



UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA – UNAD

ESCUELA DE CIENCIAS BASICAS TECNOLOGIA E INGENIERIA

GUIA COMPONENTE PRÁCTICO DEL CURSO: 301403 – PROGRAMACION ORIENTADA A OBJETOS

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA

ESCUELA DE CIENCIAS BASICAS TECNOLOGIA E INGENIERIA

## **GUÍA COMPONENTE PRÁCTICO II**

**301403 – PROGRAMACION ORIENTADA A OBJETOS**

CESAR ORLANDO JIMENEZ ANGARITA  
(Director Nacional)

Zona: Centro Bogotá Cundinamarca,

Cead: José Acevedo y Gómez

Enero 2013

## 2. ASPECTOS DE PROPIEDAD INTELECTUAL Y VERSIONAMIENTO

Curso Académico Módulo Programación Orientada a Objetos

Libros de la Biblioteca Virtual de Unad en Programación Orientada a Objetos

Consulta de Internet Programación Orientada a Objetos

Autor: Deitel y Deitel Introducción DOO con UML y los Patrones de Diseño JDBC tm, SERVLETS, JSP tm Editorial Pearson Prentice Hall Quinta Edición.

David Arnow Gerald Weiss Introducción a la Programación con Java tm Actualización a la Versión 2 de Java Editorial Addison Wesley.

Fcd Javier Ceballos Java 2 Curso de Programación 2 Edición Editorial Alfaomega Ra-Ma.

Agustin Froute Java 2 Manual Usuario tutorial 3 Edición Editorial Alfaomega Ra-Ma 5.

Herbert shildt Fundamentos de Programación Java 2 Editorial Mc Graw Hill . para conocer el lenguaje de programación JAVA.

Arnow, D., Weiss, G., Introducción a la programación con JAVA, Addison wesley, 2000.

Larman, C., UML y patrones, Prentice Hall, 1998.

Meyer, B., Construcción de software orientado a objetos, Prentice Hall, segunda edición, 1997.

Wu, T., Introducción a la programación orientada a objetos con Java, Mc Graw Hill, 2000.

Joyanes, L., Programación orientada a objetos, Segunda edición, Mc Graw Hill, 1998.

Greze Voss, Introducción Orientada a Objetos, Editorial Mc Graw Hill, 1994.

Joyanes, L., C++ a su alcance un enfoque orientado a objetos, Editorial, Mc Graw Hill, 1994.

Cesar Becerra Santamaría, C++ Una Herramienta para la Programación Orientada a Objetos, Editorial Mc Graw Hill, 1.993.

Fco Javier Ceballos, Programación Orientada a Objetos con C++, Editorial Alfaomega, 1.998.

Nathan Meyers, Programación JAVA en Linux, Editorial Prentice Hall, 2.000.

[www.lawebdelprogramador.com](http://www.lawebdelprogramador.com)

[www.programacion.com](http://www.programacion.com)

<http://www.cimat.mx/~jlabreu/CursoJava/>

<http://www.mindview.net/Books/TIJ>

<http://java.sun.com/docs/books/tutorial/>

<http://programacion.com/java/tutoriales/>

### 3. INDICE DE CONTENIDO

|  | Pág. |
|--|------|
| <b>1. PRESENTACION</b>   | 1    |
| <b>2. ASPECTOS DE PROPIEDAD INTELECTUAL Y VERSIONAMIENTO</b>   | 2    |
| <b>3. INDICE DE CONTENIDO</b>  | 3    |
| <b>4. LISTADO DE TABLAS</b>  | 4    |
| <b>5. CARACTERÍSTICAS GENERALES</b>  | 5    |
| <b>6. DESCRIPCIÓN DE PRÁCTICAS</b>   | 8    |
| <b>PRACTICA No. 01 – Estructuras no dinámicas Utilizando Java Eclipse</b>  | 8    |
| <b>Ejercicios Propuestos por el tutor:</b>   | 11   |
| <b>Ejercicios a Realizar por el Estudiante:</b>  | 36   |
| <b>PRACTICA No. 02 – Estructuras dinámicas Utilizando Java Eclipse</b>   | 37   |
| <b>Ejercicios Propuestos por el tutor:</b>   | 40   |
| <b>Ejercicios a Realizar por el Estudiante:</b>  | 73   |
| <b>PRACTICA No. 03 – Base de Datos Modelo Relacional entidad Relación Access Utilizando Java Eclipse.</b>  | 74   |
| <b>Ejercicios Propuestos por el tutor:</b>   | 76   |
| <b>Ejercicios a Realizar por el Estudiante:</b>  | 106  |
| <b>PRACTICA No. 04 – Redes Encriptamiento y Desencriptamiento de un dato, Chat cliente Servidor, computacion Grafica Utilizando Java Eclipse</b> | 109  |
| <b>Ejercicios Propuestos por el tutor:</b>   | 111  |
| <b>Ejercicios a Realizar por el Estudiante:</b>  | 119  |
| <b>7. Fuentes Documentales</b>   | 120  |

## 5. CARACTERÍSTICAS GENERALES

|  |  |
|--|--|
| <p><b>Introducción</b></p>                 | <p>Actualmente una de las áreas más importantes en la industria y el ámbito académico es la orientación a objetos, el curso que a continuación se relaciona tiene el nombre de Programación Orientada a Objetos el cual es teórico – práctico, de 3 créditos y hace parte de la formación profesional en el programa de Ingeniería de Sistemas.</p> <p>El curso está compuesto por tres unidades académicas didácticas:</p> <p>Unidad Uno comprende la introducción y elementos básicos de la programación orientada a objetos (POO) y las propiedades básicas de la POO aplicado con JAVA.</p> <p>Unidad Dos hace referencia al diseño de la estructura y comportamiento de un objeto y se inicia con los Fundamentos de la programación en el lenguaje JAVA.</p> <p>Unidad tres se enfocan en el trabajo con las clases, el manejo de la herencia y la extensión de las clases aplicado con JAVA.</p>  |
| <p><b>Justificación</b></p>                | <p>El uso de y aplicaciones de herramientas enfocadas a la POO, hacen que el profesional en Ingeniería sea capaz de diseñar y desarrollar aplicaciones que permitan dar soluciones integrales a las necesidades del medio que lo rodea utilizando un compilador de JAVA.</p> <p>El curso de POO le permite al estudiante desarrollar su creatividad con el ánimo de dar soluciones a problemas cotidianos de su trabajo, le da la capacidad de comparar la programación estructurada con la programación orientada a objetos y enfrentar los retos del cambio, le permite desarrollar su capacidad de comunicación, y su espíritu Investigativo y crítico con un lenguaje de programación orientado a objetos.</p> <p>Para lograr la apropiación de todo el contenido temático el estudiante aplicará en todo momento su autoformación y se apoyará en el tutor para la resolución de dudas, también se tendrán uso de herramientas tecnológicas para la interacción con los tutores y alumnos dentro de los laboratorios.</p> |
| <p><b>Intencionalidades formativas</b></p> | <p><b>Propósitos:</b></p> <p>Dar a conocer a los estudiantes de una forma clara los conceptos fundamentales de la Programación Orientada a Objetos en JAVA.</p> <p>Dar a Conocer a los estudiantes las ventajas de aplicar la Programación Orientada a Objetos ya que simplifica la construcción de programas y lo motiva para estudiar las características del lenguaje JAVA.</p> <p>Capacitar a los estudiantes a comprender y utilizar herramientas de Programación Orientadas a Objetos como JAVA, para el diseño de aplicaciones.</p>   |

### **Objetivos:**

Que el estudiante Describa los conceptos que caracterizan al modelo Programación Orientada a Objetos Utilizando JAVA.

