

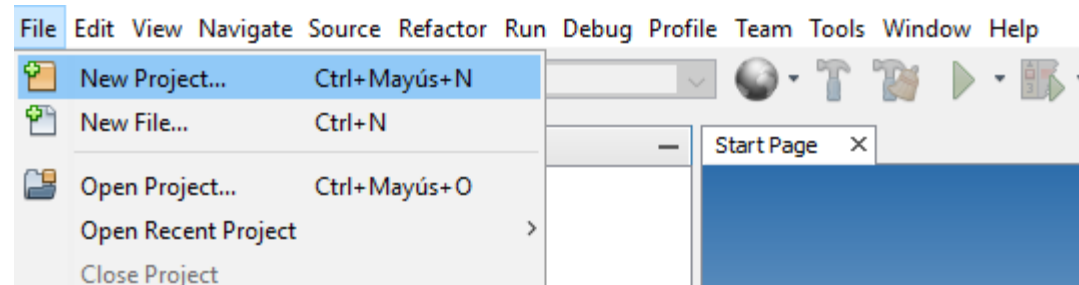
Guía de Programación: Fundamentos de Programación en Java

TEMARIO

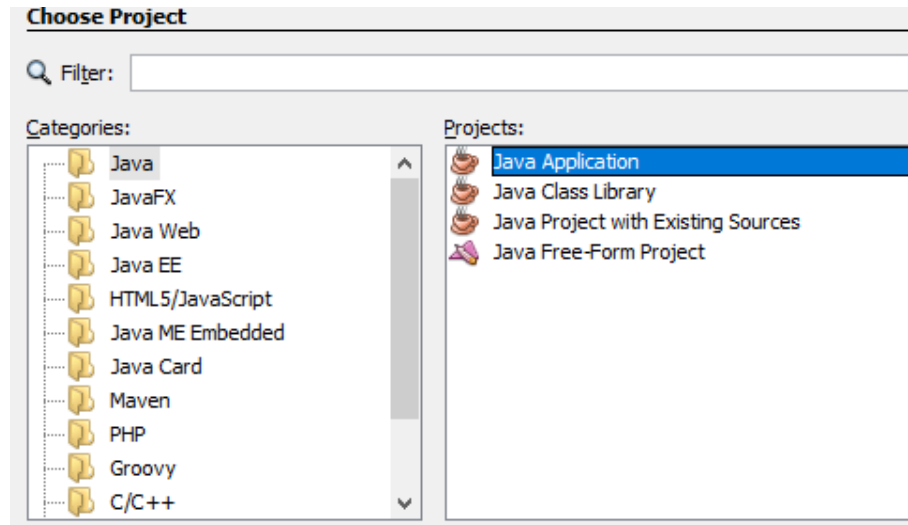
1. Variables, operadores y tipos en Java
2. Estructura Condicionales en Java
3. Estructuras repetitivas

Primeros pasos...

1.- Crear un nuevo proyecto

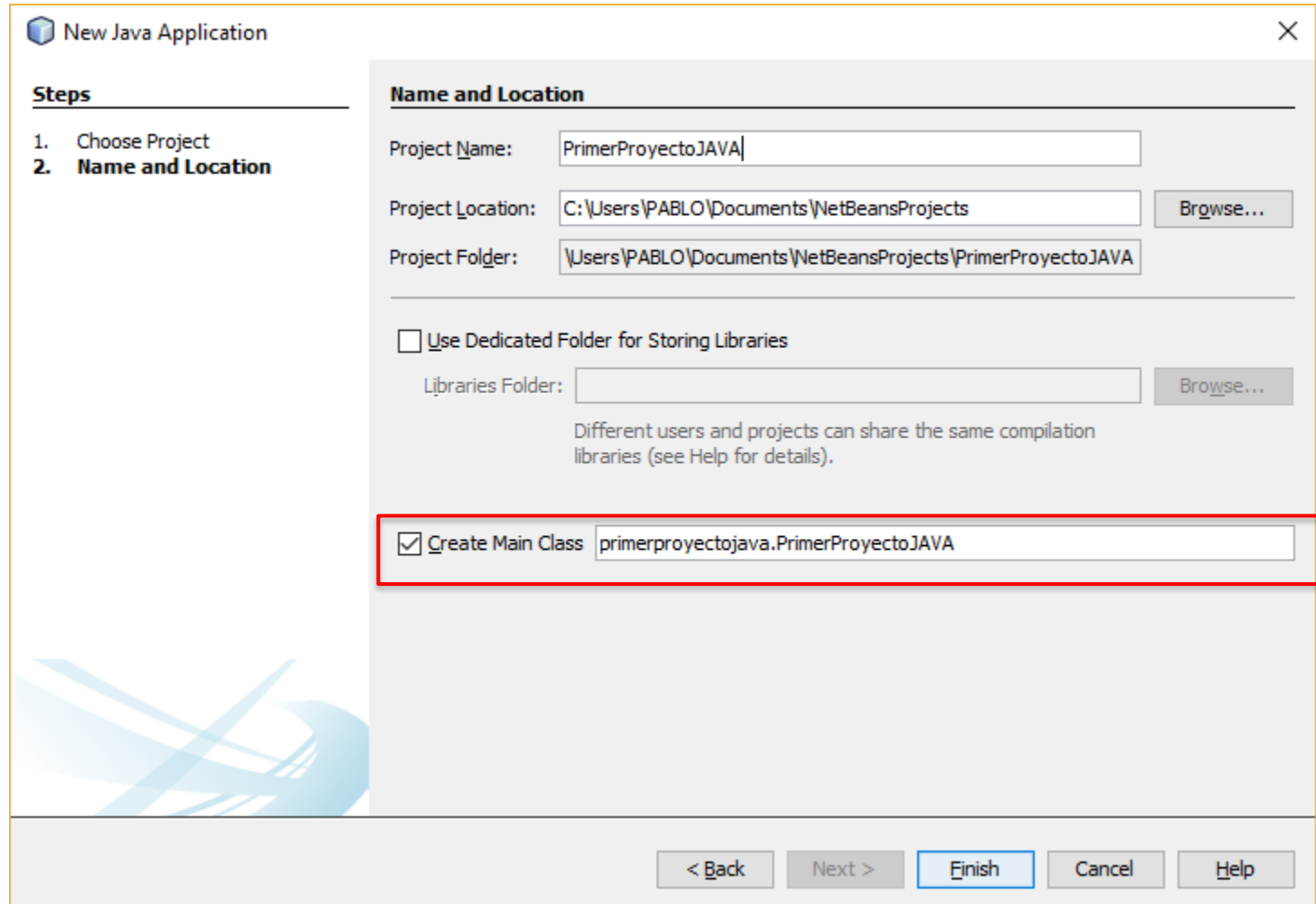


2.- Seleccionar tipo: Java Application



Primeros pasos...

