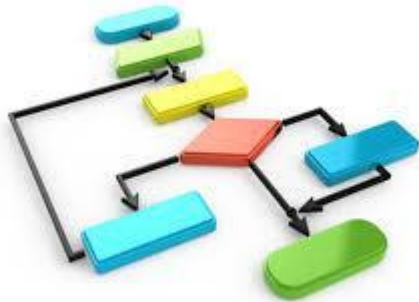
Escribir un programa que lea 10 números enteros y luego muestre cuántos valores ingresados fueron múltiplos de 3 y cuántos de 5. Debemos tener en cuenta que hay números que son múltiplos de 3 y de 5 a la vez.

```
import java.util.Scanner;  
  
public class Clase3 {  
    public static void main(String[] ar) {  
        Scanner teclado = new Scanner(System.in);
```

```

int mul3, mul5, valor, f;
mul3 = 0;
mul5 = 0;
for (f = 1; f <= 10; f++) {
    System.out.print("Ingrese un valor:");
    valor = teclado.nextInt();
    if (valor % 3 == 0) {
        mul3 = mul3 + 1;
    }
    if (valor % 5 == 0) {
        mul5 = mul5 + 1;
    }
}
System.out.print("Cantidad de valores ingresados múltiplos de 3:");
System.out.println(mul3);
System.out.print("Cantidad de valores ingresados múltiplos de 5:");
System.out.println(mul5);
}
}

```



Problemas propuestos

1 - Desarrollar un programa que solicite la carga de 10 números e imprima la suma de los últimos 5 valores ingresados.

2 - Desarrollar un programa que muestre la tabla de multiplicar del 3 (del 3 al 30)

3 - Confeccionar un programa que permita ingresar un valor del 1 al 10 y nos muestre la tabla de multiplicar del mismo (los primeros 12 términos)

Ejemplo: Si ingreso 3 deberá aparecer en pantalla los valores 3, 6, 9, hasta el 36.

4 - Escribir un programa que pida ingresar coordenadas (x,y) que representan puntos en el plano.

Informar cuántos puntos se han ingresado en el primer, segundo, tercer y cuarto

cuadrante. Al comenzar el programa se pide que se ingrese la cantidad de puntos a procesar.

Cadenas de caracteres

Las cadenas de caracteres son más difíciles de administrar que el lenguaje Python, para dar un ejemplo en Java no se puede utilizar el operador relacionar `==` para verificar si dos String son iguales.

En Java hemos visto que cuando queremos almacenar un valor entero definimos una variable de tipo `int`, si queremos almacenar un valor con decimales definimos una variable de tipo `float`. Ahora si queremos almacenar una cadena de caracteres (por ejemplo un nombre de una persona) debemos definir un objeto de la clase `String`.

Más adelante veremos en profundidad y detenimiento los conceptos de CLASE y OBJETO, por ahora solo nos interesa la mecánica para trabajar con cadenas de caracteres.

Problema 9:

Solicitar el ingreso del nombre y edad de dos personas. Mostrar el nombre de la persona con mayor edad.

```
import java.util.Scanner;

public class Clase1 {
    public static void main(String[] ar) {
        Scanner teclado=new Scanner(System.in);
        String nombre1,nombre2;
        int edad1,edad2;
        System.out.print("Ingrese el nombre:");
        nombre1=teclado.next();
        System.out.print("Ingrese edad:");
        edad1=teclado.nextInt();
        System.out.print("Ingrese el nombre:");
        nombre2=teclado.next();
        System.out.print("Ingrese edad:");
        edad2=teclado.nextInt();
        System.out.print("La persona de mayor edad es:");
        if (edad1>edad2) {
            System.out.print(nombre1);
        } else {
            System.out.print(nombre2);
        }
    }
}
```

Para almacenar un nombre debemos definir una variable de tipo `String` y su ingreso por teclado se hace llamando al método `next()` del objeto `teclado`:

```
nombre1=teclado.next();
```


La primera salvedad que tenemos que hacer cuando utilizamos el método next() es que solo nos permite ingresar una cadena de caracteres con la excepción del espacio en blanco (es decir debemos ingresar un nombre de persona y no su nombre y apellido separado por un espacio en blanco)

Veamos que existe otro método llamado nextLine() que nos permite cargar espacios en blanco pero para su uso se complica cuando cargamos otros valores de tipo distinto a String (por ejemplo int, float etc.)

