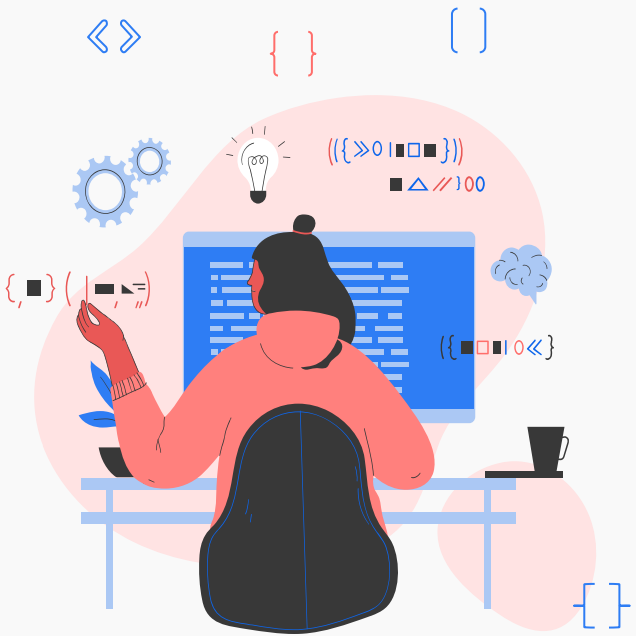
Programación

Tema 2 - Estructuras de control

Marina Hurtado Rosales
marina.hurtado@escuelaartegranada.com



Índice

Control de caracteres

Control de caracteres

Los caracteres son un tipo de dato básico de los que disponemos. Al ser un tipo de dato básico, podemos realizar **todas las operaciones** que podemos hacer con el resto de tipos de datos básicos.

Ej:

- ‘a’ + ‘h’
- ‘a’ < ‘t’

Control de caracteres

La máquina, el ordenador, no maneja los caracteres **directamente**, sino que los traduce a un valor numérico para poder procesarlos. Es lo que conocemos como **codificación (ASCII, UTF-8, ISO-8859-1, etc).**

Gracias a esta forma de manejar los caracteres podemos operar con ellos como si **fuesen números**.

Ej:

- ‘a’ + 5 = ‘f’
- ‘A’ > 40

Control de caracteres

Como vimos en el tema 1, los String son un tipo de dato especial que maneja una **colección de caracteres.**

“Hola” == ‘H’ + ‘o’ + ‘l’ + ‘a’

Dentro de Java tenemos una serie de funcionalidades que nos permiten procesar los caracteres que forman parte de los String.

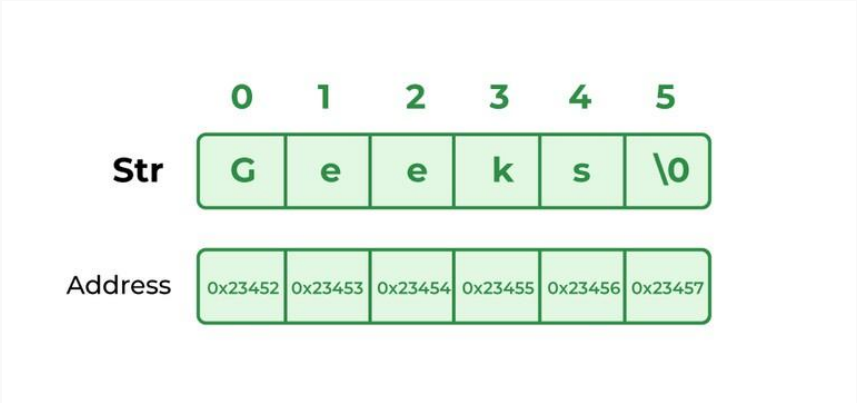
- **“hola”.toUpperCase() => “HOLA”**
- **“HOLA”.toLowerCase() => “hola”**

Control de caracteres

Dentro de un String, cada carácter tiene **asignado un número** en base a la posición que ocupa, **empezando por 0.** Es lo que se conoce como **índice.**

Para la variable **String cadena**:

- **cadena.length():** Nos devuelve la cantidad **total** de caracteres que incluye el String.
- **cadena.charAt(indice):** Nos devuelve el carácter de la cadena que tenga asignado el índice que se indica entre los paréntesis.

