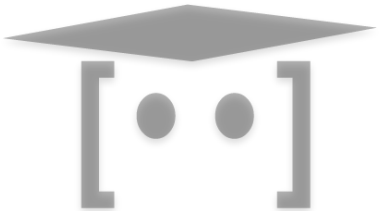
JAVA

Profesor: Nahuel Meza

Cod.Ar[••]

Repaso

¿Qué vimos la clase pasada?....



Repaso

¿Qué imprime el siguiente código?

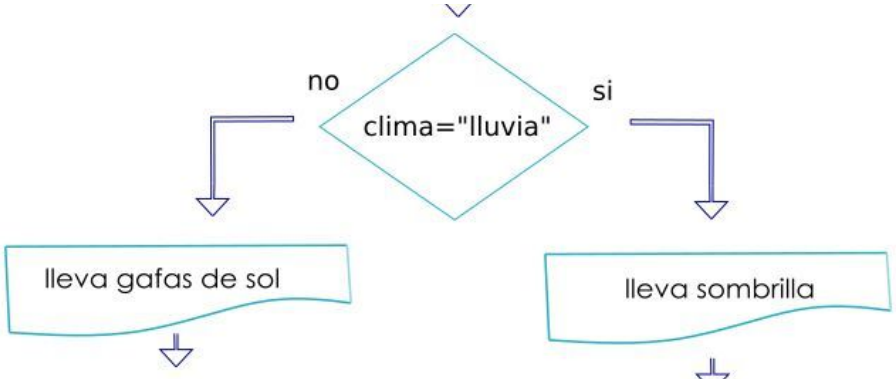
```java
boolean esFalso = false;
boolean mayorQueCinco = 10 > 5;

System.out.println(esFalso || mayorQueCinco);

System.out.println(mayorQueCinco && true);
```

¿Qué vamos a ver hoy?

- Condicionales
- Ciclos

Condicionales

Una **condición booleana** es una expresión de resultado booleano utilizado para definir **cuándo debe ejecutarse** cierta parte del código



Condicionales

Generalmente, sobre todo en los lenguajes actuales, diferenciamos estas secciones de código con **bloques de código.**

En java, los bloques de código están delimitados por corchetes {}.

Si una variable se crea en un bloque, **solo existe** en ese bloque y sub-bloques.

Condicionales

Incluso podemos observar como nuestros programas se encuentra **dentro** del bloque de codigo del metodo **main**

```
Run | Debug
public static void main(String[] args){
    // Int, String, Float, Bool
    int num = 12;
    System.out.println(num);
}
```

Condicionales

Si quisiéramos que en nuestro bloque de código se ejecute algo solamente si pasa “algo” en específico, podemos utilizar los **if/else.**

El cual define un nuevo bloque de código que se ejecuta solamente si se cumple la condición booleana especificada.

Condicionales

se puede escribir de estas tres formas.

```
if(condición){
   // solo se ejecuta si condicion == true
}
// flujo normal
```

Condicionales

se puede escribir de estas tres formas.

```
if(condición){
    // solo se ejecuta si condicion == true
}
else{
    // se ejecuta si no se cumplio la primera condicion
}
// flujo normal
```

Condicionales

se puede escribir de estas tres formas.

```
if(condición){
    // solo se ejecuta si condicion == true
}
else if(condicion2){
    // se ejecuta si no se cumplio la primera condicion
    // pero si esta nueva
}
else{
    // se ejecuta si no se cumplio ninguna
}
```

Condicionales ejemplo

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

    int a = 2;
    int b = 0;
    int resultado = 0;
    if(b == 0){
        System.err.println(x:"no se puede dividir por 0");
    }
    else{
        resultado = a / b;
    }
    System.err.println(resultado);

}
```

Condicionales ejemplo

```java
int numero = 14;

if(numero > 6){
    System.out.println(x:"El numero es mayor que 6");
}
else if(numero < 6){
    System.out.println(x:"El numero es menor que 6");
}
else{
    System.out.println(x:"El numero es 6");
}
```

Se pueden encadenar infinitos if, else if.
Se ejecuta el primero que es cierto de arriba a abajo.

Condicionales ejemplo

```java
int res = 0;
int num1 = 2;
int num2 = -3;

if(num2 > 0){
    res = num1 + num2;
}
else{
    res = num1 - num2;
}

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

Condicionales: ternario

Para este último ejemplo, existe un condicional mucho más estético y dinámico. El **operador ternario**.

