3.6.- Instrucción de repetición: mientras (while)

Sintaxis: **while (** <expr_booleana> **)** <bloque_instr>

Semántica:
1) Se evalúa la expresión booleana <expr_booleana>
2) Si es falsa se termina de ejecutar la instrucción, es decir, se sale del bucle
3) Y si es verdadera se ejecuta el bloque <bloque_instr> y se vuelve al paso 1.

Ejemplo: programa que lee un número entero `n` y devuelve el factorial `n!`

```
n! = n  x (n-1) x (n-2) x ... x 3 x 2 x 1
```

```
import acm.io.IODialog;

public class Factorial {

      public static void main(String[] args) {
            IODialog entradaSalida; // Declaración de variables
            int f, i, n; /* esto es un comentario */

            entradaSalida = new IODialog(); // Instrucciones
            n = entradaSalida.readInt("Dime un número: ");
            f=1; i=1;
            while (i<=n) {f = f * i; i = i + 1; }
            entradaSalida.println("El factorial de "+n+" es "+f); }
      }
```

NOTA: Siempre que se utilice un while hay que tener cuidado al escribir la condición de salida del bucle (la expresión booleana) para que llegue un momento en el que se evalúe a false y el while termine. En otro caso, se produciría un ciclo infinito, es decir, un programa que nunca terminara. Excepto, claro está, en el caso en que se desee hacer un programa de esas características.

El siguiente fragmento de programa escribiría infinitas veces HOLA en la pantalla.

```
      i=7;
      while (i>=5) {System.out.println("HOLA");}
```

3.7.- Instrucción de repetición: repetir-hasta (do-while)

Sintaxis: **do** <bloque_instr> **while (** <expr_booleana> **)**

Semántica:
1) Se ejecuta el bloque de instrucciones <bloque_instr>
2) Se evalúa la expresión booleana <expr_booleana>
3) Si es falsa se termina de ejecutar la instrucción, es decir, se sale del bucle
4) Y si es verdadera entonces se vuelve al paso 1.

Ejemplo: programa que lee un número entero y devuelve el factorial.

```
import acm.io.IODialog;

public class FactorialConDo {

      public static void main(String[] args) {
            IODialog entradaSalida; // Declaración de variables
            int f, i, n; /* esto es un comentario */

            entradaSalida = new IODialog(); // Instrucciones
            n = entradaSalida.readInt("Dime un número: ");
            f=1; i=1;
          do {f = f * i; i = i + 1; } while (i<=n);
            entradaSalida.println("El factorial de "+n+" es "+f); }
      }
```

Todo lo que se puede hacer con do-while se puede hacer usando while ya que

do <bloque_instr> while (<expr_booleana>)

es equivalente a:

<bloque_instr>
while (<expr_booleana>) <bloque_instr>

La diferencia entre WHILE y DO-WHILE es la siguiente:

El bloque del WHILE se ejecutará de 0 a N veces mientras que el del DO-WHILE se ejecutará de 1 a N veces. Siempre que una acción deba ejecutarse por lo menos una vez seguramente será más natural utilizar un DO-WHILE.

NOTA: Nuevamente hay que tener en cuenta que no se produzcan ciclos infinitos en el programa, esto es, que la condición de salida del bucle debe ser cierta en algún momento.

3.8.- Instrucción de repetición: para (for)

Sintaxis:

for (<variable>=<valor_inic> **;** <variable> **<=** <valor_final> **;** <variable> **++)** <bloque_instr>

Semántica:

1) Se asigna a la variable <variable> el valor <valor_inic>
2) Se evalúa la expresión booleana <variable> <= <valor_final>
3) Si es verdadera entonces se ejecuta el bloque <bloque_instr>, se incrementa la <variable> en uno y se vuelve al paso 2.
4) Y si es falsa entonces se termina de ejecutar la instrucción

Ejemplo: programa que lee un número entero por teclado y devuelve el factorial.

```
import acm.io.IODialog;

public class FactorialConFor {

      public static void main(String[] args) {
            IODialog entradaSalida; // Declaración de variables
            int f, i, n; /* esto es un comentario */

            entradaSalida = new IODialog(); // Instrucciones
            n = entradaSalida.readInt("Dime un número: ");
            f=1;
            for (i=1; i<=n; i++) f=f*i;
            entradaSalida.println("El factorial de "+n+" es "+f); }
      }
```

Todo lo que se puede hacer con for se puede hacer usando while ya que

for (<variable>=<valor_inic> ; <variable> <= <valor_final> ; <variable> ++) <bloque_instr>

es equivalente a:

<variable>=<valor_inic> ;
while (<variable> <= <valor_final>)
{ <bloque_instr>
<variable> ++; }

La diferencia entre FOR y las otras instrucciones de repetición (WHILE y DO-WHILE) es la siguiente:

Cuando se sepa a priori el número de veces que se va a ejecutar el bloque de instrucciones entonces es más natural utilizar un FOR que un WHILE o DO-WHILE donde la condición sea que una variable que actúe de contador sea menor o igual al número de veces a ejecutar.

NOTA: Nuevamente hay que tener en cuenta que no se produzcan ciclos infinitos en el programa, esto es, que la condición de salida del bucle debe ser cierta en algún momento.

NOTA: NO ES ACONSEJABLE modificar el valor de la variable que controla el número de repeticiones del FOR dentro del bloque de instrucciones ya que OSCURECE LA SEMÁNTICA DEL FOR, esto es, hace más difícil entender qué es lo que hace el FOR. Seguro que en ese caso sería más apropiado utilizar un WHILE o un DO-WHILE.