4.- Tres estructuras básicas de Dijkstra.

4.1.- Los principios de Dijkstra de programación estructurada.

Edsger Dijkstra, publicó en 1968 un arículo titulado «Go To Statement Considered Harmful» en *Communications of the ACM*. Vol. 3 en el cual comprueba desde un punto de vista práctico que solo se requieren tres estructuras de control en un lenguaje de programación para construir cualquier programa. En las ciencias computacionales modernas Dijkstra es el primer científico que reconoce el fundamento de la lógica de los programas y formalmente proporciona los cimientos de la programación.

El **teorema del programa estructurado** es un resultado en la teoría de lenguajes de programación. Establece que toda función computable puede ser implementada en un lenguaje de programación que combine subrutinas en únicamente tres formas. Esas tres formas (también llamadas estructuras de control) son:

1. Ejecutar una subrutina y luego otra subrutina (secuencia)
2. Ejecutar una de dos subrutinas, dependiendo del valor de una variable booleana (selección o IF-THEN-ELSE)
3. Ejecutar una subrutina mientras una variable booleana sea 'verdadera' (iteración, ciclo o bucle)

Las tres estructuras de control tienen las siguientes características:

1. Cada estructura tiene una sola entrada y una sola salida.
2. Una estructura se puede conectar con otra a través de su entrada o su salida.
3. La definición de una estructura es recursiva, puede contener en un bloque otra estructura que puede ser de cualquiera de los tres tipos: secuencia, selección o ciclo.

Esta sección estudia las estructuras anteriores, sus propiedades y relaciones.

4.2.- La estructura secuencial.

Ejecutar una subrutina y luego otra subrutina, imaginémonos que sostenemos nuestro programa contra un muro, y una gota de agua cae sobre el escrito, dicha gota de agua

sigue la ley de la gravedad y se mueve hacia abajo del programa, este es el orden en el que naturalmente se ejecuta un programa, en forma secuencial. Esta secuencia no se interrumpe, solo se modifica utilizando otras dos estructuras de control.

Recordando que en cada una de las estructuras de Dijkstra solo existe una forma de entrar a ella y una forma de salir la estructura de secuencia quedaría como se muestra en la figura 4.1

