INSTITUTO SAN ISIDRO

Curso de Java a distancia

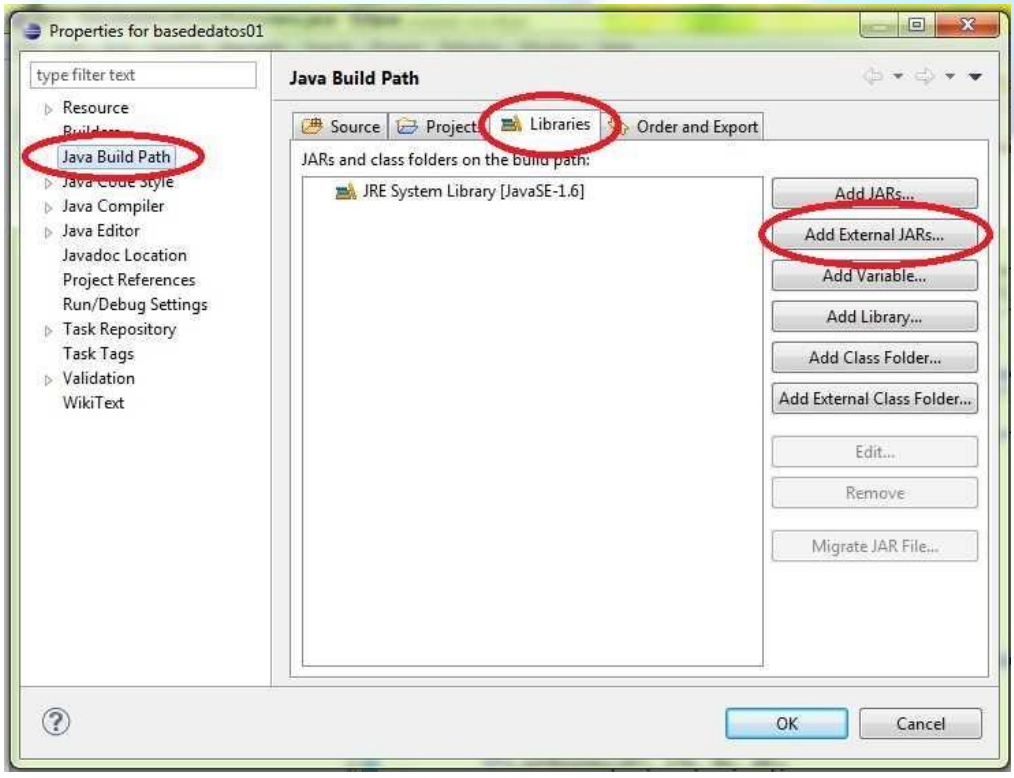
Clase 21: Alta y Consulta de una tabla de MySQL

Problema 1
Ya creamos en el concepto anterior una base de datos llamada bd1 y en la misma creamos una tabla llamada articulos.

Procederemos a implementar en Java un programa que nos permita comunicarnos con la base de datos "bd1" e insertar filas en la tabla "articulos" y posteriormente consultar su contenido.

1 - Creamos desde Eclipse un proyecto llamado "basededatos01" y seguidamente con el WindowBuilder creamos una clase llamada "Formulario".

2 - Primero debemos añadir el driver que descargamos (mysql-connector-java-5.1.18-bin.jar) presionamos el botón derecho del mouse sobre nuestro proyecto y seleccionamos la opción "Properties", aparece el siguiente diálogo:



Seleccionamos la opción "Java Build Path", de la parte central seleccionamos la pestaña "Libraries" y procedemos a presionar el botón "Add External JARs...", donde procedemos a buscar el archivo mysql-connector-java-5.1.18-bin.jar

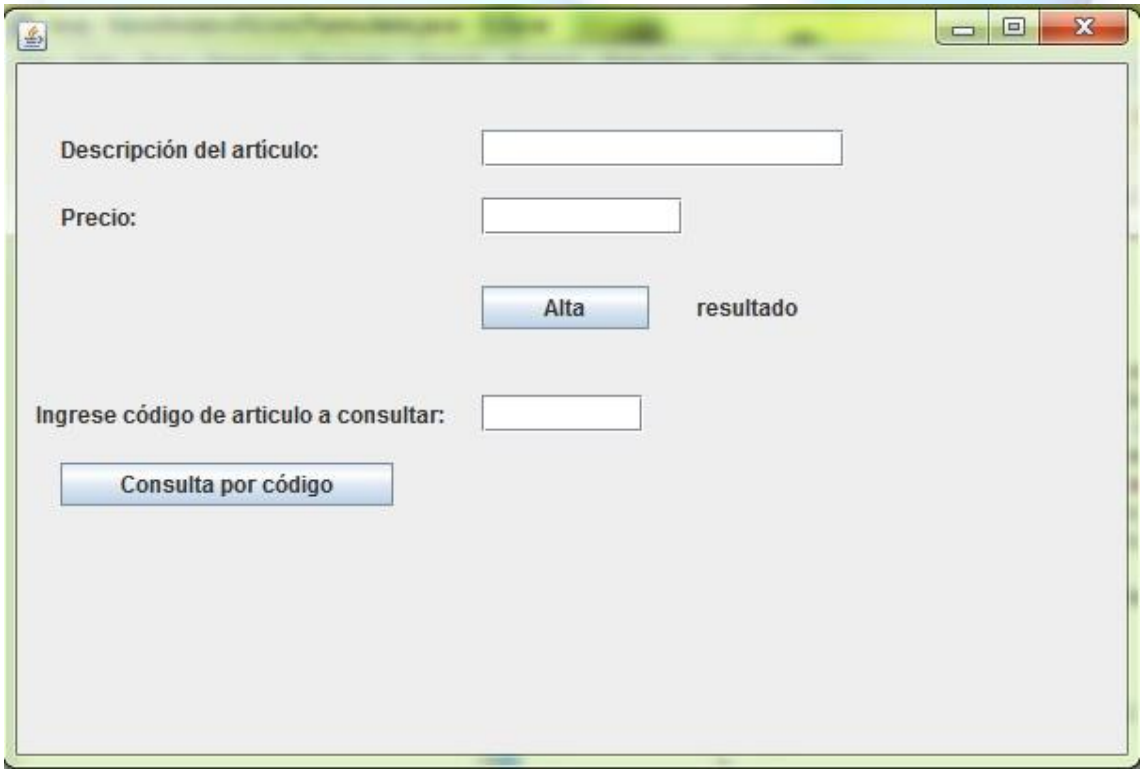
El código fuente de la clase formulario es:

