Modulo: Programación con sistemas gestores de bases de datos **Siglema: PSGB-04**

Docente: Alejo Carlos Mena Ruiz

Unidad 2 Programación de aplicaciones con conexión a bases de datos

2.2. Desarrolla la aplicación para el acceso a la base de datos mediante la integración de recursos del sistema gestor de bases de datos y los requerimientos establecidos del usuario.

Contenido temático

A) Programación de la interactividad de componentes de la aplicación.

Objetivo

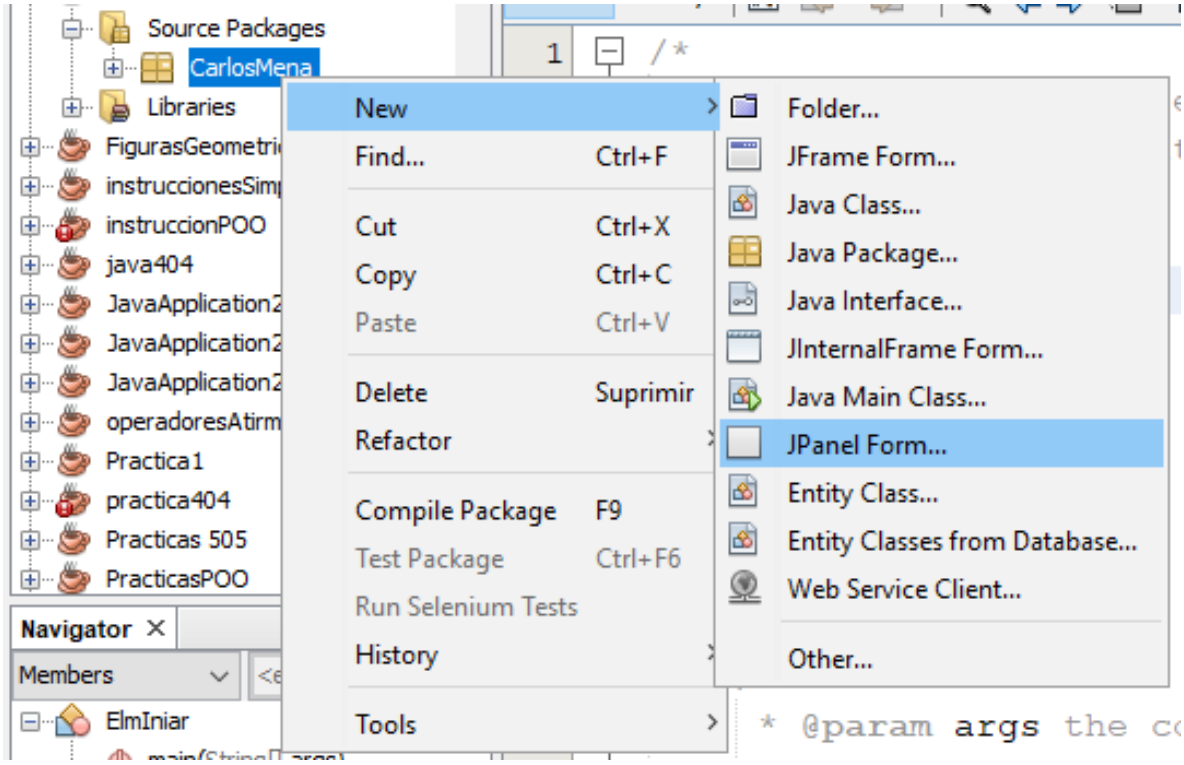
Lograr que el estudiante identifique los componentes visuales de los formularios, así como la programación de los mismos para la creación de una aplicación para el acceso a la base de datos mediante la integración de recursos del sistema

Actividad Práctica

Uso de formularios y manejo de componentes visuales.