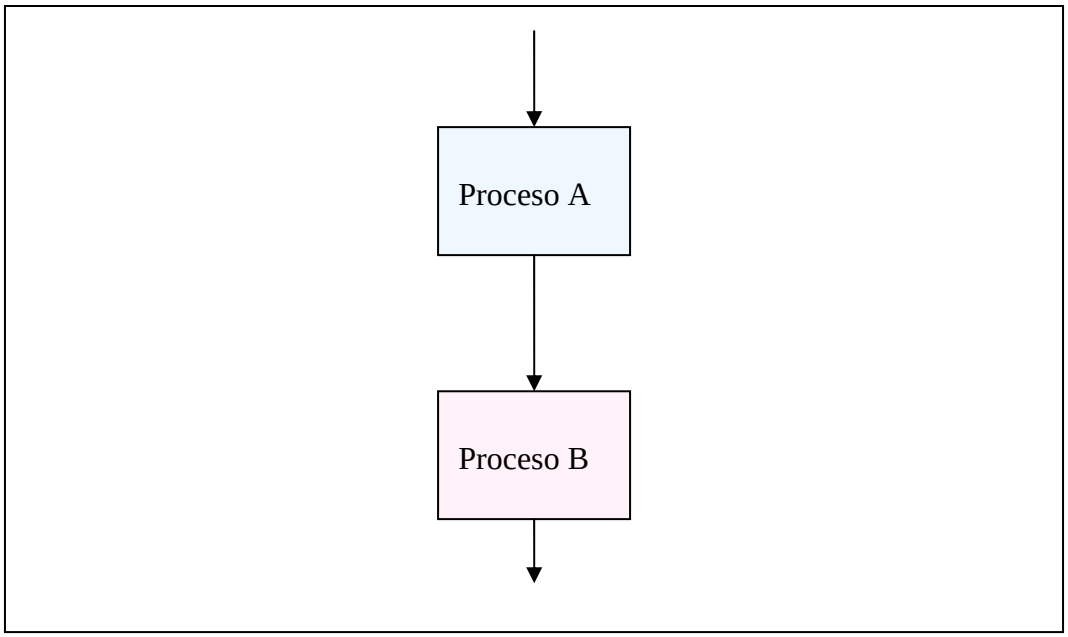

Figura 4.1 Estructura de control secuencial

Para comprender las estructuras es necesario tener una definición muy clara de lo que es un proceso.
Un proceso, por ejemplo, el Proceso A de la figura 4.1 es una o varias instrucciones válidas estas instrucciones pueden estar formadas a sus vez de estructuras y otro tipo de las mismas instrucciones, por lo que ésta su definición es una definición recursiva (que se utiliza a sí misma para definirse). Es como un empaque que dentro de él puede contener otro u otros empaques y así sucesivamente.

4.3.- La estructura de selección “if”.

La estructura de selección ejecuta solo una de dos opciones dependiendo del resultado de una comparación o el valor de una variable tipo booleana. Esto es similar a hacer una pregunta que solo tiene como posibles respuestas si o no y dependiendo de la respuesta ejecutar el proceso A o el proceso B. Podemos observar la figura 4.2 que nos muestra el diagrama de la estructura de control de selección.

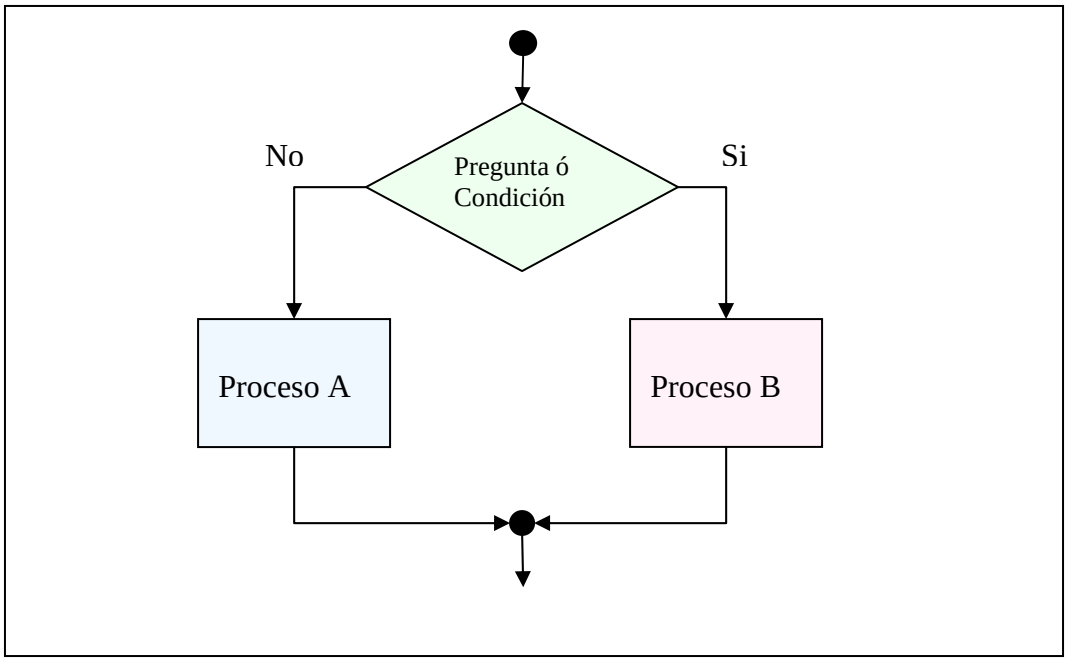

Figura 4.2 Estructura de control de selección.

