



Centro de Investigación en Computación del IPN

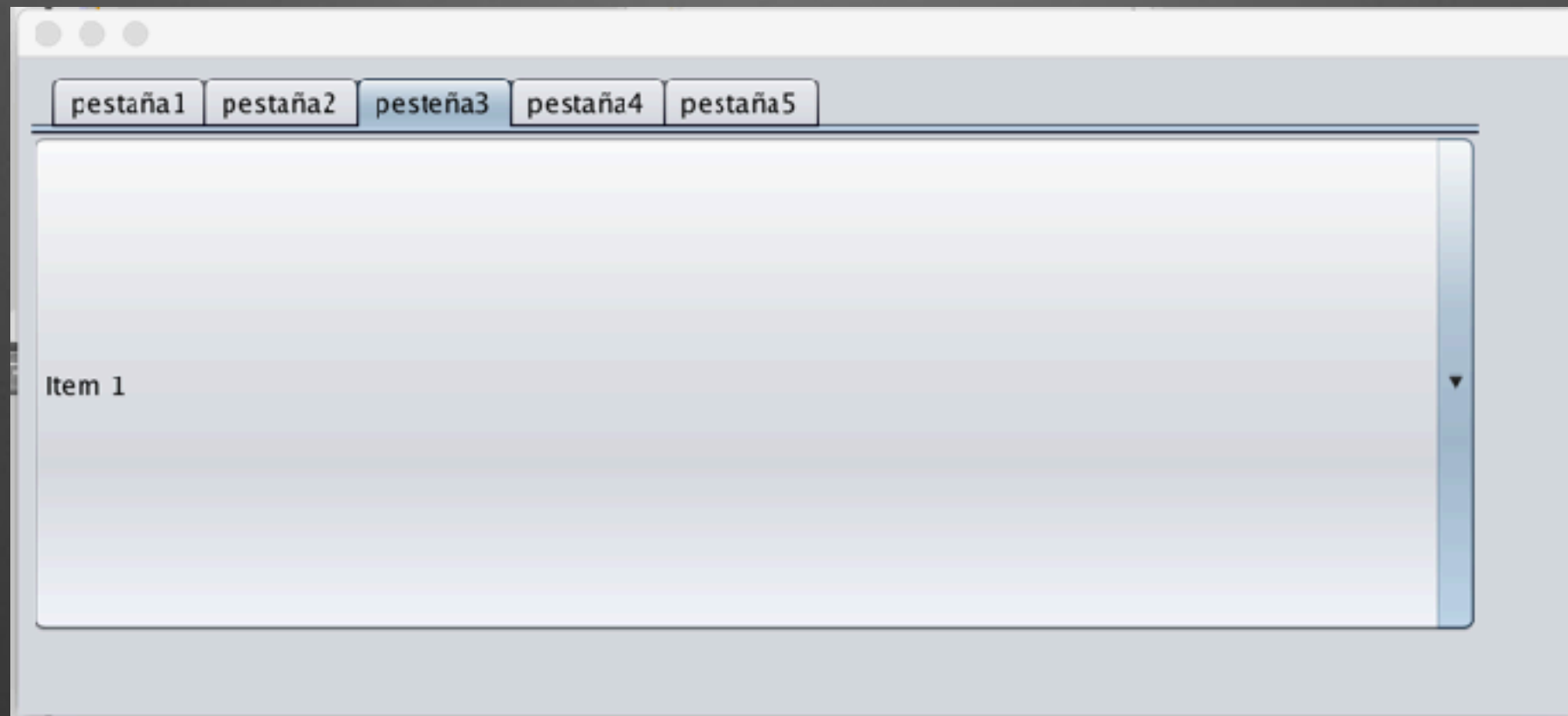
Java – Diplomado

Modulo 2

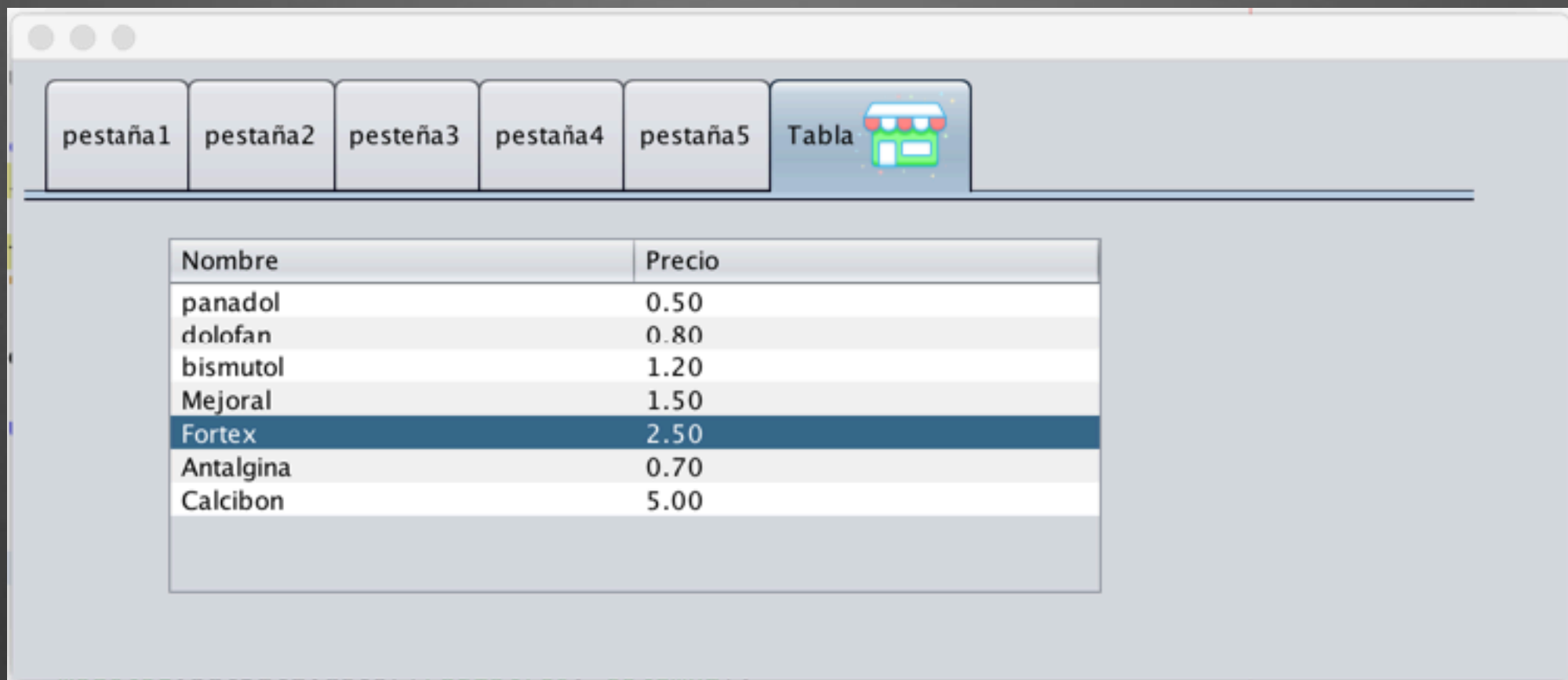
Dra. Erandi Castillo Montiel

Implementar la misma metodología pero para el menú del restaurante

Tabbed Pane



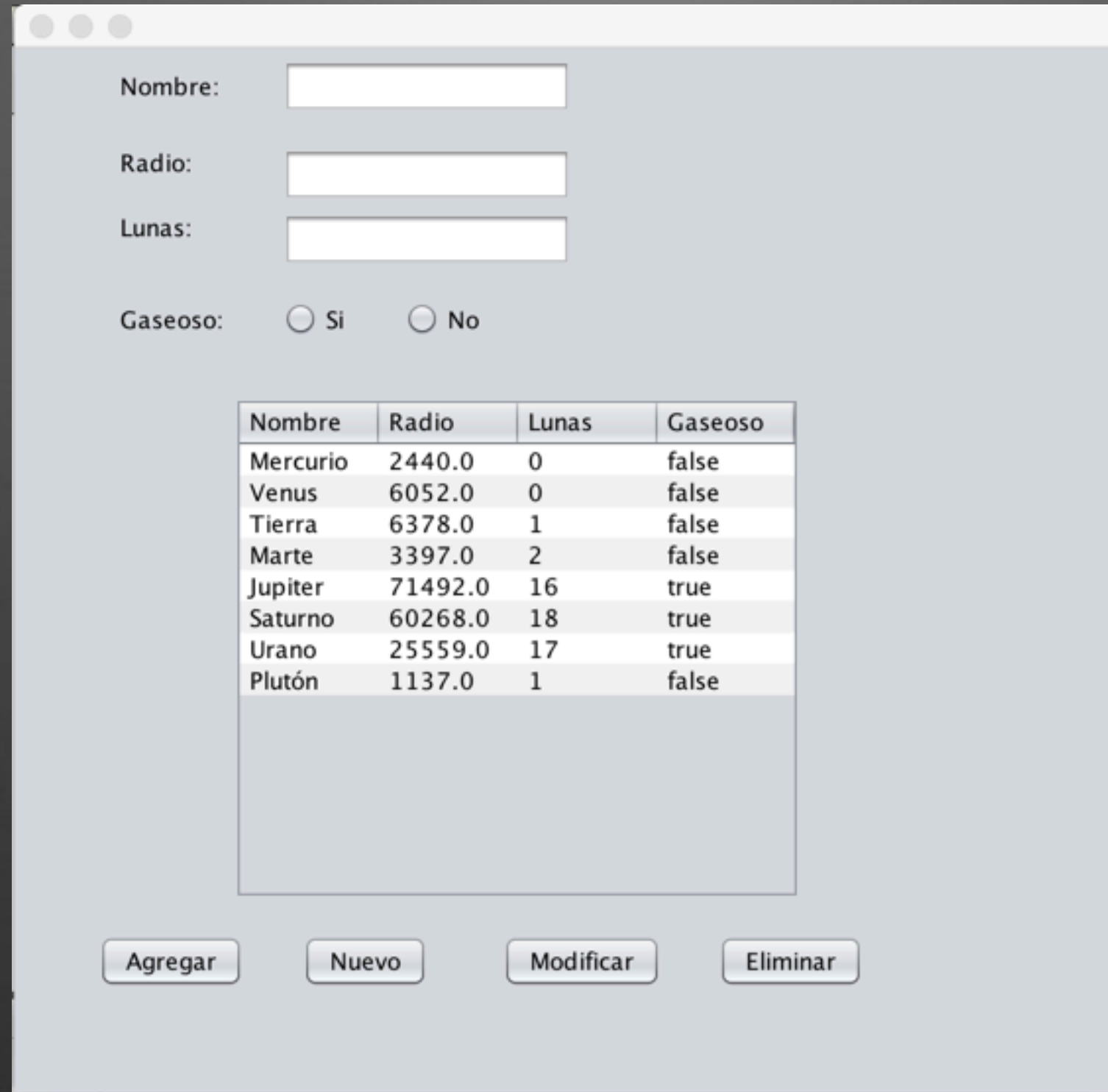
Table



Nombre	Precio
panadol	0.50
dolofan	0.80
bismutol	1.20
Mejoral	1.50
Fortex	2.50
Antalgina	0.70
Calcibon	5.00

```
public class TabbedPanel extends javax.swing.JFrame {  
    String[] columna = {"Nombre", "Precio"};  
  
    String[][] registros = {{"panadol", "0.50"}, {"dolofan", "0.80"}, {"bismutol", "1.20"},  
