FOR

Sintaxis del for

Es la instrucción de bucle más utilizada en Java por su sintaxis compacta y precisa. La clave del éxito es que en la cabecera del for se indica de forma muy concisa como empieza el bucle, como se sale y como avanza.

```
for(int i=0;i<4;i++)
    System.out.println(i + " ");
```

El formato anterior del for es el más habitual, pero la sintaxis del for es muy elástica y admite bastantes variantes, por ejemplo, se puede suprimir cualquiera de las tres expresiones de la cabecera. Son expresiones equivalentes a la anterior las siguientes:

Sin expresión de inicialización del bucle

```
int i = 0;
for(;i<4;i++)
  System.out.println(i + " ");
```

ojo, en el primer caso, la variable i es local al for, no se conoce esa variable i fuera del for, en cambio en el segundo caso, el alcance de i pertenecería a un bloque más externo. También es equivalente a lo anterior declarar i fuera pero inicializar dentro.

```
int i;
for(i=0;i<4;i++)
  System.out.println(i + " ");
```

Sin expresión de salida

```
for(int i=0;;i++){
  if(i>=4)
     break;
  System.out.println(i + " ");
}
```

Con la condición de salida vacía nos vemos forzados a utilizar break. La instrucción break en un bucle fuerza la salida de dicho bucle. La vemos, más adelante

Sin expresión de incremento de contador

```
for(int i=0;i<4;){
  System.out.println(i + " ");
   i++;
}
```

este formato nos permitiría dentro del bucle justo en que momento queremos incrementar i, por ejemplo, antes del println() con lo que varío el efecto del programa en este caso

Sin utilizar nada en la cabecera del for , el ejemplo inicial tendría este terrible aspecto:

```
class Unidad3 {
      public static void main(String[] args) {
         int i=0;
```

```
        for(;;){
            if(i>=4)
              break;
            System.out.println(i++ + " ");
        }
      }
}
```

Como último ejemplo, poco importante, utilizando el operador coma ",", puedo tener más de una expresión de inicialización y de incremento. Por ejemplo:

```
int i,j;
for(i=0,  j=10; i<j;i++,j--)
  System.out.println("i y j valen:" + i + "  y  " + j);
```

O equivalentemente

```
//int i,j;
  for(int i=0,   j=10; i<j;i++,j--)
      System.out.println("i y j valen:" + i + "  y  " + j);
```

Bucles infinitos con for

Hay muchas posibilidades la forma "clásica" es simplemente

```
for(;;){
   System.out.println("me verás hasta el fin de los tiempos");
}
```

EJERCICIOS. ¡ HACERLOS CON FOR!

Ejercicio U3_B5_1: imprimir los números del 0 al 100 ambos inclusive.

Ejercicio U3_B5_2: imprimir los números del 100 al 0 en orden decreciente.

Ejercicio U3_B5_3: imprimir la suma de los 100 primeros números naturales(del 1 al 100 ambos inclusive)

Ejercicio U3_B5_4: El siguiente código intenta resolver, sin éxito el ejercicio U3_B5_3. ¿Cuál es el problema?

```
class Unidad3{
   public static void main(String[] args) {
      int suma=0;
      for(int i=1;i<101;i++)
         suma=+i;
      System.out.print(suma);
    }
}
```

Si no ves directamente el problema, usa el tracer de Bluej o debugger netbeans para observar el comportamiento del bucle.

Ejercicio U3_B5_5: imprimir los números pares entre 0 y 100

Ejercicio U3_B5_6: imprimir los números impares hasta el 100 e imprimir cuantos

impares hay en total.

Ejercicio U3_B5_7: imprimir todos los números naturales que hay desde la unidad hasta un número introducido por teclado.

Ejercicio U3_B5_8: Imprime una tabla que muestra los números de 0 a 127 en decimal con su equivalente binario, octal y hexadecimal. Consulta en el API la clase Integer y localiza los métodos que te permiten convertir un entero en un String binario, octal y hexadecimal. Ejemplo de salida(parcial)

```
L:\Programacion>java Unidad3
decimal binario octal   hexadecimal
0       0       0       0
1       1       1       1
2       10      2       2
3       11      3       3
```

Ejercicio U3_B5_9: Imprime la cuenta de 50 a 0 de -2 en -2 de 3 formas: con while, dowhile y for. Ejemplo de salida.

```
run:
50 48 46 44 42 40 38 36 34 32 30 28 26 24 22 20 18 16 14 12 10 8 6 4 2 0
50 48 46 44 42 40 38 36 34 32 30 28 26 24 22 20 18 16 14 12 10 8 6 4 2 0
50 48 46 44 42 40 38 36 34 32 30 28 26 24 22 20 18 16 14 12 10 8 6 4 2 0
```

Ejercicio U3_B5_10: Con for imprime la progresión 1 2 4 8 16 32 ....1024. Es decir, empezando en 1, cada elemento de la progresión dobla en su valor al anterior.

