

1º CFGS DESARROLLO DE APLICACIONES MULTIPLATAFORMA

UD 3. Estructuras de control.

Módulo: Programación

Control del flujo

- Es fundamental el controlar el flujo de los programas.
- Dos formas:
 - Estructuras de **selección**:
 - if-else
 - switch
 - Estructuras de **iteración**:
 - while
 - do-while
 - for
 - **for-loop**



Estructuras de selección

I – Sentencia if

- Sintaxis:

```
if (condición) acción;
```

```
if (condición)  
    acción;
```

```
if (condición) {  
    acción1;  
    acción2;  
    ...  
    acción;  
}
```



Estructuras de selección

I – Sentencia if

- Analicemos los siguientes fragmentos de código:

```
1. int nota = teclado.nextInt();  
2. if (nota >= 5)  
3.     System.out.println("Aprobado");  
4. System.out.println("NOTA PROGRAMACION");
```

```
1. int nota= teclado.nextInt();  
2. if (nota >= 5) {  
3.     System.out.println("Aprobado");  
4.     System.out.println("NOTA PROGRAMACION");  
5. }
```



Estructuras de selección

I – Sentencia if

- Ejercicios propuestos:

1. Hacer un programa llamado *Divisible* que reciba por teclado dos números y compruebe si el primero es divisible por el segundo.
2. Hacer un programa llamado *SignoNumero* que reciba por teclado un número real e imprima por pantalla si el número es positivo, negativo o cero.



Estructuras de selección

II – Sentencia if/else

- Sintaxis:

```
if (condición) acción1; else acción2;
```

```
if (condición) acción1;  
else acción2;
```

```
if (condición)  
    acción1;  
else  
    acción2;
```

```
if (condición) {  
    acción1;  
    acción2;  
} else {  
    acción3;  
    acción4;  
}
```



Estructuras de selección

II – Sentencia if/else

- Ejemplo:

```
1. int edad = teclado.nextInt();  
2. if (edad >= 18) {  
3.     System.out.println("Mayor de edad");  
4. } else {  
5.     System.out.println("Menor de edad");  
6. }
```



Estructuras de selección

III – Sentencias if/else anidadas

- Sintaxis formato anidado:

```
if (condición1) {  
    acción1;  
} else {  
    if (condición2) {  
        acción2;  
    } else {  
        acción3;  
    }  
}
```

- Sintaxis formato secuencial:

```
if (condición1) {  
    acción1;  
} else if (condición2) {  
    acción2;  
} else {  
    acción3;  
}
```

Estructuras de selección Comparativa

- Ejemplo:

```
if (x>0) positivos = positivos + 1;  
if (x<0) negativos = negativos + 1;  
if (x==0) ceros = ceros + 1;
```

```
if (x>0) {  
    positivos = positivos + 1;  
} else if (x<0) {  
    negativos = negativos + 1;  
} else {  
    ceros = ceros + 1;  
}
```



Estructuras de selección

IV – Sentencia switch

- Sintaxis:

```
switch (variable) {  
    case etiqueta1:  
        sentencias1;  
        break;  
    ...  
    case etiquetaN:  
        sentenciasN;  
        break;  
    default:  
        sentencias_alternativas;  
}
```



Estructuras de selección

IV – Sentencia switch

- Ejemplo:

```
1. int eleccion = teclado.nextInt();  
2. switch(eleccion) {  
3. case 1:  
4.     System.out.println("Salir.");  
5. case 2:  
6.     System.out.println("Nuevo.");  
7.     break;  
8. case 3:  
9.     System.out.println("Borrar.");  
10.    break;  
11. default:  
12.     System.out.println("Opción no contemplada");  
13. }
```



Estructuras de selección

IV – Sentencia switch

- Ejercicios propuestos:

1. Hacer un programa llamado *Vocales* que reciba por teclado un carácter y muestre por pantalla si se trata de una vocal o no. Hacer el programa con sentencias if anidadas en primer lugar y después con una estructura switch.
2. Hacer un programa llamado *Notas* que lea por teclado un entero correspondiente a la nota de un examen y muestre por pantalla a qué calificación corresponde (MD=0,1,2, I=3,4, A=5, B=6, N=7,8, SB=9,10).

