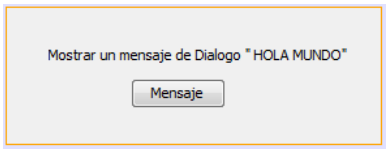
NETBEANS (JAVA)

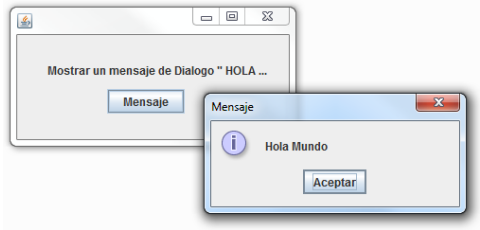
Es un entorno de desarrollo, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

EJERCICIO 1

1.- Mostrar un mensaje “Hola Mundo”

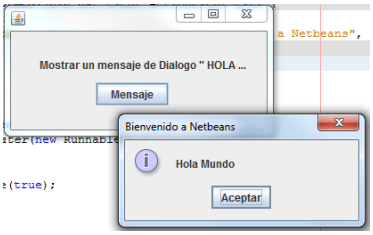



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JOptionPane.showMessageDialog(rootPane,"Hola Mundo");
}
```



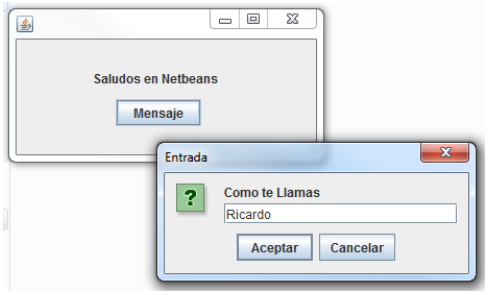
2.- Mostrar un mensaje “Hola Mundo” con Titulo Bienvenido a Netbeans

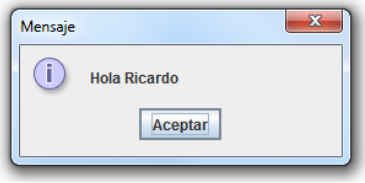
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JOptionPane.showMessageDialog(rootPane, "Hola Mundo", "Bienvenido a Netbeans", WIDTH);
}
```



3.- Mediante una caja de dialogo pedir nuestro nombre y saludarnos en otra.

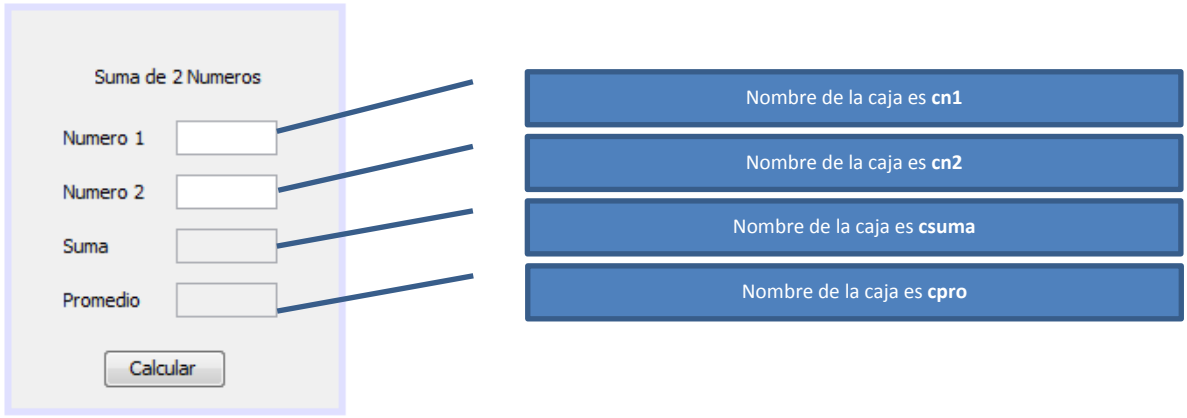
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    //Declaro una Variable String
    String a;
    //Pregunto en un cuadro de dialogo
    a= JOptionPane.showInputDialog("Como te Llamas");
    //Saludo en un cuadro de dialogo
    JOptionPane.showMessageDialog(rootPane,"Hola " + a );
}
```



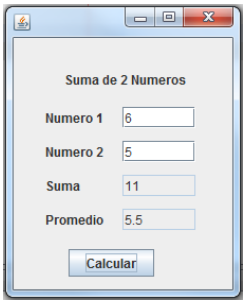


Ejercicio 2

1.- Mediante un JFrame realizar un programa que sume 2 números, muestre el resultado y su promedio.