En esta estructura solo se ejecuta una de las opciones, depende de la respuesta a la condición o pregunta que se plantea.
La sintaxis en Java para esta estructura es la siguiente:

```
If (<nombreVariableTipoboolean>)
     { //proceso B }
else
     { //proceso A}
```

O bien

```
If (<expresiónTipoboolean>)
     { //proceso B }
else
     { //proceso A}
```

Por ejemplo:
El programa que se muestra a continuación:

```
int a=0;
int b=0;
a=Ip.wlee("Valor de a ",a);
b=Ip.wlee("Valor de b ",b);
 if (a>b)
   {Ip.wescribeMensaje("a cuyo valor es "+ a +
                       " es mayor que b cuyo valor es " + b);
```

```
   }
 else
 {Ip.wescribeMensaje("a cuyo valor es "+ a +
                     " es menor o igual que b cuyo valor es " + b);
  }
```

Envía el mensaje que corresponde a los valores leídos para las variables a y b. Ver el ejemplo adjunto 400EstructuraDeSeleccion.java.

4.4.- La estructura iterativa "while".

En esta estructura se ejecuta la comparación o se examina el valor de la variable booleana según sea el caso, si es verdad se ejecuta el proceso A, al terminar el proceso A vuelve a ejecutarse la comparación o a examinar el valor de la variable booleana, si la respuesta es afirmativa vuelve a repetirse el proceso A y así sucede tantas veces como la respuesta sea afirmativa, hasta que la respuesta sea negativa se rompe el ciclo. Normalmente dentro del proceso A se llevan a cabo acciones que modifican las condiciones de la comparación o el valor de la variable booleana.

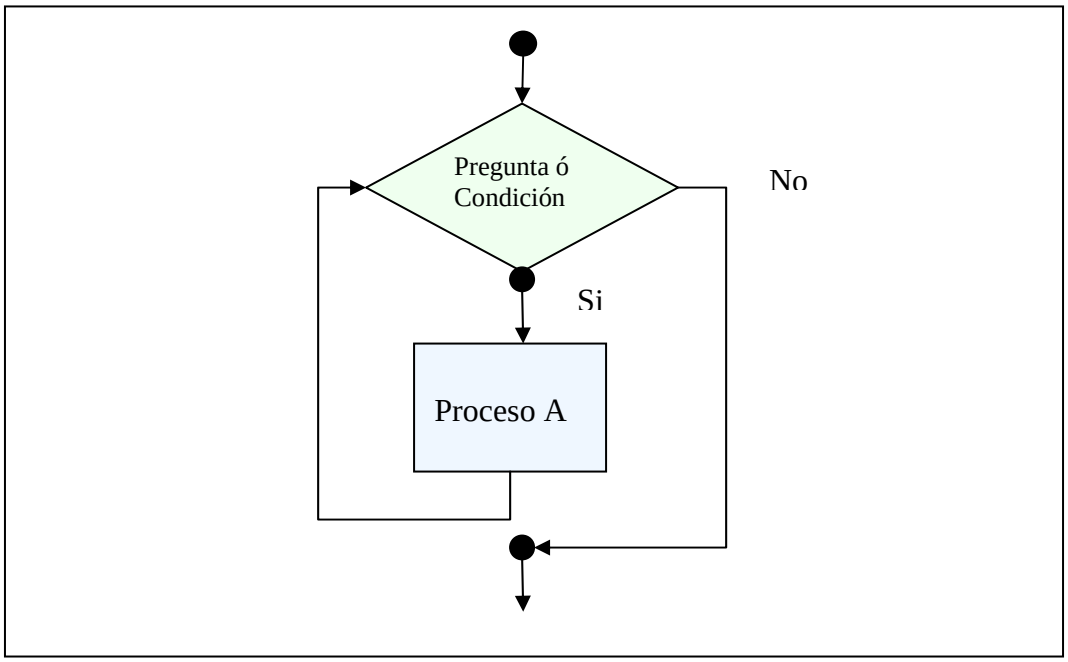

Figura 4.3 Estructura de iteración o cíclica.