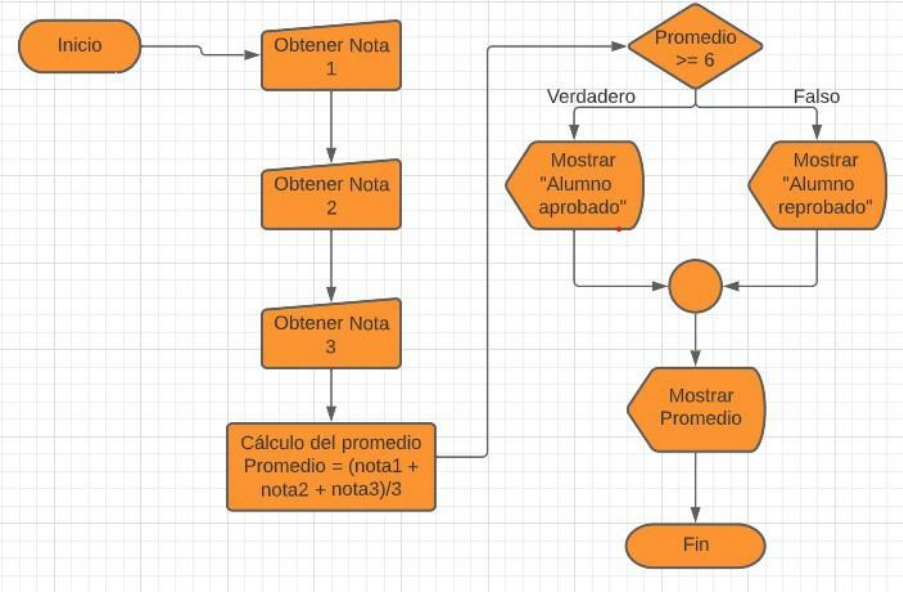


Estructuras de selección

Lógica de programación

Antes de ponerse a escribir el código, hay que organizarse.
No se puede hacer una casa empezando por el tejado, con los problemas de programación es lo mismo. Se analiza el problema y se va poco a poco.

Realiza un programa donde obtienes una serie de notas de un alumno y calculas el promedio. Si la nota es mayor a 6 avisas que ha aprobado, en caso contrario que está suspenso. En cualquier caso muestras su nota media final.