Que el estudiante Valore en que medida las técnicas Programación Orientada a Objetos favorecen la calidad del software, analizando sobre todo cómo facilitan la reutilización y extensibilidad en JAVA.

Que el estudiante entienda la importancia de trabajar Herramientas Programación Orientadas a Objetos, para el desarrollo de nuevas aplicaciones empresariales en JAVA.

Que le estudiante Aprenda un lenguaje Programación Orientada a Objetos, junto a su entorno de programación.

Que el estudiante aplique técnicas de programación Orientada a Objetos en JAVA.

Que el estudiante identifique analice y diseñe ejemplos de aplicación en JAVA

### **Metas:**

Al terminar este curso de Programación Orientada a Objetos el estudiante:

Desarrollara aplicaciones teniendo en cuenta los fundamentos teóricos, y el manejo de objetos como principal elemento de Construcción de soluciones en JAVA.

Tendrá la capacidad de analizar, diseñar y desarrollar software utilizando el entorno de Programación Orientada a Objetos con JAVA.

Adquiera destrezas en el uso de las herramientas como el JAVA para el desarrollo de aplicaciones que den soluciones confiables y permitan mejorar procesos.

### **Competencias:**

El estudiante comprende e interpreta todos los conceptos fundamentales de la Programación Orientada a Objetos, para poder analiza, diseñar, desarrollar e implantar aplicaciones más robustas y con código reutilizable con JAVA.

El estudiante adquiere destrezas en el uso de procedimientos que le permitan analizar, diseñar y desarrollar aplicaciones haciendo uso de la programación Orientada a Objetos con JAVA.

|                                    |   |
|------------------------------------|---|
|                                    | El estudiante aprende un lenguaje Programación Orientada a Objetos, junto a su entorno de programación JAVA.  |
| <b>Denominación de practicas</b>   | <p>Práctica 01: Estructuras no dinámicas Utilizando Java Eclipse</p> <p>Practica 02: Estructuras dinámicas Utilizando Java Eclipse</p> <p>Practica 03: Base de Datos Modelo Relacional entidad Relación Access Utilizando Java Eclipse</p> <p>Practica 04: Redes Encriptamiento y descriptamiento de un dato, Chat cliente Servidor Utilizando Java Eclipse</p> <p>Practica 05: Computación Grafica 2D y 3D Utilizando Jcreator Java Eclipse</p>  |
| <b>Número de horas</b>             | 4 Horas por laboratorio, 4 Cinco Laboratorios, un total 16 Horas  |
| <b>Porcentaje</b>                  | 25 Puntos de la plataforma Campus Virtual   |
| <b>Curso Evaluado por proyecto</b> | SI____ NO_X_  |
| <b>Seguridad industrial</b>        | <p>No Ingresar Alimentos líquidos o sólidos al Laboratorio. Para que se presente algún corto circuito.</p> <p>No ingresar fumando al Laboratorio. Para que moleste a los compañeros.</p> <p>No ingresar otros aparatos electrónicos como radios, Walkman, para que no interfieran en el desarrollo del laboratorio.</p> <p>No Cambiar los computadores de sitio sin autorización del monitor del laboratorio.</p> <p>No reconfigurar los equipos de computo ya que este presta servicio a toda la comunidad unadista.</p> |

## PRACTICA No. 02 – Estructuras dinámicas Utilizando Java Eclipse

|                              |  |              |  |              |   |        |   |
|------------------------------|--|--------------|--|--------------|---|--------|---|
| Tipo de practica             | <table><tr><td>Presencial</td><td></td><td>Autodirigida</td><td></td><td>Remota</td><td>X</td></tr></table>  | Presencial   |  | Autodirigida |   | Remota | X |
| Presencial                   |  | Autodirigida |  | Remota       | X |        |   |
| Porcentaje de evaluación     | 25 Puntos Plataforma campus virtual  |              |  |              |   |        |   |
| Horas de la practica         | 4 Horas Laboratorio ST 16 Horas Independientes   |              |  |              |   |        |   |
| Temáticas de la práctica     | Estructuras dinámicas Java<br>Utilizando Java Eclipse  |              |  |              |   |        |   |
| Intencionalidades formativas | <p><b>Propósito(s):</b></p> <p>Desarrollar destreza en los estudiantes en estructuras dinámicas que permitan conceptuar los fundamentos básicos en el desarrollo programas informáticos mediante la utilización de un lenguaje de programación Java.</p> <p>Comprender las técnicas básicas del diseño de estructuras dinámicas de un algoritmos y la lógica de programación en Java.</p> <p>Adquirir destrezas y habilidades en la toma decisiones, en estructuras dinámicas y mediante la mediación Tutorial, lecturas complementarias, trabajo individual y cooperativo para el desarrollo de algoritmos en Java.</p> <p>Codificar de estructuras dinámicas en un lenguaje de programación diferentes supuestos semánticos y problemas cotidianos en JAVA.</p> <p><b>Meta(s):</b></p> <p>El estudiante desarrolla el proyecto y los diferentes talleres en forma adecuada, utilizando las diferentes técnicas y estrategias estudiadas a lo largo del curso de Java.</p> <p>El estudiante está en capacidad de desarrollar “software”, partiendo de los diferentes supuestos planteados, y quedando motivado para seguir en la línea de programación de computadoras con lenguaje de programación JAVA.</p> <p>El estudiante desarrolla habilidades que lo lleven por el camino del pensamiento crítico, permitiéndoles <i>aprender, comprender, practicar y aplicar</i> nueva información, que parte de experiencias del medio ambiente, hasta llegar al auto concepto, formando un individuo critico y pensador.</p> <p><b>Competencia(s):</b></p> <p>El estudiante describe y analiza de manera adecuada y</p> |              |  |              |   |        |   |

sistematizada las técnicas y pautas para la construcción y secuencia de un algoritmo codificado en JAVA.

El estudiante diseña y aplica algoritmos y diagramas de flujo en la construcción de programas, como técnica principal en la resolución de un determinado problema en JAVA.

El estudiante mediante lecturas y mediaciones sobre técnicas de construcción de software analiza, sistematiza y pone en práctica las buenas prácticas de desarrollo de algoritmos y de programación en lenguaje JAVA.

### Fundamentación Teórica

Estructuras dinámicas Listas mediante el lenguaje de programación en JAVA.  
Estructuras dinámicas Colas mediante el lenguaje de programación en JAVA.  
Estructuras dinámicas Pilas mediante el lenguaje de programación en JAVA.  
Estructuras dinámicas Árboles mediante el lenguaje de programación en JAVA.

### Descripción de la práctica

El estudiante debe instalar una máquina virtual y Java Eclipse en su casa, este software debe ser bajado a través de internet de la página Sum MacroSystem

El estudiante debe bajar la Guía de Laboratorio para la práctica de la Plataforma campus Virtual del curso Académico de Programación Orientada a Objetos.

El estudiante mediante la herramienta de copiar y pegar debe practicar los diferentes ejercicios propuestos en esta guía para que vaya adquiriendo destreza en el lenguaje de programación Java.

El estudiante debe de estar en capacidad de investigar los diferentes códigos de java ya que todos los ejercicios que propone la guía de laboratorio están propuestos java.swing.\*, o ambiente gráfico y el estudiante debe tener las condiciones de pasarlo en java.io.\*, o y applet ambiente en código de línea todos los propuestos en estas.

