2. A01. Estruturas de control selectivas

2.1 Introdución

Na actividade que nos ocupa aprenderanse os seguintes conceptos e manexo de destrezas:

- Controlar o fluxo dun programa.
- Crear estruturas de control selectivas, tanto simples como múltiples.

2.2 Actividade

2.2.1 Introdución ao control de fluxo

Nos exemplos e programas que fixemos nas unidades anteriores, as sentencias executábanse de maneira secuencial, unha detrás doutra. Con todo, moitas veces é necesario modificar o fluxo de execución das sentencias en base a determinadas condicións. Para iso, Java proporciona estruturas de control que permiten controlar a orde na que se executan unha serie de instrucións, establecendo *bifurcacións*, *bucles* ou *conmutacións*.

As sentencias de control de fluxo en Java divídense en dous grupos:

- **Sentencias selectivas**: Controla a execución de unha ou varias instrucións segundo unha ou varias condicións. As instrucións executaranse unha única vez. Pertencen a este grupo as *bifurcacións* e os *conmutadores*.
- **Sentencias repetitivas**: Controla a execución de unha ou varias instrucións mentres se cumpra una determinada condición. As instrucións executaranse repetidas veces ata que se deixe de cumprir a condición. Pertencen a este grupo os *bucles*.

Nesta actividade estudaremos as sentencias de control selectivas.

2.2.2 Estruturas de control selectivas

Estrutura selectiva simple

As sentencias de selección simple avalían unha ou varias condicións, e se se cumpren, executaranse unhas determinadas instrucións. Se non se cumpren, non se executa nada.

A sentencia de selección simple de Java ten a seguinte sintaxe:

```
if (condición) instrución;
```

A condición ou condicións irán entre parénteses, e se necesitamos executar máis dunha instrución, empregaremos un **bloque de instrucións**, que é un conxunto de instrucións que se escriben entre chaves "{}":

```
if (condición) {
  instrución 1;
    …
  instrución m;
}  //fin do if
```

O diagrama de fluxo para esta sentencia *if* sería:

Así por exemplo, se queremos calcular o perímetro dunha circunferencia deberemos ter en conta que o radio sexa positivo:

```
if (radio > 0) {
  lonxitude = 2 * PI * radio;
  System.out.printl("A lonxitude da circunferencia é: " + radio);
}  //fin do if
```

Pode obterse máis información das estruturas de selección simple na documentación oficial de Java https://docs.oracle.com/javase/specs/jls/se9/html/jls-14.html#jls-14.9.1

Realizarase a tarefa 1 "Estrutura selectiva simple" onde o alumnado resolverá os exercicios propostos empregando estruturas de selección simples.

Estrutura selectiva dobre

As sentencias de selección dobre avalían unha ou varias condicións, e se se cumpren, executan unhas determinadas instrucións, e se non se cumpren, executará outras instrucións diferentes.

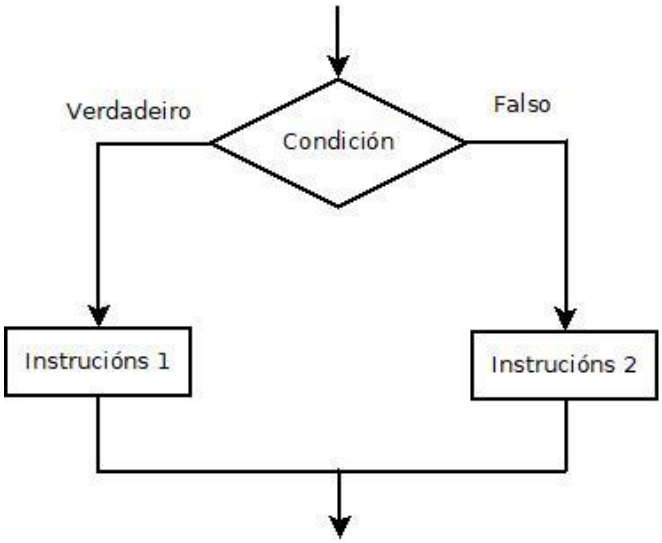
A sentencia de selección dobre en Java é a máis xeral. Ten a seguinte sintaxe:

```
if (condición) instrución;
else instrución 2;
```