3.- Escribir el nombre del proyecto y quitar check de Main Class



New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: PrimerProyectoJAVA

Project Location: C:\Users\PABLO\Documents\NetBeansProjects Browse...

Project Folder: \Users\PABLO\Documents\NetBeansProjects\PrimerProyectoJAVA

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

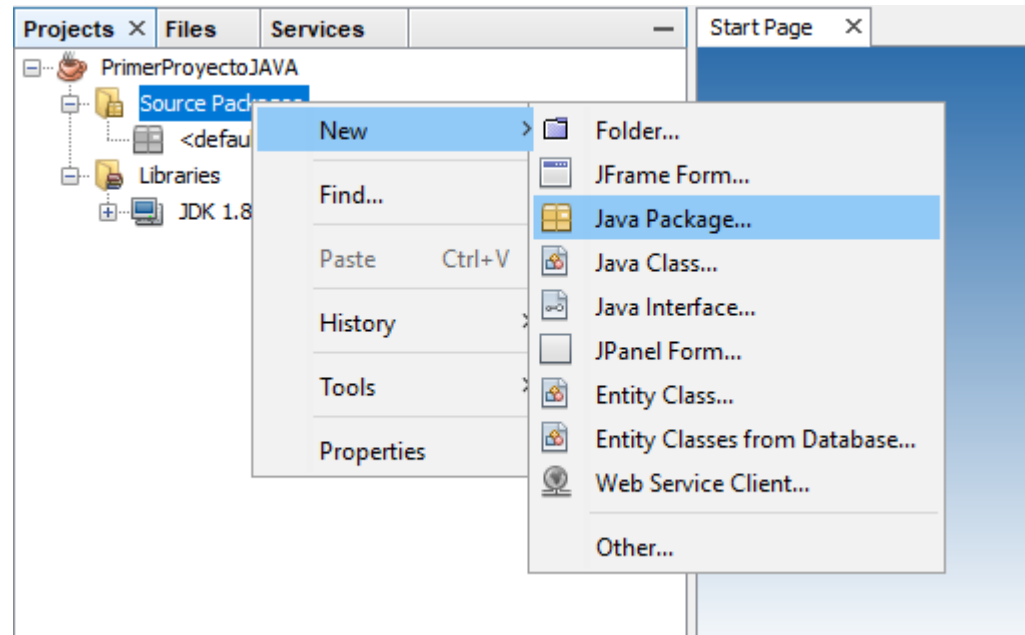
Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class primerproyectojava.PrimerProyectoJAVA

< Back Next > Finish Cancel Help

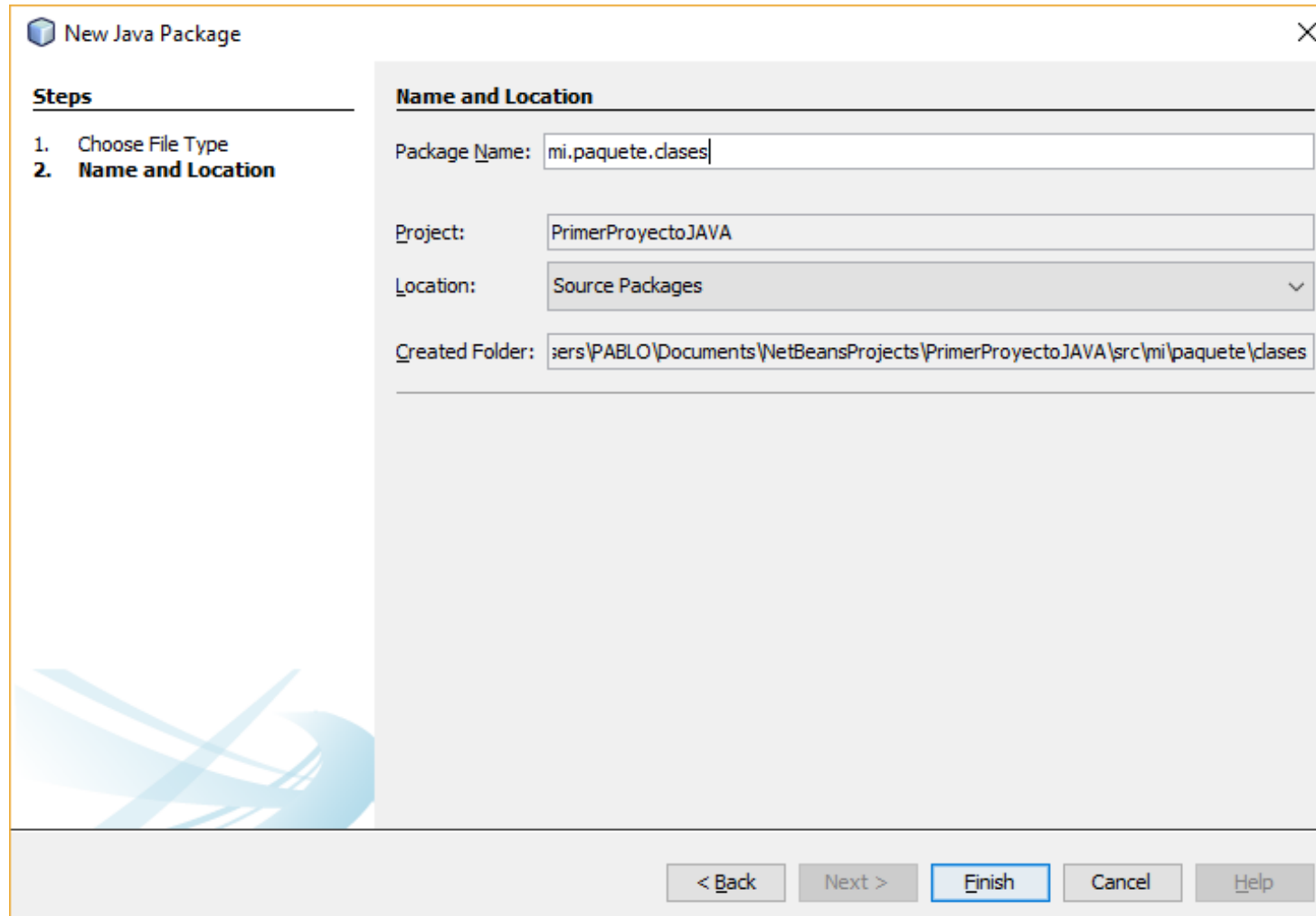
Primeros pasos...

