TEMA 3: SENTENCIAS DE CONTROL

"Programar,

¿arte o ciencia?"

SENTENCIAS DE SELECCIÓN

En java hay tres tipos de sentencias if: if, if-else, if-else-if

- **Sentencia If**

  Si se cumple la condición se ejecutan las instrucciones.

```
if (condición)
        {
        instrucciones (si sólo hay 1 no se ponen las llaves)
        }
```

Ejemplo

```
if (x<0){
                System.out.println(x);
                x++;
        }

if (x<0)
        x++;
```

- **Sentencia If-else**

  Si se cumple la condición se ejecuta el primer bloque de instrucciones, si no se cumple se ejecuta el bloque del else:

```
if (condición)  {
                instrucciones;
        }
```

```
else{
        instrucciones;
    }
```

Ejemplo

```
if (x<0)
    x++;
else
    x=x*x;
```

Sentencia If-else-if-else (if anidados)

Dentro de una sentencia if puedo introducir otra, y así sucesivamente.

```
 if (condición)
    {
        instrucciones;
    }
else
    {
        if (condición)
            {
            instrucciones;
            }
        else
            {
            Instrucciones;
            }
    }
```

Ejemplo

```
if (numero<0)
    System.out.println("Número negativo");
else
    {
    if (numero>0)
        System.out.println("Número positivo");
    else
        System.out.println("Número nulo");
    }
```

Pueden evaluarse varias condiciones, utilizando los operadores lógicos (&&, ||)

Ejemplos

1. Programa que lee desde el teclado la nota de programación de un alumno y nos dice si ha aprobado o no
2. Programa que lea una cantidad en dólares y nos haga el cambio a euros.
3. Añadir al programa anterior el código necesario para cobrar una comisión del 0,2% si cambiamos menos de 100$
4. Programa que lee dos números enteros y nos dice si son iguales o distintos
5. Programa que calcula el salario semanal de un empleado al que se le paga 15 euros por hora si éstas no superan las 35 horas. Cada hora por encima de 35 se considerará extra y se paga a 22 €.
   El programa pide las horas del trabajador y devuelve el salario que se le debe pagar

6. Programa que lee desde teclado las notas en programación y marcas de un alumno y nos dice si ha aprobado una o las dos asignaturas

7. Escribe un programa que lea una temperatura introducida a través de teclado y muestre por pantalla la actividad más apropiada para dicha temperatura teniendo en cuenta los siguientes criterios.

ACTIVIDAD	TEMPERATURA IDONEA
Natación	temp > 30
Tenis	20 < temp <= 30
Golf	10 < temp <= 20
Esquí	5 < temp <= 10
Parchís	temp <= 5

8. Construir un programa que calcule el índice de masa corporal de una persona (IMC = peso [kg] / altura^2 [m]) e indique el estado en el que se encuentra esa persona en función del valor de IMC:

Valor de IMC	Diagnóstico
< 16	Criterio de ingreso en hospital
de 16 a 17	infrapeso
de 17 a 18	bajo peso
de 18 a 25	peso normal (saludable)
de 25 a 30 sobrepeso	(obesidad de grado I)
de 30 a 35	sobrepeso crónico (obesidad de grado II)
de 35 a 40	obesidad premórbida (obesidad de grado III)
>40	obesidad mórbida (obesidad de grado IV)

Nota 1: se recomienda el empleo de sentencias if–else anidadas.

Nota 2: Los operandos (peso y altura) deben ser introducidos por teclado por el usuario.

- **Sentencia switch**

Cuando una expresión puede tener varios valores y dependiendo del valor que tome hay que ejecutar una serie de sentencias, en lugar de utilizar los if anidados podemos utilizar la sentencia switch

```
switch (variable) {
      case valor:
            instrucciones;
            break;
      case valor:
            instrucciones;
            break;
       default:
            instrucciones;
     } ;
```

Ejemplos

```
switch (posición)
{
     case 1:
          System.out.println("ORO");
          break;
     case 2:
          System.out.println("PLATA");
          break;
     case 3:
          System.out.println("BRONCE");
          break;
     default:
          System.out.println("SIN PREMIO");
          break;
}
```

1. Implementar un programa que pida por teclado la posición de un corredor en una maratón y nos diga la medalla obtenida.

2.Considera estás desarrollando un programa Java donde necesitas trabajar con **objetos de tipo Motor** (que representa el motor de una bomba para mover fluidos). Pide por teclado un número correspondiente a un motor y muestra un mensaje según las instrucciones siguientes:

a) Si el tipo de motor es 0, mostrar un mensaje por consola indicando “No hay establecido un valor definido para el tipo de bomba”.

b) Si el tipo de motor es 1, mostrar un mensaje por consola indicando “La bomba es una bomba de agua”.

c) Si el tipo de motor es 2, mostrar un mensaje por consola indicando “La bomba es una bomba de gasolina”.

d) Si el tipo de motor es 3, mostrar un mensaje por consola indicando “La bomba es una bomba de hormigón”.

e) Si el tipo de motor es 4,mostrar un mensaje por consola indicando “La bomba es una bomba de pasta alimenticia”.

f) Si no se cumple ninguno de los valores anteriores mostrar el mensaje “No existe un valor válido para tipo de bomba”.