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;


public class Formulario extends JFrame {

  private JPanel contentPane;
  private JTextField tf1;
  private JTextField tf2;
  private JLabel labelResultado;
  private JButton btnConsultaPorCdigo;
  private JLabel lblIngreseCdigoDe;
  private JTextField tf3;

  /**
   * Launch the application.
   */
  public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
      public void run() {
        try {
          Formulario frame = new Formulario();
          frame.setVisible(true);
        } catch (Exception e) {
          e.printStackTrace();
        }
      }
    });
  }

  /**
   * Create the frame.
   */
  public Formulario() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 606, 405);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    JLabel lblDescripcinDelArtculo = new JLabel("Descripción del artículo:");
```

```
    lblDescripcinDelArtculo.setBounds(23, 38, 193, 14);
    contentPane.add(lblDescripcinDelArtculo);

    tf1 = new JTextField();
    tf1.setBounds(247, 35, 193, 20);
    contentPane.add(tf1);
    tf1.setColumns(10);

    JLabel lblPrecio = new JLabel("Precio:");
    lblPrecio.setBounds(23, 74, 95, 14);
    contentPane.add(lblPrecio);

    tf2 = new JTextField();
    tf2.setBounds(247, 71, 107, 20);
    contentPane.add(tf2);
    tf2.setColumns(10);

    JButton btnAlta = new JButton("Alta");
    btnAlta.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent arg0) {
        labelResultado.setText("");
        try {
          Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root"
,"");
          Statement comando=conexion.createStatement();
          comando.executeUpdate("insert into articulos(descripcion,precio) values
("'+tf1.getText()+"',"+tf2.getText()+")");
          conexion.close();
          labelResultado.setText("se registraron los datos");
          tf1.setText("");
          tf2.setText("");
        } catch(SQLException ex){
          setTitle(ex.toString());
        }
      }
    });
    btnAlta.setBounds(247, 118, 89, 23);
    contentPane.add(btnAlta);

    labelResultado = new JLabel("resultado");
    labelResultado.setBounds(361, 122, 229, 14);
    contentPane.add(labelResultado);

    btnConsultaPorCdigo = new JButton("Consulta por código");
    btnConsultaPorCdigo.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent arg0) {
        labelResultado.setText("");
        tf1.setText("");
        tf2.setText("");
        try {
          Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root"
,"");
          Statement comando=conexion.createStatement();
          ResultSet registro = comando.executeQuery("select descripcion,precio from articulos
where codigo="+tf3.getText());
          if (registro.next()==true) {
            tf1.setText(registro.getString("descripcion"));
            tf2.setText(registro.getString("precio"));
          } else {
            labelResultado.setText("No existe un artículo con dicho código");
          }
```

```
          conexion.close();
        } catch(SQLException ex){
          setTitle(ex.toString());
        }
      }
    });
    btnConsultaPorCdigo.setBounds(23, 212, 177, 23);
    contentPane.add(btnConsultaPorCdigo);

    lblIngreseCdigoDe = new JLabel("Ingrese código de articulo a consultar:");
    lblIngreseCdigoDe.setBounds(10, 179, 243, 14);
    contentPane.add(lblIngreseCdigoDe);

    tf3 = new JTextField();
    tf3.setBounds(247, 176, 86, 20);
    contentPane.add(tf3);
    tf3.setColumns(10);
    cargarDriver();
  }

  private void cargarDriver() {
    try {
      Class.forName("com.mysql.jdbc.Driver");
    }catch(Exception ex) {
      setTitle(ex.toString());
    }
  }
}
```

La interfaz visual de la aplicación a implementar es la siguiente (se solicita el ingrese de la descripción del artículo y su precio, cuando se presiona el botón "Alta" se procede a insertar una fila en la tabla articulos de la base de datos bd1):



Expliquemos ahora el código fuente de la aplicación:

Primero debemos cargar en memoria el Driver, esto lo hacemos mediante el método cargarDriver que es llamado luego desde el constructor de la clase:

```
  private void cargarDriver() {
    try {
      Class.forName("com.mysql.jdbc.Driver");
    }catch(Exception ex) {
      setTitle(ex.toString());
    }
  }
```

Tenemos una clase llamada "Class" que tiene un método estático llamado forName, al mismo hay que pasar el nombre de la clase a importar:

```
      Class.forName("com.mysql.jdbc.Driver");
```

com.mysql.jdbc es el nombre del paquete donde se encuentra la clase Driver. Esta el la forma que importamos los driver en Java.

El método forName de la clase Class genera excepciones de tipo Excepcion que deben ser capturadas obligatoriamente (luego por eso encerramos el código en un bloque try/catch.

Si no importamos el driver desde el diálogo Properties del proyecto o indicamos en forma incorrecta el nombre del paquete o clase luego aparece en el título del JFrame un mensaje del error sucedido.

Luego desde el constructor llamamos por única vez al método cargarDriver:

```
                .....
                        tf3 = new JTextField();
                        tf3.setBounds(247, 176, 86, 20);
                        contentPane.add(tf3);
                        tf3.setColumns(10);
                        cargarDriver();
                }
```

Veamos ahora cual es el código a implementar cuando se presiona el botón "Alta":