4.- Crear un nuevo paquete (Source Package)



Primeros pasos...

5.- Escribir el nombre del paquete, las siguientes opciones mantenerlas



New Java Package

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Package Name:

Project:

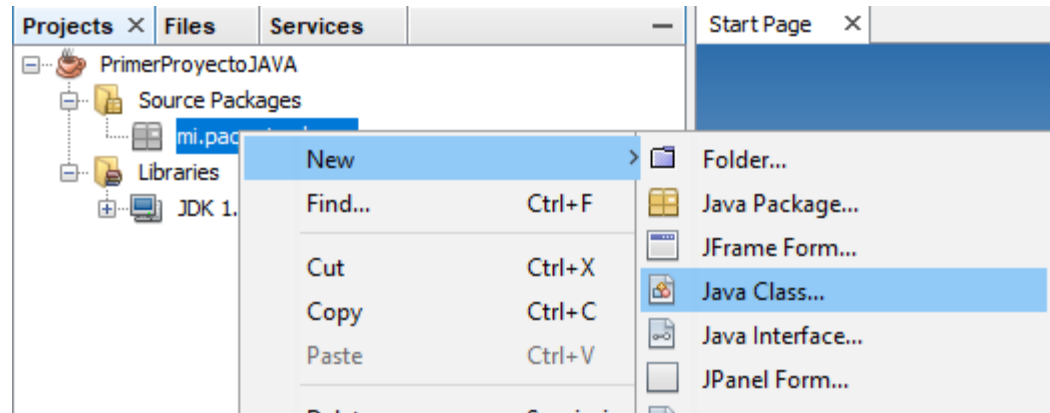
Location:

Created Folder:

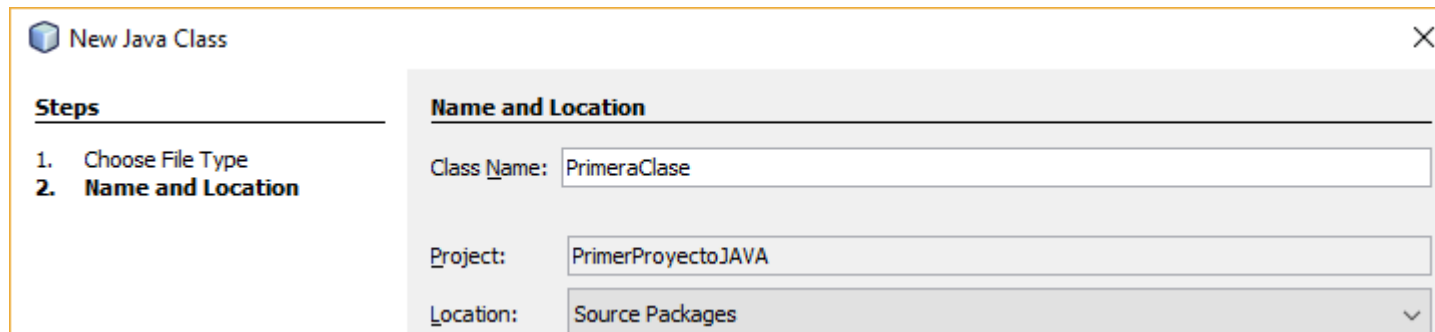
< Back Next > **Finish** Cancel Help

Primeros pasos...

6.- Crear una nueva clase, dentro del paquete creado



7.- Escribir el nombre, lo demás mantenerlo



Primeros pasos...

6.- Crear primeras instrucciones:

a) Función principal (shortcut: escribir psvm)

```
public static void main(String[] args)
{
    |
}

```

b) Instrucción para mostrar datos en consola (shortcut: escribir sout)

```
System.out.println("");

```




1.VARIABLES, OPERADORES Y TIPOS EN JAVA

1.1. VARIABLES EN JAVA

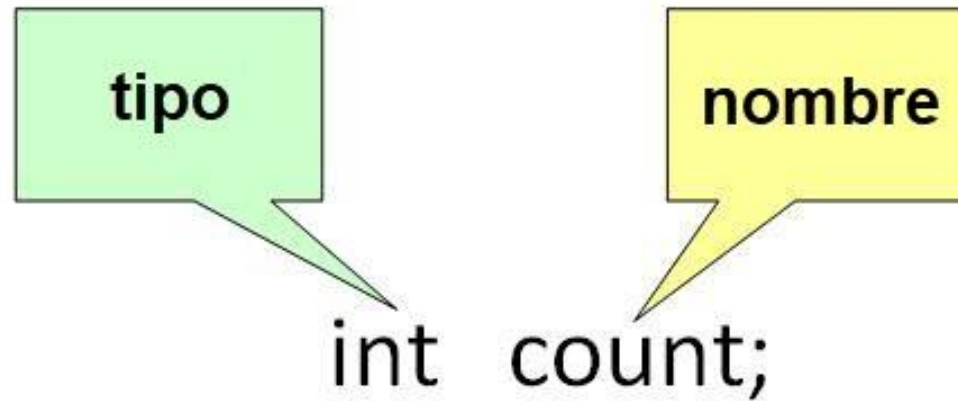


Diagram illustrating the components of a variable declaration in Java:

```
int count;
```

- tipo** (type): Points to the `int` keyword.
- nombre** (name): Points to the `count` variable name.

1.1.1 Declaración y tipos de datos primitivos

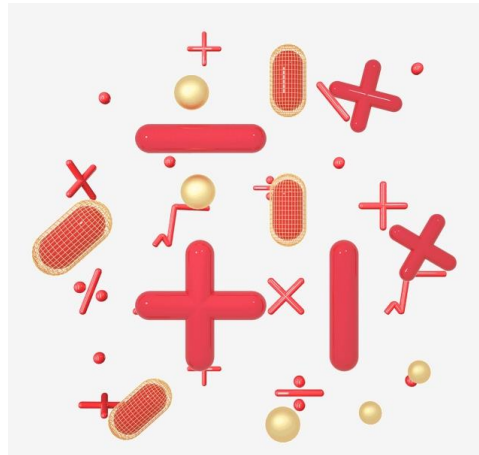
Tipos de datos primitivos de java

Dato	Tipo	Bits	Rango
carácter	char	16	0 a 65535
entero	byte	8	-128 a 127
	short	16	-32768 a 32767
	int	32	-2147483648 a 2147483647
	long	64	-9223372036854775808 a 9223372036854775807
real	float	32	-3.4×10^{38} a -1.4×10^{-45} , 1.4×10^{-45} a 3.4×10^{38}
	double	64	-1.7×10^{308} a -4.9×10^{-324} , 4.9×10^{-324} a 1.7×10^{308}
booleano	boolean	1	true, false

