Programación Avanzada

Estructuras de Control

Estructuras de Control

- Controlar el flujo es determinar el orden en el que se ejecutaran las instrucciones en nuestros programas.
- Si no existiesen las sentencias de control los programas se ejecutarían de forma secuencial
  - empezarían por la primera instrucción, e irían ejecutándose una a una hasta llegar a la última.

Estructuras de Control

- Pero, obviamente este panorama sería muy malo para el programador.
- Por un lado, en sus programas no existiría la posibilidad de elegir uno de entre varios caminos en función de ciertas condiciones (sentencias de selección).
- Y por el otro, no podrían ejecutar algo repetidas veces, sin tener que escribir el código para cada una (sentencias repetitivas).

Estructuras de Control

- **Para estos dos problemas tenemos dos soluciones:**
  - **las sentencias de control selectivas**
  - **las sentencias de control repetitivas.**

- **Estos dos conjuntos de sentencias se les llama estructuradas porque a diferencia de las simples pueden contener en su cuerpo otras sentencias.**

Estructuras de Control

- **Las sentencias selectivas se llaman así porque permiten elegir uno de entre varios caminos por donde seguirá la ejecución del programa.**
- **Esta selección viene determinada por la evaluación de una expresión lógica.**
- **Este tipo de sentencias se dividen en dos:**
  - **La sentencia if**
  - **La sentencia switch**

Condicional Sí – if, Sí … sí no… – if else

- **La idea básica de esta sentencia es la de encontrarnos ante una bifurcación de un camino, en la que seguiremos por uno u otro camino dependiendo de la respuesta a una pregunta que se halla escrita en la bifurcación.**
- **Esta sentencia evalúa una condición:**
  - **si la misma es cierta ejecuta un código**
  - **sino es cierta ejecuta otro código.**

Condicional Sí – if, Sí … sí no… – if else

```
if (condición){
    Instrucciones… (Sólo sí la “condición” es Verdadera)
}
```

```
if (condición) {
    Instrucciones… (Sólo sí la “condición” es Verdadera)
}else{
    Instrucciones… (Sólo sí la “condición” es Falsa)
}
```

Condicional Sí – if, Sí … sí no… – if else

- **Ejemplos**

```
if (a > b) {
    System.out.println("Mayor = " + a);
}
```

```
if (a > b) {
    System.out.println("Mayor = " + a);
}else{
    System.out.println("Mayor = " + b);
}
```

Condicional Sí – if, Sí … sí no… – if else

- Una variante de esto es que en lugar de dos posibilidades, se nos presenten más caminos por los que poder seguir.

- Esta sentencia evalúa una condición:
  - si la misma es cierta ejecuta un código
  - sino es cierta se plantea otra condición
    - si la misma es cierta ejecuta un código
    - sino es cierta ejecuta un código

Condicional Sí... sí no Sí ... – if else if

```
if (a > b) {
    System.out.println(“Mayor = ” + a);
}else if (a < b) {
    System.out.println(“Mayor = ” + b);
}else{
    System.out.println(a + “ = ” + b);
}
```

Condicional selectiva, switch … case

- Esta sentencia examina el valor de una variable y en base a ello ejecuta las sentencias del caso.

- Se abre un bloque y se indica la lista de los los valores posibles que puede adoptar dicha variable.
  - Casos posibles

Condicional selectiva, switch … case

- **Los valores posibles son iniciados con “case” seguido del posible valor de la variable y luego “:”**
  - **case 55:**
- **Para cada caso se define un conjunto de instrucciones que serán ejecutados si el valor de la variable corresponde con la del caso.**
- **Finalmente se utiliza vocablo “break” para terminar el caso y salir del switch.**

Condicional selectiva, switch … case

```
switch (variable){
   case valor1:
         instrucciones…
           break;
   case valor2:
         instrucciones…
           break;
…
   default:
         instrucciones …
}
```

Condicional selectiva, switch … case

- **El caso “default” (por defecto) es opcional.**
- **Sus instrucciones serán ejecutadas en caso que el valor de la variable del “switch” no coincida con ninguno de los casos anteriores.**

Condicional selectiva, switch … case

```
switch (meses){
   case 6:
            System.out.println("Pasó medio año);
            break;
   case 12:
            System.out.println("Pasó un año);
            break;
   default:
            System.out.println("Pasaron " + meses + " meses");
}
```

Condicional selectiva, switch … case

```
switch (nota){
   case 1:   case 2:   case 3:    case 4:
              System.out.println("La nota es insuficiente");
              break;
   case 5:   case 6:    case 7:
              System.out.println("La nota es aceptable");
              break;
   case 8:    case 9:    case 10:
             System.out.println("La nota es sobresaliente");
}
```

Estructuras de Control Repetitivas

- **A las sentencias repetitivas se les conoce también como sentencias iterativas ya que permiten realizar algo varias veces (repetir, iterar).**
- **Estudiaremos las siguientes:**
  - **La sentencia for**
  - **La sentencia while**
  - **La sentencia do/while**

Repetitiva para…hasta… modificar

- La estructura de control for se utiliza para repetir una o varias sentencias de código tantas veces, como veces se evalúe una condición como verdadera.

