Programación

PLG
Planning and Learning Group

Universidad Carlos III de Madrid

Tema 4: Condicionales y Bucles

Control de Flujo

- Instrucciones para romper el flujo de ejecución secuencial
- 3 tipos:
  - Condicionales: `if, switch`
  - Bucles: `while, do while, for`
  - Ramificación: `break, continue, return`

En este tema

Tema 4: Condicionales y Bucles

- Condicionales
- Bucles

Condicionales `if`

- Sintaxis

```
if ( condicion )
    sentencia;
```

```
if ( condicion )
    { sentencias }
```

- La condición siempre tiene que ser de tipo `boolean`
- Usar llaves `{}` para más de una sentencia
- Por convención el bloque condicionado de instrucciones va indentado a la derecha

Condicionales `if-else`

- Sintaxis

```
if ( condicion )
        sentencia;
else
        sentencia;
```

```
if ( condicion )
        { sentencias }
else
        { sentencias }
```

- Para sentencias o bloque de sentencias excluyentes

Encadenamiento de condicionales

```
public static void main(String[] args) {
    Scanner entrada = new Scanner(System.in);
    System.out.println("Introduzca nota del alumno");
    double nota = entrada.nextDouble();

    if (nota < 0 || nota > 10)
        System.out.println("Nota inválida");
    else if (nota >= 5)
        System.out.println("Alumno APROBADO");
    else
        System.out.println("Alumno SUSPENSO");
}
```

Condicionales `switch`

- Cuando hay que elegir entre varias alternativas
- Sobre variables `int` o compatibles. No con long.
- Vale con `String` a partir de Java7
- Cuando un valor se cumple se ejecuta hasta que encuentra un `break`
- Sintaxis:

```
switch (variable){
        case valor1:
                sentencia;
                sentencia;
                [ break; ]
        case valor2:
                sentencia;
                [break;]
        default:
                sentencia;
}
```

Ejemplo switch

```
double n1, n2, res=0;
String op;
Scanner sc = new Scanner (System.in);

System.out.println("CALCULADORA BASICA");
System.out.println("Primer número");
n1 = sc.nextDouble();
System.out.println("Operación");
op = sc.next();
System.out.println("Segundo número");
n2 = sc.nextDouble();

switch (op) {
    case "+":
        res = n1 + n2;
        break;
    case "-":
        res = n1 - n2;
        break;
    default:
        System.out.println("Operacion no reconocida");
}
System.out.println(" = " + res);
```

Ámbito de una variable

- Llamamos bloque al conjunto de sentencias entre llaves {}
- En todo programa hay al menos 2 bloques, el de la clase y el del main
- Los bloques establecen el ámbito de una variable, o porción de código donde se puede utilizar
- Ejemplo con bloques de `if` anidados

En este tema

Tema 4: Condicionales y Bucles

- Condicionales
- Bucles

Bucles

- Necesidad de repetir bloques de código
- Tipos de bucles:
  - Bucles que se repiten mientras se cumpla una condición: `while`, `do while`
  - Bucles que se repiten un número determinado de veces: `for`
- Realmente con un tipo es suficiente
- Elementos importantes:
  - Variable(s) de control: Para decidir si continuamos repitiendo o no. Debe cambiar en cada iteración
  - Condición de control: Se comprueba en cada repetición. Si es verdadera se repite el bloque otra vez

Bucles `while`

- Se repite mientras se cumpla la condición de control
- Si no se cumple inicialmente **no se ejecuta ninguna vez**
- Sintaxis:

```
while ( condicion )
      { sentencias }
```

Ejemplo `while`

Hacer un programa que sume los números pares hasta N

Ejemplo `while`

```
Scanner entrada = new Scanner(System.in);
System.out.println("Introduce número límite:");

int numeroLimite = entrada.nextInt();

int resultado = 0;
int numeroActual = 0;


while (numeroActual <= numeroLimite)
{

   resultado += numeroActual;
   numeroActual += 2;

}

System.out.println("La suma de los números pares hasta el " + numeroLimite +
                   " es " + resultado);

entrada.close();
```

Bucles `do while`

- Se repite mientras se cumpla la condición de control
- Siempre se ejecuta **al menos una vez**
- Sintaxis:

```
do
    { sentencias }
while ( condicion );
```

Ejemplo `do while`

Hacer un programa en el que el usuario introduzca un número entre 0 y 10. Si el número introducido esta fuera de ese intervalo se debe pedir nuevamente.

Ejemplo `while`

```
Scanner entrada = new Scanner(System.in);
int numero;


do
{
    System.out.println("Introduce un número entre 0 y 10:");
    numero = entrada.nextInt();
} while (numero < 0 || numero > 10);

System.out.println("El numero introducido es " + numero);

entrada.close();
```

Bucles `for`

- El más potente y versátil de las sentencias de bucles
- Suele utilizarse cuando conocemos el número de veces que queremos repetir un bloque
- Elementos:
  - Inicialización: declara y/o da valor inicial a la(s) variable(s) de control
  - Control: condición que debe cumplirse para permanecer en el bucle. Tipo `boolean`
  - Actualización: modifica las variables de control al final de cada ciclo
- Sintaxis:

`for`(*inicialización* ; *condicion_control* ; *actualización*)
{ *sentencias* }

Ejemplo `for`

Hacer un programa que sume los 100 primeros números naturales.

Ejemplo `for`

```
int suma = 0;

for (int i = 1; i <= 100; i++){
  suma = suma + i;
}
```

Bucles `for`

- Tiempo de vida de las variables de control
  - Si declaramos dentro del `for`, sólo existe en ese ámbito.

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

- Si declaramos fuera, cualquier cambio dentro del bucle afecta la variable

```
int i = 5;
for ( i = 0 ; ; )
      { sentencias }
```

Bucles `for`

- En la inicialización y la actualización se pueden poner varias variables a la vez, separadas por comas

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

Bucles `for`

- En la inicialización y la actualización se pueden poner varias variables a la vez, separadas por comas

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

- Si se declaran varias variables en la inicialización, éstas tienen que ser del mismo tipo
- Se pueden dejar vacías, tanto inicialización como control o actualización, pero se mantienen los ";"
- Modificar variables de control dentro del bucle **NO** es una buena práctica

Bucles `for` anidados

```
int exterior = 0, interior = 0;

for (int i=0; i<3; i++){

    exterior++;

    for (int j=0; j<5; j++){

        interior++;
    }
}
System.out.println("El bucle exterior repitió "+exterior+" veces");
System.out.println("El bucle interior repitió "+interior+" veces");
```

`break`

- Se utiliza en el `switch` y en bucles
- En bucles sirve para salir del bucle
- Uso no recomendado, se puede sustituir por una condición extra en el bucle

Ejemplo `break`

```
for (int i=0; i<25; i++){
     if (i == 12)
         break;
     System.out.print(i+" ");
}
System.out.println("ABCD");
```

`continue`

- Vuelve a la condición del bucle, saltando lo que queda por ejecutar dentro de él
- En caso de `for` pasa a la siguiente iteración
- Uso no recomendado, se puede resolver con un condicional dentro del bucle

Ejemplo `continue`

```
for (int i=0; i<25; i++){
     if (i == 12)
         continue;
     System.out.print(i+" ");
}
System.out.println("ABCD");
```