### Recursos a utilizar en la práctica (Equipos / instrumentos)

Computadores Modernos con buena Configuración y con software licenciado

### Software a utilizar en la práctica

Instalación de la máquina virtual e Instancias del Java Eclipse

### Metodología

Conocimiento previo para el desarrollo de la práctica. El estudiante de ver el Curso Académico de Programación Orientada a Objetos tiene que tener buenas bases de Algoritmos, Introducción a Programación, Estructura de Datos, Bases de Datos, Redes de Comunicación,



Forma de trabajo: El estudiante analizara primero todo los programas propuestos por el tutor de laboratorio los cuales están analizados, diseñados y desarrollados completamente listos para que su código sea copiado y pegado en JAVA y luego ser ejecutados, luego el estudiante de haber analizado el código se enfrentara a la tarea de desarrollar una serie de programas por sí solo.

Procedimiento: El Estudiante debe investigar como puede hacer estos mismos programas utilizando las librerías import java.io.\* y applet

### Sistema de Evaluación

El sistema de evaluación estará a cargo de los tutores de Laboratorio ya que ellos son autónomos de calificar bien sea en grupos o de forma individual el trabajo realizado en los laboratorios

### Informe o productos a entregar

Los estudiantes debe entregar los ejercicios propuestos en la guía de laboratorio en java.io.\*; y applet

### Rúbrica de evaluación

**UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA "UNAD"**

**ESCUELA CIENCIAS BASICAS TECNOLOGIA E INGENIERIA**

**PROGRAMA INGENIERIA DE SISTEMAS**

### Practica 02 Estructuras dinámicas con Java Utilizando Jcreator Le

### Rúbrica de Evaluación:

| Ítem Evaluado                                     | Valoración Baja           | Valoración Media              | Valoración Alta           | Máximo Puntaje |
|---|---------------------------|-------------------------------|---------------------------|----------------|
| Estructura del Programa codificado en Jcreator LE | Programas sin formato (0) | Programas sin ejecutables (4) | Programas ejecutables (8) | 8              |

|   |  |                                      |                                     |    |
|---|--|--------------------------------------|-------------------------------------|----|
| Redacción y Ortografía documentación del programa | Programas No documentado (0)           | Programas Medio documentado (4)      | Programas Bien documentado (8)      | 8  |
| Fines del programa                                | Programa no cumple con lo sugerido (0) | Programa regular con lo sugerido (4) | Programa Cumple con lo sugerido (9) | 9  |
| TOTAL DE PUNTOS POSIBLES                          |  |                                      |                                     | 25 |

**Nota:** Estudiante que no se presente al componente practico en las fechas propuestas por los diferentes Cead o Ceres y que no desarrollen las guías propuestas para estos laboratorios, tendrá una nota total de CERO: (0)

### Retroalimentación

La retroalimentación de los diferentes productos entregados en el laboratorio serán evaluados por el tutor de Plataforam Campus Virtual

## Ejercicios Propuestos por el tutor:

### Guía de Laboratorio Estructuras dinámicas utilizando el lenguaje de programación Java trabajos con java.swing.\*; o ambiente gráfico:

#### 1. Ejemplo Cola1:

```
import javax.swing.*;
import java.io.*;
import java.lang.Math.*;
import java.util.*;
public class Cola1
{
    static double SALDO=0;
    public static void main(String[] args)
    {
        int i,opc,nit;
        Cola A=new Cola(10);
        Cola B=new Cola(20);
        Cliente C=new Cliente();
        Object [] valores = {"1. Adicion","2. Borrar","3. Consulta Por Atender","4. Consulta Atendidos","5.Salir"};
        do
        {
            String resp=(String) JOptionPane.showInputDialog(null,"Elija la Opcion", "Entrada de
datos",JOptionPane.QUESTION_MESSAGE, null, valores,valores[0]);
            opc=Character.digit(resp.charAt(0),10);
            switch(opc)
            {
                case 1:
                    C.nit=Integer.parseInt(JOptionPane.showInputDialog(null,"Escriba el número de Nit"));
                    if(A.existe(C.nit) || B.existe(C.nit))
                        JOptionPane.showMessageDialog(null,"Existe Nit");
                    else
                        if(A.llena())
                            JOptionPane.showMessageDialog(null,"No se Puede Atender Mas");
                        else
                        {
                            C.nomCliente=JOptionPane.showInputDialog(null,"Digite Nombre");
                            C.telefono=Integer.parseInt(JOptionPane.showInputDialog(null,"Digite Telefono"));
                            A.adicion(C);
                        }
                    break;
                case 2:
                    if(A.vacia())
                        JOptionPane.showMessageDialog(null,"Cola de Atencion Vacía");
                    else
                    {
                        A.borra(C);
                        if(!B.llena())
                            B.adicion(C);
                    }
                    break;
                case 3:
                    if(A.vacia())
                        JOptionPane.showMessageDialog(null,"Cola de Atencion Vacía");
                    else
                        JOptionPane.showMessageDialog(null,A.imprime(" Atencion "));
                    break;
                case 4:
```

```

        if(B.vacia())
            JOptionPane.showMessageDialog(null,"Cola de Atendidos Vacía");
        else
            JOptionPane.showMessageDialog(null,B.imprime(" Atencion "));
        }
        }
        while(opc!=5);
    }
}
class Cliente
{
    int nit;
    String nomCliente;
    int telefono;
}
class Cola
{
    int min,max,n;
    Cliente A[];
    int i;
    public Cola(int n)
    {
        min=-1;
        max=-1;
        this.n=n;
        A=new Cliente[n];
    }
    boolean vacia()
    {
        if(min== -1)
            return true;
        else
            return false;
    }
    boolean llena()
    {
        if(max==n-1)
            return true;
        else
            return false;
    }
    void adicion(Cliente C)
    {
        if(min== -1)
            min++;
            max++;
            A[max]=new Cliente();
            A[max].nit=C.nit;
            A[max].telefono=C.telefono;
            A[max].nomCliente=C.nomCliente;
    }
    void borra(Cliente C)
    {
        C.nit=A[min].nit;
        C.telefono=A[min].telefono;
        C.nomCliente=A[min].nomCliente;
        if(min==max)
            min=max=-1;
        else
            min++;
    }
}

```

```
}  
boolean existe(int nit)  
{  
    boolean esta=false;  
    if(!vacía())  
        for(i=min;i<=max && ! esta;i++)  
            if(A[i].nit==nit)  
                esta=true;  
    return esta;  
}  
String imprime(String Aviso)  
{  
    String S=" Elementos de la Cola de: "+Aviso.toString()+"\n";  
    for(i=min;i<=max;i++)  
    {  
        S=S+" "+new String().valueOf(A[i].nit).toString();  
        S=S+" "+A[i].nomCliente;  
        S=S+" "+new String().valueOf(A[i].telefono).toString()+"\n";  
    }  
    return S.toString();  
}
```

