5.- Otras estructuras que proporciona Java.

5.1.- Estructura iterativa generalizada *"for"*.

Los lenguajes de programación suelen proporcionar estructuras de control adicionales a las tres estructuras básicas de Dijkstra cubiertas en la sección anterior, Java no es la excepción y proporciona algunas, solo discutiremos las más comunes.

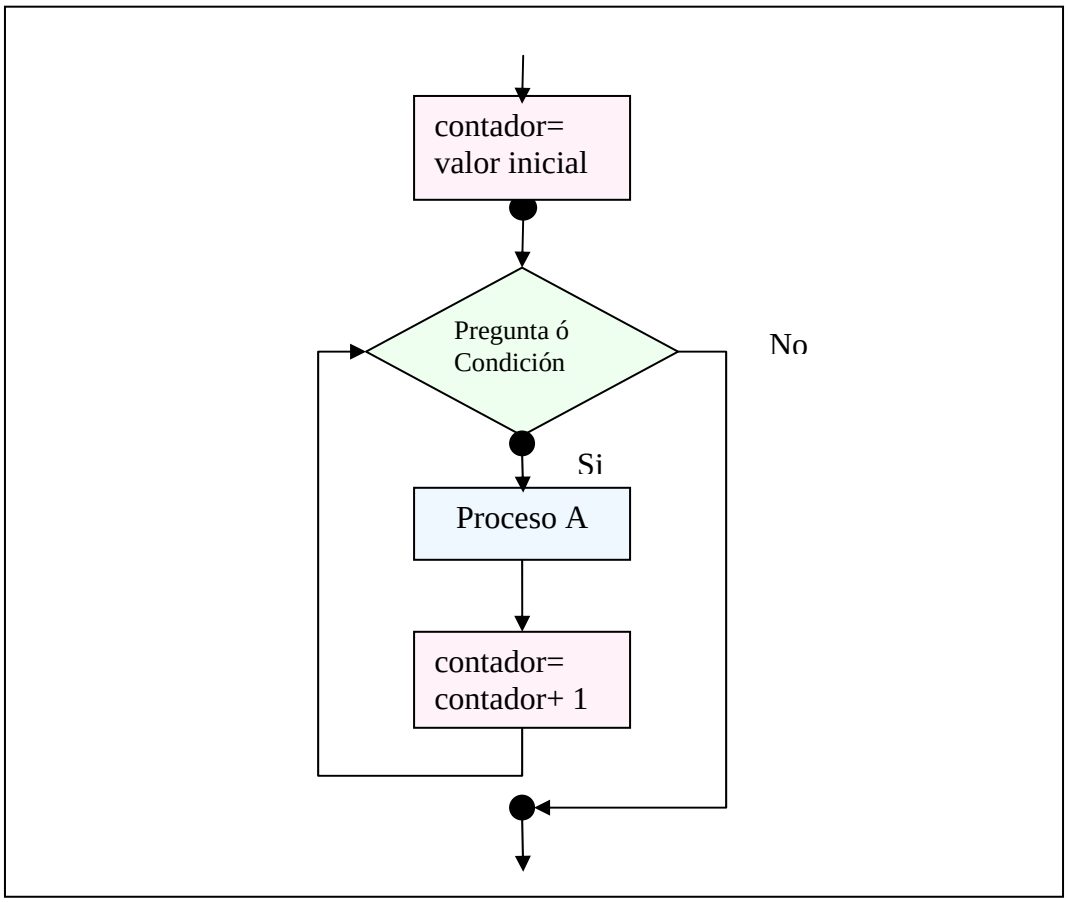

Figura 5.1 Estructura *"for"*

La primera es la estructura "for" que permite integrar contadores a la estructura de iterativa, una forma simple de la estructura for se presenta en el diagrama de la figura 5.1. La sintaxis del diagrama mostrado es la siguiente:

```
for (<contador>=<valor inicial>;<condicion>;<contador=contador + 1)
      {//Proceso A}
```

El contador toma el valor inicial, revisa la condición, si esta se cumple procede a ejecutar el proceso A, incrementa en 1 el contador, revisa de nuevo la condición si esta se cumple

ejecuta de nuevo el proceso A e incrementa en uno el contador, así hasta que la condición no se cumpla que es cuando pasa a la siguiente instrucción.