```
    JButton btnAlta = new JButton("Alta");
    btnAlta.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent arg0) {
        labelResultado.setText("");
        try {
          Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root"
,"");
          Statement comando=conexion.createStatement();
          comando.executeUpdate("insert into articulos(descripcion,precio) values
("'+tf1.getText()+"',"+tf2.getText()+")");
          conexion.close();
          labelResultado.setText("se registraron los datos");
          tf1.setText("");
          tf2.setText("");
        } catch(SQLException ex){
          setTitle(ex.toString());
        }
      }
    });
```

En el actionPerformed procedemos primero a limpiar la label que puede tener un mensaje de ejecuciones anteriores:

```
        labelResultado.setText("");
```

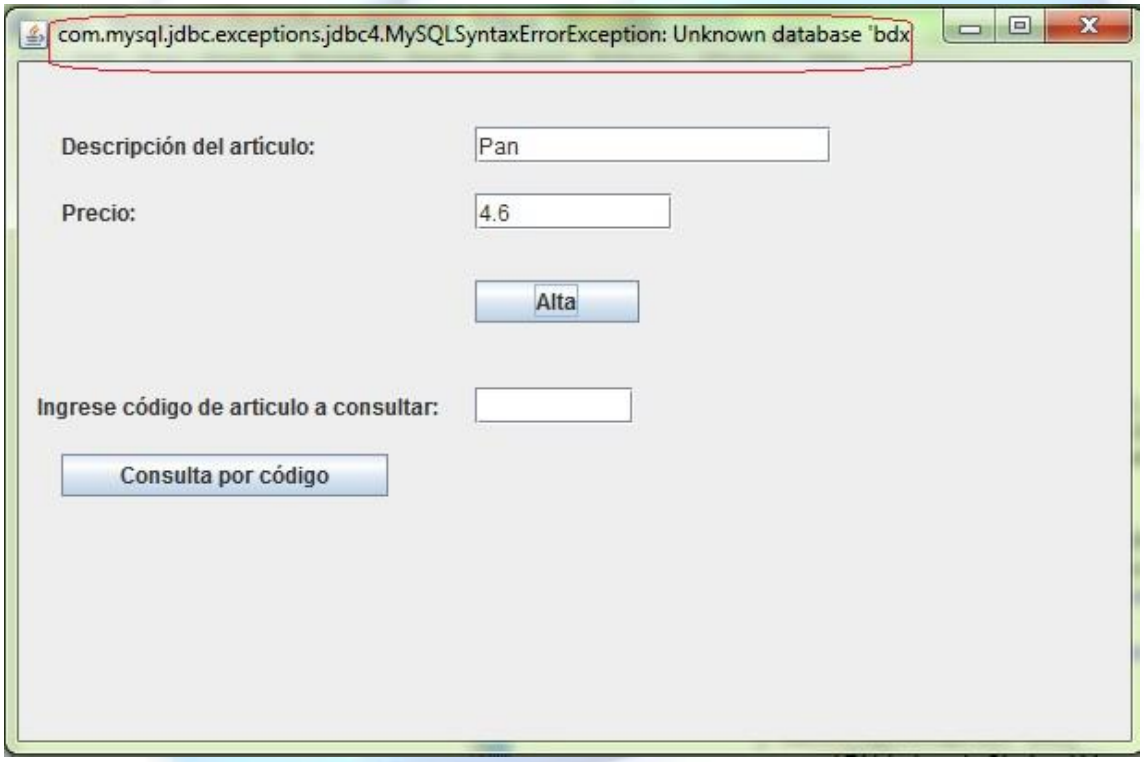
Todas las clases orientadas al acceso a base de datos generan excepciones de tipo SQLException y deben ser capturadas obligatoriamente. Lo primero que hacemos es crear un objeto de la clase Connection, para esto la clase DriverManager tiene un método llamado getConnection que retorna un objeto de la clase Connection:

```
        Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root"
,"");
```

El método getConnection debemos pasarle tres String, el primero indica el nombre de la base de datos que queremos acceder (en este caso "bd1"), el segundo parámetro es el nombre de usuario (recordemos que cuando instalamos el MySQL se crea un usuario por defecto llamado "root") y el último parámetro el la clave del usuario "root", por defecto esta clave es un String vacío.

Como podemos ver también previo a la base de datos tenemos en la cadena de conexión el nombre de nuestro servidor (localhost)

Si nos equivocamos por ejemplo con el nombre de base de datos a comunicarnos (por ejemplo cambiar "bd1" por "bdx") veremos en el título del JFrame el mensaje de error que nos devuelve el MySQL:



Luego creamos un objeto de la clase Statement a partir del objeto de la clase Connection que acabamos de crear:

```
        Statement comando=conexion.createStatement();
```

La clase Statement tiene un método llamado executeUpdate que le pasamos el comando SQL insert para agregar una fila a la tabla articulos:

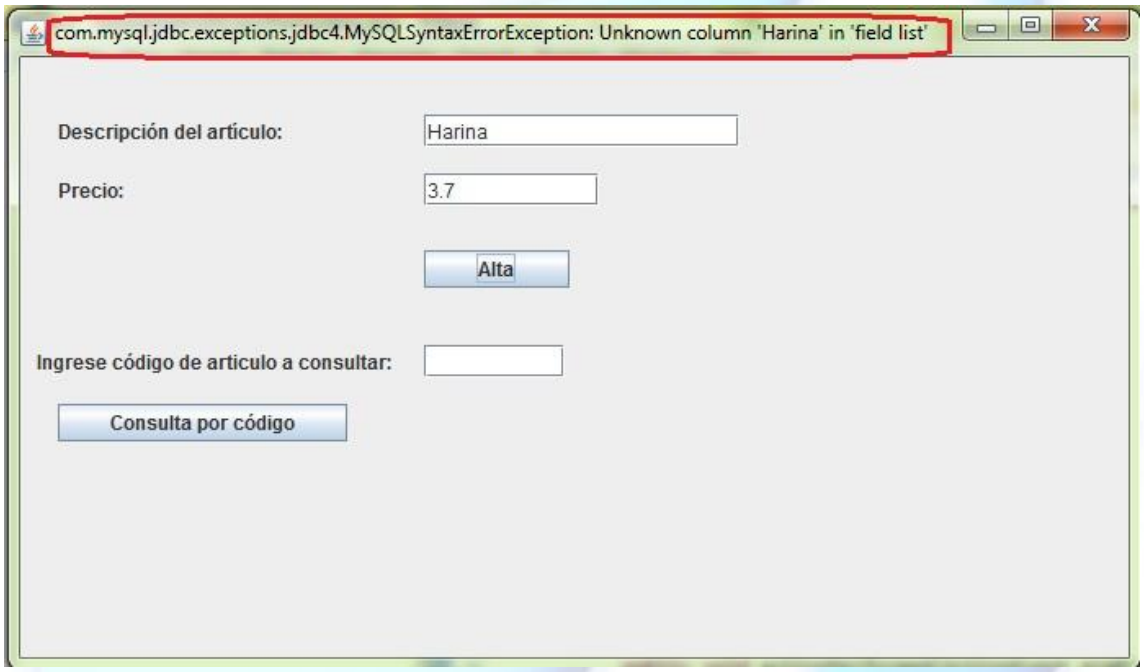
```
        comando.executeUpdate("insert into articulos(descripcion,precio) values
('"+tf1.getText()+"',"+tf2.getText()+")");
```

Como podemos ver generamos el String con el comando insert rescatando los datos de los dos controles de tipo JTextField. Es importante notar que en Java los String están encerrados entre comillas dobles y los contatenamos con el operador +. Las comillas simples son necesarias para los campos de tipo varchar de MySql (como podemos notar el lugar donde se dispondrá el texto de la descripción del artículo deben ir obligatoriamente las comillas simples):

..."'+tf1.getText()+"'...

Si nos olvidamos las comillas simples al generar el String con el comando Insert el MySQL nos devolverá un error que será capturado por el try/catch, por ejemplo si lo ejecutamos con la siguiente sintaxis (sin las comillas simples envolviendo el valor de la descripción):

```
comando.executeUpdate("insert into articulos(descripcion,precio) values
("+tf1.getText()+","+tf2.getText()+")");
```



Luego de solicitar la ejecución del comando Insert al MySQL procedemos a llamar al método close de la clase Connection:

```
conexion.close();
```

Con lo visto ya podemos agregar filas a la tabla articulos. Veamos ahora como consultar datos. El código a implementar cuando se presiona el botón "Consulta por código" es el siguiente:

```
btnConsultaPorCdigo = new JButton("Consulta por código");

btnConsultaPorCdigo.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
                labelResultado.setText("");
                tf1.setText("");
                tf2.setText("");
                try {
                        Connection
conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root" ,"");
                        Statement comando=conexion.createStatement();
                        ResultSet registro = comando.executeQuery("select
descripcion,precio from articulos where codigo="+tf3.getText());
                        if (registro.next()==true) {
                                tf1.setText(registro.getString("descripcion"));
                                tf2.setText(registro.getString("precio"));
                        } else {
```

```
                                        labelResultado.setText("No existe un artículo
con dicho código");
                                }
                                conexion.close();
                        } catch(SQLException ex){
                                setTitle(ex.toString());
                        }
                }
        });
```

De forma similar al Insert procedemos a crear un objeto de la clase Connection y otro objeto de la clase Statement:

```
                                Connection
conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root" ,"");
                                Statement comando=conexion.createStatement();
```

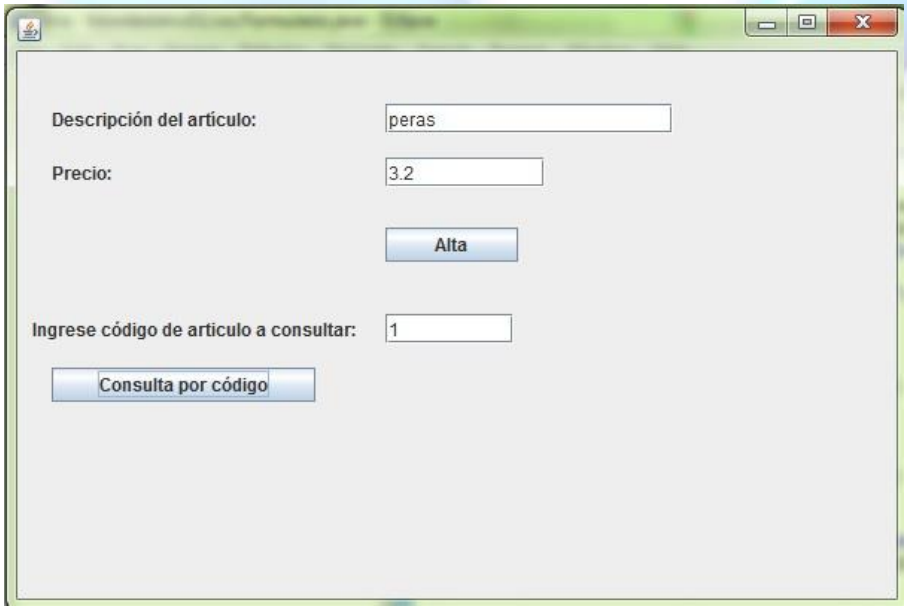
Seguidamente definimos una variable de la clase ResultSet llamada registro y llamamos al método executeQuery de la clase Statement del objeto que acabamos de crear previamente:

```
                                ResultSet registro = comando.executeQuery("select
descripcion,precio from articulos where codigo="+tf3.getText());
```

La clase ResultSet lo podemos imaginar como una tabla con todos los datos recuperados del comando SQL select que acaba de ejecutar el MySQL. En este ejemplo puede retornar una fila o ninguna ya que estamos utilizando la cláusula where y preguntando por el campo clave codigo.

Para acceder al registro devuelto debemos llamar al método next(), si retorna true es que si se recuperó una fila de la tabla articulos (es decir si existe el codigo de articulo ingresado), en caso que retorne false el método next() significa que no hay un artículo con el código que ingresamos en el control JTextField:

```
                                if (registro.next()==true) {
                                        tf1.setText(registro.getString("descripcion"));
                                        tf2.setText(registro.getString("precio"));
                                } else {
                                        labelResultado.setText("No existe un artículo
con dicho código");
                                }
```



Este proyecto lo puede descargar en un zip desde el adjunto del mail envíado: **basededatos01.zip**

Baja y modificación de datos de una tabla de MySQL

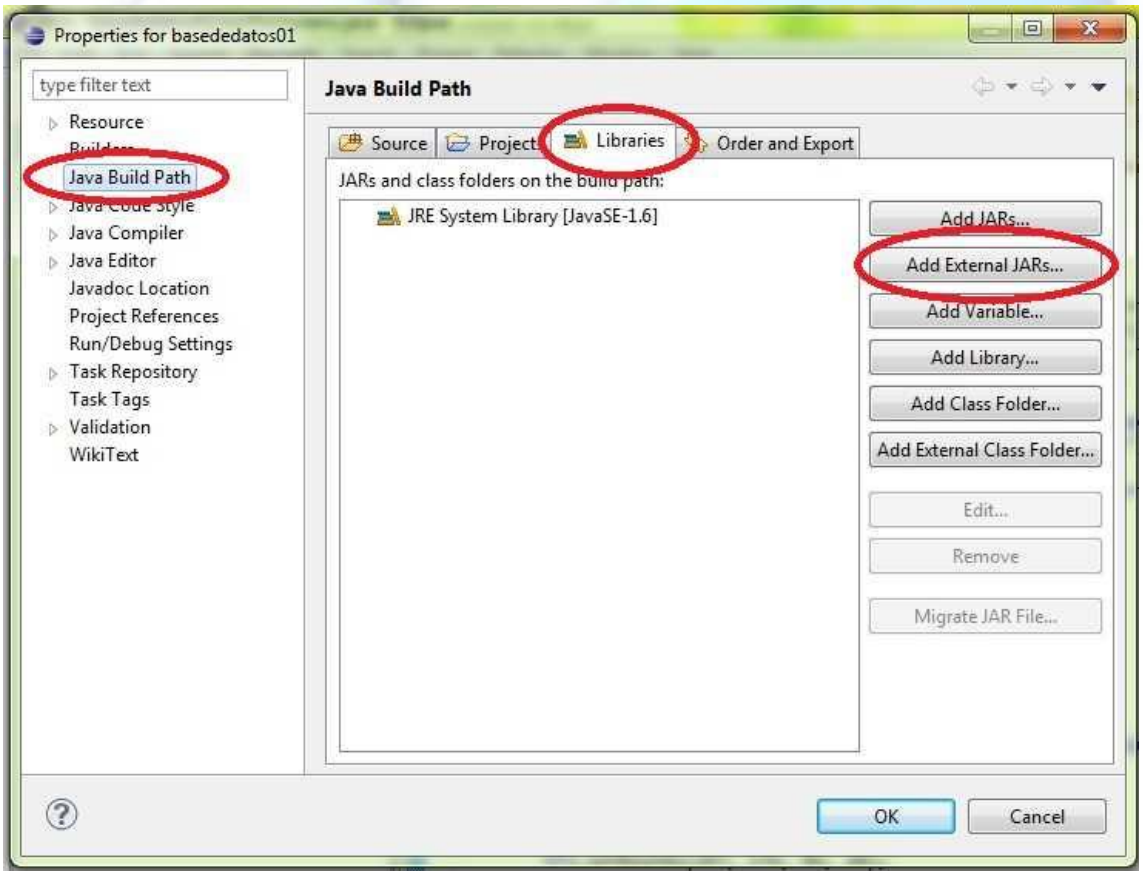
Problema 1

Ya creamos anteriormente una base de datos llamada bd1 y en la misma creamos una tabla llamada articulos.

Procederemos a implementar en Java un programa que nos permita comunicarnos con la base de datos "bd1" y consultar, borrar y modificar filas en la tabla "articulos".

1 - Creamos desde Eclipse un proyecto llamado "basededatos02" y seguidamente con el WindowBuilder creamos una clase llamada "Formulario".

2 - Primero debemos añadir el driver que descargamos (mysql-connector-java-5.1.18-bin.jar) presionamos el botón derecho del mouse sobre nuestro proyecto y seleccionamos la opción "Properties", aparece el siguiente diálogo:



Seleccionamos la opción "Java Build Path", de la parte central seleccionamos la pestaña "Libraries" y procedemos a presionar el botón "Add External JARs...", donde procedemos a buscar el archivo mysql-connector-java-5.1.18-bin.jar

El código fuente completo que resuelve este problema es:

```
import java.awt.EventQueue;
```

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;


public class Formulario extends JFrame {

  private JPanel contentPane;
  private JTextField tf1;
  private JTextField tf2;
  private JLabel labelResultado;
  private JButton btnConsultaPorCdigo;
  private JTextField tf3;

  /**
   * Launch the application.
   */
  public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
      public void run() {
        try {
          Formulario frame = new Formulario();
          frame.setVisible(true);
        } catch (Exception e) {
          e.printStackTrace();
        }
      }
    });
  }

  /**
   * Create the frame.
   */
  public Formulario() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 606, 405);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    JLabel lblDescripcinDelArtculo = new JLabel("Descripción del artículo:");
    lblDescripcinDelArtculo.setBounds(23, 38, 193, 14);
    contentPane.add(lblDescripcinDelArtculo);

    tf1 = new JTextField();
    tf1.setBounds(247, 35, 193, 20);
    contentPane.add(tf1);
    tf1.setColumns(10);
```

```
    JLabel lblPrecio = new JLabel("Precio:");
    lblPrecio.setBounds(23, 74, 95, 14);
    contentPane.add(lblPrecio);

    tf2 = new JTextField();
    tf2.setBounds(247, 71, 107, 20);
    contentPane.add(tf2);
    tf2.setColumns(10);

    labelResultado = new JLabel("resultado");
    labelResultado.setBounds(361, 122, 229, 14);
    contentPane.add(labelResultado);

    btnConsultaPorCdigo = new JButton("Consulta por código");
    btnConsultaPorCdigo.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent arg0) {
        labelResultado.setText("");
        tf1.setText("");
        tf2.setText("");
        try {
          Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root"
,"");
          Statement comando=conexion.createStatement();
          ResultSet registro = comando.executeQuery("select descripcion,precio from articulos
where codigo="+tf3.getText());
          if (registro.next()==true) {
            tf1.setText(registro.getString("descripcion"));
            tf2.setText(registro.getString("precio"));
          } else {
            labelResultado.setText("No existe un artículo con dicho código");
          }
          conexion.close();
        } catch(SQLException ex){
          setTitle(ex.toString());
        }
      }
    });
    btnConsultaPorCdigo.setBounds(25, 122, 177, 23);
    contentPane.add(btnConsultaPorCdigo);

    tf3 = new JTextField();
    tf3.setBounds(247, 123, 86, 20);
    contentPane.add(tf3);
    tf3.setColumns(10);

    JButton btnNewButton = new JButton("Borrar");
    btnNewButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent arg0) {
        labelResultado.setText("");
        try {
          Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root"
,"");
          Statement comando=conexion.createStatement();
          int cantidad = comando.executeUpdate("delete from articulos where
codigo="+tf3.getText());
          if (cantidad==1) {
            tf1.setText("");
            tf2.setText("");
            labelResultado.setText("Se borro el artículo con dicho código");
          } else {
            labelResultado.setText("No existe un artículo con dicho código");
```

```
          }
          conexion.close();
        } catch(SQLException ex){
          setTitle(ex.toString());
        }
      }
    });
    btnNewButton.setBounds(24, 156, 177, 23);
    contentPane.add(btnNewButton);

    JButton btnNewButton_1 = new JButton("Modificar");
    btnNewButton_1.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        labelResultado.setText("");
        try {
          Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root"
,"");
          Statement comando=conexion.createStatement();
          int cantidad = comando.executeUpdate("update articulos set descripcion='" + tf1.getText()
+ "'," +
                                  "precio=" + tf2.getText() + " where codigo="+tf3.getText());
          if (cantidad==1) {
            labelResultado.setText("Se modifico la descripcion y el precio del artículo con dicho
código");
          } else {
            labelResultado.setText("No existe un artículo con dicho código");
          }
          conexion.close();
        } catch(SQLException ex){
          setTitle(ex.toString());
        }
      }
    });
    btnNewButton_1.setBounds(21, 190, 179, 23);
    contentPane.add(btnNewButton_1);
    cargarDriver();
  }

  private void cargarDriver() {
    try {
      Class.forName("com.mysql.jdbc.Driver");
    }catch(Exception ex) {
      setTitle(ex.toString());
    }
  }
}
```

El código a implementar cuando se presiona el botón "Consulta por código" es el visto en el concepto anterior:

```
                btnConsultaPorCdigo = new JButton("Consulta por código");
                btnConsultaPorCdigo.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                labelResultado.setText("");
                                tf1.setText("");
                                tf2.setText("");
                                try {
                                        Connection
conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root" ,"");
                                        Statement comando=conexion.createStatement();
```

```
                                        ResultSet registro = comando.executeQuery("select
descripcion,precio from articulos where codigo="+tf3.getText());
                                        if (registro.next()==true) {
                                                tf1.setText(registro.getString("descripcion"));
                                                tf2.setText(registro.getString("precio"));
                                        } else {
                                                labelResultado.setText("No existe un artículo
con dicho código");
                                        }
                                        conexion.close();
                                } catch(SQLException ex){
                                        setTitle(ex.toString());
                                }
                        }
                });
```

Veamos el código para efectuar una baja en la tabla articulos:

```
    btnNewButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent arg0) {
        labelResultado.setText("");
        try {
          Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root"
,"");
          Statement comando=conexion.createStatement();
          int cantidad = comando.executeUpdate("delete from articulos where
codigo="+tf3.getText());
          if (cantidad==1) {
            tf1.setText("");
            tf2.setText("");
            labelResultado.setText("Se borro el artículo con dicho código");
          } else {
            labelResultado.setText("No existe un artículo con dicho código");
          }
          conexion.close();
        } catch(SQLException ex){
          setTitle(ex.toString());
        }
      }
    });
```

Luego de crear un objeto de la clase Statement procedemos a llamar al método executeUpdate con un comando SQL válido (delete from articulos where codigo= código de artículo) El código de artículo lo extraemos del tercer JTextField.
El método executeUpdate retorna un entero que representa la cantidad de registros borrados de la tabla articulos. Luego en caso que retorne un uno procedemos a mostrar en un JLabel el mensaje "Se borro el artículo con dicho código", en caso contrario mostramos el mensaje "No existe un artículo con dicho código".

Para la modificación procedemos de forma muy similar al borrado:

```
    btnNewButton_1.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        labelResultado.setText("");
        try {
          Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root"
,"");
          Statement comando=conexion.createStatement();
```

```
        int cantidad = comando.executeUpdate("update articulos set descripcion='" + tf1.getText()
+ "'," +
                                    "precio=" + tf2.getText() + " where codigo="+tf3.getText());
        if (cantidad==1) {
          labelResultado.setText("Se modifico la descripcion y el precio del artículo con dicho
código");
        } else {
          labelResultado.setText("No existe un artículo con dicho código");
        }
        conexion.close();
      } catch(SQLException ex){
        setTitle(ex.toString());
      }
    }
  });
```

Al método executeUpdate le pasamos un comando SQL de tipo update. Debemos concatenar los datos fijos del comando update con los valores que extraemos de los JTextField:

```
        int cantidad = comando.executeUpdate("update articulos set descripcion='" + tf1.getText()
+ "'," +
                                    "precio=" + tf2.getText() + " where codigo="+tf3.getText());
```

Es importante notar las comillas simples luego del caracter =, esto debido a que se trata de un campo de tipo varchar.

Nuevamente el método executeUpdate retorna la cantidad de registros modificados. En caso que retorne un 1 significa que se modificaron los datos correctamente.