Repetitiva para…hasta… modificar

for (*inicio; condición; diferencia*) *{*

Instrucciones…

}

Repetitiva para…hasta… modificar

```
for (inicio; condición; diferencia) {
  Instrucciones...
}
```

Se ejectua sólo una vez y antes de evaluar la condición por primera vez.
Se utiliza para declarar y/o inicializar las variables de la estructura

Repetitiva para…hasta… modificar

for (*inicio; condición; diferencia*) {

Instrucciones…

}

Después de ejecutar "inicio" se evalúa la "condición".
Sí es verdadera se ejecutan las "instrucciones" del bloque.
Sí es falsa no se ejecutan las "instrucciones"

Repetitiva para…hasta… modificar

for (*inicio; condición; diferencia*) {

Instrucciones…

}

Luego de ejecutar las "instrucciones" se ejecuta la "diferencia" que modifica el valor de la variable que es parte en la "condición".

Repetitiva para…hasta… modificar

for (*inicio; condición; diferencia*) {

Instrucciones…

}

Después de ejecutar "diferencia" se evalúa la "condición".
Sí es verdadera se ejecutan las "instrucciones" del bloque.
Sí es falsa no se ejecutan las "instrucciones"

Repetitiva para…hasta… modificar

```
for (inicio; condición; diferencia) {
  Instrucciones...
}
```

Los dos pasos anteriores se repiten hasta que la condición sea falsa.

Repetitiva para…hasta… modificar

```
for (int i = 1; i < 11; i++){
    System.out.print (i + "  ");
}
//Muestra: 1 2 3 4 5 6 7 8 9 10
//No muestra el número 11 porque
// 11 < 11 → FALSO!
```

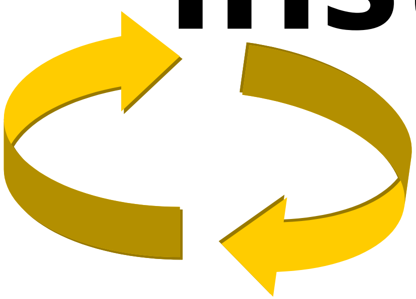
Repetitiva mientras, while ...

- **La estructura de control while (mientras) se utiliza para repetir una o varias sentencias de código tantas veces hasta que se evalúe una condición como falsa.**

Repetitiva mientras, while ...

while (condición){

instrucciones ...



}

Condición Falsa o Verdadera

Repetitiva mientras, while ...

```
int i = 2;
while (i < 50){
  System.out.print(i + "  ") ;
   i += 2 ;
}
// Muestra los números pares del 2 al 48
// No muestra el número 50 porque
// 50 < 50 → FALSO!
```

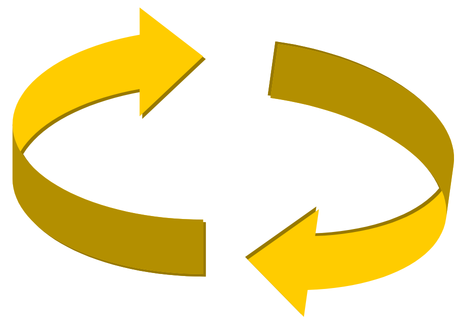
Repetitiva hacer… mientras, do...while

- La estructura de control do... while (hacer... mientras) se utiliza para repetir una o varias sentencias de código tantas veces hasta que se evalúe una condición como falsa.
- A diferencia del “while”, en el “do... while”, las sentencias se ejectuan al menos una vez sin importar si la condición es falsa la primera vez.

Repetitiva hacer… mientras, do...while

do {

instrucciones …



} while (*condición*);

Falsa o **Verdadera**

- Las "instrucciones" se ejecutan al menos una vez.

Repetitiva hacer… mientras, do...while

```
int num, suma = 0;
Scanner leer = new Scanner(System.in);
 do{
      System.out.print(“Numero: “);
      num = leer.nextInt( );
      suma += num;
 }while (num != 0);

 System.out.print(“La suma es: “ + suma);

 //Solicita el ingreso de números hasta que
 //ingrese el numero 0 (cero). Luego
 //Muestra la suma de los numeros ingresados
```