```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    int a;
    int b;
    int c;
    double d;
    a=Integer.parseInt(this.cn1.getText());
    b=Integer.parseInt(this.cn2.getText());
    c=a+b;
    d=(double)c/2;
    this.csuma.setText(String.valueOf(c));
    this.cpro.setText(String.valueOf(d));
}
```



NOTA: Tome en cuenta que una caja de texto posee un valor STRING, y para poder almacenar en una variable INT, debe convertir de String a Int.

Obtener datos de una caja de texto.

```
this.cn1.getText()
```

Convertir de String a Int

```
Integer.parseInt
```

Enviar información a una caja de texto

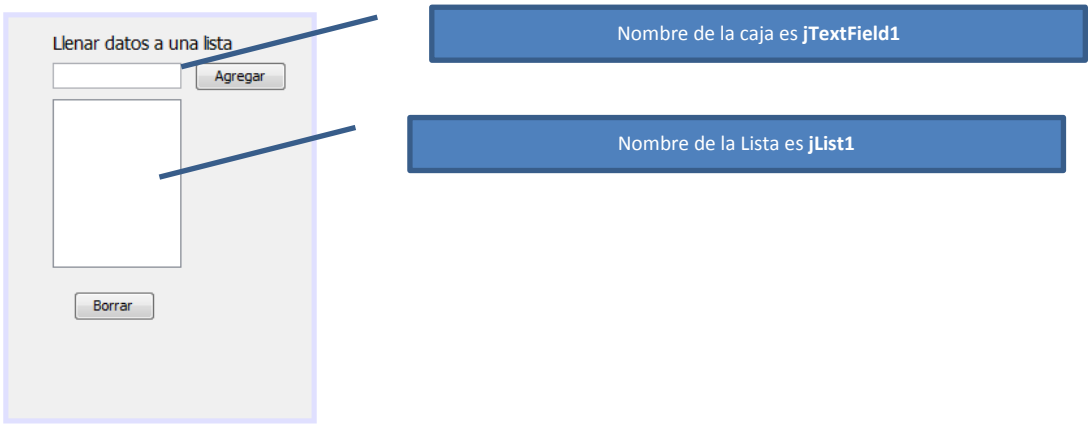
```
this.csuma.setText
```

Convertir cualquier tipo variable a String

```
String.valueOf(c)
```

Ejercicio 3

1.- Llenar datos en un JList.



Para manejar un JList debemos manejar una clase para las listas, para crear lo hacemos de esta manera.

```
DefaultListModel JList = new DefaultListModel();
```

- Codificación en el Botón **AGREGAR**

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JList.addElement(this.jTextField1.getText());
    this.jList1.setModel(JList);
}
```

- Codificación en el Botón **BORRAR**

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Creo de nuevo la lista
    JList.removeAllElements();
    this.jList1.setModel(JList);
}
```

NOTA:

Agrego datos a la lista

```
JList.addElement(this.jTextField1.getText());
```

Elimino todos los datos de la lista

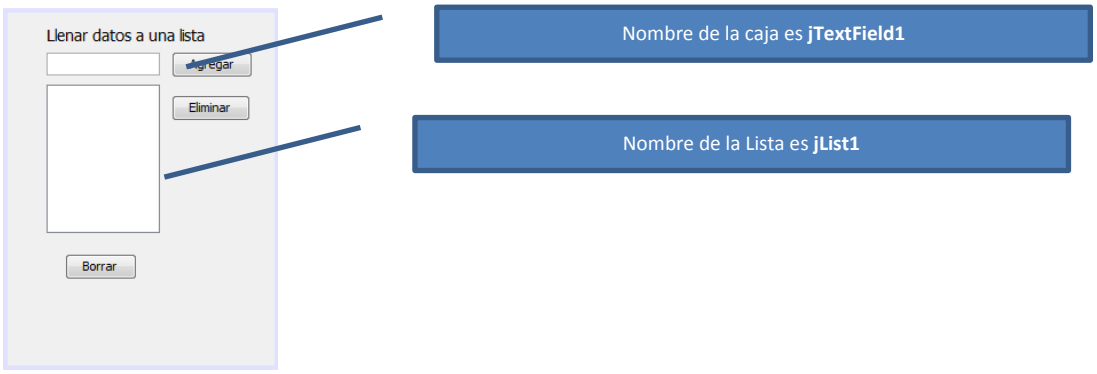
```
JList.removeAllElements();
```

Envió datos de la lista a mi elemento JList1

```
this.jList1.setModel(JList);
```

Ejercicio 4

1.- Llenar datos en un JList y eliminar un elemento seleccionado.



Para manejar un JList debemos manejar una clase para las listas, para crear lo hacemos de esta manera.

```
DefaultListModel JList = new DefaultListModel();
```

- Codificación en el Botón **AGREGAR**

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JList.addElement(this.jTextField1.getText());
    this.jList1.setModel(JList);
}
```

- Codificación en el Botón **BORRAR**

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Creo de nuevo la lista
    JList.removeAllElements();
    this.jList1.setModel(JList);
}
```

- Codificación en el Botón **ELIMINAR**

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int index = jList1.getSelectedIndex();
    if (index>=0){
        JList.remove(index);
    }else{
        JOptionPane.showMessageDialog(rootPane, "No hay elementos en la lista");
    }
}
```

NOTA:

Agrego datos a la lista

```
JList.addElement(this.jTextField1.getText());
```

Elimino todos los datos de la lista

```
JList.removeAllElements();
```

Envió datos de la lista a mi elemento JList1

```
this.jList1.setModel(JList);
```

Ejercicio 5

1.- Conectar Base de Datos en Access

Primeramente se debe importar la siguiente librería

```
import java.sql.*;
```

Con el siguiente código creamos la una conexión a una base de datos en Access ubicada en la dirección d:/uno.mdb

```
try{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
String db = "jdbc:odbc:Driver=Microsoft Access Driver (*.mdb);DBQ=d:/uno.mdb";
Connection con = DriverManager.getConnection( db, "", "");
JOptionPane.showMessageDialog(rootPane,"Conectado");
}
catch(Exception e){
JOptionPane.showMessageDialog(rootPane,"Erros es "+e);
}
```

Ejercicio 6

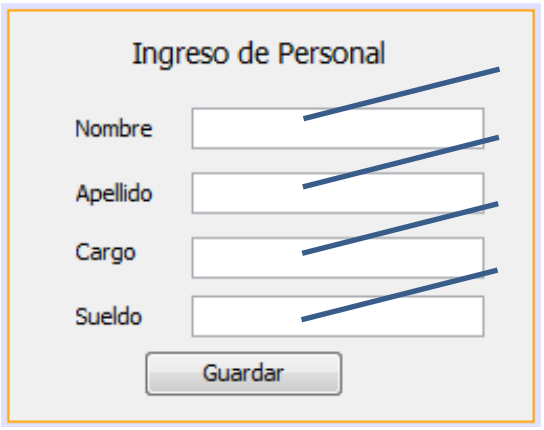
1.- Grabar información en la base de datos Anterior.

La base de datos cuyo nombre es UNO.MDB tiene la tabla personal.

personal

Nombre del campo	Tipo de datos
cod_per	Autonumeración
nom_per	Texto
ape_per	Texto
carg_per	Texto
suel_per	Número