La sintaxis en Java para esta estructura es la siguiente:

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

O bien

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

Por ejemplo las instrucciones siguientes cuentan del 1 al 5:

```
int contador=1;
while (contador <= 5)
   {Ip.escribeMensaje("Contador = "+ contador);
    contador = contador + 1;
   }
```

Observe que la instrucción **`contador = contador + 1;`** modifica las condiciones de la
comparación **`contador <= 5`** al alterar el valor del contador sumándole uno. Java proporciona facilidades de abrevitura de instrucciones de tal manera que la instrucción **`contador = contador + 1;`** puede abreviarse como **`contador ++;`**
Vea y modifique el ejemplo adjunto 401EstructuraDeIteracion.java adjunto para comprobarlo.

4.5.- La anidación de estructuras.

Por definición las estructuras son recursivas, cualquier proceso puede contener dentro de si otra u otras estructuras, el diagrama de la figura 4.4 muestra lo anterior.
En este diagrama podemos ver como el proceso B de la estructura de selección se convierte en una estructura iterativa que a su vez dentro de ella tiene un proceso C. El proceso A de la estructura de selección se ha convertido en una estructura secuencial con tres procesos en ella.
El nivel de anidación de estructuras es infinito lo que quiere decir que cualquiera de los procesos mostrados A, D , E o C pueden contener dentro de si otra u otras estructura de las tres permitidas. Note como las propiedades de las estructuras tal como la propiedad de tener una sola entrada y una sola salida se siguen conservando para las estructuras resultantes.

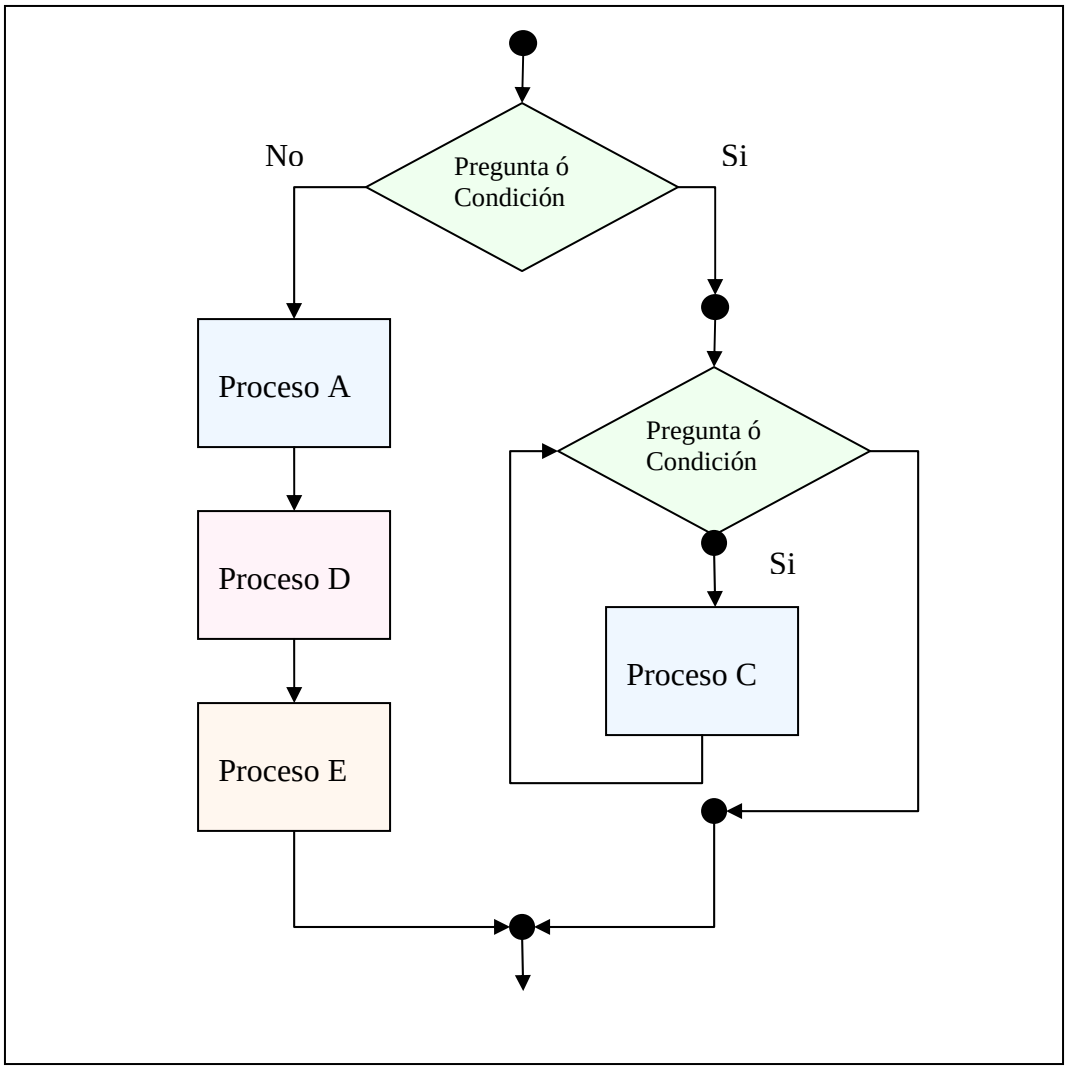

Figura 4.4 Estructuras anidadas,
ilustra la propiedad de recursividad de las estructuras de Dijkstra.

4.6.- Ejemplo integrador.

Para ilustrar el uso de las tres estructuras de Dijkstra utilizaremos el ejemplo que se muestra a continuación, este programa tiene como objetivo capturar y mostrar una lista de artículos. Al final muestra el total sin impuesto IVA, el total de impuesto IVA y el total general. Las estructuras utilizadas son la secuencial y la de iteración.

```
String nombreComercio="El Porvenir SA de CV";
String RFC="POVE1405985R7";
String CalleYNum ="Hidalgo 345 Pte.";
String Col="Col. El Futuro Incierto";
String Mpo="Monterrey, N.L.";
String CP ="65000";
String mensaje="";
String articulo="";
float  precio=0;
```

```
int contador=0;
float acumulador=0;
float porcentajeIVA = 16;
float subTotal = 0;
int limite = 10;
boolean r=true; //variable booleana si se desea agregar otro articulo
mensaje = nombreComercio;
mensaje += "\n"+ RFC;
mensaje += "\n"+ CalleYNum;
mensaje += "\n"+ Col;
mensaje += "\n"+ Mpo;
mensaje += "\n"+ CP;
mensaje += "\n\n\n";
while(r)
    {
     articulo = Ip.wlee("articulo " + (contador+1),articulo);
     precio   = Ip.wlee("precio   " + (contador+1),precio);
     mensaje += "\n" + (contador +1) + ".-" +  articulo+" "+precio;
     contador++;
     acumulador = acumulador + precio;
     r = Ip.wescribePregunta("¿Desea agregar otro articulo?");
  }
subTotal = (float) (acumulador /(1.00 + (porcentajeIVA/100)));
mensaje += "\n\n" + "Sub Total....." +  subTotal;
mensaje += "\n" + "IVA....." +  (acumulador - subTotal);
mensaje += "\n\n" + "Total....." +  acumulador;
Ip.wescribeMensaje(mensaje);
```

El diagrama general de este programa se muestra en la figura 4.5, note la estructura de secuencia al iniciar el programa y la estructura iterativa al centro del programa, seguida por otra estructura de secuencia. El programa utiliza un contador, el cual es una variable que se actualiza cada vez que se ejecuta el proceso que se encuentra dentro del ciclo. El concepto de contador es muy útil y se usa repetidamente en diversas lógicas para la solución de una gran cantidad de problemas. Compile y ejecute el programa adjunto 403ListaArticulos.java para observar como se ejecuta dicho programa.

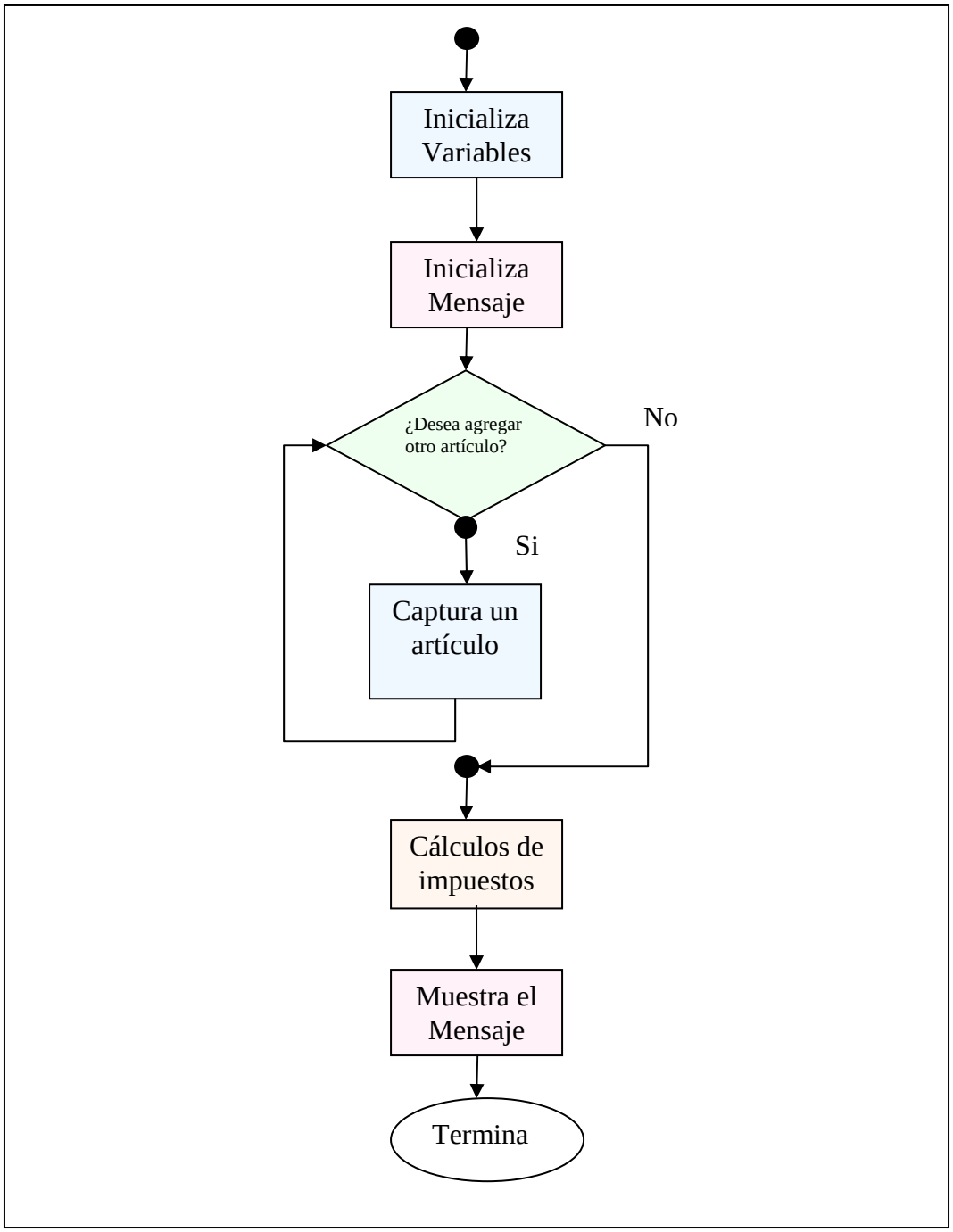

Figura 4.5 Diagrama general del programa que captura y muestra l alista de artículos utilizando estructuras secuenciales e iterativas.