Así por ejemplo si n es una variable entera con valor de 10, el siguiente lazo hace que i tome valores desde 0 hasta 9.

```
for(i=0;i<n;i++)
{Proceso A que utiliza i}
```

Existe una forma general del for y es muy flexible, el diagrama de la figura 5.2 muestra dicha forma. La sintaxis de esta forma general es la siguiente:

```
for (<Incialización>;<condición>;<actualización>)
      {//Proceso A}
```

En la inicialización se pueden incializar uno o mas contadores, separando cada instrucción por una coma y al terminar como se muestra un punto y coma. La actualización permite actualizar uno o mas contadores separando cada instrucción por una coma. De tal forma que este tipo de sintaxis permite la expresión:

```
for(i=0,j=2;i<n;i++,j=j+2)
{Proceso A que utiliza i y j}
```

I sería un contador de uno en uno y j un contador de dos en dos. Los contadores pueden declararse en la misma instrucción por ejemplo:

```
for(int i=0,int j=2;i<n;i++,j=j+2)
{Proceso A que utiliza i y j}
```

Lo que los hace variables locales al “for”, fuera del proceso A no existen, tal como se comento en el tema de alcance de variables.

Note que esta estructura siempre tiene una estructura “while” que combinada con una estructura secuencial es lógicamente equivalente. Compile y ejecute el ejemplo adjunto 501lazoFor.java que construye diversos lazos simples y el ejemplo 502lazoForGeneral.java que construye lazos con dos contadores.

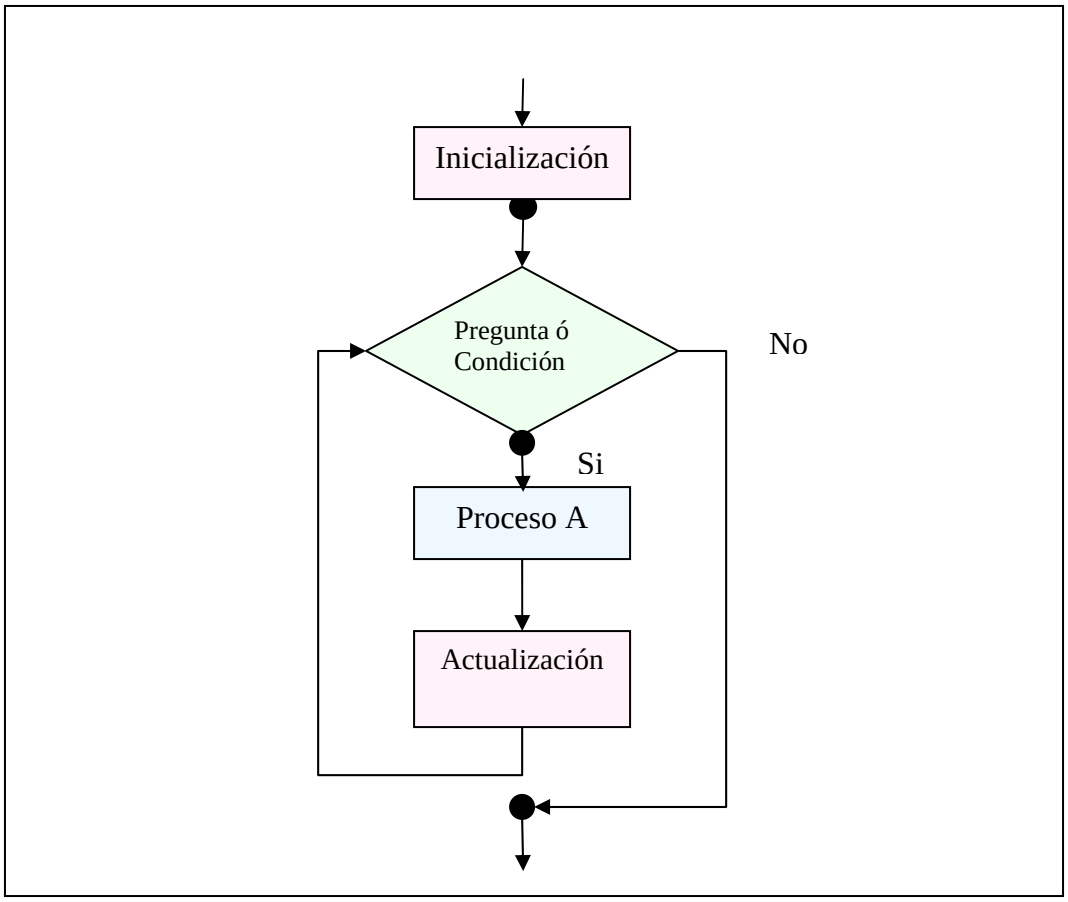

Figura 5.2 Estructura *"for"* generalizada

5.2.- Estructura iterativa de repetición *"do while"*.

La estructura *"do-while"* es muy similar a la estructura *"while"* solo que la prueba de la condición se lleva a cabo al final del ciclo, esto es el proceso A se ejecuta cuando menos una vez. La sintaxis es la siguiente:

```
do{ //proceso A }
while (<nombreVariableTipoboolean>);
```

O bien

```
do{ //proceso A }
while (<expresiónTipoboolean>);
```

El diagrama de la figura 5.3 muestra la estructura. El ejemplo adjunto 503DoWhileMenu.java muestra la implementación de un menú sencillo utilizando esta estructura.

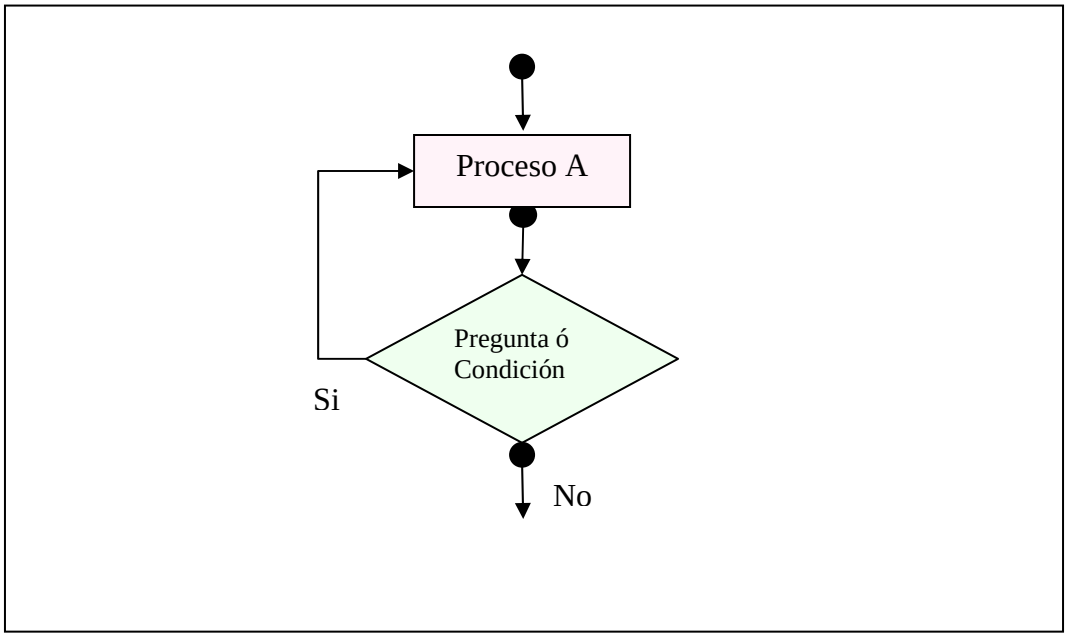

Figura 5.3 Estructura *"do - while"*

5.3.- Estructura de selección múltiple "switch-case".

Para tener opciones múltiples se cuenta con la estructura "*switch-case*" que permite tener n procesos para seleccionar entre ellos dependiendo del valor de una variable de tipo primitivo.
La sintaxis es la siguiente:

```
switch (<variable de control>)
      {  case valor1:
            Proceso 1
            break;
         case valor2:
            Proceso 2
            break;
         case valor3:
            Proceso 3
            break;
            .
            .
            .
         default:
            Proceso por defecto;
            break;
    }
```

La variable de control toma un valor y se prueba contra los valores de las cláusulas "case" si hay alguna que coincida, se ejecuta dicho proceso por ejemplo si el valor de la variable de control coincide con el valor3, entonces el proceso 3 es el que se ejecuta, si el

valor de la variable de control no coincide con ninguno de los valores, se ejecuta el proceso por defecto bajo la cláusula "*default*".
La cláusula "*default"* es opcional, si no aparece no se ejecuta ninguna instrucción cuando sea necesario ejecutar el proceso por defecto.

El diagrama de esta estructura se muestra en la figura 5.4.