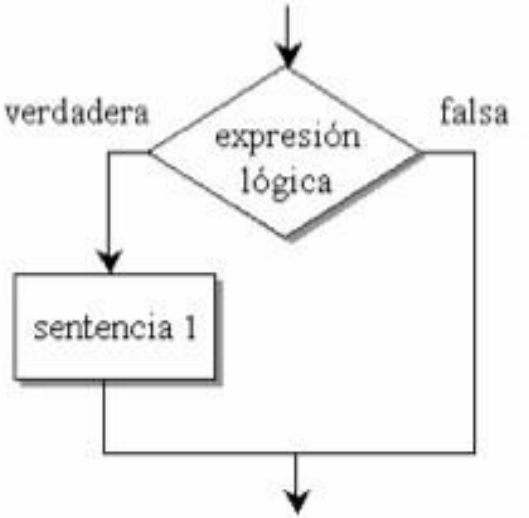


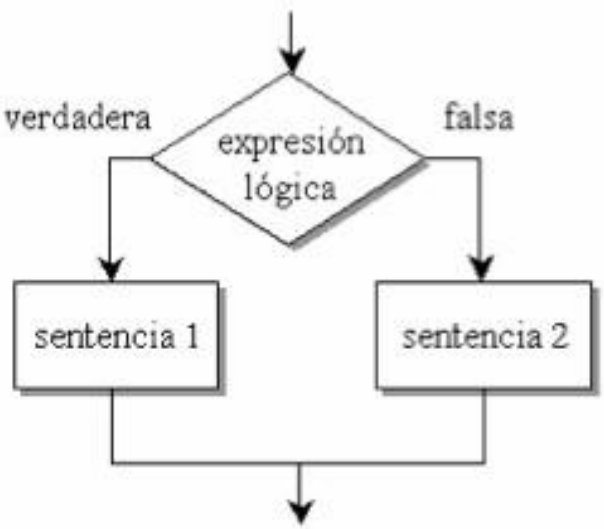
Estructuras de selección (condicionales)

- Estas estructuras permiten decidir si ejecutar o no un bloque de código entre una o más opciones.
- Para evaluar estas condiciones se utilizan los operadores lógicos y de comparación.
- **If-else**: Se ejecuta un bloque de código o no, en función de si se cumple una condición o no.
- **Switch:** Permite ejecutar un bloque de código en función de los valores que pueda tomar una expresión.

Sentencia if-else

Gracias a estas estructuras el programa puede seguir un camino u otro dependiendo del valor de un dato





Sentencia if-else

La sentencia if-else permite ejecutar un bloque de código según si se cumple una cierta condición

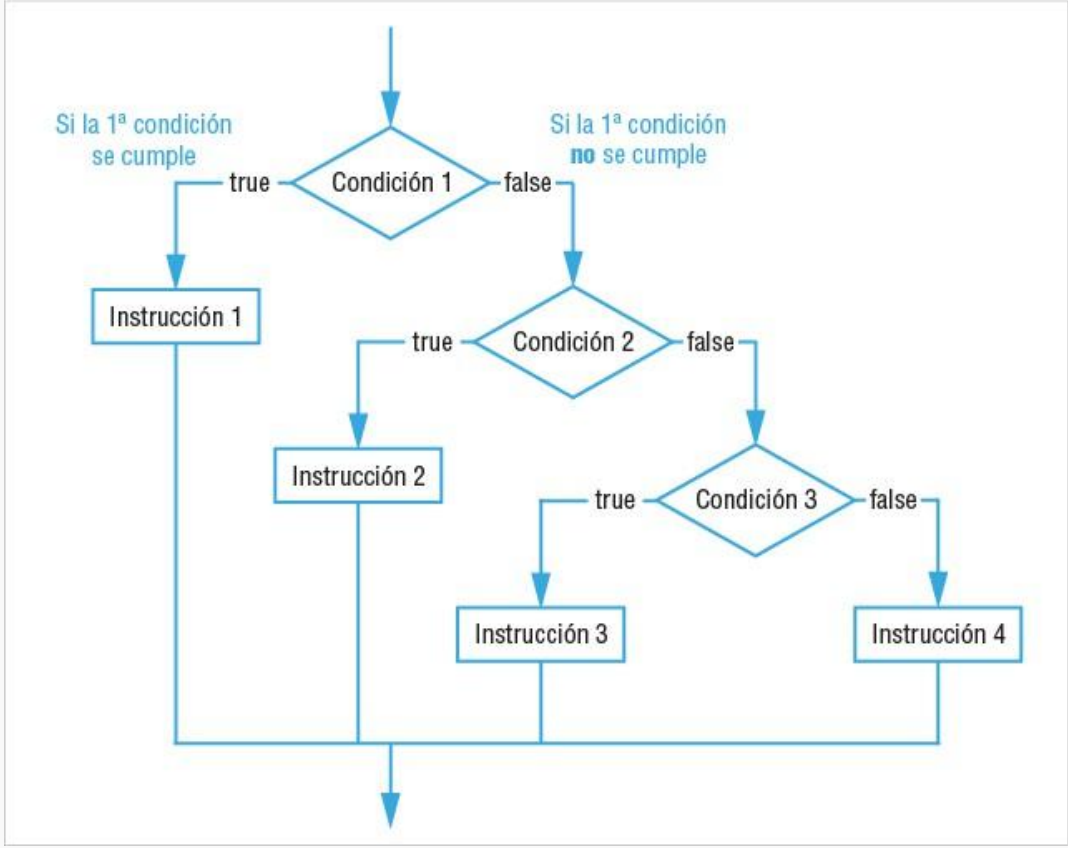
Ejemplo if

```
int hora = 12;
if (hora < 12) {
        System.out.println(“Buenos días”);
}
```