Problema 10:

Solicitar el ingreso del apellido, nombre y edad de dos personas. Mostrar el nombre de la persona con mayor edad. Realizar la carga del apellido y nombre en una variable de tipo String.

```
import java.util.Scanner;

public class Clase2 {
    public static void main(String[] ar) {
        Scanner teclado=new Scanner(System.in);
        String apenom1,apenom2;
        int edad1,edad2;
        System.out.print("Ingrese el apellido y el nombre:");
        apenom1=teclado.nextLine();
        System.out.print("Ingrese edad:");
        edad1=teclado.nextInt();
        System.out.print("Ingrese el apellido y el nombre:");
        teclado.nextLine();
        apenom2=teclado.nextLine();
        System.out.print("Ingrese edad:");
        edad2=teclado.nextInt();
        System.out.print("La persona de mayor edad es:");
        if (edad1>edad2) {
            System.out.print(apenom1);
        } else {
            System.out.print(apenom2);
        }
    }
}
```

Cuando se ingresa una cadena con caracteres en blanco debemos tener en cuenta en llamar al método nextLine()

Una dificultad se presenta si llamamos al método nextLine() y previamente hemos llamado al método nextInt(), esto debido a que luego de ejecutar el método nextInt() queda almacenado en el objeto de la clase Scanner el caracter "Enter" y si llamamos inmediatamente al método nextLine() este almacena dicho valor de tecla y continúa con el flujo del programa. Para solucionar este problema debemos generar un código similar a:

```
System.out.print("Ingrese edad:");
```

```

edad1=teclado.nextInt();

System.out.print("Ingrese el apellido y el nombre:");

teclado.nextLine();

apenom2=teclado.nextLine();

```

Como vemos llamamos al método `nextLine()` dos veces, la primera retorna la tecla "Enter" y la segunda se queda esperando que ingresemos el apellido y nombre (tener en cuenta que esto es necesario solo si previamente se llamó al método `nextInt()` o `nextFloat()`).

Problema 11:

Solicitar el ingreso de dos apellidos. Mostrar un mensaje si son iguales o distintos.

```

import java.util.Scanner;

public class Clase3 {
    public static void main(String[] ar) {
        Scanner teclado=new Scanner(System.in);
        String apellido1,apellido2;
        System.out.print("Ingrese primer apellido:");
        apellido1=teclado.next();
        System.out.print("Ingrese segundo apellido:");
        apellido2=teclado.next();
        if (apellido1.equals(apellido2)) {
            System.out.print("Los apellidos son iguales");
        } else {
            System.out.print("Los apellidos son distintos");
        }
    }
}

```

Para comparar si el contenido de dos String son iguales no podemos utilizar el operador `==`. Debemos utilizar un método de la clase String llamado `equals` y pasar como parámetro el String con el que queremos compararlo:

```

if (apellido1.equals(apellido2)) {

```

El método `equals` retorna verdadero si los contenidos de los dos String son exactamente iguales, esto hace que se ejecute el bloque del verdadero.

Recordemos que hemos utilizado el método `next()` para la carga de los String, luego esto hace que no podamos ingresar un apellido con espacios en blanco (podemos probar que si ingresamos por ejemplo "Rodriguez Rodriguez" en el primer apellido, luego se carga la cadena "Rodriguez" en la variable `apellido1` y "Rodriguez" en la variable `apellido2` (con esto hacemos notar que cada

vez que ingresamos un espacio en blanco cuando utilizamos el método `next()` los caracteres que siguen al espacio en blanco son recuperados en la siguiente llamada al método `next()`

El método `equals` retorna verdadero si los contenidos de los dos `String` son exactamente iguales, es decir si cargamos "Martinez" en `apellido1` y "martinez" en `apellido2` luego el método `equals` retorna falso ya que no es lo mismo la "M" mayúscula y la "m" minúscula.

En el caso que necesitemos considerar igual caracteres mayúsculas y minúsculas podemos utilizar el método `equalsIgnoreCase`:

```
if (apellido1.equalsIgnoreCase(apellido2)) {  
  
    System.out.print("Los apellidos son iguales sin  
tener en cuenta mayúsculas y minúsculas");  
  
} else {  
  
    System.out.print("Los apellidos son distintos sin  
tener en cuenta mayúsculas y minúsculas");  
  
}
```