Programación

UD 3: Uso de estructuras de control

Ciclo Formativo de Grado Superior
Desarrollo de Aplicaciones Web

Francisco Pérez

1.- Estructura secuencial

1.- Estructura secuencial

El orden en que se ejecutan por defecto las sentencias de un programa es secuencial. Es decir, las sentencias se ejecutan una después de otra, en el orden en que aparecen escritas dentro del programa.

Cada una de las instrucciones están separadas por el carácter punto y coma (;).

Las instrucciones se suelen agrupar en bloques.

El bloque de sentencias se define por el carácter llave de apertura ({) para marcar el inicio del mismo, y el carácter llave de cierre (}) para marcar el final.

Ejemplo:

```
{
    instrucción 1;
    instrucción 2;
    instrucción 3;
}
```

En Java si el bloque de sentencias está constituido por una única sentencia no es obligatorio el uso de las llaves de apertura y cierre ({ }), aunque sí recomendable.

1.- Estructura secuencial

Ejemplo: Programa que pide 2 números enteros y los muestra por pantalla.

```
import java.util.Scanner;

public class Secuencial {

	public static void main(String[] args){

		//Declaración de variables
		int n1, n2;
		Scanner sc = new Scanner(System.in);

		//Leer el primer número
		System.out.println("Introduce un número entero: ");
		n1 = sc.nextInt();

		//Leer el segundo número
		System.out.println("Introduce otro número entero: ");
		n2 = sc.nextInt();

		//Mostrar resultado
		System.out.println("Los números son: " + n1 + " y " + n2);
	}
}
```

2.- Estructuras de selección

2.- Estructuras de selección

La estructura de selección o condicional permite al programa bifurcar el flujo de ejecución de instrucciones dependiendo del valor de una expresión (condición).

Se puede diferenciar entre selección simple, selección doble y selección múltiple.

En java las estructuras condicionales se implementan mediante:

Selección simple:	if …
Selección doble:	if … else … operador condicional ? :
Selección múltiple:	if … else … *encadenados* (if … else if … else) switch case

2.- Estructuras de selección

La condición debe ser una expresión booleana, es decir, debe dar como resultado un valor booleano (true ó false).

Selección simple: se evalúa la condición y si ésta se cumple se ejecuta una determinada acción o grupo de acciones. En caso contrario se saltan dicho grupo de acciones.

```
if (expresión_booleana) {
        instrucción 1
        instrucción 2
        .......
}
```

Si el bloque de instrucciones tiene una sola instrucción no es necesario escribir las llaves { } aunque para evitar confusiones se recomienda escribir las llaves siempre.

2.- Estructuras de selección

Ejemplo: Programa que pide la edad al usuario y determina si es mayor de edad.

```
import java.util.Scanner;

public class Edades {

	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);
		int edad;

		System.out.println("Introduzca la edad: ");
		edad = sc.nextInt();

		//Comprobar si es mayor de edad
		if (edad >= 18){
			System.out.println("Eres mayor de edad.");
		}
	}

}
```

2.- Estructuras de selección

Selección doble:

Se evalúa la condición y si ésta se cumple se ejecuta una determinada instrucción o grupo de instrucciones. Si no se cumple se ejecuta otra instrucción o grupo de instrucciones.

```
if (expresión booleana) {
        instrucciones 1
}
else{
        instrucciones 2
}
```

Operador condicional ? :

Se puede utilizar en sustitución de la sentencia de control if-else. Los forman los caracteres ? y :

Se utiliza de la forma siguiente: `expresión1 ? expresión2 : expresión3`

Si expresión1 es cierta entonces se evalúa expresión2 y éste será el valor de la expresión condicional. Si expresión1 es falsa, se evalúa expresión3 y éste será el valor de la expresión condicional.

2.- Estructuras de selección

Selección múltiple: Se obtiene anidando sentencias if ... else. Permite construir estructuras de selección más complejas.

```
if (expresion_booleana1){
        instrucciones;
}
else if (expresion_booleana2){
        instrucciones;
}
else{
        instrucciones;
}
```

Cada else se corresponde con el if más próximo que no haya sido emparejado.

Una vez que se ejecuta un bloque de instrucciones, la ejecución continúa en la siguiente instrucción que aparezca después de las sentencias if .. else anidadas.

2.- Estructuras de selección

Ejemplo: Programa que muestra un saludo según la hora indicada.

```
import java.util.Scanner;

public class Saludo {

	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);
		int hora;

		System.out.println("Introduzca una hora (un valor entero): ");
		hora = sc.nextInt();

		if (hora >= 0 && hora < 12)
			System.out.println("Buenos días");
		else if (hora >= 12 && hora < 21)
			System.out.println("Buenas tardes");
		else if (hora >= 21 && hora < 24)
			System.out.println("Buenas noches");
		else
			System.out.println("Hora no válida");
	}

}
```

2.- Estructuras de selección

Selección múltiple. Instrucción Switch: Se utiliza para seleccionar una de entre múltiples alternativas. La forma general de la instrucción switch en Java es la siguiente:

```
switch (expresión){
        case valor 1:
                instrucciones;
                break;
        case valor 2:
                instrucciones;
                break;
        . . .
        default:
                instrucciones;
}
```

La instrucción switch se puede usar con datos de tipo byte, short, char e int. También con tipos enumerados y con las clases envolventes Character, Byte, Short e Integer.

