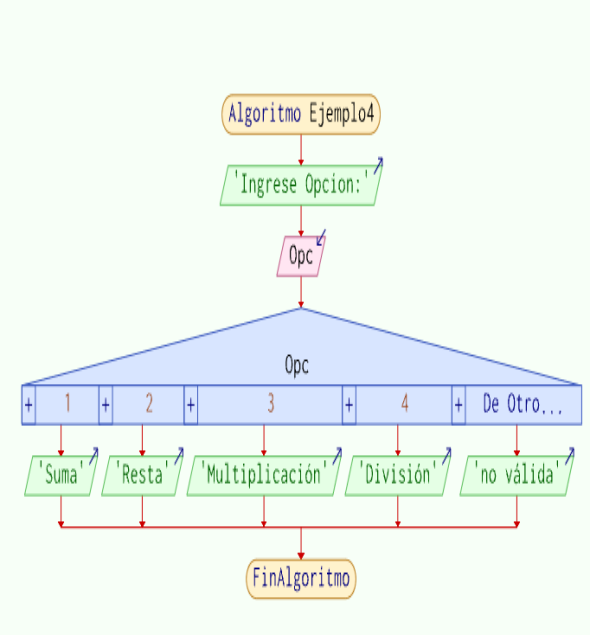
ARQUITECTURA DE
PLATAFORMAS Y SERVICIOS DE TI

III SEMESTRE



Puntos a tratar:

- Condicional múltiple:
  - if – else anidada.
  - if – else if – else.
  - switch
- Desarrollo de ejercicios de aplicación de las estructuras condicional múltiple.

Objetivo de la sesión:

Al concluir la sesión el estudiante estará en la capacidad de implementar las estructuras condicionales múltiples en su código de consola java.

ESTRUCTURA CONDICIONAL MÚLTIPLE

INSTITUTO DE EDUCACIÓN SUPERIOR TECNOLÓGICO PÚBLICO "CHINCHA"

TEMA 04

FUNDAMENTOS DE PROGRAMACIÓN

NG. BARRIOS QUISPE, RICHARD JHONSON
Ingeniero de sistemas

2025

Tema 04:

ESTRUCTURA CONDICIONAL MÚLTIPLE

1. CONDICIONAL MÚLTIPLE

La estructura condicional múltiple es aquella que se utiliza cuando hay más de 2 alternativas (rompe el booleano true/false).
Existen 3 formas de lograr este objetivo y son:

1.1. if – else ANIDADA

Esta forma de implementar una estructura condicional múltiple se basa en escribir una (o varias) funciones condicionales (simples o dobles) dentro de otras estructuras condicionales (simples o dobles).

1.1.1. SINTAXIS

SINTAXIS

```
if (condición){
     if (condición){
          Instrucción1_camino_true;
          .          .          . ;
          .          .          . ;
          .          .          . ;
          InstrucciónN_camino_true;
     }
     else{
          Instrucción1_camino_false;
          .          .          . ;
          .          .          . ;
          .          .          . ;
          InstrucciónN_camino_false;
     }
}
else{
     if (condición){
          Instrucción1_camino_true;
          .          .          . ;
          .          .          . ;
          .          .          . ;
          InstrucciónN_camino_true;
     }
     else{
          Instrucción1_camino_false;
          .          .          . ;
          .          .          . ;
          .          .          . ;
          InstrucciónN_camino_false;
     }
}
```

En muchas ocasiones resulta conveniente insertar un if dentro de otro if. Si, por ejemplo, quisiéramos saber si Juan es mayor, menor o de la misma edad que Ana, normalmente recurriríamos a la utilización de if anidados:

```
1    public class if5 {
2      public static void main (String[] args) {
3        int EdadJuan = 30, EdadAna =25;
4
5        if (EdadJuan<EdadAna)
6          System.out.println ("Juan es mas joven que Ana") ;
7        else
8          if (EdadJuan==EdadAna)
9            System.out.println ("Juan tiene la edad de Ana") ;
10         else
11           System.out.println ("Juan es mayor que Ana") ;
12     }
13   }
```

En la clase if5, en la línea 3, se declaran dos variables de tipo int y se inicializan a 30 y 25; representarán las edades de Juan y de Ana. En la línea 5 se pregunta si la edad de Juan es menor que la de Ana; si lo es, se imprime el mensaje adecuado, y si no, pueden ocurrir dos cosas: que tengan la misma edad o que Ana sea mayor; esto obliga a realizar en este punto una nueva pregunta: ¿Tienen la misma edad?, la línea 8 realiza esta pregunta dentro de la rama else del primer if.

El ejemplo anterior se podría haber resuelto sin hacer uso de sentencias condicionales anidadas.

```
1    public class if6 {
2      public static void main (String [] atgs) {
3        int EdadJuan = 30, EdadAna =25;
4
5        if (EdadJuan < EdadAna)
6          System.out.println ("Juan es mas joven que Ana") ;
7
8        if (EdadJuan > EdadAna)
9          System.out.println ("Juan es mayor que Ana") ;
10
11       if (EdadJuan == EdadAna)
12         Systm.out.println ("Juan tiene la edad de Ana") ;
13
14     }
15   }
```

La solución de la clase if6 resuelve le mismo problema que el ejemplo de la clase if5; además la solución es más legible y fácil de depurar en casi de errores, sin embargo, es importante darse cuenta que en este último caso el ordenador siempre tiene que evaluar tres condiciones, mientras que en la solución aportada en if5 basta con evaluar 1 ó 2 condiciones (dependiendo de las edades de Juan y Ana), por lo cual los if anidados proporcionan una solución más eficiente.

Ejemplos:

El primer ejemplo presenta la forma más simple de instrucción condicional.

```
1   public class if1 {
2     public static void main  (String []  args) {
3       int EdadJuan = 20, EdadAna =25;
4
5        if (EdadJuan><EdadAna)
6          System.out.println ("Juan es mas joven que Ana") _;
7     }
8   }
```

En el segundo ejemplo empleamos las dos ramas de if:

```
1    public classs if2  {
2      public static void main (String [] args) {
3         int EdadJuan = 20, EdadAna =25;
4
5         if (EdadJuan<EdadAna)
6           System.out.println ("Juan es mas joven que Ana") ;
7         else
8            System.out.println ("Juan no es mas joven que Ana");
9       }
10   }
```

1.2. if – else if - else

Esta instrucción es una variante de una condicional múltiple anidada, se puede decir que fusiona los anidamientos.

Consta de 3 tipos de bloques, un bloque tipo **if** se utiliza para definir la condición elegida como el inicio de una serie de condiciones, luego de 1 a varios bloques tipo **else if**, cada uno de estos bloques se ejecuta sólo cuando se cumple la condición que se ha definido para él; finalmente un bloque tipo **else**, este se ejecuta cuando ninguno de los bloques precedentes se ejecutaron es decir cuando no se cumplió ninguna de las condiciones

1.2.1. SINTAXIS

SINTAXIS

```
if (condición_if) {
      Instrucción1_camino_if;
      …;
      …;
      …;
      InstrucciónN_camino_if;
}
else if(condición_else_if_1) {
      Instrucción1_camino_else_if_1;
      …;
      …;
      …;
      InstrucciónN_camino_else_if_1;
}
…
…
…
```

```
else if(condición_else_if_N) {
      Instrucción1_camino_else_if_N;
      …;
      …;
      …;
      InstrucciónN_camino_else_ifN;
}
else{
      Instrucción1_camino_else;
      …;
      …;
      …;
      InstrucciónN_camino_else;
}
```

1.3. switch

Esta instrucción a diferencia de todas las otras condicionales:

NO EVALUA UNA EXPRESIÓN, SINO UNA **VARIABLE**.

