## IV. Estructuras de control:

La secuencia lógica de ejecución de un programa puede ser alterada cuando el usuario lo requiera, a través del uso de las estructuras de control.

Las estructuras de control de la programación son: **secuencia**, **Alternativas** y **ciclos de repetición.**

### Estructura de Secuencia

Es aquella en donde las sentencias se ejecutan una detrás de otra o sea en forma consecutiva.

Las sentencias pueden ser:

- **Simple: Está constituida por una sola instrucción** (Ejm. Sentencia de declaración, lectura, imprimir y asignación)
- **Compuesta: Está constituida por varias instrucciones** (Ejm. if, Switch, do while , while y for).

| **Formato de la estructura de secuencia** | **Ejemplo** | **Funcionamiento** |
|---|---|---|
| Sentencia_1;<br>Sentencia_2;<br>Sentencia_3;<br>Sentencia_4;<br>Sentencia_5;<br>Sentencia_6; | 1. int n, d;<br>2. d=d+1;<br>3. n=*2;<br>4. if ( n% d == 0)<br>4.1System.out.println(n+ "es divisible por " + d);<br>5. System.out.println("El valor de d = "+d);<br>6. System.out.println("El valor de n = "+n); | Las sentencias se ejecutan secuencialmente(consecutivamente) |

Las estructuras de alternativa y repetición son aquellas sentencias que evalúan una expresión lógica, la cual puede dar como respuesta **true o false**, dependiendo de la condición evaluada.

Quiere decir que para trabajar con las estructuras de alternativas y las de repetición requerimos de los operadores relacionales y lógicos.

### Operadores Relacionales y Lógicos

Se utilizan para formar expresiones de comparación cuyos resultados se dan en estados de **true** y **false (boolean).**

**Operadores relacionales**

| Operador relacional | Descripción |
|---|---|
| < | menor que |
| <= | menor o igual a |
| > | mayor que |
| >= | mayor o igual a |
| == | igual a |
| != | no igual a, distinto que |

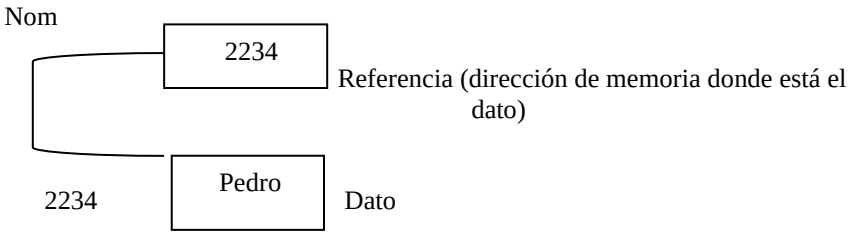
Cuando se comparan los objetos con ( = = o !=) no se compara el contenido.
Para comparar el contenido (dato) se utiliza el método equals

El método equals () se encuentra en la clase String y es **public final,** por lo que no se necesita instanciar un objeto a la clase para utilizar sus métodos. La clase String está en el paquete java.lang

**Representación interna de un dato tipo String**

un dato de tipo String es un objeto)

String nom="Pedro";




| Formato | Ejemplo |
|---|---|
| public boolean equals(String cadena) | String p1="Aquí";<br>String p2= "Aquí";<br>if (p1.equals(p2) )<br>System.out.println ("Los objetos tienen el contenido igual");<br><br>else<br>System.out.println ("Los objetos no tienen el contenido igual"); |

## Operadores lógicos

| Operador | Descripción |
|---|---|
| && | AND en corto circuito( si el primer operando es false, entonces el segundo no es evaluado |
| \|\| | Or en corto circuito (si el primer operando es true el segundo no se evalúa) |
| ! | NOT (invierte el resultado) |
| & | AND lógico (Da como resultado true, si ambas comparaciones son true) |
| \| | OR lógico(Si una de las comparaciones es true , el resultado es true) |

**Ejemplo 1:**

(dnom != 0 && (num/dnom) > 10) si dnom es cero el resultado es false, se evitará división por cero

(dnom != 0 & (num/dnom) > 10) si dnom es cero el resultado es ERROR

**Ejemplo 2:**

((x>0) && (Math.sqrt(x) >= 2) si x es negativa el resultado es false

((x>0) & (Math.sqrt(x) >= 2) si x es negativa el resultado es ERROR

**Ejemplo 3:**

(c = = 1 & m1++ < 100) si c es igual a 1 el resultado es cierto o falso, se aplicara el incremento a m1

**Ejemplo 4:**

(a > 3 & x == 3) si a >3 y el resultado es cierto o falso, se compara si x == 3

### Jerarquía de los operadores relacionales y lógicos:

1) ( )
2) !
3) >, <, >=, <=
4) = =, !=
5) &
6) |
7) &&
8) ||

Al trabajar con las estructuras de alternativas y repetición también requerimos saber que es un bloque de instrucciones.