## 2. Ejemplo Cola2:

```
import javax.swing.*;
import java.io.*;
import java.lang.Math.*;
import java.util.*;
public class Cola2
{
    public static void main(String[] args)
    {
        int i,opc,nit;
        BiCola A=new BiCola(10);
        Info C=new Info();
        Object [] valores = {"1. Adicion Mas Reciente","2. Adicion Menos Reciente","3. Borrar Menos
Reciente","4. Borrar Mas Recientes","5. Consulta","6.Salir"};
        do
        {
            String resp=(String) JOptionPane.showInputDialog(null,"Elija la Opcion", "Entrada de
datos",JOptionPane.QUESTION_MESSAGE, null, valores,valores[0]);
            opc=Character.digit(resp.charAt(0),10);
            switch(opc)
            {
                case 1:
                    if(A.llena())
                        JOptionPane.showMessageDialog(null,"No se Puede Atender Mas");
                    else
                    {
                        C.nit=Integer.parseInt(JOptionPane.showInputDialog(null,"Escriba el número de Nit"));
                        C.nomCliente=JOptionPane.showInputDialog(null,"Digite Nombre");
                        C.telefono=Integer.parseInt(JOptionPane.showInputDialog(null,"Digite Telefono"));
                        A.adicion_max(C);
                    }
                    break;
                case 2:
                    if(A.llena())
                        JOptionPane.showMessageDialog(null,"No se Puede Atender Mas");
                    else
                        if(!A.tope() && !A.minimo())
                        {
                            C.nit=Integer.parseInt(JOptionPane.showInputDialog(null,"Escriba el número de Nit"));
                            C.nomCliente=JOptionPane.showInputDialog(null,"Digite Nombre");
                            C.telefono=Integer.parseInt(JOptionPane.showInputDialog(null,"Digite Telefono"));
                            A.adicion_min(C);
                        }
                    break;
                case 3:
                    if(A.vacia())
                        JOptionPane.showMessageDialog(null,"Cola de Atencion Vacía");
                    else
                    {
                        A.borra_min();
                    }
                    break;
                case 4:
                    if(A.vacia())
                        JOptionPane.showMessageDialog(null,"Cola de Atencion Vacía");
                    else
                    {
                        A.borra_max();
                    }
            }
        }
    }
}
```

```

        break;
    case 5:
        if(A.vacia())
            JOptionPane.showMessageDialog(null,"BiCola Vacía");
        else
            JOptionPane.showMessageDialog(null,A.imprime());
        }
    }
    while(opc!=6);
}

class Info
{
    int nit;
    String nomCliente;
    int telefono;
}

class BiCola
{
    int min,max,n;
    Info A[];
    int i;
    public BiCola(int n)
    {
        min=-1;
        max=-1;
        this.n=n;
        A=new Info[n];
        for(i=0;i<n;i++)
            A[i]=new Info();
    }
    boolean vacia()
    {
        if(min==-1)
            return true;
        else
            return false;
    }
    boolean llena()
    {
        if(min==0 && max==n-1)
            return true;
        else
            return false;
    }
    boolean tope()
    {
        if(max==n-1)
            return true;
        else
            return false;
    }
    boolean minimo()
    {
        if(min==0)
            return true;
        else
            return false;
    }
}

void adicion_max(Info C)

```

```

{
    if(min== -1)
        min=0;
    max++;
    A[max].nit=C.nit;
        A[max].telefono=C.telefono;
        A[max].nomCliente=C.nomCliente;
    }
void adicion_min(Info C)
{
    min--;
    A[min].nit=C.nit;
        A[min].telefono=C.telefono;
        A[min].nomCliente=C.nomCliente;;
    }
void borra_min()
{
    if(min==max)
        min=max=-1;
    else
        min++;
}
void borra_max()
{
    if(min==max)
        min=max=-1;
    else
        max--;
}
String imprime()
{
    String S=" Elementos de la Cola \n";
    for(i=min;i<=max;i++)
    {
        S=S+" "+new String().valueOf(A[i].nit).toString();
        S=S+" "+A[i].nomCliente;
        S=S+" "+new String().valueOf(A[i].telefono).toString()+"\n";
    }
    return S.toString();
}
}
    
```



### 3. Ejemplo Cola3:

```
import javax.swing.*;
import java.io.*;
import java.lang.Math.*;
import java.util.*;
public class Cola3
{
    public static void main(String[] args)
    {
        int i,opc,nit;
        ColaCir A=new ColaCir(3);
        Info C=new Info();
        Object [] valores = {"1. Adicion","2.Eliminar","3. Consulta","4.Salir"};
        do
        {
            String resp=(String) JOptionPane.showInputDialog(null,"Elija la Opcion", "Entrada de
datos",JOptionPane.QUESTION_MESSAGE, null, valores,valores[0]);
            opc=Character.digit(resp.charAt(0),10);
            switch(opc)
            {
                case 1:
                    if(A.llena())
                        JOptionPane.showMessageDialog(null,"No se Puede Atender Mas");
                    else
                    {
                        C.nit=Integer.parseInt(JOptionPane.showInputDialog(null,"Escriba el número de Nit"));
                        C.nomCliente=JOptionPane.showInputDialog(null,"Digite Nombre");
                        C.telefono=Integer.parseInt(JOptionPane.showInputDialog(null,"Digite Telefono"));
                        A.adicion(C);
                    }
                    break;
                case 2:
                    if(A.vacia())
                        JOptionPane.showMessageDialog(null,"Cola de Atencion Vacía");
                    else
                    {
                        A.borrar();
                    }
                    break;
                case 3:
                    if(A.vacia())
                        JOptionPane.showMessageDialog(null,"Cola Vacía");
                    else
                        JOptionPane.showMessageDialog(null,A.imprime());
                }
                while(opc!=4);
            }
        }
    }
}
class Info
{
    int nit;
    String nomCliente;
    int telefono;
}
class ColaCir
{
    int min,max,n;
    Info A[];
```

```

int i;
String S=" ";
public ColaCir(int n)
{
    min=-1;
    max=-1;
    this.n=n;
    A=new Info[n];
    for(i=0;i<n;i++)
        A[i]=new Info();
}
boolean vacia()
{
    if(min==-1)
        return true;
    else
        return false;
}
boolean llena()
{
    if((min==0 && max==n-1) || (max==min-1))
        return true;
    else
        return false;
}
void adicion(Info C)
{
    if(min==-1)
        min=0;
    if(max==n-1)
        max=0;
    else
        max++;
    A[max].nit=C.nit;
    A[max].telefono=C.telefono;
    A[max].nomCliente=C.nomCliente;;
}
void borrar()
{
    if(min==max)
        min=max=-1;
    else
        if(min==n-1)
            min=0;
        else
            min++;
}
String imprime()
{
    S=" Elementos de la Cola \n";
    if(max<min)
    {
        consulta(min,n-1);
        consulta(0,max);
    }
    else
        consulta(min,max);
    return S.toString();
}
void consulta(int inf,int sup)

```

```
{  
    for(i=inf;i<=sup;i++)  
    {  
        S=S+" "+new String().valueOf(A[i].nit).toString();  
        S=S+" "+A[i].nomCliente;  
        S=S+" "+new String().valueOf(A[i].telefono).toString()+"\n";  
    }  
}
```



## 4. Ejemplo Cola 3 Archivos

```
import javax.swing.*;
import java.io.*;
import java.lang.Math.*;
import java.util.*;
public class ColaArch
{
    static double SALDO=0;
    public static void main(String[] args)
    {
        int i,opc,nit;
        Cola A=new Cola(10);
        if(A.salir())
            System.exit(1);
        A.cargar();
        Object [] valores = {"1. Adicion","2. Borrar","3. Consulta","4.Salir"};
        do
        {
            String resp=(String) JOptionPane.showInputDialog(null,"Elija la Opcion", "Entrada de
datos",JOptionPane.QUESTION_MESSAGE, null, valores,valores[0]);
            opc=Character.digit(resp.charAt(0),10);
            switch(opc)
            {
                case 1:
                    if(A.llena() || !A.hayMas())
                        JOptionPane.showMessageDialog(null,"No se Puede Atender Mas");
                    else
                        A.adicion();
                    break;
                case 2:
                    if(A.vacia())
                        JOptionPane.showMessageDialog(null,"Cola de Atencion Vacia");
                    else
                        A.borrar();
                    break;
                case 3:
                    if(A.vacia())
                        JOptionPane.showMessageDialog(null,"Cola de Atencion Vacia");
                    else
                        JOptionPane.showMessageDialog(null,A.imprime());
                    break;
            }
            while(opc!=4);
            A.cerrar();
            System.exit(1);
        }
    }
}
class Cola
{
    ArchivoBase P=new ArchivoBase();
    int min,max,n,d,numreg=-1;
    int A[];
    int i;
    public Cola(int n)
    {
        min=-1;
        max=-1;
        this.n=n;
    }
}
```

```

        A=new int[n];
    }
    void cargar()
    {
        P.tabla();
    }
    boolean salir()
    {
        if(!P.lectura("cliente.txt"))
            return true;
        cargar();
        if(P.tope()==0)
            return true;
        return false;
    }
    boolean vacia()
    {
        if(min== -1)
            return true;
        else
            return false;
    }
    boolean llena()
    {
        if(max==n)
            return true;
        else
            return false;
    }
    boolean hayMas()
    {
        if(numreg<P.tope())
            return true;
        else
            return false;
    }
    void adicion()
    {
        if(min== -1)
            min++;
        max++;
        numreg++;
        A[max]=numreg;
    }
    void borrar()
    {
        if(min==max)
            min=max=-1;
        else
            min++;
    }
    String imprime()
    {
        String S=" Elementos de la Cola de: \n";
        for(i=min;i<=max;i++)
        {
            S=S+P.consulta(A[i])+"\n";
        }
        return S.toString();
    }