3. Codificar un programa que simule el funcionamiento de una calculadora que puede realizar las cuatro operaciones aritméticas básicas (suma, resta, producto y división) con valores numéricos enteros. El usuario debe especificar la operación con el primer carácter: S o s para la suma, R o r para la resta, P, p, M o m para el producto y D o d para la división.

EJERCICIOS

Práctica_3
Práctica_4

CONVERSIÓN DE TIPOS (CASTING)

Existen dos tipos de conversiones, implícitas y explícitas. Sólo se permite la conversión de tipos entre tipos numéricos

Implícitas: se realizan de forma automática entre dos tipos de datos numéricos diferentes cuando la variable destino (izquierda) tiene más precisión que la variable destino (derecha).

Ejemplo

```
int x=3;
float y;
y=x;
```

Explícitas: el programador fuerza la conversión de un tipo en otro mediante una operación llamada cast : (tipo) expresión

Ejemplo

```
float kil;
double cal;
kil = (float) cal;
```

Siempre se convierte de mayor a menor, y sólo en el caso en que sea posible, si no lo es el error puede dar en compilación o en ejecución. Para convertir un real a entero se desecha la parte fraccionaria:

(int)3.8 da como resultado 3

OPERACIONES MATEMÁTICAS

Potencia: Math.pow(2,5) -→ 2^5
Redondeo: Math.round(6.3) → 6
Raiz : Math.sqrt(4) → 2
Valor absoluto : Math.abs(-4) →4

SENTENCIAS DE ITERACIÓN (BUCLES)

Las sentencias de repetición son utilizadas cuando una o varias instrucciones tienen que ser ejecutadas cero, una o más veces.

- **Bucle While**

Se utiliza cuando se tiene que ejecutar un grupo de sentencias un número determinado de veces (no tiene por qué saberse cuantas serán). Las instrucciones

se ejecutan mientras la condición del bucle sea verdadera. Dicha condición se evalúa antes de entrar y cada vez que se ejecuta el bloque de instrucciones. El formato es el siguiente:

```
while (condición)
    {
        instrucciones;
    }
```

```
int numero=1;
while (numero<=10)
    {
    System.out.println(numero);
    numero++;
}
```

Ejemplos:

1. Programa que lee un número entero positivo mayor que 2. Si el número introducido no cumple estas condiciones se pide de nuevo al usuario que lo introduzca.
2. Programa que escribe los pares menores o iguales que un número introducido, este debe ser positivo y mayor que 2
3. Programa que escribe los números menores que 100

- **Bucle Do-While**

```
do
 {
  instrucciones
 }
while (condición);
```

La estructura es igual a la anterior, pero la comprobación de la condición sólo se hace al final del bucle, con lo que siempre se ejecutarán las instrucciones al menos una vez.

```
int numero=1;
do{
    System.out.println(numero),
    numero++;
    }
while(numero<=10);
```

Ejemplos

1. Programa que escribe los números enteros positivos menores que 10
2. Programa que permite al usuario elegir entre una de las cuatro operaciones básicas: +, - ,*, /. El proceso se repite tantas veces como quiera el usuario.

Bucle For

Se utiliza cuando una serie de sentencias se ejecutan un número fijo de veces.

```
for (expresión inicial; expresión de comprobación; expresión de incremento)
    {
            instrucciones;
    }
```

- Expresión inicial = inicializa la variable del bucle al valor con el que empezamos a ejecutarlo.
- Expresión de comprobación = mientras que se cumple la condición se ejecuta el bucle
- Expresión de incremento = incrementa la variable o las variables en cada ejecución de las instrucciones del bucle.

Este bucle se puede conseguir con el while

```
int numero;
for (numero=1; numero<=10;numero++)
        System.out.println(numero);
```

Hemos hecho el mismo ejemplo con los tres tipos de bucles.

Ejemplos

1. Programa que muestra los cuadrados de los 10 primeros números naturales

Los siguientes formatos son válidos

```
int num=10
for(;num>0;num=num-2)

int x, y;
for (x=0, y=1; x < 10 &&  y < 10; x++, y=y+3)
```

Ejemplos:

1. Programa que escribe todas las tablas de multiplicar

SENTENCIAS DE SALTO

- **break:** Se puede utilizar en cualquiera de los bucles vistos. El programa al ejecutar esta instrucción se sale del bucle que está ejecutando. Puede utilizarse también en sentencias de selección (la hemos visto en switch)