Ejercicio U3_B5_11: Haz 20 tiradas de un dado(usar Random) e imprime su resultado en 4 filas de 5 resultados cada una como en el ejemplo.

```
3 2 5 4 4
2 4 1 4 1
2 4 1 5 1
5 5 6 4 6
```

Cada ejecución, evidentemente, tendrá valores diferentes. En el ejemplo anterior, la primera tirada fue un 3, luego salió un 2, luego 5, 4, 4, 2, etc. Es decir cada 5 tiradas cambiamos de fila

Ejercicio U3_B5_12: Simulamos tirar un dado usando la clase Random. Tiramos el dado 6000000 de veces(6_000_000). Vamos almacenando la cantidad de veces que sale cada cara. Si realmente se generan números aleatorios los valores obtenidos para cada cara serán aceptablemente próximos. Ejemplo:

```
L:\Programacion>java Unidad3
Cara 1: 999999
Cara 2: 999790
Cara 3: 999623
Cara 4: 1000511
Cara 5: 1000461
Cara 6: 999616
```

Ejercicio U3_B5_13: Mejorar el ejercicio Ejercicio U2_B11_E2 de "cara o cruz" de forma que pueda repetir tiradas hasta que se canse el usuario

```
run:
¿Cara o Cruz?
Cara
ACIERTO
¿otra tirada(si/no)?
si
¿Cara o Cruz?
Cruz
FALSO
¿otra tirada(si/no)?
no
BUILD SUCCESSFUL (total time:
```

Ejercicio U3_B5_14: Utilizando for anidados consigue la siguiente impresión

11
12
13
21
22
23
31
32
33

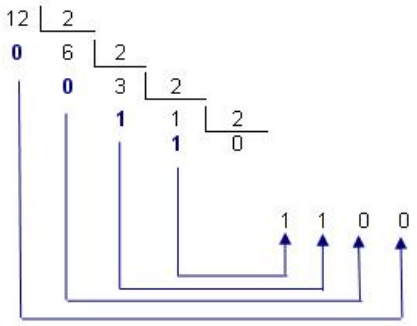
Ejercicio U3_B5_E15: Imprime las tablas de multiplicar del 1 al 9. Utilizar for anidado. Ejemplo de impresión

```
C:\Users\niinjashyper>java Unidad3.java
Tabla de multiplicar del 1
----------------------------
        1x1=1
        1x2=2
        1x3=3
        1x4=4
        1x5=5
        1x6=6
        1x7=7
        1x8=8
        1x9=9

Tabla de multiplicar del 2
----------------------------
        2x1=2
        2x2=4
        2x3=6
        2x4=8
        2x5=10
        2x6=12
        2x7=14
        2x8=16
        2x9=18

Tabla de multiplicar del 3
----------------------------
        3x1=3
        3x2=6
        3x3=9
        3x4=12
        3x5=15
        3x6=18
        3x7=21
        3x8=24
        3x9=27

Tabla de multiplicar del 4
----------------------------
        4x1=4
        4x2=8
        4x3=12
        4x4=16
        4x5=20
        4x6=24
```

Ejercicio U3_B5_E16: Utilizando doble anidación (3 niveles de for) consigue todas las combinaciones posibles de las letras A, B, y C.

```
lozano@infol06:~$ java  Unidad2
AAA
AAB
AAC
ABA
ABB
ABC
ACA
ACB
ACC
BAA
BAB
BAC
BBA
BBB
BBC
BCA
BCB
BCC
CAA
CAB
CAC
CBA
CBB
CBC
CCA
CCB
CCC
```

Ejercicio U3_B5_17: Transcribir a java el clásico algoritmo de divisiones sucesivas por 2 para pasar decimal a binario. Se para de dividir cuando el número a dividir es cero y el resultado es la concatenación de los restos de las sucesivas divisiones. Ejemplo:



Ejercicio U3_B5_18:

Consigue imprimir un triángulo con base de 10 asteriscos con el siguiente aspecto. Debes resolver el problema sin utilizar concatenación de Strings.

```
E:\Programacion>java Unidad3
*
**
***
****
*****
******
*******
********
*********
**********

E:\Programacion>
```

Con la concatenación de objetos strings …es muy fácil. Este pequeño recurso simplifica mucho la lógica

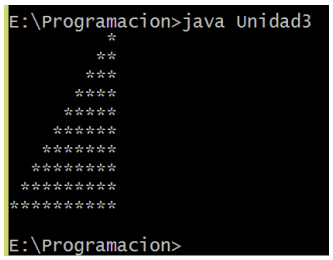
```
class Unidad3{
   public static void main(String[] args) {
      String asteriscos="";
      for(int i=0;i<10;i++){
         asteriscos+="*";
         System.out.println(asteriscos);
      }
   }
}
```

Pero antiguamente no había objetos Strings, la única solución era la impresión secuencial de carácter en carácter y además sería un lujo mantener en memoria objetos String. Hoy en día para resolver algo complicado, es posible que haya un objeto que me ayuda a simplificar la lógica, como ocurre en el ejemplo anterior. Por eso, en muchas áreas de programación ya no se exige tanto a los programadores en destrezas lógicas y en cambio se exigen destrezas con el trabajo con objetos. Esto no quiere decir que las destrezas lógicas no sean importantes, si no que en muchos contextos son menos importantes que antaño.

Imprimir secuencialmente de * en * me fuerza a utilizar un for anidado y a razonar mucho con los contadores de los bucles, que es un habilidad muy importante en programación y estos ejercicios nos ayudan a desarrollarla.

Ejercicio U3_B5_19:

Con las mismas consideraciones que en el ejercicio anterior, usando impresión de * en * consigue:



Observa que en todas las filas, excepto en la última tengo que imprimir espacios antes de *

Ejercicio U3_B5_20:

Consigue: