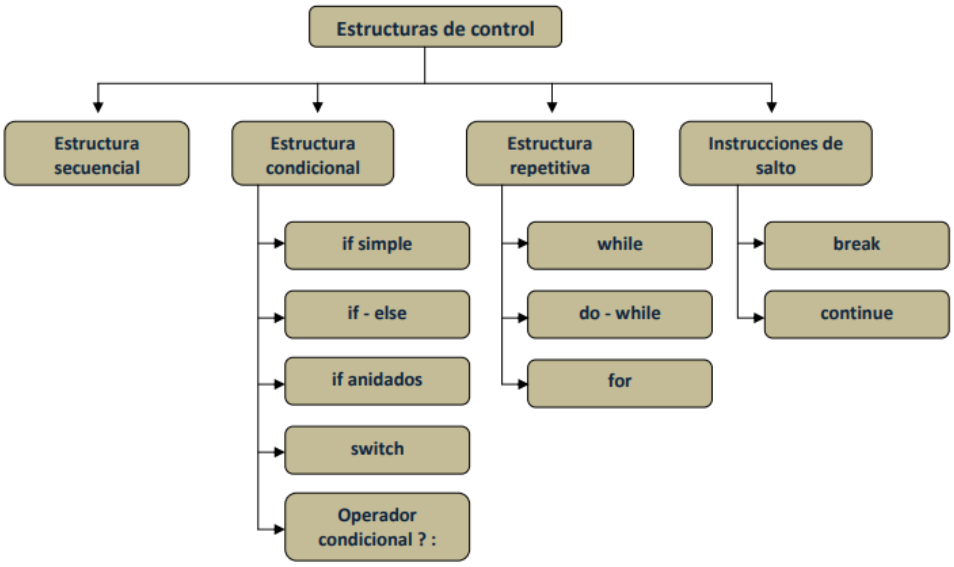
ESTRUCTURAS DE CONTROL EN JAVA

Los programas contienen instrucciones que se ejecutan una a continuación de la otra, siguiendo el orden en que se han escrito. En la gran mayoría de programas será necesario "romper" la secuencia de ejecución lineal descendente añadiendo variantes en el flujo de ejecución.

- **Bifurcaciones** en los programas que permitan varios caminos posibles con diferentes instrucciones **(Estructuras condicionales).**

- **Repetición** de una serie de instrucciones que han de ejecutarse varias veces antes de que el programa siga su curso **(Estructuras repetitivas o bucles).**



Por defecto, las instrucciones de un programa se ejecutan en **orden secuencial**, una detrás de otra, según aparecen escritas dentro del programa. Las instrucciones se agrupan en bloques de código entre corchetes **{}**

Ejemplo (1)

```
/*
  Programa que pide un nombre (string) por teclado y lo muestra por pantalla formando un mensaje.
*/
        import java.util.Scanner;  // importa la clase Scanner del paquete java.util – NO HACER NUNCA import java.util.*
         public class EjemSecuencial1 {
                    public static void main(String[] args) {
                    Scanner sc=new Scanner(System.in);  //crea un objeto de tipo Scanner (flujo de entrada) conectado al teclado
                    System.out.print("Hola, ¿Cómo te llamas?");
                    String miNombre=sc.nextLine();
                    /* el objeto Scanner sc llama a su método nextLine() para solicitar por teclado un String (cadena de caracteres)
                       El método nextLine() recogerá todo lo que tecleemos hasta que pulsemos RETORNO y lo asignará a la
                       variable de tipo String miNombre
                    */
                    System.out.println("Así que te llamas "+ miNombre);
                    sc.close();
               }
         }
```

Ejemplo (2)

```
/*
  Programa que lee dos números por teclado y los muestra por pantalla.
*/

        import java.util.Scanner;  // importa la clase Scanner del paquete java.util

        public class EjemSecuencial2 {
           public static void main(String[] args){
              //declaración de variables
              int n1, n2;
              Scanner sc = new Scanner(System.in);
              //lee el primer número por teclado usando el método nextInt() que sirve para recoger un dato int desde teclado
              System.out.print("Introduce un número entero: ");
              n1 = sc.nextInt();
              //lee el segundo número por teclado
              System.out.print("Introduce otro número entero: ");
              n2 = sc.nextInt();
              //mostrar resultado
              System.out.println("Ha introducido los números: " + n1 + " y " + n2);    //Atentos al uso de + con Strings
              sc.close () ;
           }
        }
```

Ejemplo (3)