1.1.2. Ejemplo sobre declaración y tipos de datos primitivos

```
public static void main(String[] args) {  
    int edad; // Tipo de dato y nombre de variable  
    double precio = 0.0; // inicializar una variable al declarar  
    // Tipos de datos primitivos - Enteros  
    byte a; // 8 bits  
    short b; // 16 bits  
    int c; // 32 bits  
    long d; // 64 bits  
    // Tipos de datos primitivos - Punto flotante o decimales  
    float x; // 32 bits  
    double y; // 64 bits  
    char r = 'R'; // Tipo de dato primitivo - Textual  
    boolean estaAbierto = true; // // Tipos de datos primitivos - para valores lógicos true o false  
}
```

1.2. OPERADORES ARITMÉTICOS EN JAVA



1.2.1 Operadores matemáticos estándar

Operadores aritméticos

Operador	Significado	Ejemplo	Resultado
-	Resta	$a - b$	Resta de a y b
+	Suma	$a + b$	Suma de a y b
*	Multiplicación	$a * b$	Producto de a por b
/	División	a / b	Cociente de a entre b
%	Residuo	$a \% b$	Residuo de a entre b

1.2.2 Incrementos y decrementos

Operador	Nombre	Expresión de Ejemplo	Explicación
++	prefijo de incremento	++a	Incrementa a en 1, luego se usa el nuevo valor de a en la expresión en la que a aparece.
++	postfijo de incremento	a++	Usa el valor actual de a en la expresión en la que a aparece, y luego incrementa a a en 1.
--	prefijo de decremento	--b	Decrementa b en 1, luego el nuevo valor es usado en la expresión en la que b aparece.
--	postfijo de decremento	b--	Use el valor actual de b en la expresión en la que b aparece, y luego decremente a b en un 1.

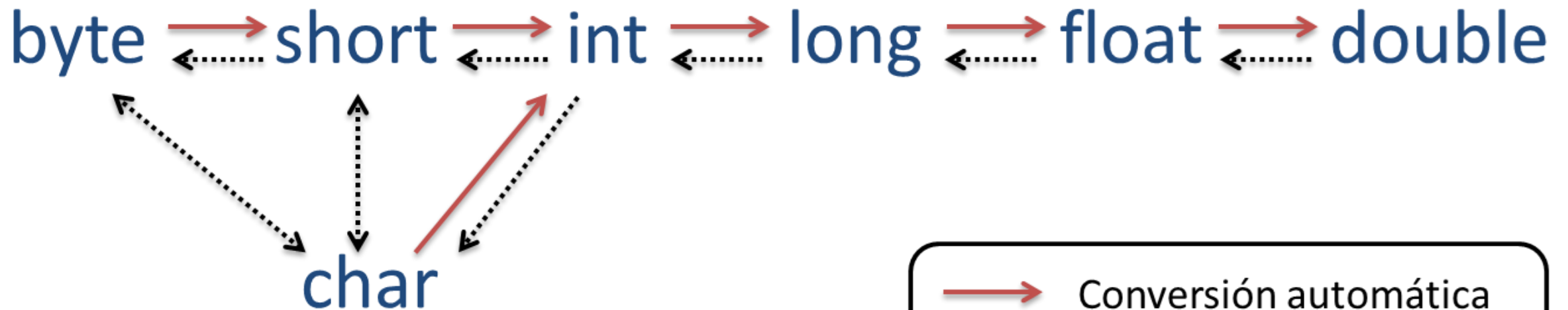
1.2.3 Incrementos y decrementos adicionales

$+$ =	$a + = 1$	$a = a + 1;$
$-$ =	$b - = 2$	$b = b - 2;$
$*$ =	$c + = 3$	$c = c + 3;$
$/$ =	$d / = 4$	$d = d / 4;$
$\%$ =	$e \% = 5$	$e = e \% 5;$

1.2.4. Ejemplo sobre Incremento y decremento

```
public static void main(String[] args) {  
    int c = 15;  
    int x, y, z, w;  
    x = c++; // A x se asigna el valor de c y luego se incrementa c  
    y = c; // A y se asigna el nuevo valor de c ya incrementado  
    z = ++c; // A z se asigna el nuevo valor de c ya incrementado  
    w = c; // a w se asigna el mismo valor asignado a z  
    System.out.println(x + ", " + y + ", " + z + ", " + w);  
}
```

1.3. CONVERSIÓN DE TIPOS DE DATOS EN JAVA



→ Conversión automática
←..... Requiere utilizar un cast

1.3.1. Conversión implícita

- Si al hacer la conversión de un tipo a otro se dan las 2 siguientes
- premisas:
 - Los dos tipos son compatibles.
 - El tipo de la variable destino es de un rango mayor al tipo de la variable que se va a convertir.
- Ejemplo:
 - `int pequeño = 6;`
 - `long grande = pequeño;`

1.3.2. Conversión explícita

- Si no es posible una conversión implícita será necesario hacer un casting de tipos. En caso de tipos de datos numéricos la conversión es por estrechamiento es decir recortando el dato para meterlo en la nueva variable.
- Ejemplo conversión explícita de tipos de datos enteros:
 - long grande = 6L;
 - int pequeño = **(int)** grande;

1.3.3. Ejemplo sobre Conversión de tipos

```
public static void main(String args[]) {  
    // Conversión implícita o automática  
    short corto = 34;  
    int largo = corto;  
    // Conversión explícita o casting  
    long largaso = 2345L;  
    int cortito = (int) largaso;  
    // Promoción en expresiones  
    short a=45,b=98;  
    // Error de compilación cuando c = a+b; por posible pérdida de datos  
    short c = (short) (a+b);  
}
```



2. ESTRUCTURAS CONDICIONALES EN JAVA

2.1. Operadores relacionales en Java

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
<code>==</code>	igual que	<code>7 == 38</code>	false
<code>!=</code>	distinto que	<code>'a' != 'k'</code>	true
<code><</code>	menor que	<code>'G' < 'B'</code>	false
<code>></code>	mayor que	<code>'b' > 'a'</code>	true
<code><=</code>	menor o igual que	<code>7.5 <= 7.38</code>	false
<code>>=</code>	mayor o igual que	<code>38 >= 7</code>	true