Este proyecto lo puede descargar en un zip desde el adjunto del mail envíado: **basededatos02.zip**

Estructura selectiva switch

La estructura selectiva switch nos puede remplazar en algunas circunstancias una serie de if anidados, como veremos no es en todos los casos, solo en los casos que se verifican igualdades con ciertos valores.

Para ver su funcionamiento y sintaxis primero lo resolveremos el problema utilizando instrucciones if y luego empleando el switch.

Problema:

Simular el tirado de un dado 1000 veces. Mostrar luego la cantidad de veces que salieron cada valor. Resolverlo empleando tanto con if como con switch.

Programa:

```
public class SimulacionDados {

    public static void main(String[] args) {
        int lado1 = 0;
        int lado2 = 0;
        int lado3 = 0;
        int lado4 = 0;
        int lado5 = 0;
        int lado6 = 0;
        for (int f = 0; f < 1000; f++) {
```

```
        int dado = (int) (Math.random() * 6) + 1;
        if (dado == 1)
            lado1++;
        else if (dado == 2)
            lado2++;
        else if (dado == 3)
            lado3++;
        else if (dado == 4)
            lado4++;
        else if (dado == 5)
            lado5++;
        else if (dado == 6)
            lado6++;
    }
    System.out.println("Resultado de la primer simulación");
    System.out.println("Cantidad de veces que salieron 1:" + lado1);
    System.out.println("Cantidad de veces que salieron 2:" + lado2);
    System.out.println("Cantidad de veces que salieron 3:" + lado3);
    System.out.println("Cantidad de veces que salieron 4:" + lado4);
    System.out.println("Cantidad de veces que salieron 5:" + lado5);
    System.out.println("Cantidad de veces que salieron 6:" + lado6);

    // Empleando la estructura swich
    lado1 = 0;
    lado2 = 0;
    lado3 = 0;
    lado4 = 0;
    lado5 = 0;
    lado6 = 0;
    for (int f = 0; f < 1000; f++) {
        int dado = (int) (Math.random() * 6) + 1;
        switch (dado) {
        case 1:
            lado1++;
            break;
        case 2:
            lado2++;
            break;
        case 3:
            lado3++;
            break;
        case 4:
            lado4++;
            break;
        case 5:
            lado5++;
            break;
        case 6:
            lado6++;
            break;
        }
    }
    System.out.println("Resultado de la segunda simulación");
    System.out.println("Cantidad de veces que salieron 1:" + lado1);
    System.out.println("Cantidad de veces que salieron 2:" + lado2);
    System.out.println("Cantidad de veces que salieron 3:" + lado3);
    System.out.println("Cantidad de veces que salieron 4:" + lado4);
    System.out.println("Cantidad de veces que salieron 5:" + lado5);
    System.out.println("Cantidad de veces que salieron 6:" + lado6);
}
```

```
}
```

Como podemos ver el algoritmo queda más legible empleando la estructura selectiva switch:

```
switch (dado) {
case 1: lado1++;
        break;
case 2: lado2++;
        break;
case 3: lado3++;
        break;
case 4: lado4++;
        break;
case 5: lado5++;
        break;
case 6: lado6++;
        break;
}
```

La sintaxis del switch es la siguiente:

Luego de la palabra clave switch debemos indicar entre paréntesis el nombre de la variable a analizar (puede ser una variable entera, char o String).

Entre llaves debemos indicar luego de la palabra clave 'case' el valor a comparar. Si la variable a analizar tiene dicho valor pasa a ejecutar todas las instrucciones que indiquemos luego de los dos puntos.

Es importante terminar con la palabra clave break la finalización del bloque de un 'case', con otro ejemplo veremos más adelante que sucede si no especificamos el break.

default

Así como una instrucción if tiene en forma opcional el bloque del else, el comando switch tiene en forma opcional el bloque 'default'.

Problema:

Generar 100 valores aleatorios comprendidos entre 1 y 10. Contar cuantos se generaron con los valores 1,5,10 y los que no son ni 1,5 o 10.