Estructuras de selección. Errores frecuentes



- Uso del operador asignación (=) en lugar del de comparación de igualdad (==).
- Llaves abriendo bloques que no se cierran.
- Utilizar una sentencia **if** que ejecute varias instrucciones y olvidar las llaves.
- Ampliar una sentencia **if** que inicialmente consta de una sola instrucción (sin llaves) y olvidar añadir las llaves al incluir las nuevas instrucciones.
- Ausencia de **break** en estructuras de selección **switch**.

Estructuras de iteración

Bucles.

- Un bucle es un bloque de código que se repite un nº de veces mientras se cumpla una condición.
- Las sentencias que se repiten se denominan **cuerpo del bucle**.
- Cada repetición se denomina **iteración**.
- La condición de repetición se denomina **condición de permanencia**.
- En cada iteración se evalúa la condición de permanencia, de modo que si es cierta (*true*) se repite la ejecución del cuerpo. Si la condición es falsa el bucle finaliza, continuando la ejecución del programa.



Estructuras de iteración

- Tipos de bucles en java:
 - while
 - do-while
 - For
 - for-loop
- Son equivalentes.
- Se diferencian en la forma de evaluar la condición de permanencia, y dónde y cuándo se realiza.



Estructuras de iteración

I – Bucle while

- Evaluación de la condición de permanencia: **previa** a la ejecución del cuerpo del bucle.
- Nº de repeticiones: 0..n
- Sintaxis:

```
while (condición) {  
    sentencia1;  
    sentencia2;  
    ...  
    sentenciaN;  
}
```



Estructuras de iteración

I – Bucle while

- Ejemplo1:

```
public class BucleWhile {  
    public static void main(String[] args) throws IOException {  
        //Acceso al teclado  
        Scanner sc = new Scanner(System.in));  
  
        //Lectura anticipada  
        System.out.println("Escribe un n° (0 para salir): ");  
        int num=sc.nextInt();  
        //Si el número introducido no es 0 se pide otro  
        while(num!=0) {  
            System.out.println("Escribe un n° (0 para salir): ");  
            int num=sc.nextInt();  
        }  
        System.out.println("Fin del programa");  
    } //main  
} //class
```



Estructuras de iteración

I – Bucle while

- Mismo ejemplo con **banderas**:

```
public class BucleWhile {  
    public static void main(String[] args) throws IOException {  
        //Acceso al teclado  
        Scanner sc = new Scanner(System.in));  
  
        //Lectura anticipada  
        System.out.println("Escribe un n° (0 para salir): ");  
        int num=sc.nextInt();  
        boolean salir=false; //bandera que controla la permanencia en el bucle  
        while(!salir) { //mientras no haya que salir  
            System.out.println("Escribe un n° (0 para salir): ");  
            int num=sc.nextInt();  
            if(num==0) salir=true; //cambiamos la bandera  
        }  
        System.out.println("Fin del programa");  
    } //main  
} //class
```

Estructuras de iteración

I – Bucle while

- Ejemplo2:

```
public class Asteriscos {  
  
    public static void main(String[] args) {  
        //acceso al teclado  
        ...  
        int n=sc.nextInt();  
        while(n>0) {  
            System.out.print("*");  
            n--;  
        }  
    } //main  
}  
} //class
```



Estructuras de iteración

II – Bucle do/while

- Evaluación de la condición de permanencia: **después** de la ejecución del cuerpo del bucle.
- Nº de repeticiones: 1..n
- Sintaxis:

```
do {  
    sentencia1;  
    sentencia2;  
    ...  
    sentenciaN;  
} while (condición);
```