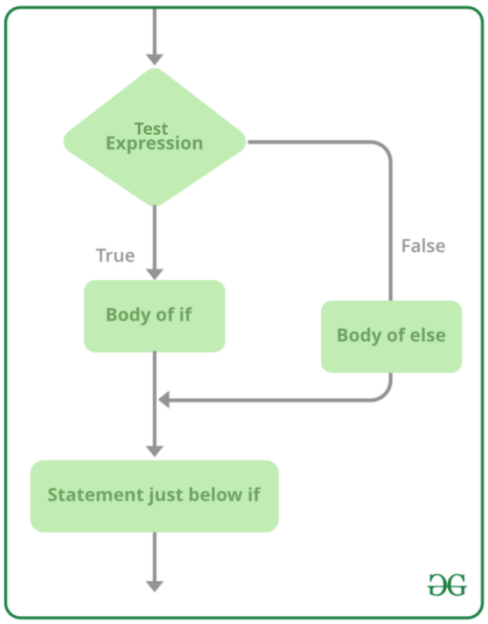
```
/*
 Mostrar la suma, resta y multiplicación de dos valores de tipo double que se introducen por teclado.
 */
        import java.util.Scanner;

        public class EjemSecuencial3{
           public static void main(String[] args){
              Scanner sc = new Scanner(System.in);
              double num1, num2;
              System.out.print("Introduce el primer número:");
              num1 = sc.nextDouble();
              System.out.print("Introduce el segundo número:");
              num2 = sc.nextDouble();
              System.out.println("Los números introducidos son: " + num1 + "  " + num2);
              System.out.println("La suma de: " + num1 + " + " + num2 + " = " + (num1+num2));
              System.out.println("La resta de: " + num1 + " - " + num2 + " = " + (num1-num2));
              System.out.println("La multiplicación de: " + num1 + " * " + num2 + " = " + num1*num2);
              sc.close () ;
            }
        }
```

ESTRUCTURAS CONDICIONALES.

Permiten modificar el orden de ejecución de las instrucciones del programa creando bifurcaciones o caminos alternativos de ejecución, que dependerán del cumplimiento o no de una **CONDICIÓN**. De ahí el nombre de **estructuras condicionales**. La condición que se comprueba para decidir si se toma un camino u otro debe ser una expresión condicional booleana, cuyo resultado será true/false.

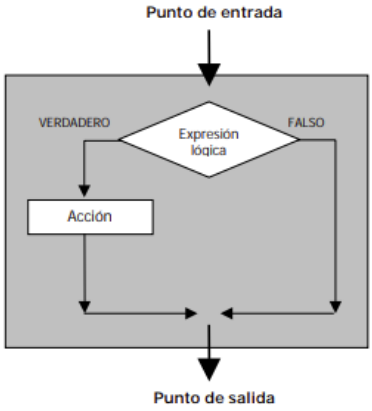
De modo genérico, diremos que el programa llega a un punto en el que se evalúa una condición booleana, y en ese punto se bifurca, tomando un camino u otro según sea el resultado de evaluar la condición.



Nos podemos encontramos varias **alternativas**:

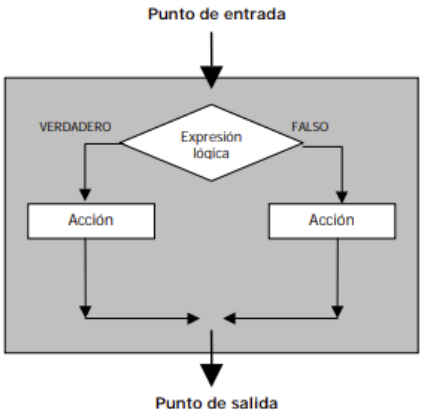
CONDICIONAL SIMPLE – if

```
if (condicion) {
  // Bloque de sentencias – condición true
 }
```



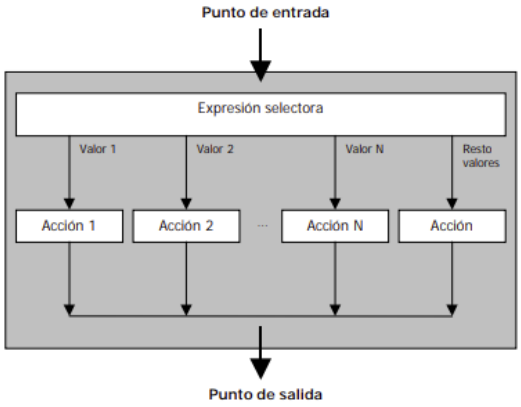
CONDICIONAL DOBLE – if … else

```
if (condicion) {
      // Bloque de sentencias – condición true
 }else {
     //Bloque de sentencias - condición false
 }
```