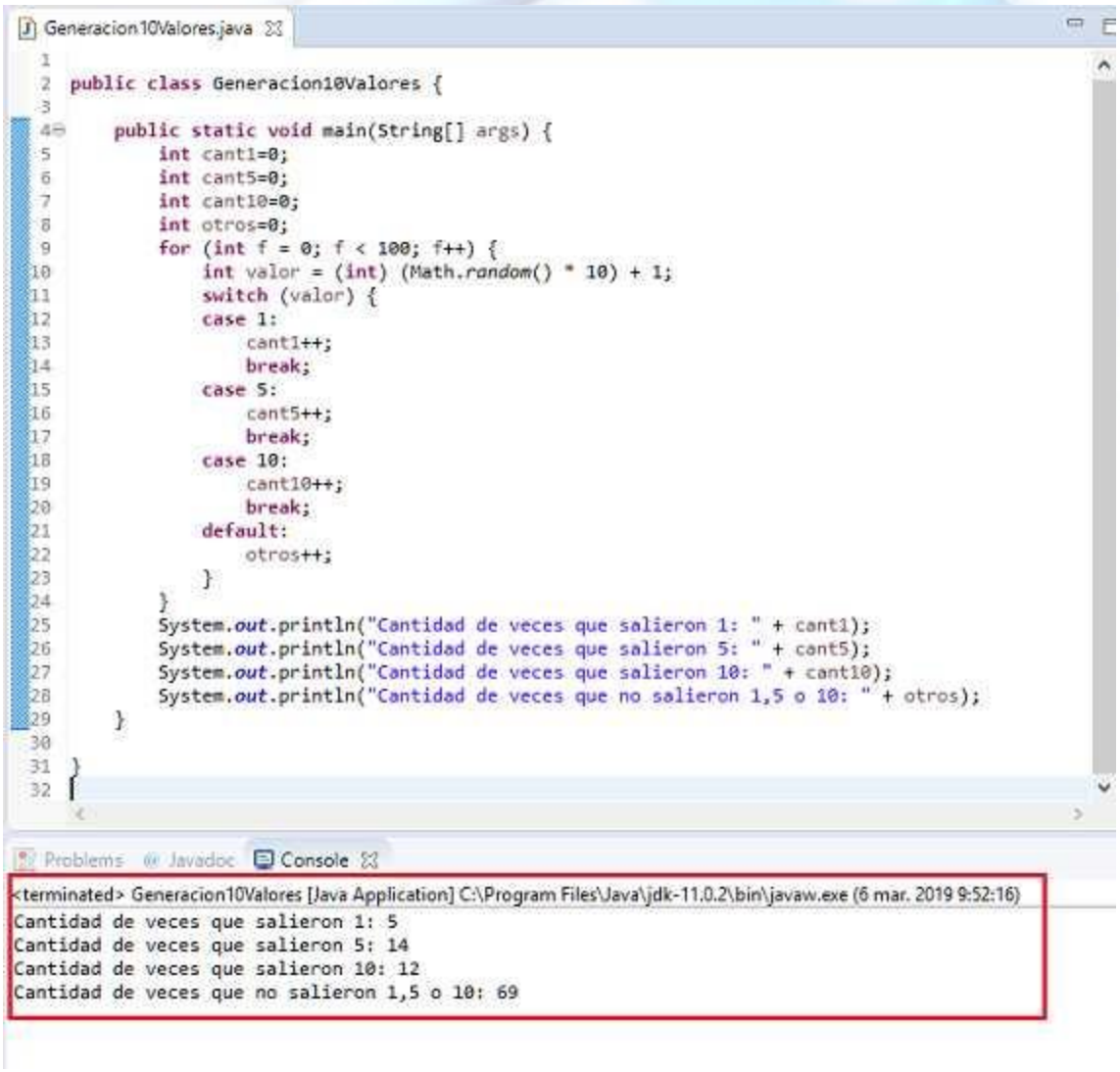
Programa:

```
public class Generacion10Valores {

    public static void main(String[] args) {
        int cant1=0;
        int cant5=0;
        int cant10=0;
        int otros=0;
        for (int f = 0; f < 100; f++) {
            int valor = (int) (Math.random() * 10) + 1;
            switch (valor) {
            case 1:
                cant1++;
                break;
            case 5:
                cant5++;
                break;
            case 10:
                cant10++;
```

```
            break;
         default:
            otros++;
         }
      }
      System.out.println("Cantidad de veces que salieron 1: " + cant1);
      System.out.println("Cantidad de veces que salieron 5: " + cant5);
      System.out.println("Cantidad de veces que salieron 10: " + cant10);
      System.out.println("Cantidad de veces que no salieron 1,5 o 10: " + otros);
   }

}
```

En este caso la estructura selectiva 'switch' tiene el bloque del 'default'. Todas las veces que en la variable valor no se almacene un 1,5 o 10 se ejecutarán las instrucciones que especifiquemos luego de la palabra clave 'default'.

La ejecución del programa nos genera una salida similar a:



Sin palabra clave 'break'

Si no disponemos la palabra clave 'break' en un 'case' luego se ejecuta el siguiente 'case', por ejemplo:

```
      switch (valor) {
```

```
        case 1:
           cant1++;
        case 5:
           cant5++;
        case 10:
           cant10++;
        default:
           otros++;
        }
```

Si la variable 'valor' almacena un 1 luego se ejecutan las instrucciones seguidas al primer caso:

```
        case 1:
           cant1++;
```

Al no tener un break se ejecuta el segundo caso, es decir se incrementa la variable 'cant5' en uno:

```
        case 5:
           cant5++;
```

También al no tener break el segundo caso se ejecuta el tercer 'case' e inclusive el 'default'.

Este comportamiento no tiene sentido y es un error lógico olvidarnos los break.

En algunas situaciones puede tener sentido no disponer 'break', veamos un ejemplo.

Problema:

Simular el tirado de un dado 1000 veces. Mostrar luego la cantidad de veces que salieron valores pares e impares.

Programa:

```
public class DadosParesImpares {
   public static void main(String[] args) {
      int pares = 0;
      int impares = 0;
      for (int f = 0; f < 1000; f++) {
         int dado = (int) (Math.random() * 6) + 1;
         switch (dado) {
         case 1:
         case 3:
         case 5:
            impares++;
            break;
         case 2:
         case 4:
         case 6:
            pares++;
            break;
         }
      }
      System.out.println("Cantidad de veces que salieron pares: " + pares);
      System.out.println("Cantidad de veces que salieron impares: " + impares);
   }
}
```

Es muy común emplear bloques de 'case' vacíos sin emplear la palabra clave 'break'. Luego si la variable 'dado' toma el valor 1 o 3 luego se ejecuta el 'case 5' incrementando el contador

'impares'. En el 'case 5' es importante notar que si finalizamos el bloque de instrucciones con un 'break'.

variable de tipo String

En la estructura selectiva switch podemos emplear una variable de tipo String.

Problema:

Almacenar en un vector los nombres de días que trabaja un empleado. Contar la cantidad de días que pertenecen a días laborables y fin de semana.

Programa:

```
public class Dias {

   public static void main(String[] ar) {
      String[] trabajo = { "lunes", "miércoles", "sábado", "domingo" };
      int laborables = 0;
      int finsemana = 0;
      for (int f = 0; f < trabajo.length; f++)
         switch (trabajo[f]) {
         case "lunes":
         case "martes":
         case "miércoles":
         case "jueves":
         case "viernes":
            laborables++;
            break;
         case "sábado":
         case "domingo":
            finsemana++;
            break;
         }
      System.out.println("Cantidad de días que trabaja en días hábiles: " + laborables);
      System.out.println("Cantidad de días que trabaja en fin de semana: " + finsemana);
   }

}
```

En el switch cada vuelta del for accede a un elemento del vector 'trabajo' que son de tipo String.

Muchas gracias hasta la próxima clase.

Alsina 16 [B1642FNB] San Isidro | Pcia. De Buenos Aires |Argentina |

TEL.: [011] 4742-1532 o [011] 4742-1665 |

www.institutosanisidro.com.ar info@institutosanisidro.com.ar