Ejemplo if-else

```
int hora = 12;
if (hora < 12) {
        System.out.println(“Buenos días”);
}
else {
                System.out.println(“Buenas tardes o
noches”);
}
```

Sentencias if-else anidadas



Sentencia if-else

La sentencia if-else permite ejecutar un bloque de código según si se cumple una cierta condición

Ejemplo if-else-if

```
int hora = 12;
if (hora < 12) {
        System.out.println(“Buenos días”);
}
else if (hora <20 ){
                System.out.println(“Buenas tardes”);
}
else {
                System.out.println(“Buenas noches”);
}
```

Ejemplos de sentencias if-else

Ejemplo if-else-if

```
int hora = 12;
if (hora < 5) {
        System.out.println(“Deberías dormir”);
}
else if (hora < 12 ){
                System.out.println(“Buenos días”);
}
else if (hora < 20){
                System.out.println(“Buenas tardes”);
}
else if (hora <= 23){
                System.out.println(“Buenas noches”);
}
else{
                System.out,println(“Esa hora no existe”);
}
```

Ejemplo if-else-if

```
int dinero = 100;
if (dinero == 0) {
        System.out.println(“Eres pobre”);
}
else if (dinero > 0 && dinero < 1000 ){
                System.out.println(“Sigues siendo
pobre);
}
else if (dinero >= 1000 && < 2000){
                System.out.println(“Ánimo” );
}
else{
                System.out,println(“Guau!”);
}
```