Formulario JFrame



Nombre de la caja es **cnom**

Nombre de la caja es **cape**

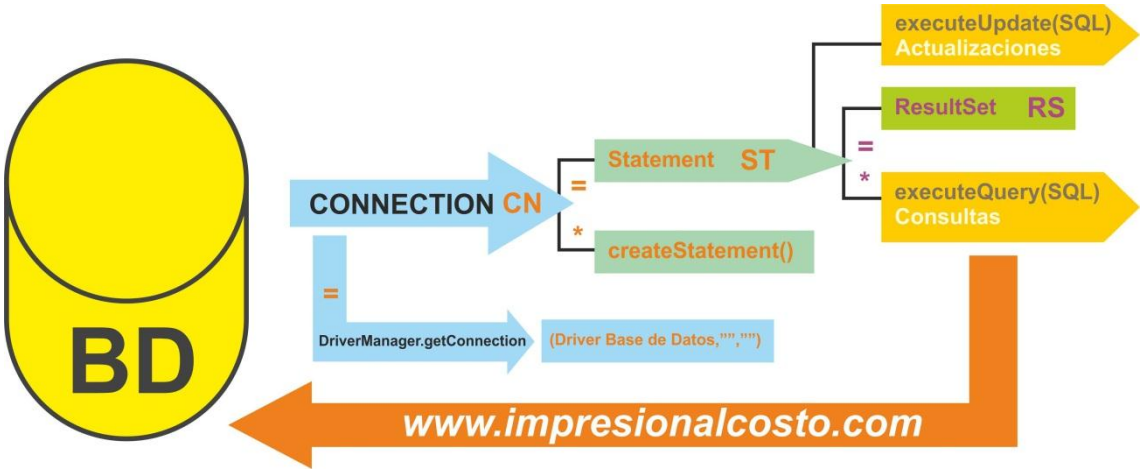
Nombre de la caja es **ccar**

Nombre de la caja es **csuel**

Codificación en el Botón **GUARDAR**

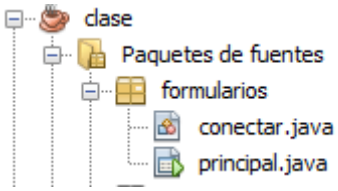
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
     // TODO add your handling code here:
    try{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String db = "jdbc:odbc:Driver=Microsoft Access Driver (*.mdb);DBQ=d:/uno.mdb";
    Connection con = DriverManager.getConnection( db, "", "");
    Statement s = con.createStatement();
    String sql = "insert into personal(nom_per,ape_per,carg_per,suel_per) values"
    + "('"+ this.cnom.getText() +"','"+ this.cape.getText() +"','"+ this.ccar.getText() +"',"
    + "'"+ this.csuel.getText() +"')";
    s.executeUpdate(sql);
     JOptionPane.showMessageDialog(rootPane,"Dato Guardado");
    }
    catch(Exception e){
    JOptionPane.showMessageDialog(rootPane,"Erros es "+e);
    }
}
```

ESTRUCTURA DE UNA CONEXIÓN EN NETBEANS



Ejercicio 7

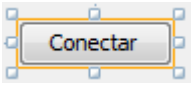
1. Conectar a una Base de Datos dentro de una Clase



Tenemos 2 archivos dentro del paquete formularios

```
public class conectar {
    Connection conect = null;
   public Connection conexion()
    {
        try {
           Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
           String strConect = "jdbc:odbc:Driver=Microsoft Access Driver"
                   + " (*.mdb);DBQ=d:/cwnetbeans/cw.mdb";
          conect = DriverManager.getConnection(strConect,"","" );
          JOptionPane.showMessageDialog(null,"Conectado");
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null,"Error "+e);
        }
        return conect;
    }
}
```

2. Llamar a la clase desde un formulario mediante un Botón