```

```

        String Consulta()
        {
            return P.Consultabla();
        }
        void cerrar()
        {
            P.cerrar();
        }
    }
class ArchivoBase
{
    long longreg=34;
    StringBuffer linea=new StringBuffer();
    RandomAccessFile arch;
    long pos=-1; //Numero de Registro
    long k;
    int numreg=-1;
    Hashtable H=new Hashtable();
    boolean lectura(String Nom)
    {
        try
        {
            arch=new RandomAccessFile(Nom.toString(),"r");
        }
        catch(FileNotFoundException e)
        {
            System.out.println("No Existe Archivo");
            return false;
        }
        return true;
    }
    void tabla()
    {
        pos=0;
        numreg=-1;
        try
        {
            do
            {
                numreg++;
                H.put(new String().valueOf(numreg),new String().valueOf(pos));
                pos=pos+(longreg*2);
            }
            while(pos<arch.length());
        }
        catch(IOException e)
        {
        }
    }
    void bajar()
    {
        try
        {
            arch.seek(pos);
            for(k=pos;k<pos+longreg;k++)
            {
                if(k==7 || k==27)
                    linea.append(" ");
                linea.append(arch.readChar());
            }
        }
    }
}

```

```

    }
    catch(IOException e)
    {
    }
}
String consulta(int n)
{
    String S=new String().valueOf(n);
    linea=new StringBuffer();
    pos=Long.parseLong((String) H.get(S));
    bajar();
    return linea.toString();
}
long longitud()
{
    long k=0;
    try
    {
        k=arch.length();
    }
    catch(IOException e){}
    return k;
}
int tope()
{
    return numreg;
}
String Consultabla()
{
    Enumeration codigo=H.keys();
    String S=null;
    linea=new StringBuffer();
    linea.append("Codigo      Nombre      Telefono \n");
    while(codigo.hasMoreElements())
    {
        S=(String) codigo.nextElement();
        pos=Long.parseLong((String) H.get(S))*longreg;
        bajar();
        linea.append("\n");
    }
    return linea.toString();
}
void cerrar()
{
    try
    {
        arch.close();
    }
    catch(IOException e){}
}
}

```

## 5. Ejemplo de Lista

```
import javax.swing.*.*;
class Nodo {
    int info;
    Nodo(int info) {
        this.info = info;
    }
    Nodo sig;
}
class Listas
{
    Nodo p,q,r;
    Nodo cab=null;
    boolean busca_ant(int x)
    {
        p=q=cab;
        while(p!=null && p.info!=x)
        {
            q=p;
            p=p.sig;
        }
        if(p!=null)
            return true;
        else
            return false;
    }
    void recorre_ant()
    {
        p=q=cab;
        while(p!=null)
        {
            q=p;
            p=p.sig;
        }
    }
    void adicion(int x)
    {
        if(!busca_ant(x))
        {
            r=new Nodo(x);
            if(cab==null)
                cab=r;
            else
                q.sig=r;
            r.sig=null;
        }
    }
    void borrar(int x)
    {
        if(busca_ant(x))
        {
            if(p==cab)
                cab=cab.sig;
            else
                q.sig=p.sig;
        }
    }
    String consulta()
    {

```



```
String S="CONSULTA\n";
p=cab;
while(p!=null)
{

    S+=p.info+"\n";
    p=p.sig;
}
return S.toString();
}
}
public class Lista
{

    public static void main(String[]args)
    {

        int i,opc,info;
        Listas L=new Listas();
        Object [] valores = {"1. Adicion","2. Borrar","3. Consulta","4.Salir"};
        do
        {
            String resp=(String) JOptionPane.showInputDialog(null,"Elija la Opcion", "Entrada de
datos",JOptionPane.QUESTION_MESSAGE, null, valores,valores[0]);
            opc=Character.digit(resp.charAt(0),10);
            switch(opc)
            {
                case 1:
                    info=Integer.parseInt(JOptionPane.showInputDialog(null," Digite Info a Adicionar:"));
                    L.adicion(info);

                    break;
                case 2:
                    info=Integer.parseInt(JOptionPane.showInputDialog(null," Digite Info a Borrar:"));
                    L.borrar(info);

                    break;
                case 3:
                    JOptionPane.showMessageDialog(null,L.consulta());
                    break;
            }
        }

        while(opc!=4);
        System.exit(1);
    }
}
```

## 6. Ejemplo de Lista Circular

```
import javax.swing.*;

class Nodo {
    int info;
    Nodo(int info) {
        this.info = info;
    }
    Nodo sig;
}

class Listas
{
    Nodo p,q,r;
    Nodo cab=null;
    Nodo ult=null;
    boolean vacia()
    {
        if(cab==null)
            return true;
        else
            return false;
    }
    void adicion(int x)
    {
        r=new Nodo(x);
        if(cab==null)
            cab=r;
        else
            ult.sig=r;
        ult=r;
        r.sig=cab;
    }
    void borrar()
    {
        if(cab==ult)
            cab=ult=null;
        else
        {
            cab=cab.sig;
            ult.sig=cab;
        }
    }
    String consulta()
    {
        String S="C O N S U L T A \n";
        p=cab;
        do
        {
            S+=p.info+"\n";
            p=p.sig;
        }
        while(p!=cab);
        return S.toString();
    }
}

public class ListaCircular
{
    public static void main(String[] args)
    {
        int i,opc,info;
```

```

        Listas L=new Listas();
        Object [] valores = {"1. Adicion","2. Borrar","3. Consulta","4.Salir"};
        do
        {
            String resp=(String) JOptionPane.showInputDialog(null,"Elija la Opcion", "Entrada de
datos",JOptionPane.QUESTION_MESSAGE, null, valores,valores[0]);
            opc=Character.digit(resp.charAt(0),10);
            switch(opc)
            {
                case 1:
                    info=Integer.parseInt(JOptionPane.showInputDialog(null," Digite Info a Adicionar:"));
                    L.adicion(info);
                    break;
                case 2:
                    if(!L.vacia())
                        L.borrar();
                    break;
                case 3:
                    if(!L.vacia())
                        JOptionPane.showMessageDialog(null,L.consulta());
                    break;
            }
        }
        while(opc!=4);
        System.exit(1);
    }
}