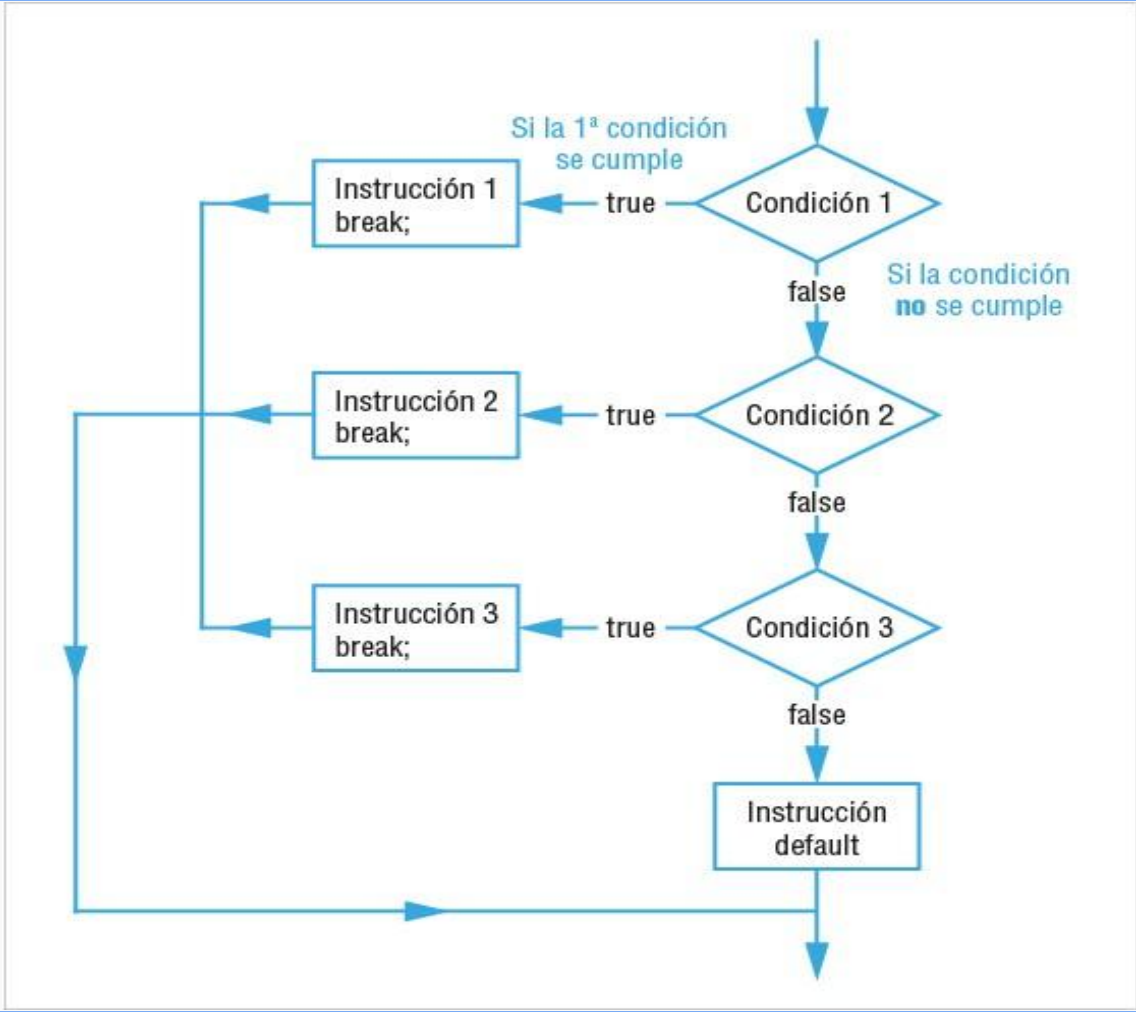
Resumen if-else

```
if (condición1){
        Operación al cumplirse condición1
}
else if(condición2){
        Operación al cumplirse condición2
} //Pones tantos else if como necesites
else{
        Operación si no se cumple las condiciones
anteriores
}
```

Sentencia switch

Esta sentencia se utiliza cuando una variable puede tener muchos valores diferentes y en función de cada uno de ellos el programa sigue un flujo u otro.

```
int dia;
System.out.println("Introduce el dia de la semana");
dia=sc.nextInt();
switch(dia)
{
  case 1:
     System.out.println("Lunes");
  break;
  case 2:
     System.out.println("Martes");
  break;
  ...
  default:
     System.out.println("No es un dia de la semana");
}
```


Si la 1ª condición se cumple
Condición 1
true
Instrucción 1
break;
false
Si la condición no se cumple
Condición 2
true
Instrucción 2
break;
false
Condición 3
true
Instrucción 3
break;
false
Instrucción default

Comparación switch y if-else if- else

```java
String materia;
materia=sc.next();
if(materia.equals("matematicas"))
{
   System.out.println("Va de numeros");
}else if(materia.equals("lengua")){
   System.out.println("Va de letras");
}else if(materia.equals("ingles")){
   System.out.println("Va de hablar");
}else if(materia.equals("ciencias")){
   System.out.println("Va de investigar");
else{
   System.out.println("No sé de que va");
}
```

```java
String materia;
materia=sc.next();
switch(materia)
{
   case "matematicas":
     System.out.println("Va de numeros");
     break;
   case "lengua":
     System.out.println("Va de letras");
     break;
   case "ingles":
     System.out.println("Va de hablar");
     break;
   case "ciencias":
     System.out.println("Va de investigar");
     break;
   default:
     System.out.println("No sé de que va");
}
```

Ejercicio práctico: calculadora +

Mejora el ejercicio de la calculadora usando la sentencia Switch.