A partir de Java 7 también pueden usarse datos de tipo String en un switch.

2.- Estructuras de selección

Ejemplo: Programa que pide el número del mes y muestra el nombre correspondiente.

```
import java.util.Scanner;

public class Meses {

	public static void main(String[] args) {

		int mes;
		Scanner sc = new Scanner(System.in);

		System.out.print("Introduzca un numero de mes: ");
		mes = sc.nextInt();

		switch (mes)
		{
			case 1:   System.out.println("ENERO");      break;
			case 2:   System.out.println("FEBRERO");    break;
			case 3:   System.out.println("MARZO");      break;
			case 4:   System.out.println("ABRIL");      break;
			case 5:   System.out.println("MAYO");       break;
			case 6:   System.out.println("JUNIO");      break;
			case 7:   System.out.println("JULIO");      break;
			case 8:   System.out.println("AGOSTO");     break;
			case 9:   System.out.println("SEPTIEMBRE"); break;
			case 10:  System.out.println("OCTUBRE");    break;
			case 11:  System.out.println("NOVIEMBRE");  break;
			case 12:  System.out.println("DICIEMBRE");  break;
			default : System.out.println("Mes no válido");
		}
	}
}
```

2.- Estructuras de selección

Instrucción switch VS if… else… encadenados:
Programa que pide el número del mes y muestra el nombre correspondiente.

```
import java.util.Scanner;

public class Meses {

	public static void main(String[] args) {

		int mes;
		Scanner sc = new Scanner(System.in);

		System.out.print("Introduzca un numero de mes: ");
		mes = sc.nextInt();

		switch (mes)
		{
			case 1:   System.out.println("ENERO");      break;
			case 2:   System.out.println("FEBRERO");    break;
			case 3:   System.out.println("MARZO");      break;
			case 4:   System.out.println("ABRIL");      break;
			case 5:   System.out.println("MAYO");       break;
			case 6:   System.out.println("JUNIO");      break;
			case 7:   System.out.println("JULIO");      break;
			case 8:   System.out.println("AGOSTO");     break;
			case 9:   System.out.println("SEPTIEMBRE"); break;
			case 10:  System.out.println("OCTUBRE");    break;
			case 11:  System.out.println("NOVIEMBRE");  break;
			case 12:  System.out.println("DICIEMBRE");  break;
			default : System.out.println("Mes no válido");
		}
	}
}
```

```
if (mes == 1){
		System.out.println("ENERO");
}else if (mes == 2){
		System.out.println("FEBRERO");
}else if (mes == 3){
		System.out.println("MARZO");
}else if (mes == 4){
		System.out.println("ABRIL");
}else if (mes == 5){
		System.out.println("MAYO");
}else if (mes == 6){
		System.out.println("JUNIO");
}else if (mes == 7){
		System.out.println("JULIO");
}else if (mes == 8){
		System.out.println("AGOSTO");
}else if (mes == 9){
		System.out.println("SEPTIEMBRE");
}else if (mes == 10){
		System.out.println("OCTUBRE");
}else if (mes == 11){
		System.out.println("NOVIEMBRE");
}else if (mes == 12){
		System.out.println("DICIEMBRE");
}else{
		System.out.println("Mes no válido");
}
```

3.- Estructuras de repetición

3.- Estructuras de repetición

Permiten ejecutar de forma repetida un bloque específico de instrucciones.

Las instrucciones se repiten mientras o hasta que se cumpla una determinada condición.

Esta condición se conoce como **condición de salida**.

Tipos de estructuras repetitivas:

- ✓ **while**
- ✓ **do – while**
- ✓ **for**

3.- Estructuras de repetición

While

Las instrucciones se repiten mientras la condición sea cierta. La condición se comprueba al principio del bucle por lo que las acciones se pueden ejecutar 0 ó más veces.

La ejecución de un bucle while sigue los siguientes pasos:

1.- Se evalúa la condición.

2.- Si el resultado es false las instrucciones no se ejecutan y el programa sigue ejecutándose por la siguiente instrucción a continuación del while.

3.- Si el resultado es true se ejecutan las instrucciones y se vuelve al paso 1

3.- Estructuras de repetición

Ejemplo: Función que imprime línea a línea un fichero de texto completo.

```
public void imprimirFichero() throws FileNotFoundException{

        Scanner teclado = new Scanner(new File("fichero.txt"));

        //Mientras hay líneas por leer…
        while(teclado.hasNext()){

                //…leer la línea del fichero…
                String s = teclado.nextLine();

                //…e imprimirla por pantalla.
                System.out.println( s );
        }
}
```

3.- Estructuras de repetición

Do while

Las instrucciones se ejecutan mientras la condición sea cierta.

La condición se comprueba al final del bucle por lo que el bloque de instrucciones se ejecutarán al menos una vez. Esta es la diferencia fundamental con la instrucción while. Las instrucciones de un bucle while es posible que no se ejecuten si la condición inicialmente es falsa.