CONDICIONAL MÚLTIPLE – switch

```
switch (variable) {
   case valor1:
        // Bloque de sentencias 1
        break;
   case valor2:
        // Bloque de sentencias 2
        break;
   case valor3:
        // Bloque de sentencias 3
        break;
   default:
        // Bloque de sentencias
   }
```



Ejemplo (4) – Estructura condicional SIMPLE

```
/*
  Programa que pide una nota por teclado y muestra un mensaje en caso de Aprobado
 */
        import java.util.Scanner;

        public class CondicionalSimple1 {
            public static void main( String[] args ){
                Scanner sc = new Scanner(System.in );
                System.out.print("Teclea la Nota: ");
                int nota = sc.nextInt();
                if (nota >= 5){
                    System.out.println("Enhorabuena, has aprobado!!");
                }
                sc.close();
            }
        }
```

Si un bloque de instrucciones contiene una sola instrucción no sería necesario escribir las llaves { }, aunque para evitar confusiones **se recomienda escribirlas siempre.**

Ejemplo (5) – Estructura condicional DOBLE

```
/*
   Programa que pide una nota por teclado y muestra si se ha aprobado o no
 */

        import java.util.Scanner;

        public class CondicionalDoble1{
            public static void main( String[] args ){
                Scanner sc = new Scanner(System.in);
                System.out.print("Teclea la Nota: ");
                int nota = sc.nextInt();
                if (nota >= 5){
                    System.out.println("Enhorabuena!!, Has aprobado");
                } else {
                   System.out.println("Lo siento, has suspendido");
                }
                sc.close();
            }
        }
```

Ejemplo (6) – Estructura condicional DOBLE

```
/*
  Programa que pide un número por teclado y muestra si es par o impar
 */
        import java.util.Scanner;

        public class CondicionalDoble2 {
            public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                int num;
                System.out.print("Introduzca un número entero: ");
                num = sc.nextInt();
                if (num % 2 == 0) {
                    System.out.println("El número " + num+ " ES PAR");
                } else {
                   System.out.println("El número " + num+ " ES IMPAR");
                }
                sc.close();
            }
        }
```

Ejemplo (7) – Estructura condicional ANIDADA

```
/*
  programa que solicita por teclado una nota numérica (con decimales) y escribe en pantalla la calificación en texto
*/
        public class CondicionalAnidado1 {
            public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                System.out.print("Introduzca una nota entre 0 y 10: ");
                float nota = sc.nextFloat();
                System.out.println("La calificación del alumno es ");
                if (nota < 0 ||nota > 10) {
                    System.out.println("Nota no válida");
                } else if (nota == 10) {
                            System.out.println("Matrícula de Honor");
                        } else if (nota >= 9) {
                                    System.out.println("Sobresaliente");
                                } else if (nota >= 7) {
                                            System.out.println("Notable");
                                        } else if (nota >= 6) {
                                                    System.out.println("Bien");
                                                } else if (nota >= 5) {
                                                            System.out.println("Suficiente");
                                                        } else {
                                                            System.out.println("Suspenso");
                                                        }
                sc.close();
            }
        }
```

Ejemplo (8) – Estructura condicional de Selección MÚLTIPLE (Switch). Se utiliza para seleccionar una de entre **múltiples** opciones posibles. **Es una alternativa práctica y sencilla a los if .. else anidados**. El flujo de ejecución del programa lo determina el valor de una variable o expresión. El tipo de esta variable o expresión puede ser:

- Tipo primitivo: byte, short, char, int, o sus clases **envolventes (ni float ni double están permitidos)**
- Tipo String o tipos enumerados (enum).

Para el ejemplo anterior de las notas, vamos a utilizar ahora una estructura **Switch.** Para ello lo tenemos que hacer con notas de tipo **int** (sin decimales). Más adelante cuando hablemos de la clase **Math,** aprenderemos a pasar de decimal a entero truncando, redondeando y muchas cosas más.

```
import java.util.Scanner;
public class SeleccionMultiple1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Introduzca una nota entera entre 0 y 10: ");
        int nota = sc.nextInt();
        System.out.println("La calificación del alumno es ");
        switch (nota) {
            case 0: case 1: case 2: case 3: case 4:
                System.out.println("SUSPENSO");
                 break;
             case 5:
                System.out.println("SUFICIENTE");
                 break;
            case 6:
               System.out.println("BIEN");
                break;
            case 7: case 8:
                System.out.println("NOTABLE");
                break;
            case 9: case 10:
                System.out.println("SOBRESALIENTE");
                break;
           default :
              System.out.println("NOTA NO VÁLIDA");
        }
        sc.close();
    }
}
```