Se utiliza cuando el **contenido** de una variable depende del resultado de una condición booleana.

```
var = condicion ? res1 : res2;
```

Condicionales: ternario

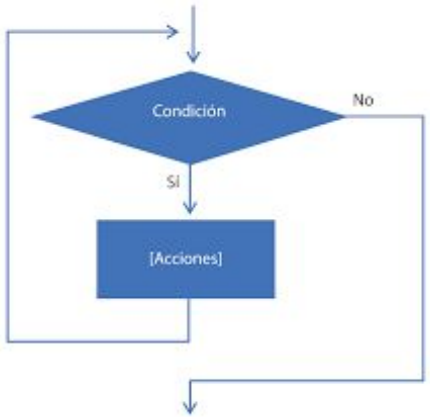
```
int res = 0;
int num1 = 2;
int num2 = -3;

res = num2 > 0 ? num1+num2 : num1-num2;
```

Condicionales

Los condicionales nos permiten controlar el **flujo** del programa. Otorgando una herramienta de **dinamismo y robustez** al mismo

Ciclos

Los ciclos, permiten que un bloque de código se **repita constantemente** siempre y cuando se cumpla su **condición booleana.**



Existen 3 formas principales de escribir estos ciclos.

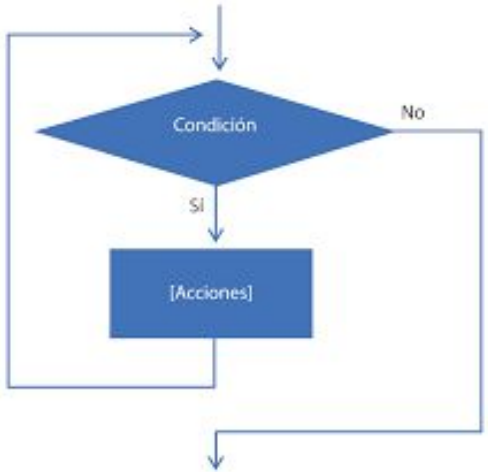
Ciclos: while

El **while**, primero chequea si su condición se cumple. De ser así ejecuta el bloque de código. (*) Al terminar de ejecutar chequea nuevamente la condición, y si se cumple ejecuta nuevamente el bloque de código. De no ser así, se sigue con el programa fuera del bloque.

Se repite (*) hasta que no se cumpla la condición.

Ciclos: while

```
while(condicion){
   // bloque de código
}
// flujo siguiente
```

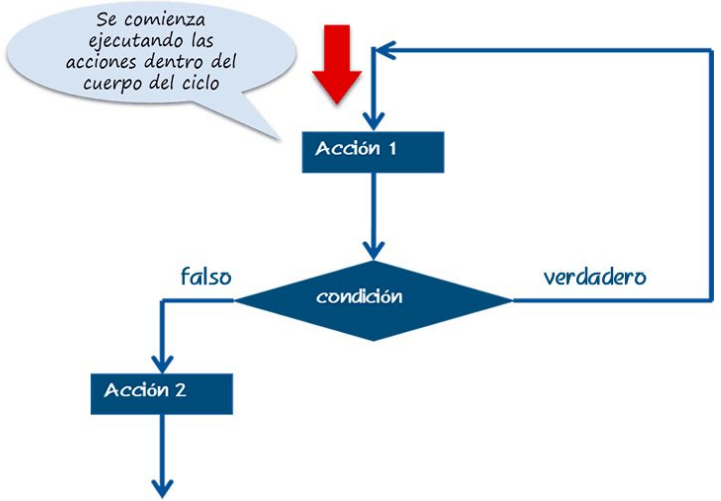


Ciclos: do ... while

El **do..while**, funciona de manera similar al while, pero permite que se ejecute siempre la primera vez del bloque de código, luego empieza a chequear la condición para saber si repetir o no.

Ciclos: do ... while

```
do{
   // bloque de código
} while (condicion)
// flujo siguiente
```

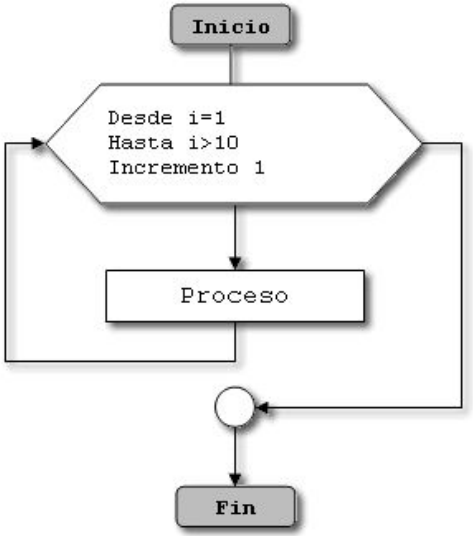


Ciclos: for

El **for**, a diferencia de los demás, la condición booleana depende de una variable que se crea en la primera instancia del ciclo, y se modifica antes de empezar la siguiente ejecución, y luego se chequea la condición.

Ciclos: for

```
for(incializacion; condicion; modificacion){
    // bloque
}
// flujo siguiente
```



break / continue

Dos expresiones claves son **break** y **continue** para cualquier tipo de ciclo en java. Con solo escribirlas en una línea de código dentro del bloque del ciclo sucede que:

- *break*: se **termina el ciclo** abruptamente
- *continue:* se **termina la ejecución del bloque** y se procede a chequear la condición del ciclo

Ejemplos de ciclos

Pasamos a ver ejemplos de ciclos y estas dos palabras reservadas.