1. Muestra un mensaje de bienvenida de la calculadora.
2. Introduces por consola los dos valores numéricos
3. Muestras un mensaje mostrando un panel de opciones que dependiendo del número introducido, haga una operación u otra **(suma, resta, producto, cociente y módulo)**
4. Introduces por consola el número de ese panel de opciones
5. Con una sentencia Switch, realizas las operaciones y muestras por pantalla el valor obtenido

```
run:
Bienvenido/a a la calculadora del futuro
Introduce el primer valor
5
Introduce el segundo valor
6
Aqu tienes el panel de opciones:
Pulsa 1 para sumar
Pulsa 2 para restar
1
El resultado de la suma es 11.0
BUILD SUCCESSFUL (total time: 10 seconds)
```

Estructuras de repetición

Estructuras de repetición

Nos permiten repetir un trozo de código un número determinado de veces. Este número de veces vendrá dado por una condición (o varias). Hay que tener cuidado en no crear un bucle infinito.

Estas estructuras permiten ejecutar un bloque de código mientras se cumpla una condición. Su uso nos ofrece una serie de ventajas:

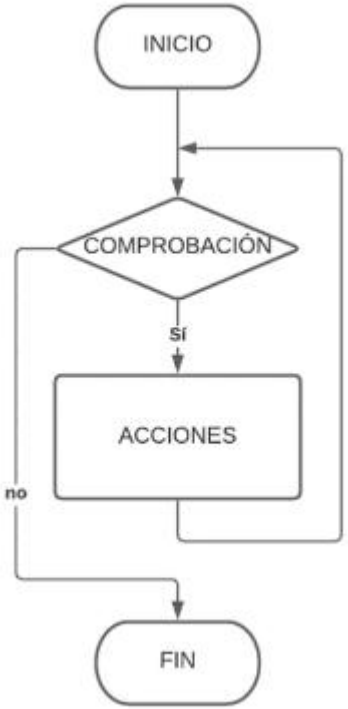
- Ahorran tiempo.
- Reduce posibles errores. Mejora la depuración.
- Hacen el código más legible.
- Reutilización de código.

Estructuras de repetición: bucle while

Ejecuta un bloque de código especificado mientras una condición sea true
Nos permite repetir un bloque de código 0 o más veces.

1. Comprueba si la condición es verdadera o falsa.
2. Si la condición es verdadera ejecuta el bloque de código.
    - Cuando termina de ejecutar el bloque vuelve al punto uno.
3. Si la condición es falsa no entra al bucle y ejecuta la siguiente instrucción.

```
int contador = 1;
while( contador <= 50 )
{
    System.out.println("El número es: " + contador);
    contador++;
}
```

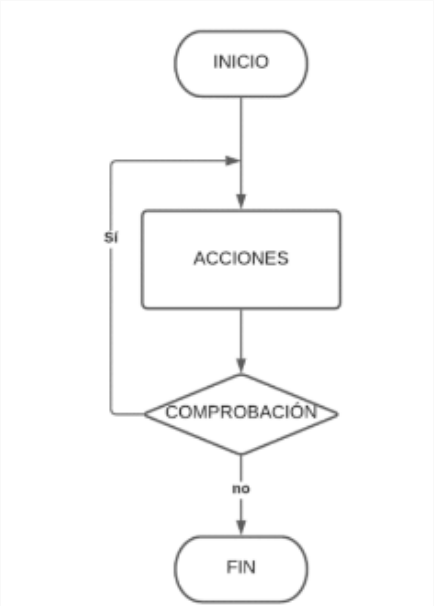


Estructuras de repetición: bucle do-while

Es una variante del bucle while. Ejecuta el bloque dé código una vez, y entonces comprueba si se cumple la condición.
Nos permite repetir un bloque de código 1 o más veces, diferencia con while es que se va a ejectutar mínimo una vez

1. Entra al bucle y ejecuta el bloque de código.
2. Al terminar la ejecución del bloque comprueba si la condición es verdadera o falsa.
   - Si la condición es verdadera vuelve al punto 1.
   - Si la condición es falsa no entra al bucle y ejecuta la siguiente instrucción

```
int contador = 1;
do
{
    System.out.println("El número es: " + contador);
    contador++;
} while( contador <= 50 );
```



Ejercicio práctico: calculadora ++

Seguimos mejorando la calculadora. Esta vez tendrá un bucle para que realicemos las operaciones que queramos.