Ejemplo (9) – Programa que solicita por teclado un día (en número) y muestra por pantalla el día en nombre

```
import java.util.Scanner;
public class SeleccionMultiple2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Introduzca un numero de día de la semana: ");
        int dia = sc.nextInt();
        switch (dia) {
            case 1:
                System.out.println("LUNES");
                break;
           case 2:
              System.out.println("MARTES");
              break;
           case 3:
              System.out.println("MIERCOLES");
              break;
           case 4:
              System.out.println("JUEVES");
              break;
           case 5:
              System.out.println("VIERNES");
              break;
           case 6:
              System.out.println("SABADO");
              break;
           case 7:
              System.out.println("DOMINGO");
              break;
           default :
              System.out.println("DÍA NO VÁLIDO");
        }
    }
}
```

Ejemplo (10). Estructura de selección múltiple (3): programa que pide por teclado dos números enteros y un operador **+, -, *, /** y muestra el resultado de la operación según el operador introducido. En caso de división, se comprueba si el divisor es cero, en ese caso la división no se puede realizar y se muestra un mensaje.

```
//Programa que pide dos números enteros y un operador y muestra el resultado de la operación

import java.util.Scanner;

public class SeleccionMultiple3 {
    public static void main(String[] args) {
        int num1, num2, resultado = 0 ;
        char operador;
        Scanner sc = new Scanner(System.in);
        System.out.print("Teclea el primer operando:");
        num1 = sc.nextInt();
        System.out.print("Teclea el segundo operando:");
        num2 = sc.nextInt();
        System.out.print("¿Qué operador quieres utilizar? (+,-,*,/):");
        operador = sc.next().charAt(0);
```

/* Para leer por teclado el operador a emplear, recurrimos al método **sc.next()** que nos permite leer por teclado una cadena hasta el primer **espacio en blanco.** Con el método **charAt(0)** tomamos el primer carácter de la cadena.
En este ejemplo usamos **next()** en vez de nextLine() por culpa del "malfuncionamiento" característico de los **nextInt()** previos, que leen el entero que tecleamos, dejando en el "buffer" de teclado el carácter de salto de línea "/n" */

```
        switch (operador) {
          case '-' :
            resultado = num1 - num2;
            break;
          case '+' :
            resultado = num1 + num2;
            break;
          case '*' :
            resultado = num1 * num2;
            break;
          case '/' :
            resultado = num1 / num2;
            break;
          default :
            System.out.println("\nOperador no valido");
        }
        System.out.println("\nEl resultado es: " + resultado);
        sc.close();
    }
}
```

Es **importante** entender que una vez que la ejecución del programa entra en uno de los *case*, el programa sigue ejecutándose desde ese punto hasta que encuentre el primer *break* (estén dentro del mismo *case* o no).

OPERADOR CONDICIONAL TERNARIO ? :

El operador condicional ternario se puede utilizar en sustitución de la instrucción condicional ***if-else***.

expresión1 **?** expresión2 **:** expresión3 **Equivale a:**

```
if (expresión1){
      expresión2
} else {
     Expresión3
}
```

Ejemplo (11): programa que pide por teclado un número entero y muestra si es positivo o negativo. Consideramos el cero como positivo.

```
import java.util.Scanner;

public class OperadorTernario {
  public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);
     System.out.print("teclea un número: ");
     int num = sc.nextInt();
     System.out.println(num >= 0 ? "POSITIVO" : "NEGATIVO");
    //usamos el operador ternario directamente en el método println
     sc.close();
  }
}
```

La línea *System.out.println(num >= 0 ? "POSITIVO" : "NEGATIVO");* **es equivalente a:**

```
if(num >= 0){
   System.out.println("POSITIVO");
}else{
   System.out.println("NEGATIVO");
}
```

Ejemplo (12): programa que pide por teclado un número entero y muestra si es par o impar.

```
import java.util.*;
public class OperadorTernario2 {
    public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       System.out.print("Teclea un número: ");
       int num = sc.nextInt();
       System.out.println(num%2 == 0 ? "PAR" : "IMPAR");
       //Si el resto (operador %)  de dividir un número entre 2 es 0 entonces el PAR, sino es IMPAR
       sc.close();
    }
}
```