- **continue:** Se puede utilizar en cualquiera de los bucles vistos. No en sentencias de selección. Sirve para saltar al principio de un bucle dejando el resto del mismo sin ejecutar

```
int  num,cuadrado;
num=Integer.parseInt(br.readLine().trim());
for(int x=1;x<=num;x++)
        {
        cuadrado=x*x;
        if ( cuadrado>1000)
                break;
        }

int i=0;
while (i<10)
        {
        i++;
        if (i==5)
                continue;
        System.out.println(i);
        }
```

En el caso de que tengamos bucles anidados, se permite el uso de etiquetas para poder decidir de cuál de ellos salimos

- **break con etiqueta**

```
int i=0;
buclesalir:
while (i<100)
        {
        i++;
        For (int j=0;j<i;j++)
                {
                System.out.println("*");
                If (i==5)
```

```
                        break buclesalir;
                    }
                }
```

- **break y continue con etiquetas**

```
uno: for( )
       {
    dos: for( )
         {
         continue;          // seguiría en el bucle interno
         continue uno;      // seguiría en el bucle principal
         break uno;         // se saldría del bucle principal
         }
       }
```

```
 int i=0;
bucleSalir;
while (i<20)
         {
         i++;
         for (int k=1; k < (20-i); k+=2)
            {
            If (i%2==0)
                        continue bucleSalir;
            System.out.print(“_“);
            }
         for (int j=0; j<I, j++)
            System.out.print(“*”);
         System.out.println(“”);
         if (i==19)
            break bucleSalir;
         }
```

EJERCICIOS

Práctica_5
Práctica_6

Funciones

Una función es un conjunto de instrucciones que realizan una tarea concreta dentro de un programa y que pueden devolver un valor.

Supongamos que calculo el factorial de un número, si más tarde en el mismo programa tengo que calcular otro, repito código. En cambio puedo codificar en una función una sola vez las instrucciones que calculan el factorial de un número , y llamar a esta función tantas veces cómo quiera.

En un programa se pueden utilizar tantas funciones como quiera y pueden llamarse entre sí.
Las funciones pueden distribuirse en cualquier orden aunque generalmente aparece primeramente la función main. Para utilizar una función hay que:

- Definirla
- Invocarla

Estructura de la funciones

- **Definición:** normalmente se coloca al terminar el main (aunque pueden aparecer indistintamente e cualquier orden). Aparece con la siguiente sintaxis:

```
Tipo_valor nombre (lista de parámetros)     {
                declaraciones;
                instrucciones;
                return (valor);   (no obligatoria)
        }
```

si no se devuelve nada el tipo es void

```
void suma(int a, int b)
{
System.out.print("Suma: "+(a+b));
}
```

esta función devuelve un entero

```
int suma (int a, int b)
{
 return a+b;
}
```

- **Llamada:** aparece dentro del main o de otra función de la siguiente forma:

```
Nombre_función (valores)
```

En el mismo orden que en la función, y del mismo tipo

suma (5, 4);

En java a las funciones se las llama también métodos. En la clase del main tienen que ir precedidos de la palabra reservada static.

Ejemplos

1. Programa que lee un número entero menor que 100 (N) y mediante una función que lo recibe como parámetro escribe "N veces: módulo ejecutándose".

2. Programa que lee el precio de un producto, su precio rebajado y calcula el descuento(en %) realizado mediante una función

   (precioO-precioF)*100/precio

3. Programa que lee un año (cifra de 4 dígitos) y nos dice si es bisiesto. Utilizar una función

   a%4==0 && a%100!=0 || a%400==0

4. Variar el ejemplo anterior para leer todos los números que el usuario quiera

NOTAS:

1. En el código de una función siempre hay que ser consecuentes con la declaración que se haya hecho de ella. Por ejemplo, si se declara una función para que devuelva un entero, es imprescindible que se coloque un return final para salir de esa función, independientemente de que haya otros en medio del código que también provoquen la salida de la función. En caso de no hacerlo se generará un Warning, y el código Java no se puede compilar con Warnings.

```
int func()
{
if( a == 0 )
   return 1;
return 0;    // es imprescindible porque se retorna un entero
}
```

2. La instrucción return sirve para devolver un valor y para salir de una función

3. La **visibilidad** de una variable se define como la zona del programa en la que es reconocida dicha variable. En java el ámbito de una variable es el bloque de llaves en el que está declarado. Fuera de ese bloque no es visible

Ej. Mostrar los 20 primeros primos usando el método esPrimo.

```
Public class Primos{
        Public static void main(String st [] ){
                boolean primo=true;
                Int numPrimos=0,numero=1;
                Final int totalPrimos=20;

                While( numPrimos<totalPrimos){
                        if (esPrimo(numero) ){
                                System.out.println(numero);
                                numPrimos++;
                                }
                        numero++;
                        }
        }
        public static boolean esPrimo(int n){
                for(int i=2; i<n; i++)
                        if (n%i==0)
                                return false;
                return true;
        }
}
```

EJERCICIOS

Práctica_7