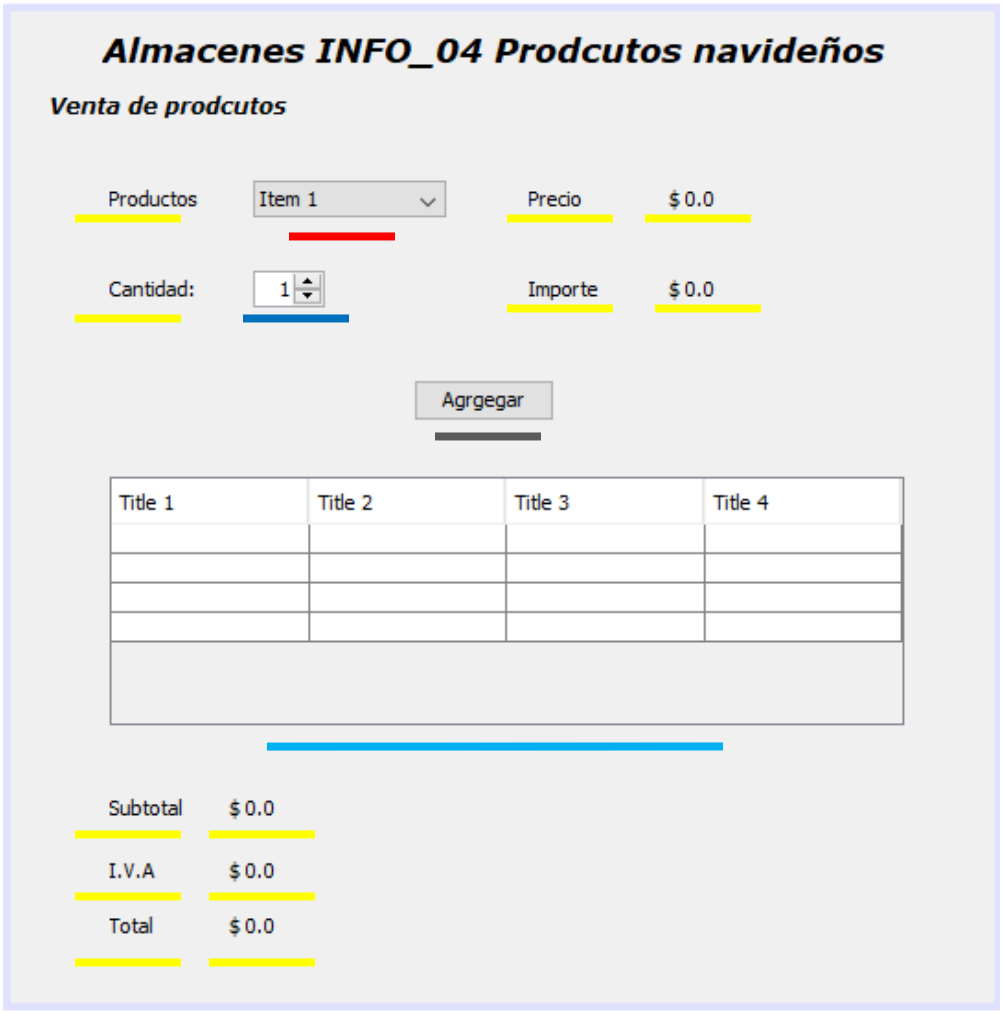
Abrir el IDE NetBeans / File / Java / JavaAplication / Nombre del alumno / Finish

Crea un JFrameForm en el paquete creado



Coloca el nombre: Uso_de_formularios

Genera el siguiente Diseño



Label

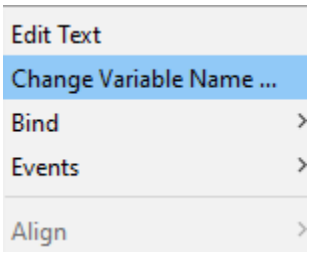
ConboBox

Spinner

Buttom

Table

Cambia el nombre de los componentes, recuerda colocar un nombre corto que los identifique