```
conectar a = new conectar();
```

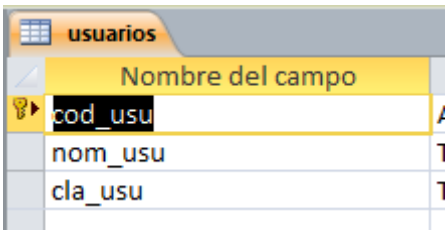
En esta línea de código nos permite llamar a la clase **CONECTAR.JAVA** y asignarle a una variable **a.**

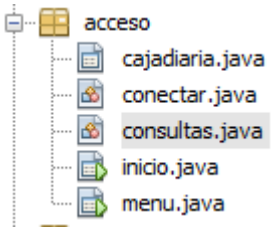
```
a.conexion();
```

Nos permite ejecutar el objeto **CONECTAR.JAVA** y ejecutar la clase **conexión()**.

Ejercicio 8

1. Ingreso al sistema mediante una clave guardada previamente en una Base de Datos.



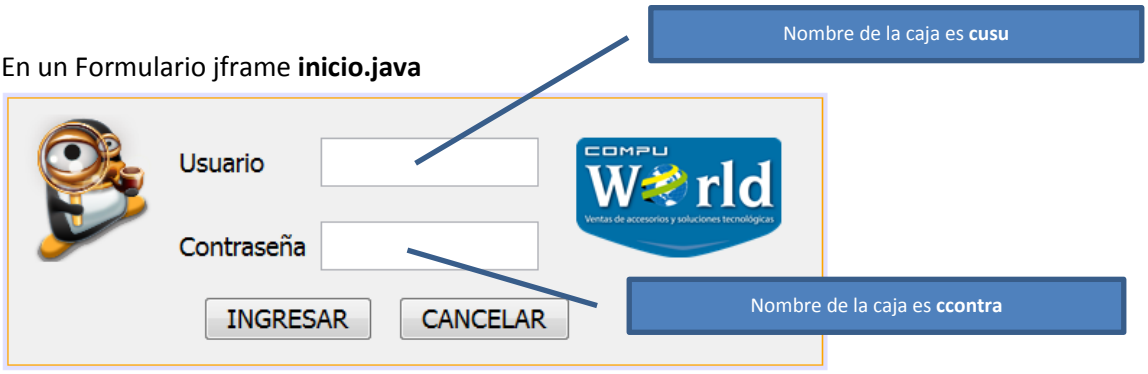


En el objeto **Consultas.java**

```
public class consultas {
    //Llamamo al objeto conectar en la variable con
    conectar con=new conectar();
    //Creamos variables de conexion
    Connection conect=null;
    ResultSet rs=null;
    Statement st=null;
    String query;

  public boolean verificar(String user,String pwr){
        int sw=0;
        query="select * from usuarios where "
                + "nom_usu='"+user+"' and cla_usu='"+pwr+"'";
        try{
            conect=con.conexion();
            st=conect.createStatement();
            rs=st.executeQuery(query);
                while(rs.next()){
                    if (rs.getString(1)==null)
                        sw=0;
                    else
                        sw=1;
                }
        }catch(SQLException e){
          JOptionPane.showMessageDialog(null,"ERROR: " + e);
        }
        if (sw==1) return true;
        else return false;
    }
}
```

2. En un Formulario jframe **inicio.java**

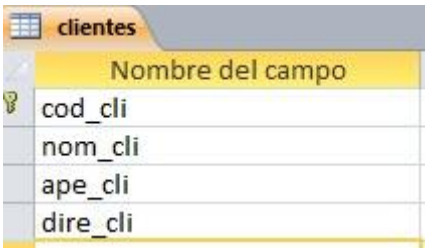


3. En el Botón Ingresar codificamos lo siguiente

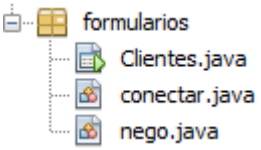