Estructuras de iteración

II – Bucle do/while

- Ejemplo1:

```
public class BucleDoWhile {  
    public static void main(String[] args) throws IOException {  
        //Acceso al teclado  
        Scanner sc = new Scanner(System.in));  
  
        //Si el número introducido no es 0 se pide otro  
        do{  
            System.out.println("Escribe un nº (0 para salir): ");  
            int num=sc.nextInt();  
            }while(num!=0);  
            System.out.println("Fin del programa");  
        } //main  
} //class
```



Estructuras de iteración

II – Bucle do/while

- Ejemplo2:

```
public class Asteriscos {  
  
    public static void main(String[] args) {  
        //acceso al teclado  
        ...  
        int n=sc.nextInt());  
        do{  
            System.out.print("*");  
            n--;  
        } while(n>0);  
    } //main  
  
} //class
```



Estructuras de iteración

III – Bucle for

- Sintaxis:

```
for (inicialización; condición; incremento) {  
    sentencias  
}
```

- **Inicialización:** sentencia que se ejecuta sólo una vez, antes de entrar en el cuerpo del bucle. Normalmente se declara e inicializa la variable contador, en cuyo caso el ámbito de la variable se limita al cuerpo del bucle. Se pueden incluir varias sentencias separadas por el operador coma “,”
- **Condición:** de permanencia del bucle. Si es falsa no se produce la siguiente iteración. Se evalúa antes de entrar en el cuerpo del bucle, por lo que la cardinalidad del bucle **for** es 0..n.
- **Incremento:** instrucción que se ejecuta por cada iteración del bucle. En este apartado se suele incrementar o decrementar la variable contador. También puede dejarse en blanco. Se pueden poner varias sentencias separadas por el operador coma “,”



Estructuras de iteración

III – Bucle for

- Ejemplo:

	for (int i=1; i<=100; i++) { System.out.println(i); }
int i =1; for (; i<=100; i++) { System.out.println(i); }	int i =1; for (; i<=100;) { System.out.println(i); i++; }
int i =1; for (; i<=100;) { System.out.println(i++); }	boolean seguir = true; for (int i=1; seguir;) { System.out.println(i++); if (i==100) seguir = false; }



Estructuras de iteración

III – Bucle for



- Importante:
 - Ojo con poner ";" (llamada **sentencia nula**) detrás del paréntesis del bucle, ya que el ";" es una sentencia válida en Java y la iteración la haría sobre dicha instrucción (la cual no haría nada).
 - Evitar la modificación de las variables que determinan el comportamiento del bucle (contador o condición de salida). Considerarlas como **variables de sólo lectura**.

Estructuras de iteración

III – Bucle for

- Mala “praxis”:

```
for(int i=1; i<=100; i++) {  
    System.out.println(i);  
    if(i==100) i=101; ←  
}
```



Estructuras de iteración

III – Bucle for

- Ejercicios propuestos:

1. Hacer un programa que sume los números pares positivos menores o iguales que 100 utilizando un bucle for.
2. Hacer un programa que imprima por pantalla todas las potencias de 2 entre 1 y 100 utilizando un bucle for.



Estructuras de iteración

Importante



- La condición de terminación del bucle debe hacerse false en un número finito de pasos, sino tendríamos un **bucle infinito**.
Normalmente, la condición de permanencia depende del valor de una variable, de modo que si dentro del cuerpo del bucle no se modifica nunca esa variable una vez dentro del bucle no se podrá salir de él.
- Este tipo de bucles donde existe una variable que determina la continuación del bucle se denominan **bucles controlados por centinela**. Donde el **centinela** es el valor que debe alcanzar la variable para que la condición de permanencia no se cumpla. En el *Ejemplo 1* el valor centinela es el 0.
- Se suelen utilizar mucho los operadores incremento (++) y decremento (--) en todos los bucles.

¿Bucles más adecuados?

- Aunque los tres tipos de bucles son equivalentes...
 - Normalmente, los bucles `while` y `do-while` se emplean cuando no se conoce previamente el número de iteraciones que se van a realizar. Y la condición de salida se genera internamente en el cuerpo del bucle.
 - Si desconocemos el número de iteraciones y el cuerpo del bucle ha de ejecutarse al menos una vez usaremos un bucle `do-while`.
 - Cuando se conoce de antemano el número de iteraciones, se usa preferentemente el bucle `for`.