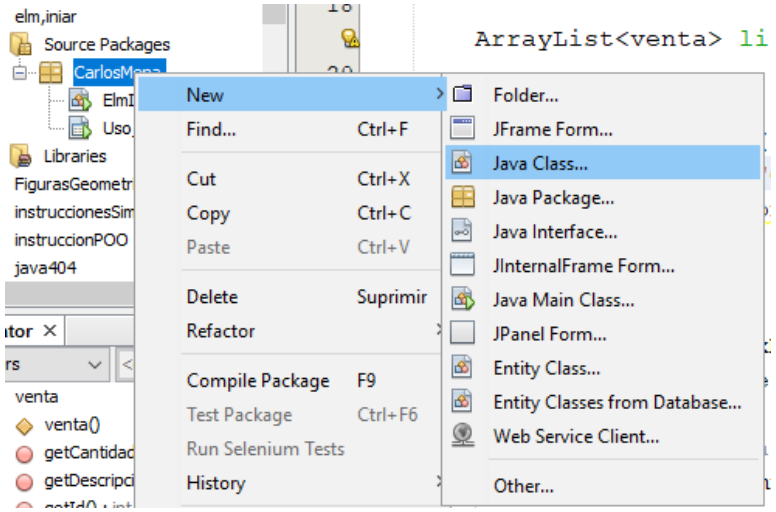
Ingresa a la Pestaña Source ubica la ubicación de la sintaxis y coloca el código en amarillo

```
initComponents();
this.setTitle("Caja 03");
this.setLocationRelativeTo(null);
```

La primera sintaxis coloca Titulo a nuestra aplicación

La segunda sintaxis centra la ventana de mi aplicación en mi monitor

Genera en el paquete una clase denominada Venta

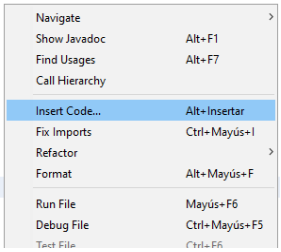


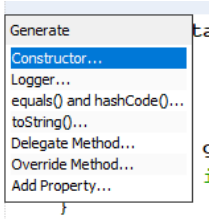
Dentro de la clase declara las siguientes variables:

```
public class venta {

    int id;
    int cantidad;
    double precio;
    double importe;
    String descripcion;

    }
```

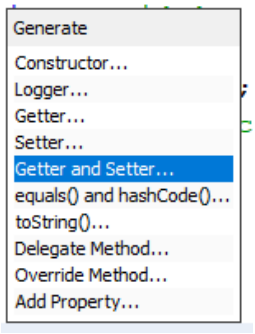
Después de haber declarado las variables selecciona con el botón derecho del Mouse y selecciona InsertCode

Primero genera un Constructor Vacío





Segundo genera los métodos Getter and Setter, selecciona todas las casillas



Regresar a la codificación y edición del formulario y declara las siguientes variables

```
public class Formulario extends javax.swing.JFrame {

    String productos[]={"Jabon 123", "ACEITE"};
    double precios[]={34,48};
    double precio=0;
    int cantidad=0;
```

La primera variable corresponde a los productos que serán registrados por las ventas, ingresa

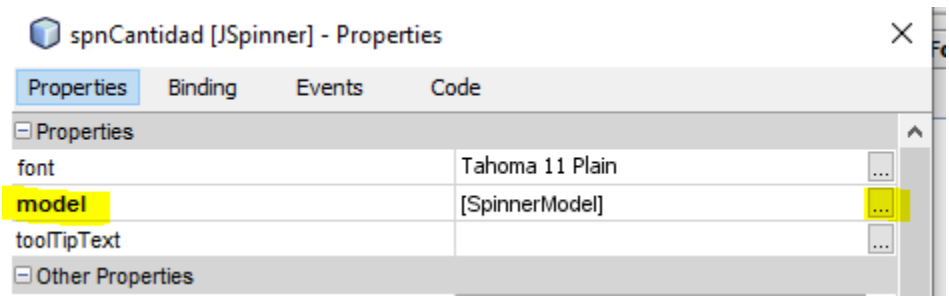
20 productos.