2.2. Operadores lógicos en Java

OPERADOR	NOMBRE	EJEMPLO	DEVUELVE VERDADERO CUANDO...
&&	y	<code>(7 > 2) && (2 < 4)</code>	las dos condiciones son verdaderas
	o	<code>(7 > 2) (2 < 4)</code>	al menos una de las condiciones es verdadera
!	no	<code>!(7 > 2)</code>	la condición es falsa

2.2. Operadores lógicos en Java

- Operadores de igualdad
 - Se lleva a cabo utilizando operadores de igualdad y desigualdad.
 - `exprIzquierda == exprDerecha`, se evalúa como verdadero (true) si `exprIzquierda` y `exprDerecha` son iguales
 - `exprIzquierda != exprDerecha`, se evalúa como verdadero (true) si `exprIzquierda` y `exprDerecha` no son iguales

2.3. Estructura if/else en Java

```
if (condición) {  
    bloque-de-sentencias-if  
}  
  
else {  
    bloque-de-sentencias-else  
}
```

2.3. Estructura if/else en Java

- Instrucción if (simple)
 - Principal forma de llevar a cabo la toma de decisiones en los programas. Su forma básica es:

```
boolean referencial = true;
boolean referencia2 = false;

if( referencial == referencia2)
{
    //secuencia de instrucciones
    //si la expresion es verdadera
}
```

operadores lógicos

```
int numero1 = 50;
int numero2 = 150;

if( numero1 <= numero2)
{
    //secuencia de instrucciones
    //si la expresion es verdadera
}
```

operadores aritméticos

2.3.1. Ejemplo sobre if/else en Java

```
public static void main(String[] args){  
    // Dado dos números indicar cual es el mayor  
    java.util.Scanner entrada = new java.util.Scanner(System.in);  
    System.out.println("Ingrese el primer número: ");  
    int a = entrada.nextInt();  
    System.out.println("Ingrese el segundo número: ");  
    int b = entrada.nextInt();  
    if (a>b)  
        System.out.println("El mayor es: "+a);  
    else  
        System.out.println("El mayor es: "+b);  
}
```

Estructuras de Control de flujo: Estructuras Selectivas

- Instrucción if (doble)
 - Opcionalmente podemos utilizar la expresión if-else

```
boolean referencia1 = true;
boolean referencia2 = false;

if( referencia1 == referencia2)
{
    //secuencia de instrucciones
    //si la expresion es verdadera
}

else
{
    //secuencia de instrucciones
    //caso contrario
}
```

operadores lógicos

```
int numero1 = 50;
int numero2 = 150;

if( numero1 <= numero2)
{
    //secuencia de instrucciones
    //si la expresion es verdadera
}

else
{
    //secuencia de instrucciones
    //caso contrario
}
```

operadores aritméticos

2.4. Estructura anidada if/else en Java

```
if(condicion){  
  acción...  
  if(condicion){  
    acción...  
  }else{  
    acción...  
  }  
}else{  
  acción...  
}
```

2.4. Estructura anidada if/else en Java

- Instrucción if (Múltiple)
 - Opcionalmente podemos anidar if-else o else if

```
int numero1 = 50;
int numero2 = 150;

if( numero1 <= numero2)
{
    //secuencia de instrucciones
    //si la expresion es verdadera
}

else
{
    if(numero1>numero2)
    {
        //secuencia de instrucciones
        //en forma secuencial
    }
}
```

```
int numero1 = 50;
int numero2 = 150;

if( numero1 <= numero2)
{
    //secuencia de instrucciones
    //si la expresion es verdadera
}

else if(numero1>numero2)
{
    //secuencia si cumple
    //con la siguiente condicion
}

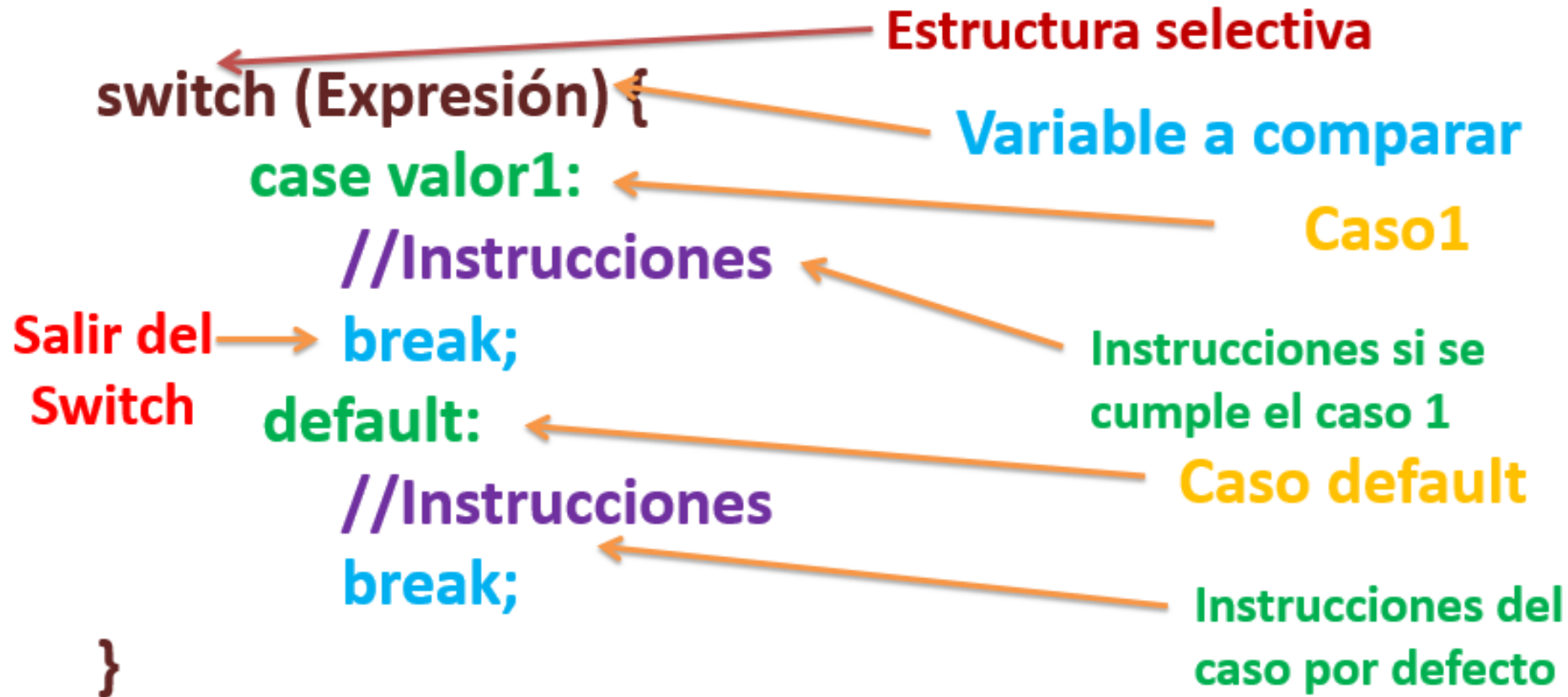
else if(numero1>=2)
{
    //secuencia si cumple
    //con la siguiente condicion
}

}
```