```
consultas acc = new consultas();
if(acc.verificar(this.cusu.getText(),this.ccontra.getText())){
JOptionPane.showMessageDialog(rootPane,"Clave Correcta");
}
else{
JOptionPane.showMessageDialog(rootPane,"Clave InCorrecta");
}
```

Ejercicio 9

1. Guardar los datos en una Base de Datos a través de una clase – TABLA **clientes**



2. Utilizando el objeto **conectar**
   (*Línea de Códigos para conectar a la base de datos – Ejercicio 7*).
   2.1. Vamos a crear el objeto **negó (Para realizar sentencias SQL)**
   2.2. Creamos un formulario **clientes**



2.1. nego.java

Insert into tabla (campos) values (valores)

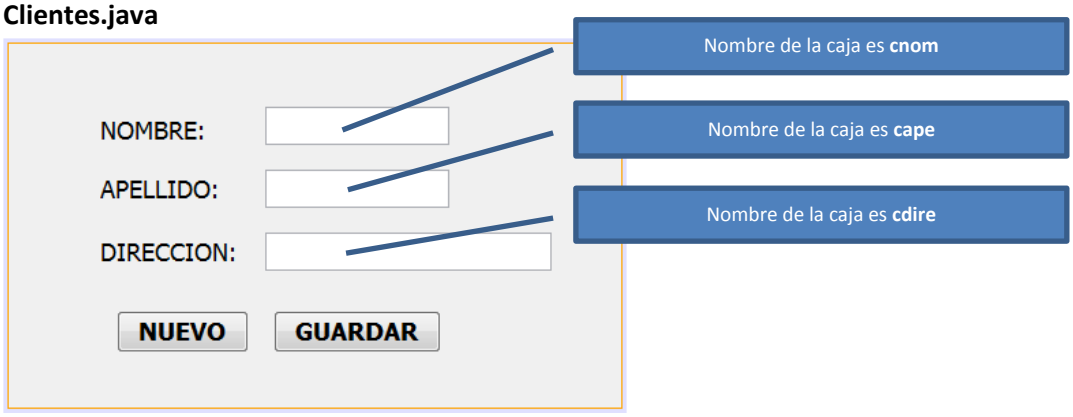
```
public class nego {
    //Llamamo al objeto conectar en la variable con
    conectar con=new conectar();
    //Creamos variables de conexion
    Connection conect=null;
    Statement st=null;
    ResultSet rs=null;
    String query;

    public void guardar(String tabla,String campos,String valores)
    { try {
            query="insert into "+tabla+"("+campos+") values("+valores+")";
            conect=con.conexion();
            st=conect.createStatement();
            st.executeUpdate(query);
            JOptionPane.showMessageDialog(null,"Registro Guardado");
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null,"Erros Sql "+ ex);
        }

    }
}
```

El Objeto es **nego** y dentro de este tiene una clase que se llama **guardar** el cual va a recibir 3 valores, 1. nombre de la tabla(**tabla**), 2. Los campos de la Tabla(**campos**), 3. Valores a Guardar(**valores**).

2.2. Clientes.java



GUARDAR

```
//Variales para adjuntar
String vnom,vape,vdire,campos,valores;
vnom=this.cnom.getText();
vape=this.cape.getText();
vdire=this.cdire.getText();
//Adjunto los nombre de los campos
campos="nom_cli,ape_cli,dire_cli";
//Adjunto los valores a guardar
valores="'"+vnom+"','"+vape+"','"+vdire+"'";
//Llamo a mi objeto nego
nego n=new nego();
//Dentro de nego, llamo a la clase guardar
//la cual me solicita los 3 valores
n.guardar("clientes", campos, valores);
```

NUEVO

```
this.cnom.setText("");
this.cape.setText("");
this.cdire.setText("");
```

VECTORES

Ejercicio 10

1. Realizar un programa que permita llenar un vector de 5 posiciones.

Crea un vector cuyo nombre es array

```
int array[];
array=new int[5];
```

Otra manera de Crear un vector cuyo nombre es array

```
int array [] = new int[5];
```

Sirve para recibir un flujo de caracteres

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
```

Sirve para llenar un vector de 5 posiciones

```
for (int x=0;x<=4;x++){
    System.out.println("Ingrese Numero en la posicion "+"["+(x+1)+"]");
    String nombre=br.readLine();
    array[x]=Integer.parseInt(nombre);
}
```

Muestra los números guardas en el vector de 5 posiciones

```
for (int y=0;y<=4;y++){
    System.out.println(array[y]);
}
```

2. Realizar un programa que permita llenar un vector de 10 posiciones con números pares.

```
int a[]=new int[10];
int par=2;
for (int x=0;x<10;x++){
    a[x]=par;
    par=par+2;
}
for (int y=0;y<10;y++){
    System.out.println(a[y]);
}
```

3. Crear un vector de 10 posiciones con números múltiplos del 4

```
int a[]=new int[10];
int par=4;
for (int x=0;x<10;x++){
    a[x]=par;
    par=par+4;
}
for (int y=0;y<10;y++){
    System.out.println(a[y]);
}
```

