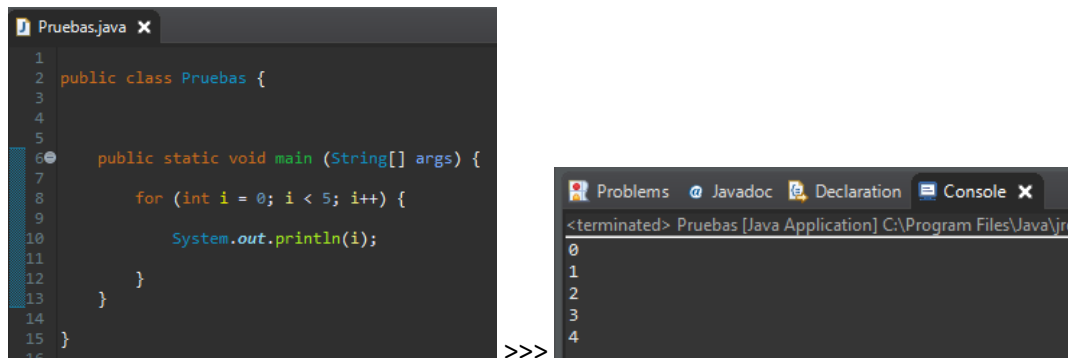


Los bucles *for* funcionan igual que un bucle *while*, pero son mucho más cómodos de usar a la hora de recorrer *arrays*, cosa que ya veremos en su PDF, tienen la siguiente estructura:



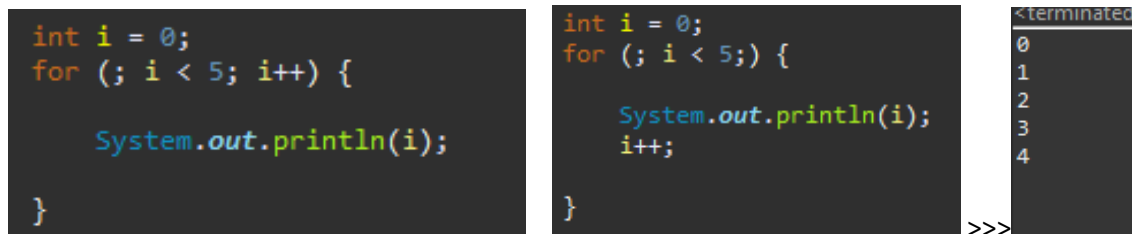
```
1 public class Pruebas {
2
3
4
5
6
7     public static void main (String[] args) {
8         for (int i = 0; i < 5; i++) {
9             System.out.println(i);
10        }
11    }
12
13
14
15 }
```

>>>

```
<terminated> Pruebas [Java Application] C:\Program Files\Java\jre
0
1
2
3
4
```

Tienen la siguiente forma -> *for* (*SENTENCIA*; *CONDICION*; *SENTENCIA*) { *//hacer algo* }

Pongo **sentencia** porque no es obligatorio que vaya la declaración de una variable, aunque es lo más común, por ponernos estrictos no hace falta que haya ni siquiera una sentencia, **lo único obligatorio es que se encuentre los dos ;**, pongo dos ejemplos donde se ve esto:



```
int i = 0;
for (; i < 5; i++) {

    System.out.println(i);

}
```

```
int i = 0;
for (; i < 5;) {

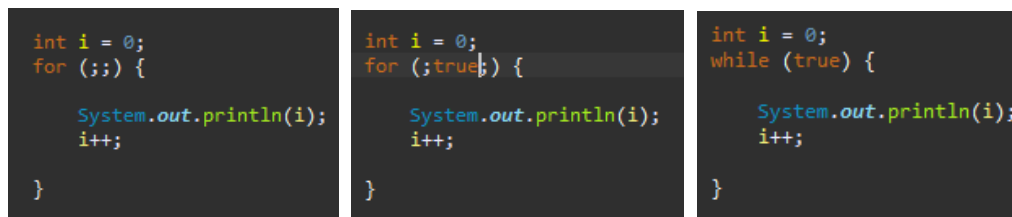
    System.out.println(i);
    i++;

}
```

>>>

```
<terminated>
0
1
2
3
4
```

Como veis en la imagen del medio (la que no tiene ninguna sentencia dentro del *for*) tiene la misma forma que un bucle *while*, exactamente el mismo, si cambiasemos *for* → *while*, y quitasemos los dos ; de dentro del *for*, tendríamos un bucle *while*, tampoco hace falta poner una condición a nuestro bucle *for*, pero si no se lo ponemos, tendremos un bucle infinito, es como poner *while* (*true*):



```
int i = 0;
for (;;) {

    System.out.println(i);
    i++;

}
```

```
int i = 0;
for (;true;) {

    System.out.println(i);
    i++;

}
```

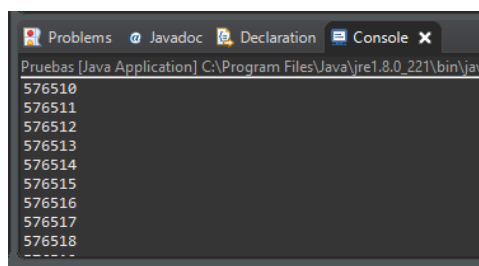
```
int i = 0;
while (true) {

    System.out.println(i);
    i++;

}
```

Estos 3 bloques de código son **equivalentes**, y dan el mismo resultado.

>>>



```
Pruebas [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\jav
576510
576511
576512
576513
576514
576515
576516
576517
576518
-----
```

Como veis no se termina.

Voy a poner un bucle *while* y un *for* que hacen exactamente lo mismo:

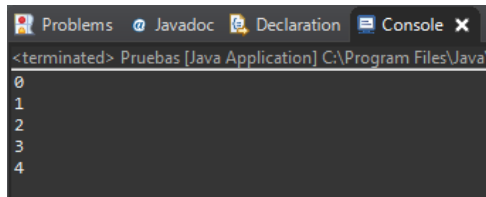
```
int i = 0;
while (i < 5) {

    System.out.println(i);
    i++;
}
```

```
for (int i = 0; i < 5; i++){

    System.out.println(i);
}
```

>>>



```
Problems Javadoc Declaration Console x
<terminated> Pruebas [Java Application] C:\Program Files\Java\
0
1
2
3
4
```

En un *for* ocurre lo siguiente:

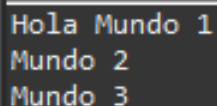
Se ejecuta la primera sentencia **una sola vez**, mientras se cumpla la condición se ejecuta el bloque de código que contiene, y **la segunda sentencia se ejecuta siempre al final del todo**, justo antes de comprobar la condición.

Por último quiero enseñar que puede haber cualquier sentencia dentro del *for*:

```
int i = 0;
for (System.out.print("Hola "); i < 3; System.out.println(i)) {

    System.out.print("Mundo ");
    i++;
}
```

>>>



```
Hola Mundo 1
Mundo 2
Mundo 3
```

Cabe mencionar que **solo** puede haber **una única sentencia** en la declaración de sentencias del *for*, una o ninguna, y que en la parte de la condición, ésta puede ser tan larga como queramos, al igual que en la condición de los *while* y los *ifs*, pero en la UPM no les gusta que se pongan condiciones muy largas, normalmente se pone **una sola condición**, y si son algoritmos de búsqueda en un *array* o similar podemos poner hasta dos condiciones, pero eso ya lo veremos en su apartado.