2.4.1. Ejemplo sobre if/else anidado en Java

```
public static void main(String[] args){  
    // Dado el número de mes indicar su cantidad de días  
    Scanner entrada = new Scanner(System.in);  
    System.out.println("Ingrese el número de mes: "); int mes = entrada.nextInt();  
    if (mes==1 || mes==3 || mes==5 || mes==7 || mes==8 || mes==10 || mes==12){  
        System.out.println("El mes tiene 31 días");  
    } else if (mes==2){  
        System.out.println("El mes tiene 28 o 29 días");  
    } else if (mes==4 || mes==6 || mes==9 || mes==11){  
        System.out.println("El mes tiene 30 días");  
    } else { System.out.println("Número de mes invalido..."); }  
}
```


2.5. Estructura switch en Java



2.5. Estructura switch en Java

- Instrucción switch
 - Se utiliza para elegir entre valores pequeños de tipo entero. Su forma general es:

```
switch (edad)
{
    case 25:
        System.out.println("es la edad 25");
        break;
    case 30:
        System.out.println("no es la edad");
        break;
    default:
        System.out.println("caso por default");
        break;
}
```

2.5.1. Ejemplo sobre switch en Java

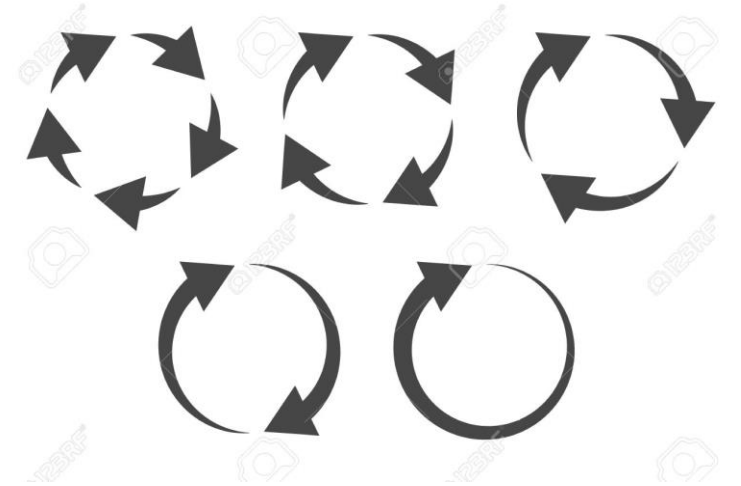
```
public static void main(String[] args){  
    // Dado el número de mes indicar su cantidad de días  
    Scanner entrada = new Scanner(System.in);  
    System.out.println("Ingrese el número de mes: ");  
    int mes = entrada.nextInt();  
    switch (mes){  
        case 1:  
        case 3:  
        case 5:  
        case 7:  
        case 8:  
        case 10:  
        case 12:  
            System.out.println("El mes tiene 31 días");  
            break;
```

```
        case 2:  
            System.out.println("El mes tiene 28 o 29 días");  
            break;  
        case 4:  
        case 6:  
        case 9:  
        case 11:  
            System.out.println("El mes tiene 30 días");  
            break;  
        default:  
            System.out.println("El número de mes no es válido...");  
            break;  
    }  
}
```

Operador condicional ?

- Operador condicional
 - En reemplazo de la sentencia if-else, podemos utilizar el operador condicional con los caracteres ? Y :
 - expresion1 ? expresion2 : expresion3
 - Si expresion1 es verdadera, entonces expresion2 es el valor condicional, en caso contrario expresion3 es el valor condicional

```
int bb=10;  
System.out.println(bb%2==0?"par":"impar");
```