En la siguiente versión del programa se agrega la utilización de la estructura de selección, pues se añade la captura del precio unitario por artículo, así como la determinación del precio unitario más alto y el precio unitario más bajo y el cálculo del promedio de los precios unitarios. Se han agregado los acumuladores necesarios, un acumulador es una variable a la cual se le suma cierta cantidad cada vez que se ejecuta el proceso que se encuentra dentro del ciclo. Al igual que el contador un acumulador es un concepto importante y de uso muy amplio. Esta segunda versión del programa la encuentra en 404ListaArticulosPreciosUnitarios.java y se muestra a continuación.

```
String nombreComercio="El Porvenir SA de CV";
String RFC="POVE1405985R7";
String CalleYNum ="Hidalgo 345 Pte.";
String Col="Col. El Futuro Incierto";
String Mpo="Monterrey, N.L.";
String CP ="65000";
String mensaje="";
String articulo="";
float  precio=0;
float  cantidad=0;
float  precioUnitario=0;
String articuloMayorPrecio="";
float mayorPrecioUnitario=0;
String articuloMenorPrecio="";
float menorPrecioUnitario=999999;
int contador=0;
float acumulador=0;
float acumuladorPreciosUnitarios=0;
float promedioPreciosUnitarios=0;
float porcentajeIVA = 16;
float subTotal = 0;
int limite = 10;
boolean r=false;
mensaje = nombreComercio;
mensaje += "\n"+ RFC;
mensaje += "\n"+ CalleYNum;
mensaje += "\n"+ Col;
mensaje += "\n"+ Mpo;
mensaje += "\n"+ CP;
mensaje += "\n\n\n";
while(!r)
    {
     articulo = Ip.wlee("articulo " + (contador+1),articulo);
     cantidad=Ip.wlee("cantidad   " + (contador+1),cantidad);
     precioUnitario=Ip.wlee("preciounitario"+
     (contador+1),precioUnitario);
     precio = cantidad * precioUnitario;
     mensaje += "\n" + (contador +1) + ".-" +  articulo+
                  " cantidad "+cantidad+
                      " precio unitario " + precioUnitario+
                  " precio total " +precio;
     contador++;
     //Revisar para encontrar el articulo de mayor precio
     if(mayorPrecioUnitario < precioUnitario )
           {
                mayorPrecioUnitario = precioUnitario;
                articuloMayorPrecio =  articulo;
           }
     //Revisar para encontrar el articulo de menor precio
```

```
      if(menorPrecioUnitario > precioUnitario )
            {
                  menorPrecioUnitario = precioUnitario;
                  articuloMenorPrecio =  articulo;
            }
     acumulador = acumulador + precio;
     acumuladorPreciosUnitarios=acumuladorPreciosUnitarios+
     precioUnitario;
      r = Ip.wescribePregunta("Deseas Terminar?");
   }
subTotal = (float) (acumulador /(1.00 + (porcentajeIVA/100)));
promedioPreciosUnitarios  = acumuladorPreciosUnitarios /contador;
mensaje += "\n\n" + "Sub Total....." +  subTotal;
mensaje += "\n" + "IVA....." +  (acumulador - subTotal);
mensaje += "\n\n" + "Total....." +  acumulador;
mensaje += "\n\n\n\nArticulo de mayor precio " + articuloMayorPrecio+
             " precio unitario " + mayorPrecioUnitario;
mensaje += "\nArticulo de menor precio " +  articuloMenorPrecio+
             " precio unitario " + menorPrecioUnitario;
mensaje += "\nPromedio de precios Unitarios " +
promedioPreciosUnitarios;
Ip.wescribeMensaje(mensaje);
```