```

## 7. Ejemplo de Lista y Cola

```
import javax.swing.*;

class Alumno {
    int Cod;
    String Nombre;
    Alumno(int Cod,String Nombre) {
        this.Cod = Cod;
        this.Nombre = Nombre;
    }
    Alumno sig;
}

class Programa
{
    int Id;
    String Nombre;
    Programa sig;
    Alumno p,q,r;
    Alumno cab=null;
    Alumno ult=null;
    Programa(int Id,String Nombre) {
        this.Id = Id;
        this.Nombre = Nombre;
    }
    boolean vacia()
    {
        if(cab==null)
            return true;
        else
            return false;
    }
    void adicion(int x,String y)
    {
        r=new Alumno(x,y);
        if(cab==null)
            cab=r;
        else
            ult.sig=r;
        ult=r;
        r.sig=null;
    }
    void borrar()
    {
        if(cab==ult)
            cab=ult=null;
        else
            cab=cab.sig;
    }
    String consulta()
    {
        String S="CONSULTA \n";
        p=cab;
        while(p!=null)
        {
            S+=p.info+"\n";
            p=p.sig;
        }
        return S.toString();
    }
}
```

```

}
public class ListaCola
{
    public static void main(String[] args)
    {
        int i,opc,info;
        Listas L=new Listas();
        Object [] valores = {"1. Adicion","2. Borrar","3. Consulta","4.Salir"};
        do
        {
            String resp=(String) JOptionPane.showInputDialog(null,"Elija la Opcion", "Entrada de
datos",JOptionPane.QUESTION_MESSAGE, null, valores,valores[0]);
            opc=Character.digit(resp.charAt(0),10);
            switch(opc)
            {
                case 1:
                    info=Integer.parseInt(JOptionPane.showInputDialog(null," Digite Info a Adicionar:"));
                    L.adicion(info);

                    break;
                case 2:
                    if(!L.vacia())
                        L.borrar();

                    break;
                case 3:
                    JOptionPane.showMessageDialog(null,L.consulta());
                    break;
            }
        } while(opc!=4);
        System.exit(1);
    }
}

```

## 8. Ejemplo Lista Encadenada

```
import javax.swing.*;

class Nodo {
    int info;
    Nodo(int info) {
        this.info = info;
    }
    Nodo sig;
    Nodo ant;
}

class Listas
{
    Nodo p,q,r;
    Nodo cab=null;
    Nodo ult=null;
    boolean vacia()
    {
        if(cab==null)
            return true;
        else
            return false;
    }
    boolean esta(int x)
    {
        p=q=cab;
        while(p!=null && p.info<x)
        {
            q=p;
            p=p.sig;
        }
        if(p!=null && p.info==x)
            return true;
        else
            return false;
    }
    void primero(int x)
    {
        r=new Nodo(x);
        cab=ult=r;
        r.ant=null;
        r.sig=null;
    }
    void insertar(int x)
    {
        r=new Nodo(x);
        q.sig=r;
        p.ant=r;
        r.ant=q;
        r.sig=p;
    }
    void menor(int x)
    {
        r=new Nodo(x);
        r.sig=cab;
        r.ant=null;
        cab.ant=r;
        cab=r;
    }
    void mayor(int x)
```

```

{
    r=new Nodo(x);
    ult.sig=r;
    r.ant=ult;
    r.sig=null;
    ult=r;
}
void adicion(int x)
{
    if(vacia())
        primero(x);
    else
        if(x>ult.info)
            mayor(x);
        else
            if(x<cab.info)
                menor(x);
            else
                if(!esta(x))
                    insertar(x);
}
void borrar(int x)
{
    if(esta(x))
    {
        if(p==cab)
        {
            if(cab==ult)
                cab=ult=null;
            else
            {
                cab=cab.sig;
                cab.ant=null;
            }
        }
        else
            if(p==ult)
            {
                ult=ult.ant;
                ult.sig=null;
            }
            else
            {
                r=p.sig;
                q.sig=r;
                r.ant=q;
            }
        }
    }
}
String consulta_asc()
{
    String S="CONSULTA \n";
    p=cab;
    while(p!=null)
    {
        S+=p.info+"\n";
        p=p.sig;
    }
    return S.toString();
}

```

```

    }
    String consulta_desc()
    {
        String S="C O N S U L T A \n";
        p=ult;
        while(p!=null)
        {

            S+=p.info+"\n";
            p=p.ant;
        }
        return S.toString();
    }
}
public class ListaEncadenada
{
    public static void main(String[]args)
    {
        int i,opc,info;
        Listas L=new Listas();
        Object [] valores = {"1. Adicion", "2. Borrar", "3. Consulta Ascendente", "4. Consulta
Descendente", "5.Salir"};
        do
        {
            String resp=(String) JOptionPane.showInputDialog(null, "Elija la Opcion", "Entrada de
datos",JOptionPane.QUESTION_MESSAGE, null, valores,valores[0]);
            opc=Character.digit(resp.charAt(0),10);
            switch(opc)
            {
                case 1:
                    info=Integer.parseInt(JOptionPane.showInputDialog(null, " Digite Info a Adicionar:"));
                    L.adicion(info);

                    break;
                case 2:

                    info=Integer.parseInt(JOptionPane.showInputDialog(null, " Digite Info a Borrar:"));
                    L.borrar(info);

                    break;
                case 3:
                    JOptionPane.showMessageDialog(null,L.consulta_asc());
                    break;
                case 4:
                    JOptionPane.showMessageDialog(null,L.consulta_desc());
                    break;
            }
        }

        while(opc!=5);
        System.exit(1);
    }
}

```



## 9. Ejemplo de Lista y Pila

```
import javax.swing.*;
class Nodo {
    int info;
    Nodo(int info) {
        this.info = info;
    }
    Nodo sig;
}
class Listas
{
    Nodo p,r;
    Nodo cab=null;
    boolean vacia()
    {
        if(cab==null)
            return true;
        else
            return false;
    }
    void adicion(int x)
    {
        r=new Nodo(x);
        r.sig=cab;
        cab=r;
    }
    void borrar()
    {
        cab=cab.sig;
    }
    String consulta()
    {
        String S="C O N S U L T A \n";
        p=cab;
        while(p!=null)
        {
            S+=p.info+"\n";
            p=p.sig;
        }
        return S.toString();
    }
}
public class ListaPila
{
    public static void main(String[]args)
    {
        int i,opc,info;
        Listas L=new Listas();
        Object [] valores = {"1. Adicion","2. Borrar","3. Consulta","4.Salir"};
        do
        {
            String resp=(String) JOptionPane.showInputDialog(null,"Elija la Opcion", "Entrada de
datos",JOptionPane.QUESTION_MESSAGE, null, valores,valores[0]);
            opc=Character.digit(resp.charAt(0),10);
            switch(opc)
            {
                case 1:
                    info=Integer.parseInt(JOptionPane.showInputDialog(null," Digite Info a Adicionar:"));

```

```
        L.adicion(info);  
  
break;  
case 2:  
    if(!L.vacia())  
        L.borrar();  
  
break;  
case 3:  
    JOptionPane.showMessageDialog(null,L.consulta());  
    break;  
    }  
    }  
    while(opc!=4);  
    System.exit(1);  
    }  
}
```