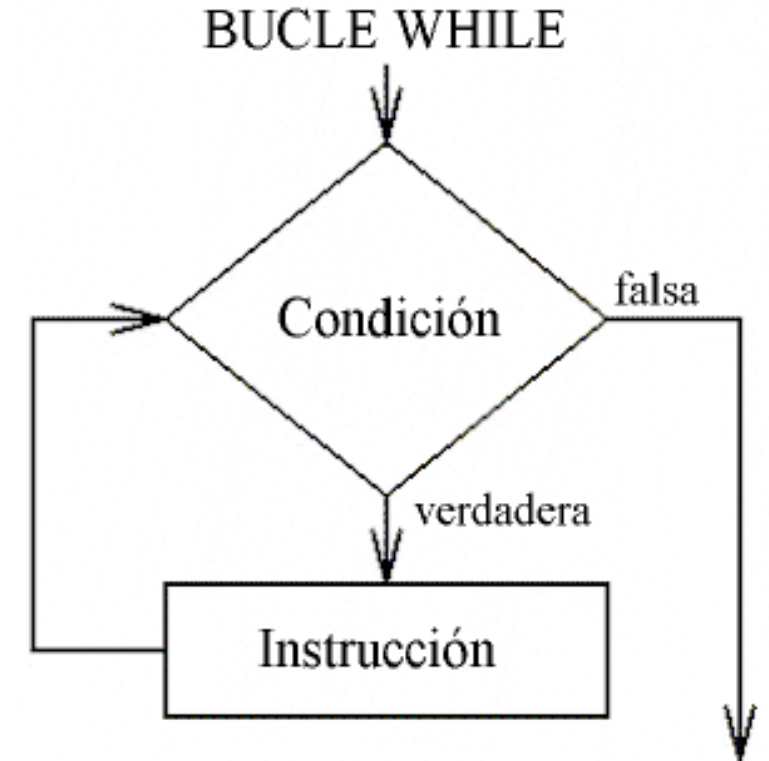
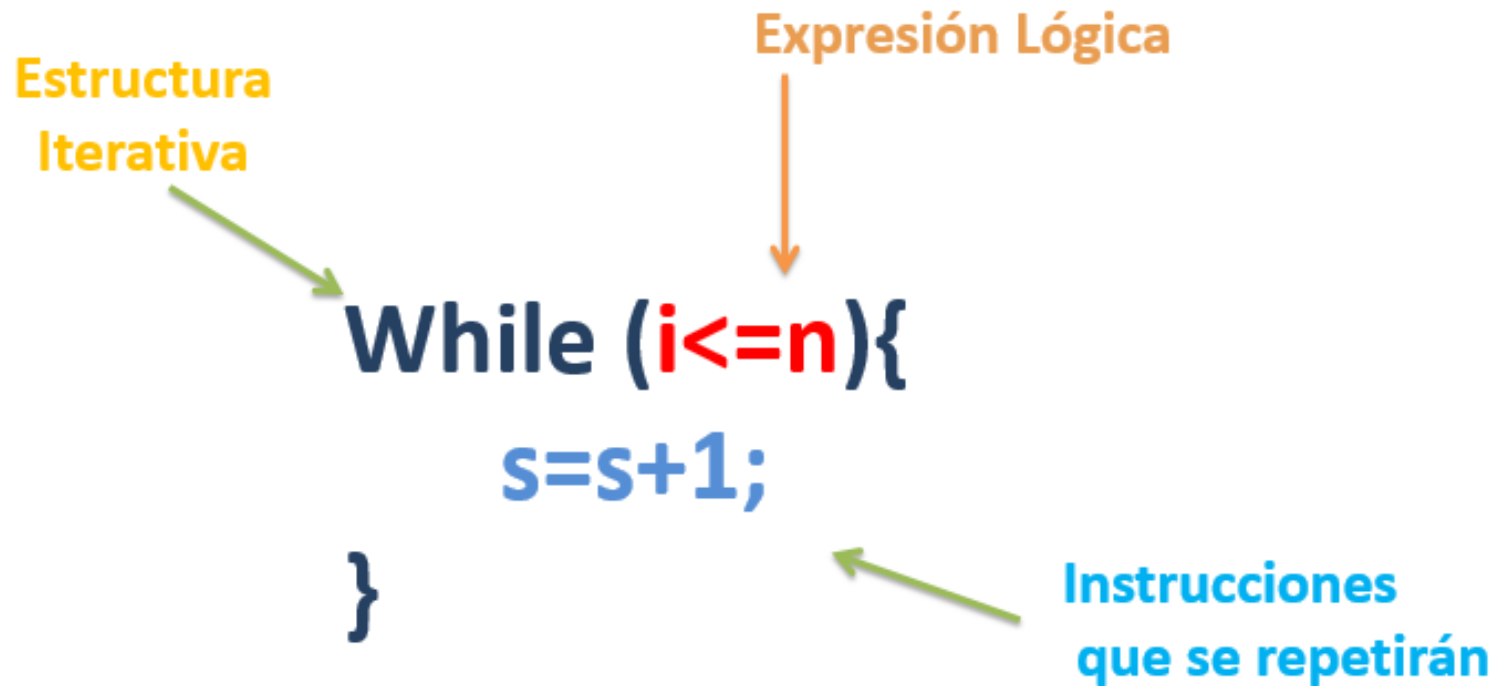
3. ESTRUCTURAS REPETITIVAS EN JAVA

Estructura Repetitiva: Instrucción while

- Sintaxis:

```
While (condicion)
{
    Instrucción 1;
    Instrucción 2;
    ...
}
```
- Condición es una expresión booleana (puede ser verdadera o falsa) que se evalúa al principio del bucle y antes de cada iteración de las sentencias.
- Si condición es verdadera, se ejecuta el bloque de sentencias y se vuelve al principio del bucle.
- Si la condición es falsa, no se ejecuta el bloque de sentencias y se continúa con la siguiente sentencia del programa.

3.1. Estructura While en Java



3.1.1. Ejemplo sobre while en Java

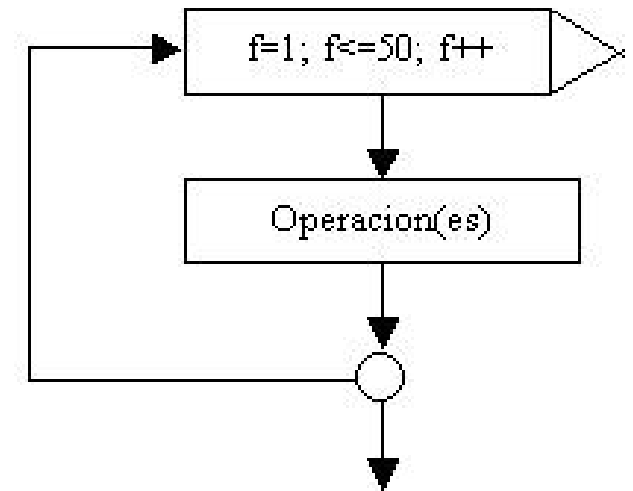
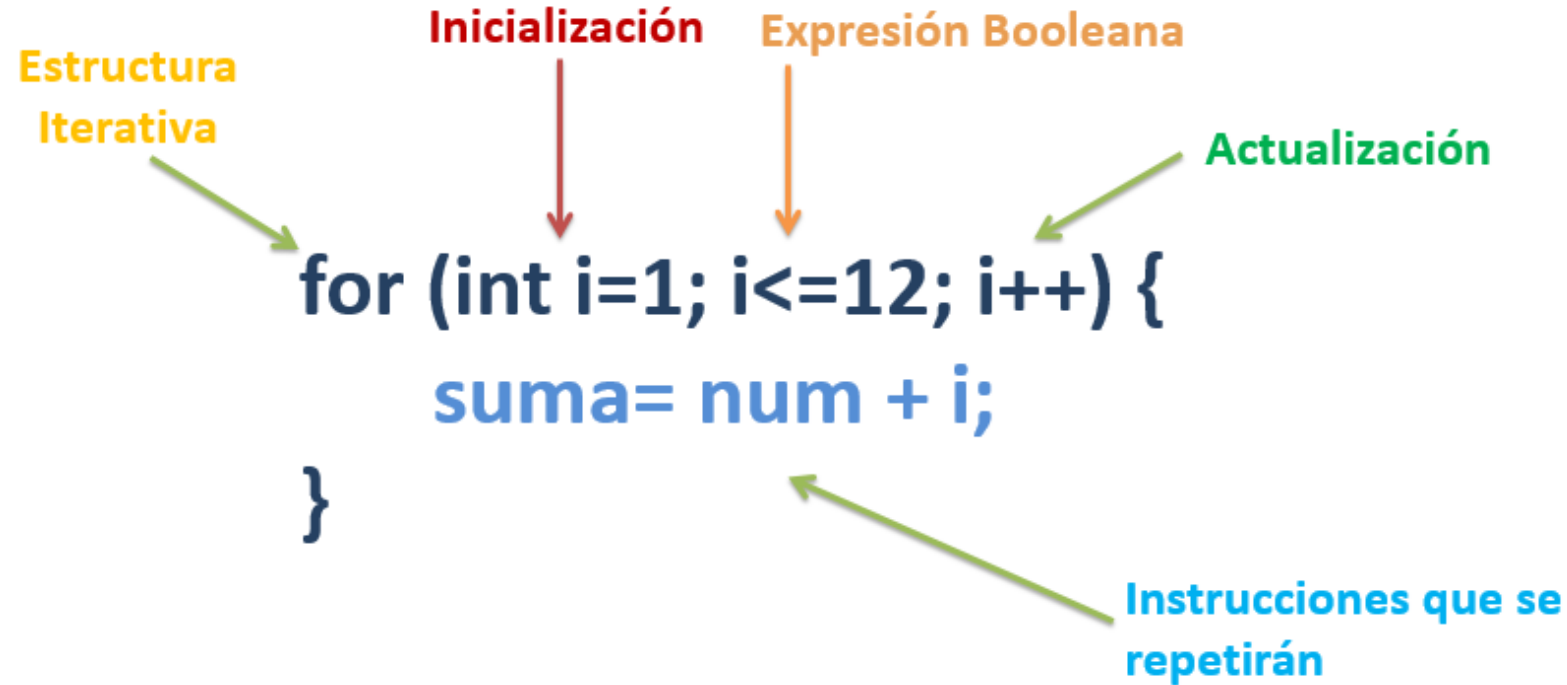
```
public static void main(String[] args) {  
    // Determinar el mayor de n números  
    Scanner entrada = new Scanner(System.in);  
    System.out.println("Ingrese n: ");    int n = entrada.nextInt();  
    int i = 0; // Contador  
    int mayor = 0;  
    while (i < n) {  
        i++;  
        System.out.println("Ingrese un número: "); int numero = entrada.nextInt();  
        if (numero > mayor) mayor = numero;  
    }  
    System.out.println("El mayor de n números es: "+mayor);  
}
```