1. Muestras un mensaje de bienvenida.
2. Introduces por consola los dos valores numéricos.
3. Muestras un mensaje con un panel con todas las opciones que tenga la calculadora
4. Introduces por consola la opción elegida.
5. Usas un bucle **while** o **do-while** con varios **if** o un **switch** que realice las operaciones.
    a. En ese bucle debe haber una opción de volver a introducir valores numéricos nuevos
    b. Debes poder salir de ese bucle.

Recordatorio: al final del bucle debe mostrar de nuevo el panel de opciones y deben introducir de nuevo la opción elegida, si no, el bucle será infinito.

Ejercicio práctico: calculadora ++

```
Bienvenido/a a la calculadora del futuro
Introduce el primer valor
5
Introduce el segundo valor
4
Aqui tienes el panel de opciones:
Pulsa 1 para sumar
Pulsa 2 para restar
Pulsa 3 para cambiar los valores
Pulsa 4 para salir
```

Estructuras de repetición: bucle for

Este tipo de bucle se utiliza cuando sabemos exactamente cuantas veces queremos que se ejecute el bloque.

Se declara utilizando 3 sentencias:

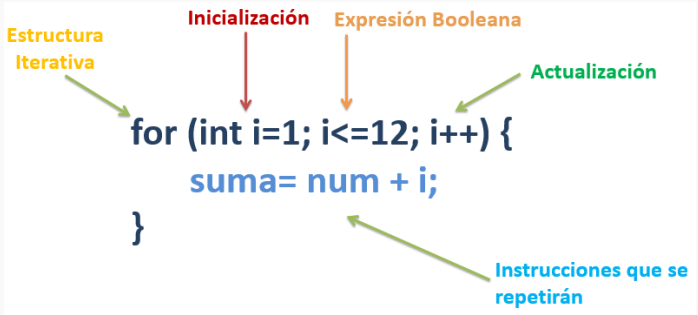
- Sentencia 1: se ejecuta una vez antes de entrar al bucle.
- Sentencia 2: condición para seguir ejecutando el bucle.
- Sentencia 3: se ejecuta cada vez que se realiza una vuelta del bucle, después del bloque de código.

```
for (sent 1; sent 2; sent 3){

    instrucciones;

}
```

Estructuras de repetición: bucle for

Está controlado por tres partes:

1. Un contador, inicializado normalmente a cero.
2. Una condición de parada, que tendrá que ver con el contador.
3. Un incremento del contador, puede ser de uno en uno, adelante, hacia atrás, o de un número x en x.



Ejercicio práctico: suma de pares

Crea un programa que te sume los números pares de un rango de números que le proporciones

1. Introduces por consola el rango de números que quieras
2. Creas un bucle **for** que recorra el rango que has puesto anteriormente
3. Dentro del bucle haces una comprobación de los números que son pares (pista: un nº es par cuando lo divides entre 2 y el resto te sale 0)
4. Realizar la suma (es acumulativa)

```
Introduce el valor inicial;
4
Introduce el valor final:
45
El resultado de sumar los numeros pares desde el numero 4 hasta el numero 45 es: 504
BUILD SUCCESSFUL (total time: 6 seconds)
```