La ejecución de un bucle do - while sigue los siguientes pasos:

1. Se ejecutan las instrucciones a partir de do{
2. Se evalúa la condición.
3. Si el resultado es *false* el programa sigue ejecutándose por la siguiente instrucción a continuación del while.
4. Si el resultado es true se vuelve al paso 1

3.- Estructuras de repetición

Ejemplo: Función que calcula el perímetro de un círculo.

```
public double perimetroCirculo() {

        Scanner teclado = new Scanner(System.in);

        double radio;

        do {
                System.out.print("Teclee un valor del radio > 0: ");
                radio = teclado.nextDouble();

        } while (radio <= 0);

        return 2 * Math.PI * radio;
}
```

3.- Estructuras de repetición

For

Hace que una instrucción o bloque de instrucciones se repitan un número determinado de veces mientras se cumpla la condición.

La estructura general de una instrucción for en Java es la siguiente:

```
for( inicialización; condición; incremento/decremento){
        instrucción 1;
        .....
        instrucción N;
}
```

A continuación de la palabra for y entre paréntesis debe haber siempre tres zonas separadas por punto y coma:

- zona de inicialización.
- zona de condición
- zona de incremento ó decremento.

3.- Estructuras de repetición

Ejemplo: Función que recorre e imprime un vector de enteros.

```
public void imprimirVector(int[] v) {

        //Recorrer el vector desde la posición 0 hasta longitud-1...
        for (int i = 0; i < v.length; i++) {

                //...e imprir el valor de la posición i-ésima
                System.out.println( v[i] );
        }
}
```

3.- Estructuras de repetición

Bucles infinitos

Java permite la posibilidad de construir bucles infinitos, los cuales se ejecutarán indefinidamente, a no ser que provoquemos su interrupción. Tres ejemplos:

```
for(;;){
    instrucciones
}
```

```
for(;true;){
    instrucciones
}
```

```
while(true){
    instrucciones
}
```

Los bucles infinitos también suelen producirse de forma involuntaria al no modificar correctamente la condición de salida de un bucle. Es un ERROR común al iniciarse en Programación.

3.- Estructuras de repetición

Bucles anidados

Bucles anidados son aquellos que incluyen instrucciones for, while o do-while unas dentro de otras.

Debemos tener en cuenta que las variables de control que utilicemos deben ser distintas.

Los anidamientos de estructuras tienen que ser correctos, es decir, que una estructura anidada dentro de otra lo debe estar totalmente.

Ejemplo:

```
public void imprimirVector(int[][] m) {

        //Recorrer todas las filas de la matriz...
        for (int i = 0; i < m.length; i++) {

                //Para cada fila, recorrer las columnas...
                for (int j = 0; j < m.length; j++) {

                        //...e imprir el valor de la posición (i, j)
                        System.out.println( m[i][j] );
                }
        }
}
```

4.- Estructuras de salto

4.- Estructuras de salto

Las palabras reservadas *break* y *continue*, se utilizan en Java para detener completamente un bucle (break) o detener únicamente la iteración actual y saltar a la siguiente (continue).

Normalmente si usamos break o continue, lo haremos dentro de una sentencia if, que indicará cuándo debemos detener el bucle al cumplirse o no una determinada condición.

La gran diferencia entre ambos es que, *break*, detiene la ejecución del bucle y salta a la primera línea del programa tras el bucle y *continue*, detiene la iteración actual y pasa a la siguiente iteración del bucle sin salir de él (a menos, que el propio bucle haya llegado al límite de sus iteraciones).

4.- Estructuras de salto

break

La sentencia break se utiliza para interrumpir la ejecución de una estructura de repetición o de un switch. Cuando se ejecuta el break, el flujo del programa continúa en la sentencia inmediatamente posterior a la estructura de repetición o al switch.

Ejemplo:

```
for (int i = 0; i < 10; i++) {
    System.out.println("Dentro del bucle");
    break;
    System.out.println("Nunca lo escribira");
}
System.out.println(“Tras el bucle”);
```

Cuando el programa alcance la sentencia *break*, detendrá la ejecución del bucle y saldrá de él. Por tanto el resultado de la ejecución del programa será:

```
Dentro del Bucle
Tras el bucle
```

El bucle for, sólo ejecuta una iteración y después termina, continuando el programa desde la primera línea tras el bucle.

4.- Estructuras de salto

continue

La sentencia continue únicamente puede aparecer dentro de una estructura de repetición. El efecto que produce es que se deja de ejecutar el resto del bucle para volver a evaluar la condición del bucle, continuando con la siguiente iteración, si el bucle lo permite.

Ejemplo:

```
for (int i = 0; i < 10; i++) {
    System.out.println("Dentro del bucle");
    continue;
    System.out.println("Nunca lo escribira");
}
```

Cada vez que el programa alcance la sentencia *continue*, saltará a una nueva iteración del bucle y por tanto la última sentencia *System.out.println* nunca llegará a escribirse.

El resultado del programa será escribir 10 veces “Dentro del bucle”.