Estructura Repetitiva: Instrucción for

- La instrucción for se ejecuta realizando primero la inicialización , después mientras que condición es verdadero se llevan a cabo las instrucciones y luego se realiza el incremento o decremento.
- Es posible anidar instrucción for
- Ejemplo:

```
for(int m=1;m<=20;m++)  
{  
    System.out.println(m) ;  
}
```

3.2. Estructura for en Java



3.2.1. Ejemplo sobre for en Java

```
public static void main(String[] args) {  
    // Determinar el mayor de n números  
    Scanner entrada = new Scanner(System.in);  
    System.out.println("Ingrese n: "); int n = entrada.nextInt();  
    int mayor = 0;  
    for (int i = 0; i < n; i++) {  
        System.out.println("Ingrese un número: "); int numero = entrada.nextInt();  
        if (numero > mayor) {  
            mayor = numero;  
        }  
    }  
    System.out.println("El mayor de n números es: " + mayor);  
}
```

Estructura Repetitiva: Instrucción do

- Sintaxis:

```
do
{
    Instrucción 1;
    Instrucción 2;
    ...
}
while (condición);
```

- El bloque de instrucciones se repite MIENTRAS que la condición sea verdadera.
- Es una estructura repetitiva que se ejecuta al menos una vez, a diferencia de las anteriores estructuras.
- La comprobación se ejecuta después de haber realizado el grupo de instrucciones especificadas.

3.3. Estructura do-while en Java

Estructura
Iterativa

do{

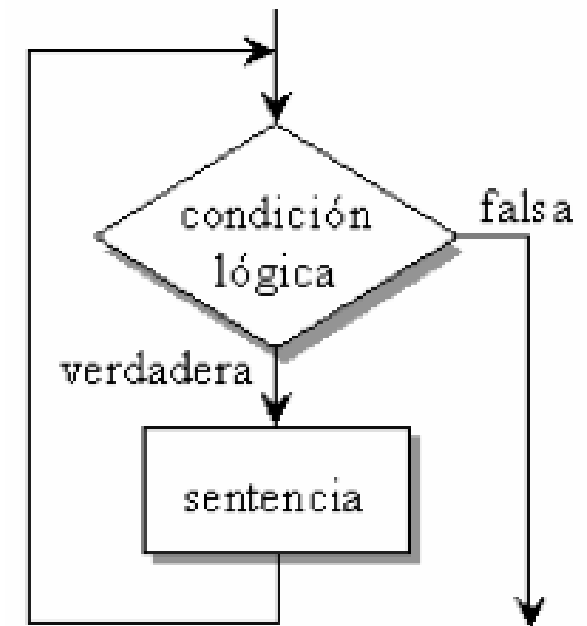
s=s+1;

}

While (nota<=0 || nota>=20)

Instrucciones
que se repetirán

Expresión Lógica



3.3.1. Ejemplo sobre do-while en Java

```
public static void main(String[] args) {  
    // Determinar el mayor de n números  
    Scanner entrada = new Scanner(System.in);  
    System.out.println("Ingrese n: "); int n = entrada.nextInt();  
    int mayor = 0;  
    int i = 0;  
    do {  
        i++;  
        System.out.println("Ingrese un número: "); int numero = entrada.nextInt();  
        if (numero > mayor) { mayor = numero; }  
    } while (i<n);  
    System.out.println("El mayor de n números es: " + mayor);  
}
```

3.4. Comparando estructura de bucle

- Usa el bucle while para iterar indefinidamente sentencias de cero o más veces.
- Use el bucle do-while para iterar indefinidamente instrucciones de uno o más veces.
- Utilice el bucle for para repetir las instrucciones un número predefinido de veces.

Clase: SCANNER

- Para leer podemos usar el método nextXxx() donde Xxx indica el tipo, por ejemplo nextInt() para leer un entero, nextDouble() para leer un double, etc
- Ejemplo

```
Scanner sc = new Scanner(System.in);  
System.out.println("Ingrese un numero entero");  
int a = sc.nextInt();
```


Entrada y Salida de Datos (Ejercicio)

- Desarrolle un programa que permita ingresar el nombre del usuario, luego muestre un mensaje de bienvenida: “Bienvenido XXX” , a continuación solicite ingresar la edad para mostrar el mensaje “Gracias XXX, su edad ingresada es YYY”. Finalmente el programa debe mostrar un mensaje indicando cual será la edad del usuario dentro de 10 años: “Estimado XXX su edad dentro de 10 años será YYY”.