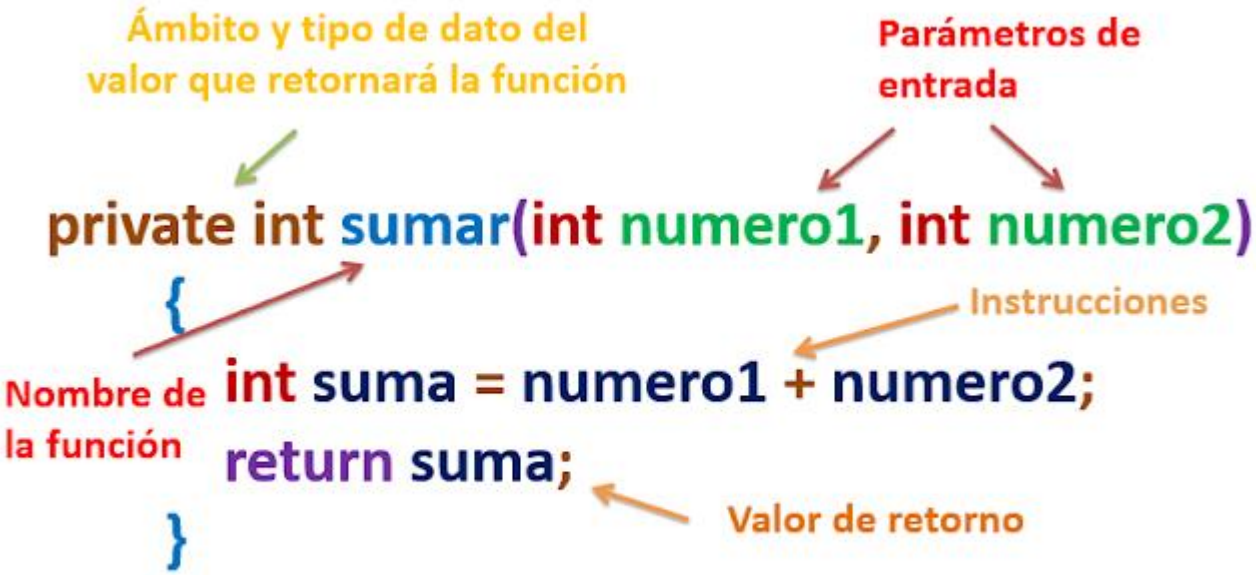
Funciones

Funciones

Hay ocasiones donde ejecutaremos el mismo bloque de código varias veces. Para no repetir código, crearemos una función con ese bloque de código y llamaremos a la función cuántas veces queremos.

- Reutilización
- Estructuración: ayuda a dividir el código en partes más pequeñas
- Mantenimiento
- Claridad

Funciones


Ámbito y tipo de dato del valor que retornará la función
Parámetros de entrada
private int sumar(int numero1, int numero2)
{
Instrucciones
Nombre de la función
int suma = numero1 + numero2;
return suma;
Valor de retorno
}

Funciones

```java
public class EjemploFuncion {

    // Definición de una función que suma dos números
    public static int sumar(int a, int b) {
        int resultado = a + b;
        return resultado;
    }

    public static void main(String[] args) {
        // Llamada a la función sumar y almacenamiento del re
        int numero1 = 5;
        int numero2 = 7;
        int resultadoSuma = sumar(numero1, numero2);

        // Imprimir el resultado
        System.out.println("La suma de " + numero1 + " y " +
    }
}
```

Las funciones pueden tener cualquier tipo:

- int
- double
- String
- ...

En caso que no devuelvan nada, son de tipo **void**

Ejercicio práctico: calculadora+++

Última mejora a nuestra calculadora. Como hemos visto en la versión “calculadora ++”, hemos repetido código varias veces, por ejemplo cuando nos muestra el panel de opciones o al introducir los números por consola.

1. Añade varias funciones a tu calculadora y llama a dichas funciones donde sea necesario
    a. 1 función de tipo “void” que contenga el panel de opciones (los System.out.println)
    b. 1 función tipo “double” que contenga el código para introducir el primer número
    c. 1 función tipo “double” que contenga el código para introducir el segundo número.
    d. Funciones tipo “double” o “int” que contenga el código para realizar:
        - Suma
        - Resta
        - Producto
        - Cociente
        - Resto de la división