4. Crear 2 vectores de 5 posiciones y obtener un tercer vector con la multiplicación de los 2 vectores

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
int a[]=new int[5];
int b[]=new int[5];
int c[]=new int[5];
for (int x=0;x<=4;x++){
    System.out.print("Vector 1 Posicion ["+(x+1)+"]: " );
    String dato=br.readLine();
    a[x]=Integer.parseInt(dato);
}
 for (int x=0;x<=4;x++){
    System.out.print("Vector 2 Posicion ["+(x+1)+"]: " );
    String dato=br.readLine();
    b[x]=Integer.parseInt(dato);
}
for (int y=0;y<=4;y++){
    c[y]=a[y]*b[y];
}
for (int z=0;z<=4;z++){
    System.out.print("V1 P"+(z+1)+"["+a[z]+"] x "+"V2 P"+(z+1)+"["+b[z]+"] = " );
    System.out.println(c[z]);
}
```

5. Transponer un vector de 5 posiciones.

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
int a[]=new int[5];
int b[]=new int[5];
for (int x=0;x<=4;x++){
    System.out.print("Vector 1 Posicion ["+(x+1)+"]: " );
    String dato=br.readLine();
    a[x]=Integer.parseInt(dato);
    b[4-x]=a[x];
}
for (int z=0;z<=4;z++){
    System.out.print("Vector 2 Posicion ["+(z+1)+"]: " );
    System.out.println(b[z]);
}
```

6. Cuál es el número que más se repite en un vector de 5 posiciones.

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
int a[]=new int[5];
int aux[]=new int[5];
for (int x=0;x<=4;x++){
    System.out.print("Vector 1 Posicion ["+(x+1)+"]: " );
    String dato=br.readLine();
    a[x]=Integer.parseInt(dato);
}
for(int i=0;i<=4;i++){
    for(int j=0;j<=4;j++){
        if(a[i]==a[j]){
        aux[i]++;
        }}}
int mayor=0;
int index=0;

for (int i=0;i<=4;i++){
    if (aux[i]>mayor){
    mayor=aux[i];
    index=i;
    }}
System.out.print("Mas se repite ");
System.out.println(a[index]);
```

7. Sumar los elementos de un vector de 5 posiciones

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
int a[]=new int[5];
int aux=0;
for (int x=0;x<=4;x++){
    System.out.print("Vector 1 Posicion ["+(x+1)+"]: " );
    String dato=br.readLine();
    a[x]=Integer.parseInt(dato);
    aux=aux+a[x];
}
    System.out.print("La suma de los elementos es ");
    System.out.println(aux);
```

8. Sacar el promedio de un vector de 5 posiciones

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
int a[]=new int[5];
int sum=0;
double pro=0;
for (int x=0;x<=4;x++){
    System.out.print("Vector 1 Posicion ["+(x+1)+"]: " );
    String dato=br.readLine();
    a[x]=Integer.parseInt(dato);
    sum=sum+a[x];
}
    pro=(double)sum/5;
    System.out.print("El promedio es ");
    System.out.println(pro);
```

9. Intercalar 2 vectores de 5 posiciones a uno tercer vector de 10 posiciones

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
int a[]=new int[5];
int b[]=new int[5];
int c[]=new int[10];
for (int x=0;x<=4;x++){
    System.out.print("Vector 1 Posicion ["+(x+1)+"]: " );
    String dato=br.readLine();
    a[x]=Integer.parseInt(dato);
    System.out.print("Vector 2 Posicion ["+(x+1)+"]: " );
    String dato2=br.readLine();
    b[x]=Integer.parseInt(dato2);
}
int y=0;
for (int x=0;x<=9;x=x+2){
    c[x]=a[y];
    c[x+1]=b[y];
    y++;
}
for (int x=0;x<=9;x++){
    System.out.print("Vector ");
    System.out.println(c[x]);
}
```

10. Contar cuantos números pares y cuantos impares hay en un vector de 10 posiciones

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
int a[]=new int[5];
int pares=0;
int inpares=0;
for (int x=0;x<=4;x++){
    System.out.print("Vector 1 Posicion ["+(x+1)+"]: " );
    String dato=br.readLine();
    a[x]=Integer.parseInt(dato);
    if (a[x] % 2 ==0 ){
     pares++;
    }else{
     inpares++;
    }
}
System.out.println("Pares son: "+pares);
System.out.println("Inpares son: "+inpares);
```