### Bloque de instrucciones:

El bloque de instrucciones se especifica con **{**

_____;
_____;
_____;

}

## 4.1 Estructuras de alternativas

**4.1.1 Sentencia if:** La expresión condicional que controla if debe dar como resultado un valor booleano (true o false).

| Formato del if simple | Ejemplo | Funcionamiento |
|---|---|---|
| if (condición)<br>sentencia(s); | int n, d;<br><br>if ( n% d == 0)<br>System.out.println(n+ "es divisible por " + d);<br>//finalización del if<br><br>n=n+1; | Si la expresión condicional es verdadera, las sentencia(s) a continuación del if son ejecutadas.<br>Y después continua por su secuencia normal |

| Formato del if doble | Ejemplo | Funcionamiento |
|---|---|---|
| if (condición)<br>sentencia(s);<br>else<br>sentencia(s); | int edad;<br><br>if (edad > 18)<br>a="Eres mayor de edad ";<br>else<br>a= "No llegas aún a ser mayor";<br>//finalización del if | Si la expresión condicional es verdadera, la(s) sentencia(s) a continuación del if son ejecutadas; o si es falso, se ejecutan la(s) sentencia(s) a continuación del else .<br>Y después en ambos casos continúa por su secuencia normal. |

| Formato del if anidado | Ejemplo | Funcionamiento |
|---|---|---|
| if (condición1)<br>sentencia(s);<br>else<br>if (condicion2)<br>sentencia(s);<br>else<br>if (condicion3)<br>sentencia(s);<br>else<br>sentencia(s); | int x ;<br>if (x = = 1)<br>return "x es uno ";<br>else<br>if (x = = 2)<br>return "x es dos ";<br>else<br>if (x = = 3)<br>return "x es tres ";<br>else<br>if (x = = 4)<br>return "x es cuatro ";<br>else<br>return "x no está entre 1 y 4 "; | .if de alternativa múltiple<br>Maneja varias condiciones diferentes y del conjunto de condiciones se ejecuta una de ellas.<br>Y después continúa por su secuencia normal. |

| | | |
|---|---|---|
| | //finalización del if | |

**4.1.2 Sentencia Switch:** permite a un programa seleccionar entre muchas alternativas. Ahora el switch solo evalúa por igualdad y contra una lista de constantes.

| Formato | Ejemplo | Descripción |
|---|---|---|
| switch (expresión)<br>{case constante_1:<br>sentencia1;<br>break;<br>case constante_2:<br>sentencia2;<br>break;<br>case constante_n:<br>sentencia_n;<br>break;<br>.<br>.<br>default:<br>sentencia_nn;<br>}<br><br>**El break o el default es opcional** | int x;<br>switch (x )<br>{ case 0: retornar (“x es cero”);<br>break;<br>case 1: retornar (“x es uno”);<br>break;<br>case 2: retornar (“x es dos”);<br>break;<br>case 3: retornar (“x es tres”);<br>break;<br>case 4: retornar (“x es cuatro”);<br>break;<br>default: retornar (“x es cinco o más”);<br>} //finalización del switch | **expresión**<br>debe ser de tipo char, byte, short o int.<br><br>**constante 1 a n**<br>deben ser de un tipo compatible con la expresión.<br>Cuando una condición se cumple, las sentencias asociadas con aquel casen se ejecuta hasta que el break se encuentre y se va al final del switch. Dos constantes casen no pueden tener idéntico valor en el mismo switch.<br><br>**default**<br>es opcional. es ejecutado si ninguna constante no corresponde con la expresión. |

| Ejemplo | Ejemplo |
|---|---|
| int x;<br>switch ( x )<br>{ case 1:<br>case 2:<br>case 3: System.out.println (“x es uno,dos o tres”);<br>break;<br>case 4: System.out.println (“x es cuatro”);<br>break;<br>} | char x,y;<br>switch (x)<br>{case ‘A’:<br>System.out.println (“Esta a es del switch externo”);<br>switch (y)<br>{ case ‘A’: System.out.println (“Esta a es del switch interno”);<br>break;<br>case ‘B’: m++;<br>} //Finalización del switch interior<br>break;<br>case ‘B’: n++;<br>} // Finalización del switch externo |