En la última versión del programa se permite la captura de varias listas de artículos, simulando que se atiende a varios clientes y que cada cliente lleva una lista distinta.
Para ello se ha agregado un ciclo externo que permite volver a iniciar una lista nueva una vez que se ha concluido la anterior. Se utiliza una variable booleana adicional para preguntar si hay más clientes (más listas). El código lo puede encontrar en el programa 405ListaArticulosVariosClientes.java.

```
String nombreComercio="El Porvenir SA de CV";
String RFC="POVE1405985R7";
String CalleYNum ="Hidalgo 345 Pte.";
String Col="Col. El Futuro Incierto";
String Mpo="Monterrey, N.L.";
String CP ="65000";
String mensaje="";
String articulo="";
float  precio=0;
float  cantidad=0;
float  precioUnitario=0;
String articuloMayorPrecio="";
float mayorPrecioUnitario=0;
String articuloMenorPrecio="";
float menorPrecioUnitario=999999;
int contador=0;
float acumulador=0;
float acumuladorPreciosUnitarios=0;
float promedioPreciosUnitarios=0;
```

```
float porcentajeIVA = 16;
float subTotal = 0;
int limite = 10;
boolean hayMasArticulos=true; //hay mas articulos
boolean hayMasClientes=true; //hay mas clientes
while (hayMasClientes)
    {//Inicializar las variables que cambian para cada cliente
      articuloMenorPrecio="";
      articuloMayorPrecio="";
      mayorPrecioUnitario=0;
      menorPrecioUnitario=999999;
      contador=0;
      acumulador=0;
      acumuladorPreciosUnitarios=0;
      subTotal = 0;
      mensaje = nombreComercio;
      mensaje += "\n"+ RFC;
      mensaje += "\n"+ CalleYNum;
      mensaje += "\n"+ Col;
      mensaje += "\n"+ Mpo;
      mensaje += "\n"+ CP;
      mensaje += "\n\n\n";
      hayMasArticulos=true;
      while(hayMasArticulos)
          {
           articulo = Ip.wlee("articulo " + (contador+1),articulo);
           cantidad=Ip.wlee("cantidad   " + (contador+1),cantidad);
           precioUnitario=Ip.wlee("precio unitario   " +
                (contador+1),precioUnitario);
           precio = cantidad * precioUnitario;
           mensaje += "\n" + (contador +1) + ".-" +  articulo+
                       " cantidad "+cantidad+
                           " precio unitario " + precioUnitario+
                       " precio total " +precio;
           contador++;
           //Revisar para encontrar el articulo de mayor precio
           if(mayorPrecioUnitario < precioUnitario )
                 {
                      mayorPrecioUnitario = precioUnitario;
                      articuloMayorPrecio =  articulo;
                 }
           //Revisar para encontrar el articulo de menor precio
           if(menorPrecioUnitario > precioUnitario )
                 {
                      menorPrecioUnitario = precioUnitario;
                      articuloMenorPrecio =  articulo;
                 }
          acumulador = acumulador + precio;
          acumuladorPreciosUnitarios=acumuladorPreciosUnitarios+
               precioUnitario;
          hayMasArticulos = Ip.wescribePregunta(
          "hay mas articulos?");
        }
      subTotal = (float) (acumulador /(1.00 + (porcentajeIVA/100)));
      promedioPreciosUnitarios  = acumuladorPreciosUnitarios
          /contador;
      mensaje += "\n\n" + "Sub Total....." +  subTotal;
      mensaje += "\n" + "IVA....." +  (acumulador - subTotal);
      mensaje += "\n\n" + "Total....." +  acumulador;
```

```
                mensaje += "\n\n\n\nArticulo de mayor precio " +
            articuloMayorPrecio+ " precio unitario " +
            mayorPrecioUnitario;
          mensaje += "\nArticulo de menor precio " + articuloMenorPrecio+
                      " precio unitario " + menorPrecioUnitario;
          mensaje += "\nPromedio de precios Unitarios " +
            promedioPreciosUnitarios;
          Ip.wescribeMensaje(mensaje);
          hayMasClientes = Ip.wescribePregunta("hay mas clientes?");
         }//Termina de procesar un cliente
        }//Termina de procesar el programa
```

El ejemplo 406RaicesCuadratica.java muestra otra lógica, en la cual un lazo contiene una estructura de selección simple. Otro ejemplo con estructuras anidadas simples se puede observar en el programa 407Pitagoras.java, que aplica el teorema de Pitágoras para el cálculo de los lados de un triángulo rectángulo.

4.7.- Actividades:
Compile y ejecute los ejemplos adjuntos.

4.8.- 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.