Ou ben:

```
if (condición) {
  instrución 1;
  …
  instrución n;
}
else {
  instrución 1;
  …
  instrución m;
}
```

O diagrama de fluxo para esta sentencia *if –else* sería:



Así por exemplo, se consideramos o exercicio anterior no que calculamos o perímetro dunha circunferencia, poderíamos sacar por pantalla unha mensaxe en caso de que o radio non sexa positivo:

```
if (radio > 0) {
  lonxitude = 2 * PI * radio;
  System.out.printl("O perímetro da circunferencia é: " + radio);
}
else {
 System.out.printl("Non se pode calcular o perímetro da circunferencia.
Radio negativo");
}
```

Pode obterse máis información das estruturas de selección dobre na documentación oficial de Java https://docs.oracle.com/javase/specs/jls/se9/html/jls-14.html#jls-14.9.2

Realizarase a tarefa 2 "Estrutura selectiva dobre" onde o alumnado resolverá os exercicios propostos empregando estruturas de selección dobres.

Estrutura selectiva múltiple

As sentencias de selección múltiple, tamén chamadas de "bloques alternativos", avalían o valor dunha variable ou unha ou varias condicións, e en función do resultado executan o conxunto de instrucións asociado ao resultado da avaliación. Pode existir un bloque de instrucións que se execute en caso de que non se executase ningún outro.

A sentencia de selección múltiple en Java realízase con dúas estruturas:

- Con sentencias *if-else* aniñadas.
- Coa sentencia *switch case*, cando o número de bloques de instrucións sexa superior a tres, ou catro, e a selección a poidamos facer en base ao valor enteiro ou carácter resultado da avaliación.

Para aniñar sentencias *if-else* seguirase a seguinte sintaxe:

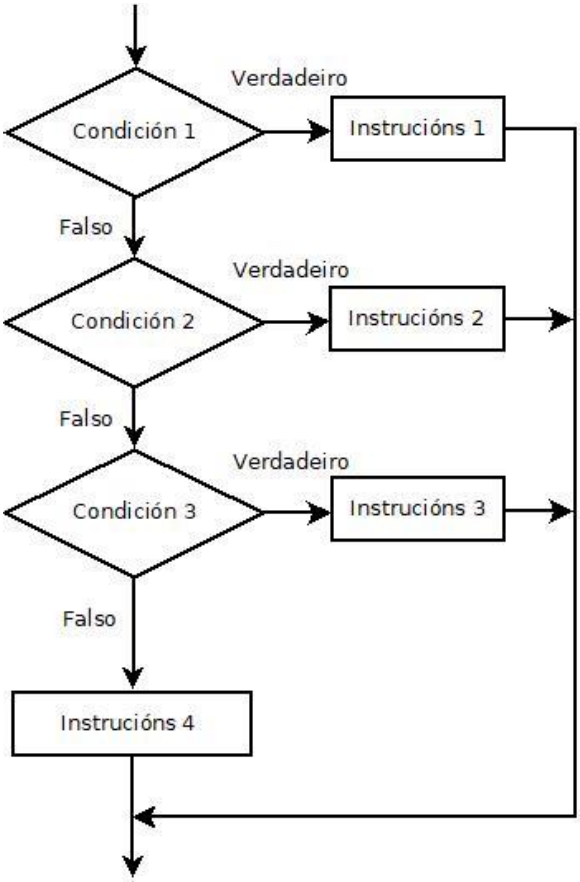
```
if (condición) instrución;
else if (condición 2) instrución2;
….
else instruciónN;
```

Ou ben:

```
if (condición) {
  instrución 1;
  …
  instrución n;
}
else if (condición 2) {
  instrución 1;
  …
  instrución m;
}
…
else {
  instrución 1;
  …
  instrución z;
}
```

Neste caso avaliaranse todas as condicións ata que se atope unha verdadeira.