## 10. Ejemplo de Multilista

```
import javax.swing.*;

class Alumno {
    int Cod;
    String Nombre;
    Alumno(int Cod,String Nombre) {
        this.Cod = Cod;
        this.Nombre = Nombre;
    }
    Alumno sig;
}

class Programa
{
    int Id;
    String Nombre;
    Programa sig;
    Alumno p,q,r;
    Alumno cab=null;
    Alumno ult=null;
    Programa(int Id,String Nombre) {
        this.Id = Id;
        this.Nombre = Nombre;
    }
    boolean vacia()
    {
        if(cab==null)
            return true;
        else
            return false;
    }
    void adicion(int x,String y)
    {
        r=new Alumno(x,y);
        if(cab==null)
            cab=r;
        else
            ult.sig=r;
        ult=r;
        r.sig=null;
    }
    void borrar()
    {
        if(cab==ult)
            cab=ult=null;
        else
            cab=cab.sig;
    }
    String consulta()
    {
        String S="CONSULTA DE ALUMNOS\n";
        p=cab;
        while(p!=null)
        {
            S+=p.Cod+" "+p.Nombre+"\n";
            p=p.sig;
        }
        return S.toString();
    }
}
```

```

}
class Facultad
{
    Programa p,q,r;
    Programa cab=null;
    Programa ult=null;
    boolean vacia()
    {
        if(cab==null)
            return true;
        else
            return false;
    }
    void adicion(int x,String y)
    {
        r=new Programa(x,y);
        if(cab==null)
            cab=r;
        else
            ult.sig=r;
        ult=r;
        r.sig=null;
    }
    void borrar()
    {
        if(cab==ult)
            cab=ult=null;
        else
            cab=cab.sig;
    }
    boolean existe(int id)
    {
        p=cab;
        while(p!=null && p.Id!=id)
            p=p.sig;
        if(p==null)
            return false;
        else
            return true;
    }
    void borrarAlumno()
    {
        if(!p.vacia())
            p.borrar();
    }
    void adicionAlumno(int x,String y)
    {
        p.adicion(x,y);
    }
    String consulta()
    {
        String S="CONSULTA DE PROGRAMAS \n";
        p=cab;
        while(p!=null)
        {
            S+=p.Id+" "+p.Nombre+"\n";
            p=p.sig;
        }
        return S.toString();
    }
}

```

```

    }
    String consultaTodo()
    {
        String S="CONSULTA DE PROGRAMAS \n" ;
        S+=p.Id+ " "+p.Nombre+"\n";
        S+=p.consulta();
        return S.toString();
    }
}
public class MultiLista
{
    public static void main(String[]args)
    {
        int i,opc,cod,id;
        String Nombre;
        Facultad L=new Facultad();
        Object [] valores = {"1. Adicion Programa","2. Borrar Programa","3. Consulta Programa","4.Adicion
Alumno","5.Borrar Alumno","6. Consulta","7.Salir"};
        do
        {
            String resp=(String) JOptionPane.showInputDialog(null,"Elija la Opcion", "Entrada de
datos",JOptionPane.QUESTION_MESSAGE, null, valores,valores[0]);
            opc=Character.digit(resp.charAt(0),10);
            switch(opc)
            {
                case 1:
                    cod=Integer.parseInt(JOptionPane.showInputDialog(null," Digite Programa a Adicionar:"));
                    Nombre=JOptionPane.showInputDialog(null," Digite Nombre A Adicionar:");
                    L.adicion(cod,Nombre);

                    break;
                case 2:
                    if(!L.vacia())
                        L.borrar();

                    break;
                case 3:
                    JOptionPane.showMessageDialog(null,L.consulta());
                    break;
                case 4:
                case 5:
                case 6:
                    cod=Integer.parseInt(JOptionPane.showInputDialog(null," Digite Programa:"));
                    if(L.existe(cod))
                        switch(opc)
                        {
                            case 4:
                                cod=Integer.parseInt(JOptionPane.showInputDialog(null," Digite Alumno a Adicionar:"));
                                Nombre=JOptionPane.showInputDialog(null," Digite Nombre A Adicionar:");
                                L.adicionAlumno(cod,Nombre);

                                break;
                            case 5:
                                L.borrarAlumno();
                                break;
                            case 6:
                                JOptionPane.showMessageDialog(null,L.consultaTodo());
                        }
                    }
            }
        } while(opc!=7);
        System.exit(1);
    }
}

```

```
}
```

## 11. Ejemplo de Pila

```
import javax.swing.*;
import java.io.*;
import java.lang.Math.*;
import java.util.*;
public class Pila1
{
    static double SALDO=0;
    public static void main(String[] args)
    {
        int i,opc,nit;
        Pila A=new Pila(10);
        Info C=new Info();
        Object [] valores = {"1. Apilar","2. Desempila","3. Consulta","4.Salir"};
        do
        {
            String resp=(String) JOptionPane.showInputDialog(null,"Elija la Opcion", "Entrada de
datos",JOptionPane.QUESTION_MESSAGE, null, valores,valores[0]);
            opc=Character.digit(resp.charAt(0),10);
            switch(opc)
            {
                case 1:
                    if(A.llena())
                        JOptionPane.showMessageDialog(null,"No se Puede Atender Mas");
                    else
                    {
                        C.nit=Integer.parseInt(JOptionPane.showInputDialog(null,"Escriba el número de Nit"));
                        C.nomCliente=JOptionPane.showInputDialog(null,"Digite Nombre");
                        C.telefono=Integer.parseInt(JOptionPane.showInputDialog(null,"Digite Telefono"));
                        A.apila(C);
                    }
                    break;
                case 2:
                    if(A.vacia())
                        JOptionPane.showMessageDialog(null,"Pila de Atencion Vacía");
                    else
                    {
                        A.desempila();
                    }
                    break;
                case 3:
                    if(A.vacia())
                        JOptionPane.showMessageDialog(null,"Pila Vacía");
                    else
                        JOptionPane.showMessageDialog(null,A.imprime());
                    }
                }
            while(opc!=4);
        }
    }
}
class Info
{
    int nit;
    String nomCliente;
    int telefono;
```

```

}
class Pila
{
    int cab,n;
    Info A[];
    int i,k;
    public Pila(int n)
    {
        cab=-1;
        this.n=n;
        A=new Info[n];
        for(i=0;i<n;i++)
            A[i]=new Info();
    }
    boolean vacia()
    {
        if(cab==n-1)
            return true;
        else
            return false;
    }
    boolean llena()
    {
        if(cab==n-1)
            return true;
        else
            return false;
    }
    void apila(Info C)
    {
        cab++;
        A[cab].nit=C.nit;
        A[cab].telefono=C.telefono;
        A[cab].nomCliente=C.nomCliente;
    }
    void desempila()
    {
        cab--;
    }
    String imprime()
    {
        String S=" Elementos de la Pila\n";
        for(i=cab;i>=0;i--)
        {
            S=S+" "+new String().valueOf(A[i].nit).toString();
            S=S+" "+A[i].nomCliente;
            S=S+" "+new String().valueOf(A[i].telefono).toString()+"\n";
        }
        return S.toString();
    }
}

```

## 12. Ejemplo de Arbol Binario

```
import javax.swing.*.*;
import java.util.*.*;
class ArbolBin
{
    NodoArbol raiz=null;
    NodoArbol p,q,r;
    Stack Pila=new Stack();
    int fila=0;
    boolean existe(int x)
    {
        q=p=raiz;
        while(p!=null && x!=p.info)
        {
            q=p;
            if(x<p.info)
            {
                p=p.izq;
            }
            else
            {
                p=p.der;
            }
        }

        if(p==null)
            return false;
        else
            return true;
    }
    boolean adicion(int x)
    {
        if(existe(x))
            return false;
        // System.out.println("adicion de: "+x+"raiz: "+raiz+" q:"+q+" p: "+p);
        r=new NodoArbol(x);
        r.izq=r.der=null;
        if(raiz==null)
        {
            raiz=r;
        }
        else
        {
            if(x<q.info)
            {
                q.izq=r;
            }
            else
            {
                q.der=r;
            }
        }
        return true;
    }
    boolean masderecha(NodoArbol s)
    {
        Pila=new Stack();
        while(s!=null)
        {
            Pila.push((Object) s);
        }
    }
}
```



```

s=s.der;
}
if(Pila.empty())
    return false;
s=(NodoArbol) Pila.pop();
if(p==raiz)
    raiz=s;
else
{
    if(q.izq==p)
        q.izq=s;
    else
        q.der=s;
}
if(Pila.empty())
    return true;
r=(NodoArbol) Pila.pop();
r.der=s.izq;
s.izq=p.izq;
s.der=p.der;
return true;
}
boolean masizquierda(NodoArbol s)
{
    Pila=null;
    while(s!=null)
    {
        Pila.push((Object) s);
        s=s.izq;
    }
    if(Pila.empty())
        return false;
    s=(NodoArbol) Pila.pop();
    if(p==raiz)
        raiz=s;
    else
    {
        if(q.izq==p)
            q.izq=s;
        else
            q.der=s;
    }
    if(Pila.empty())
        return true;
    r=(NodoArbol) Pila.pop();
    r.izq=s.der;
    s.izq=p.izq;
    s.der=p.der;
    return true;
}
void caso1Borrado()// P. Es Nodo Terminal
{
    if(p==raiz)
        raiz=null;
    else
    {
        if(q.izq==p)
            q.izq=null;
        if(q.der==p)
            q.der=null;
    }
}