Equivalencia entre bucles

- Ejemplo:

Programa que pida números al usuario mayores que cero hasta que el acumulado supere un valor máximo también fijado por el usuario.

Se mostrará ese acumulado y el valor máximo en cada petición de un nuevo número.



Equivalencia entre bucles

- Bucle while:

```
Sanner sc = new Sanner(System.in);  
//variables  
int numero=0;  
int maximo=0;  
int acumulado=0;  
//pedimos el valor máximo  
System.out.println("Introduce valor límite: ");  
maximo=sc.nextInt();  
while(acumulado <= maximo) {  
    //pedimos número  
    System.out.println("Introduce número: ");  
    numero=sc.nextInt();  
    //acumulamos  
    acumulado=acumulado + numero;  
    //mostramos como va el proceso  
    System.out.println("Válor límite: " + maximo);  
    System.out.println("Acumulado hasta el momento: " + acumulado);  
}
```

Equivalencia entre bucles

- ¿Bucle do-while?:
- ¿Bucle for?

Bucles anidados

- Los bucles pueden anidarse, es decir, incluirse unos bucles dentro de otros.
- El efecto del anidamiento es que por cada iteración de un bucle externo el bucle interno ejecuta un ciclo de iteraciones.
- Es decir, si tenemos dos bucles anidados con tres iteraciones en el externo y 10 iteraciones en el interno, el número de iteraciones será de 3 iteraciones para el bucle externo y $3 * 10 = 30$ iteraciones para el interno.
- Cada vez que el bucle interno termine su ciclo de iteraciones el control del programa pasa al bucle externo realizándose otra iteración externa.



Bucles anidados

- Ejemplo1: Imprimir 3 series de números del 1 al 10.

```
for(int i=1; i<=3; i++) {  
    System.out.print(i+": ");  
  
    for(int j=1; j<=10; j++) {  
        System.out.print(j + " ");  
    }  
    System.out.println();  
}
```



Bucles anidados

- Ejemplo2: Imprimir los números del 1 al 30 en 3 líneas.

```
for(int i=0; i<=2; i++) {  
    System.out.print((i+1) + ":" );  
  
    for(int j=1; j<=10; j++) {  
        System.out.print((10*i+j)+ " ");  
    }  
    System.out.println();  
}
```



Salida anticipada de bucles

- Java dispone de las sentencias **break** y **continue** para abandonar la ejecución “normal” de un bucle, forzando su salida.
- Irán siempre dentro de un if.
- Es aconsejable **no usarlas**.



Salida anticipada de bucles

I - Break

- Sale completamente del bucle.
- Ignorando la ejecución de cualquier código dentro del bucle.
- Continua en la siguiente línea después del bucle.



Salida anticipada de bucles

I - Break

- Ejemplo:

```
Calendar cal = Calendar.getInstance();
cal.setTimeInMillis(System.currentTimeMillis());
int dia = cal.get(Calendar.DAY_OF_WEEK);

for (int i = 1; i <= 7; i++) {
    if (dia == i){
        System.out.println("Hoy es el " + i +
                           "º dia de la semana.");
        break;
    }
    System.out.println("Dia " + i);
}

System.out.println("Seguimos...");
```



Salida anticipada de bucles

I - Continue

- Detiene la ejecución de la iteración actual y pasa a la siguiente.
- Mismo ejemplo:

```
Calendar cal = Calendar.getInstance();
cal.setTimeInMillis(System.currentTimeMillis());
int dia = cal.get(Calendar.DAY_OF_WEEK);

for (int i = 1; i <= 7; i++) {
    if (dia == i){
        continue;
    }
    System.out.println("Dia " + i);
}

System.out.println("Seguimos...");
```





Actividades



Ejercicios Tema3

gf

Centro de Enseñanza
Gregorio Fernández