O diagrama de fluxo para esta sentencia *if –else* aniñada sería:



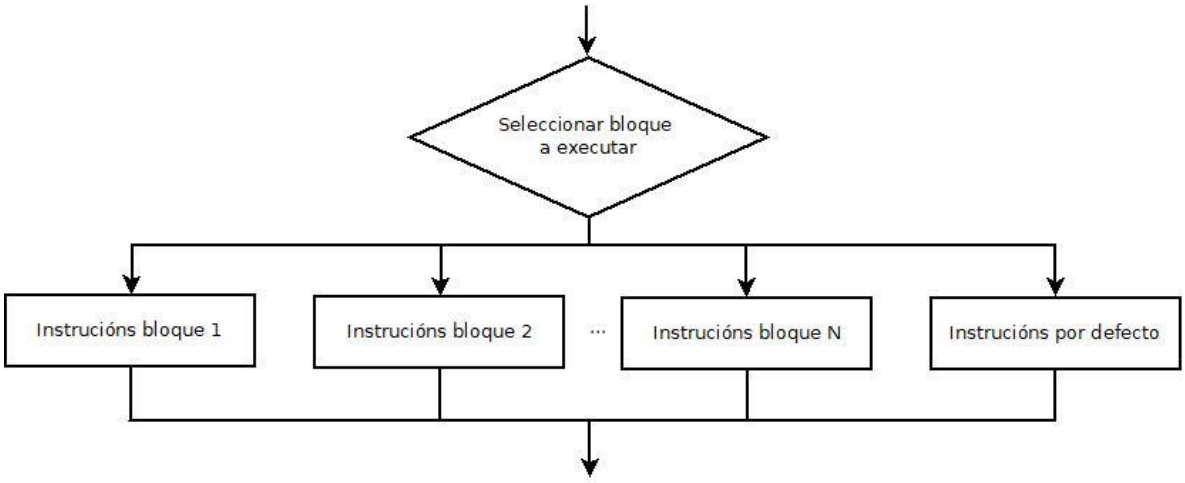
Por exemplo, imos a ler tres números por teclado e a amosar por pantalla o maior dos tres:

```
public static void main(String[] args) {
	java.util.Scanner scan = new java.util.Scanner(System.in);
	int numero1,numero2,numero3,maior;

	System.out.print("\Inserta o primeiro número:");
	numero1=scan.nextInt(); //Lemos o primeiro número
	System.out.print("\Inserta o segundo número:");
	numero2=scan.nextInt(); //Lemos o segundo número
	System.out.print("\Inserta o terceiro número:");
	numero3=scan.nextInt(); //Lemos o terceiro número

	if (numero1>=numero2 && numero1>=numero3)
		maior=numero1;
	else
		if (numero2>=numero1 && numero2>=numero3)
		maior=numero2;
		else
			maior=numero3;

	System.out.print("\n\n\t O número maior é " + maior);
}
```

A sentencia *switch case*, a diferenza da anterior, soamente avalía unha vez, independentemente do número de bloques que teña a estrutura. A súa sintaxe é a seguinte:

```
switch (variable ou expresión){
        case valor1:
                instrución 1;
                 …
                instrución n;
                break;
        case valor2:
                instrución 1;
                 …
                instrución m;
                break;
        …
        case valorN:
                instrución 1;
                 …
                instrución p;
                break;
        [default]: //executarase cando non coincida con ningún dos anteriores
                instrución 1;
                 …
                instrución z;
}  //fin do switch
```

O diagrama de fluxo para esta sentencia *switch-case* sería:



Por exemplo, imos a amosar por pantalla a cualificación dunha nota a partir da puntuación introducida por teclado:

```
public static void main(String[] args) {
        java.util.Scanner scan = new java.util.Scanner(System.in);
        int puntuación;
```

```
        System.out.print("\Introduce a puntuación:");
        puntuación=scan.nextInt(); //Lemos a puntuación

        switch (puntuación){
        case 5:
               System.out.println("APROBADO");
        break;
        case 6:
               System.out.println("BEN");
        break;
        case 7: case 8:
               System.out.println("NOTABLE");
        break;
        case 9: case 10:
               System.out.println("SOBRESAINTE");
        break;
        default:
               System.out.println("SUSPENSO");
        }  //fin do switch
}
```

Pode obterse máis información das estruturas de selección múltiple na documentación oficial de Java https://docs.oracle.com/javase/specs/jls/se9/html/jls-14.html#jls-14.11

Realizarase a tarefa 3 “Estrutura selectiva múltiple” onde o alumnado resolverá os exercicios propostos empregando estruturas de selección múltiple.