Y para esa variable se define una lista de **CASOS (bloques case)** que vienen a ser un listado de posibles valores que puede asumir la variable evaluada; existe además el bloque **default**, que se ejecutará cuando el dato ingresado en la variable evaluada no se encuentre en ninguno de los casos definidos.

1.3.1. SINTAXIS

SINTAXIS

```
switch (Variable) {
      case [Valor1]:
            Instrucciones_Valor1;
            break;
      case [Valor2]:
            Instrucciones_Valor2;
            break;
            …
            …
            …
      case [ValorN]:
            Instrucciones_ValorN;
            break;
      default:
            Instrucciones_default;
            break;
}
```

2. DESARROLLO DE EJERCICIOS DE APLICACIÓN DE LA ESTRUCTURA CONDICIONAL MÚLTIPLE.

EJER001: Nuestro primer ejemplo (Switch1) nos muestra una instrucción switch (línea 5) con una expresión de tipo int. Según el valor de la expresión sea 1, 2, ó 3, se imprimirán diferentes mensajes. Cuando el valor de la expresión es diferente a 1, 2 y 3 (línea 18) se imprime un mensaje genérico (línea 19). En el ejemplo, si la instrucción 12 no existiera, se imprimirían dos mensajes: "Medalla de plata" y "Medalla de bronce".

CODIFICACIÓN:

```
1   public class Switch1 {
2     public static void main  (String []  args) {
3       int Puesto = 2;
4       switch (Puesto) {
5           case 1:
6                 System.out.println ("Medalla de ORO");
7                 break;
8           case 2:
9                 System.out.println ("Medalla de PLATA");
10                break;
11          case 3:
12                System.out.println ("Medalla de BRONCE");
13                break;
14          default:
15                System.out.println ("Gracias por participar");
16                break;
17       }
18    }
19  }
```

EJER002: **EJEMPLO DE RESOLUCIÓN DE PROBLEMAS: (Mensaje días de la semana)**

La estructura de la instrucción *switch* es muy sencilla en nuestro ejemplo, basta con una cláusula *case* por cada día laborable y una cláusula *default* para el sábado y domingo. También pude haber colocado dos cláusulas *case* para los días 6 y 7 y emplear la opción *default* para tratar el posible error en el que *DiaSemana* contenga un valor distinto del 1 al 7.

CODIFICACIÓN:

```
1   public class Switch7 {
2     public static void main  (String []  args) {
3       byte DiaSemana = 4;
4       switch (DiaSemana) {
5           case 1:
6                 System.out.println ("¡Qué duro es el Lunes!");
7                 break;
8           case 2:
9                 System.out.println ("Día de hacer deporte");
10                break;
11          case 3:
12                System.out.println ("Mitad de la semana");
13                break;
```

```
14          case 4:
15                System.out.println ("Jueves día de los solteros");
16                break;
17          case 5:
18                System.out.println ("¡Viernes bendito!");
19                break;
20          default:
21                System.out.println ("¡Fin de semana de juerga!");
22                break;
23       }
24    }
25 }
```

EJERCICIOS DE AUTOESTUDIO PARA EL SIGUIENTE TEMA:

1. Mostrar en pantalla la tabla de multiplicar del 1 al 18.

2. Mostrar los N primeros términos de la siguiente serie, indicando además la suma de los mismos: 7, 9, 12, 16, 21, …

3. Determinar el monto a pagar, semanalmente por una empresa que tiene 20 trabajadores los cuales ganan determinada tarifa por hora normal y por hora extra su tarifa se duplica. Se consideran horas extras aquellas que excedan de 40 a la semana.

4. Un estudiante ha registrado 10 instituciones que dictan un curso de computación de su interés, a diferentes costos. Desea determinar el costo promedio del curso, el costo más elevado, el costo más bajo y el nombre de la institución que ofrece el costo más bajo. Se sabe además que ningún costo llega a 4 cifras.

5. Cree un algoritmo que permita ordenar de "mayor" a menor" 3 números, ingresando los números en una sola variable.

Material redactado con información proveniente de diversas fuentes.