```

```

    }
}
void caso2Borrado()//Hijo Unico. P.der es Null ó P.izq es Null
{
    if(p==raiz)
    {
        if(p.izq==null)
            raiz=p.der;
        else
            raiz=p.izq;
    }
    else
    {
        if(p.izq==null)
        {
            if(q.izq==p)
                q.izq=p.der;
            else
                q.der=p.der;
        }
        else //p.der==null
        {
            if(q.izq==p)
                q.izq=p.izq;
            else
                q.der=p.izq;
        }
    }
}
void caso3Borrado()// Dos Hijos. P.der y P.izq no es null
{
    if(!masderecha(p.izq))
        if(!masizquierda(p.der))
            System.out.println(" No se pudo Borrar");
}
boolean borrar(int x)
{
    if(!existe(x))
        return false;
    if(p.izq!=null && p.der!=null)
        caso3Borrado();
    else
        if(p.izq==null && p.der==null)
            caso1Borrado();
        else
            caso2Borrado();
    return true;
}
void preorden(NodoArbol p)
{
    if(p!=null)
    {
        System.out.println(p.info);
        preorden(p.izq);
        preorden(p.der);
    }
}
void inorden(NodoArbol p)
{
    if(p!=null)

```

```

    {
        inorden(p.izq);
        System.out.println(p.info);
        inorden(p.der);
    }
}
void posorden(NodoArbol p)
{
    if(p!=null)
    {
        posorden(p.izq);
        posorden(p.der);
        System.out.println(p.info);
    }
}
void conteoNodo(NodoArbol p,int A[])
{
    if(p!=null)
    {
        A[0]++;
        conteoNodo(p.izq,A);
        conteoNodo(p.der,A);
    }
}
void contarNodos()
{
    int A[]=new int[2];
    p=raiz;
    conteoNodo(p,A);
    System.out.print("El numero de nodos: "+A[0]);
}
void imprimirInorden()
{
    System.out.println("Impresion de Arbol en InOrden");
    p=raiz;
    inorden(p);
}
void imprimirPreorden()
{
    System.out.println("Impresion de Arbol en PreOrden");
    p=raiz;
    preorden(p);
}
void imprimirPosorden()
{
    System.out.println("Impresion de Arbol en PosOrden");
    p=raiz;
    posorden(p);
}
}

class NodoArbol
{
    int info;

    NodoArbol(int x)
    {
        info=x;
    }
}

```

```

        NodoArbol izq,der;
    }
    public class Arbol
    {
        static double SALDO=0;
        public static void main(String[]args)
        {
            int info,opc;
            ArbolBin A=new ArbolBin();
            Object [] valores = {"1. Adicion","2. Borrar","3. Consulta En PreOrden","4. Consulta en InOrden","5.
Consulta en PosOrden","6. Conteo de Nodos","7.Salir"};
            do
            {
                String resp=(String) JOptionPane.showInputDialog(null,"Elija la Opcion", "Entrada de
datos",JOptionPane.QUESTION_MESSAGE, null, valores,valores[0]);
                opc=Character.digit(resp.charAt(0),10);
                switch(opc)
                {
                    case 1:
                        info=Integer.parseInt(JOptionPane.showInputDialog(null,"Escriba info a Borrar: "));
                        if(!A.adicion(info))
                            JOptionPane.showMessageDialog(null,"Existe Info");
                        break;
                    case 2:
                        info=Integer.parseInt(JOptionPane.showInputDialog(null,"Escriba info a Adicionar: "));
                        if(!A.borrar(info))
                            JOptionPane.showMessageDialog(null,"No Existe Info");
                        break;
                    case 3:
                        A.imprimirPreorden();
                        break;
                    case 4:
                        A.imprimirInorden();
                        break;
                    case 5:
                        A.imprimirPosorden();
                        break;
                    case 6:
                        A.contarNodos();
                        break;
                }
            } while(opc!=7);
            System.exit(0);
        }
    }

```

## Ejercicios a Realizar por el Estudiante:

El estudiante después de adquirir destreza y conocimiento acerca de lenguaje de programación JAVA debe realizar todos los programas propuesto por el tutor y debe pasarlos con código comando utilizando las librerías `java.io.*` y `applet` ; ya que los espuestos antriormente viene desarrollados en `java.swing.*`; o ambiente grafico.

## 7. FUENTES DOCUMENTALES

Curso Académico Módulo Programación Orientada a Objetos

Libros de la Biblioteca Virtual de Unad en Programación Orientada a Objetos

Consulta de Internet Programación Orientada a Objetos

Autor: Deitel y Deitel Introducción DOO con UML y los Patrones de Diseño JDBC tm, SERVLETS, JSP tm Editorial Pearson Prentice Hall Quinta Edición.

David Arnow Gerald Weiss Introducción a la Programación con Java tm Actualización a la Versión 2 de Java Editorial Addison Wesley.

Fcd Javier Ceballos Java 2 Curso de Programación 2 Edición Editorial Alfaomega Ra-Ma.  
Agustín Froute Java 2 Manual Usuario tutorial 3 Edición Editorial Alfaomega Ra-Ma 5.

Herbert Schildt Fundamentos de Programación Java 2 Editorial Mc Graw Hill . para conocer el lenguaje de programación JAVA.

Arnow, D., Weiss, G., Introducción a la programación con JAVA, Addison Wesley, 2000.

Larman, C., UML y patrones, Prentice Hall, 1998.

Meyer, B., Construcción de software orientado a objetos, Prentice Hall, segunda edición, 1997.

Wu, T., Introducción a la programación orientada a objetos con Java, Mc Graw Hill, 2000.

Joyanes, L., Programación orientada a objetos, Segunda edición, Mc Graw Hill, 1998.

Greß Voss, Introducción Orientada a Objetos, Editorial Mc Graw Hill, 1994.

Joyanes, L., C++ a su alcance un enfoque orientado a objetos, Editorial, Mc Graw Hill, 1994.

Cesar Becerra Santamaría, C++ Una Herramienta para la Programación Orientada a Objetos, Editorial Mc Graw Hill, 1.993.

Fco Javier Ceballos, Programación Orientada a Objetos con C++, Editorial Alfaomega, 1.998.

Nathan Meyers, Programación JAVA en Linux, Editorial Prentice Hall, 2.000.

[www.lawebdelprogramador.com](http://www.lawebdelprogramador.com)

[www.programacion.com](http://www.programacion.com)

<http://www.cimat.mx/~jlabreu/CursoJava/>

<http://www.mindview.net/Books/TIJ>

<http://java.sun.com/docs/books/tutorial/>

<http://programacion.com/java/tutoriales/>