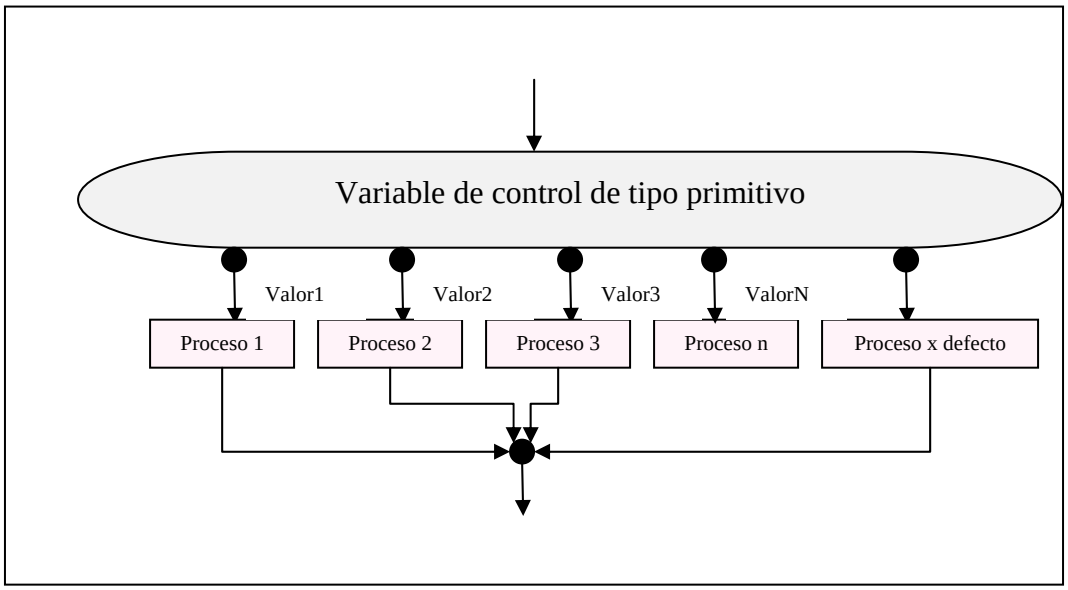

Figura 5.4 Estructura *"switch-case"*

Esta estructura puede ser construida y obtener una lógica equivalente utilizando varias estructuras de selección. El ejemplo 504SwitchCaseMenu.java implementa un menú sencillo combinando las estructuras *do-while y switch-case*. Por otra parte el ejemplo 505ComparacionDeCadenas.java combina las cadenas y la estructura *switch-case.*

5.4.- Instrucción de captura de excepciones: try-catch.

Contamos adicionalmente con una estructura de prevención de errores, es una instrucción para vigilar un grupo de instrucciones y si hay algún error tomar acciones correctivas. Su sintaxis y funcionamiento a continuación:

```
try {
      //Proceso a Vigilar
    }
catch ( <tipoErrorEsperado> <nombreVariableConDatosError>)
    {
      //acciones correctivas
```

}

Así por ejemplo el programa

```
int x=0;
int y=5;
try { y = y + 1;
      y = y/x;
    }
catch (Exception e)
 {Ip.wescribeMensaje("Ha ocurrido un error el mensaje del error es \n"
           + e.getMessage());
  Ip.wescribeMensaje("Se asigna un valor de cero a la variable y");
 }
```

No detiene su ejecución al dividir entre cero, solo envía los mensajes informando del error y asigna un valor a la variable con problemas. El ejemplo adjunto 506TryCatch.java muestra como se utiliza esta instrucción.

5.5.- Actividades

Compile y ejecute los ejemplos adjuntos 507MultiplosDeUnNumero.java, 508NumeroPrimo.java y 509NumerosPrimos.java que presentan la solución de un problema complejo en tres pasos, ilustrando en todo momento los principios de cada estructura.

5.6.- Ejercicios Propuestos:

Escriba un programa que lea numero, nombre y salario de un grupo de empleados que laboran en una empresa, posteriormente el programa deberá imprimir la lista capturada y el total de los salarios de todos los empleados.

Modifique el programa anterior para que calcule el impuesto a pagar sabiendo que el impuesto es el 30% del salario de un empleado, su programa deberá imprimir el importe del impuesto de todos los empleados, la suma de todos los salarios y el total que incluye los salarios más el impuesto.

Agregue el código necesario para que su programa imprima las listas de varias empresas.