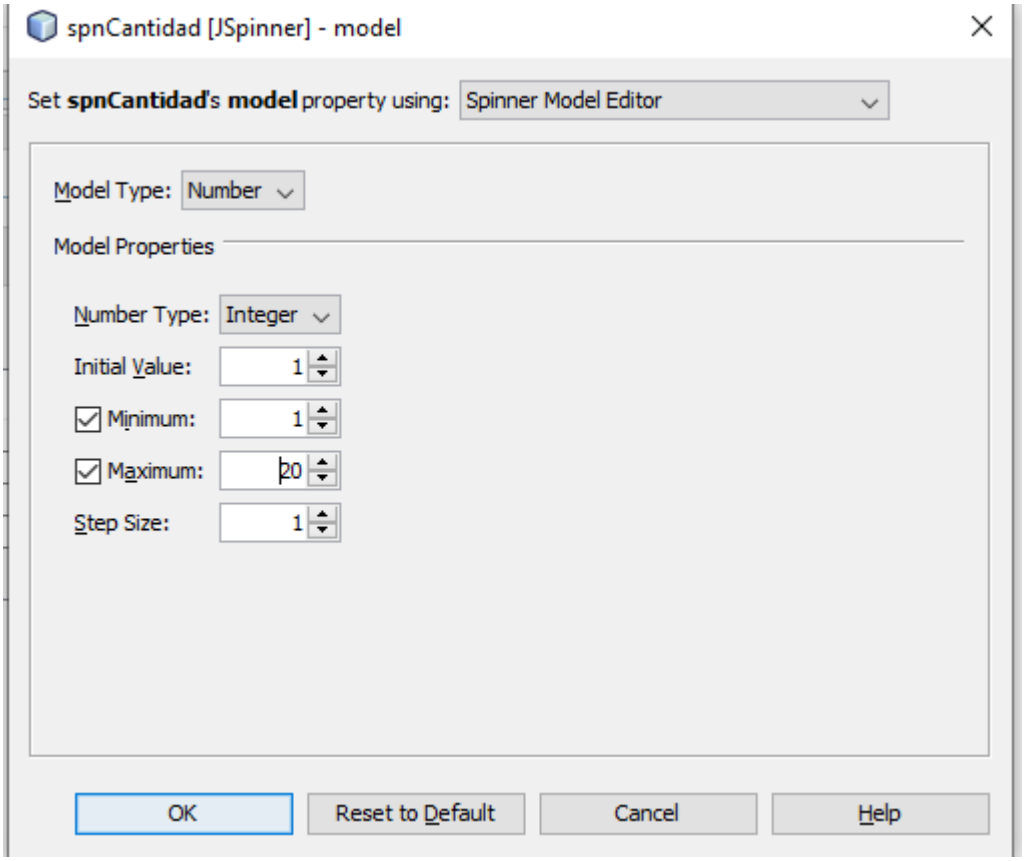
La variable precios debe de coincidir con los precios de cada uno de los productos declarados.

Para rellenar el ComboBox lama a la clase DefaultComboBoxModel y crear el objeto comboModel

```
initComponents();
this.setTitle("Caja 03");
this.setLocationRelativeTo(null);


//CONBOBOX
DefaultComboBoxModel comboModel=new DefaultComboBoxModel(productos);
cnbProductos.setModel(comboModel);
```

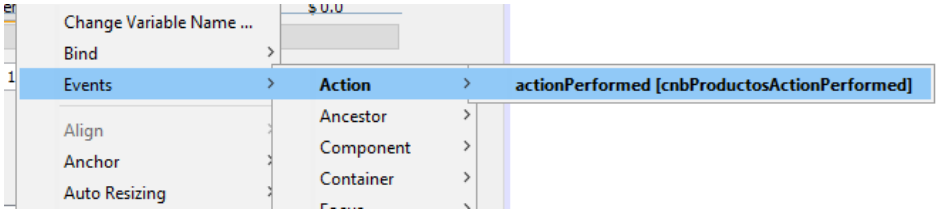
Configura el Spinner con los siguientes valores



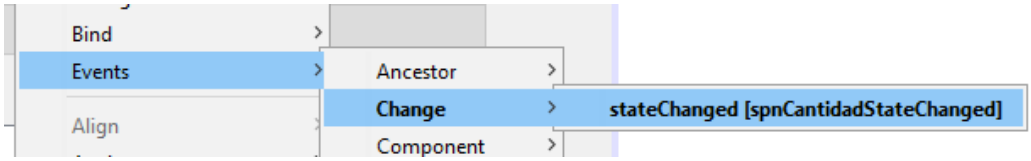


Asigna los siguientes eventos seleccionando el elemento ComboBox y Spinner (cada uno) botón derecho del Mouse

ComboBox



Spinner



Llamar a la clase DefaultTableModel y crear un objeto de la clase para realizar la edición de la tabla.

```
String productos[]={"Jabon 123", "ACEITE"};
double precios[]={34,48};
double precio=0;
int cantidad=0;

DefaultTableModel modelo=new DefaultTableModel();   //tabla
```

Declara las instrucciones para poder modificar la estructura de la tabla

```
//CONBOBOX
DefaultComboBoxModel comboModel=new DefaultComboBoxModel(productos);
cnbProductos.setModel(comboModel);

//Edicion de la tabla //Columnas//
modelo.addColumn("Descripcion");
modelo.addColumn("Precio U");
modelo.addColumn("Cantidad");
modelo.addColumn("Importe");
```

Después agrega una de las estructuras de datos ArrayList, que permite modificar y almacenar dinámicamente una colección de objetos que tomaremos de la Clase Venta para el llenado de la tabla

```
DefaultTableModel modelo=new DefaultTableModel();  //tabla

ArrayList<venta> listaVenta=new ArrayList<venta>();   //arreglo pra rellenar la tabla
```

Programar el botón agregar

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // btn sgregar

    venta venta=new venta();
    venta.setId(cnbProductos.getSelectedIndex());
    venta.setDescripcion(cnbProductos.getSelectedItem().toString());
    venta.setPrecio(precio);
    venta.setCantidad(cantidad);
    venta.setImporte(precio*cantidad);

    listaVenta.add(venta);

    actualizarTabla();
    borrar();

}
```

Programar el método borrar ()

```
public void borrar(){
    precio=0;
    cantidad=1;
    cnbProductos.setSelectedIndex(0);
    spnCantidad.setValue(1);
    calcularPrecio();

}
```

Se crea el método actualizar tabla, para mostrar los elementos agregados y mostrar los valores de subtotal, IVA y Total

```
public void actualizarTabla(){
    while(modelo.getRowCount()>0){
    modelo.removeRow(0);
    }

    double subtotal=0;

    for(venta v: listaVenta){
        Object x[]=new Object[4];
        x[0]=v.getDescripcion();
        x[1]=v.getPrecio();
        x[2]=v.getCantidad();
        x[3]=v.getImporte();

        subtotal+=v.getImporte();

        modelo.addRow(x);
    }

        double iva=subtotal*.016;
        double total=subtotal+iva;


        lblSubtotal.setText(aMoneda(subtotal));
        lblIva.setText(aMoneda(iva));
        lblTotal.setText(aMoneda(total));

        tablaProductos.setModel(modelo);
}
```

Realiza las pruebas necesarias y corrige los posibles errores de lógica de programación o sintaxis que pudieran existir.

¡Éxito

Tú puedes!