        {"Mejoral", "1.50"}, {"Fortex", "2.50"}, {"Antalgina", "0.70"}, {"Calcibon", "5.00"} };  
  
    DefaultTableModel modelo1 = new DefaultTableModel();  
  
    public TabbedPanel() {  
        initComponents();  
  
        ImageIcon imagen = new ImageIcon(getClass().getResource("/imagenes/farmacy2.png"));  
        jTabbedPane1.setIconAt(5, imagen);  
        jTable1.setModel(modelo1);  
        //asigna losvalores a la tabla  
        modelo1.setDataVector(registros, columna);  
    }  
}
```

JTable - Planetas



Nombre:

Radio:

Lunas:

Gaseoso: ☐ Si ☐ No

Nombre	Radio	Lunas	Gaseoso
Mercurio	2440.0	0	false
Venus	6052.0	0	false
Tierra	6378.0	1	false
Marte	3397.0	2	false
Jupiter	71492.0	16	true
Saturno	60268.0	18	true
Urano	25559.0	17	true
Plutón	1137.0	1	false

Carga automática de la tabla

En la clase carga

```
public DefaultTableModel cargaTablaPlanetas(){  
    DefaultTableModel model = new DefaultTableModel();  
    String[] columnNames = {"Nombre", "Radio", "Lunas", "Gaseoso"};  
  
    Object[][] data = {  
        {"Mercurio", 2440.0, 0, false},  
        {"Venus", 6052.0, 0, false},  
        {"Tierra", 6378.0, 1, false},  
        {"Marte", 3397.0, 2, false},  
        {"Jupiter", 71492.0, 16, true},  
        {"Saturno", 60268.0, 18, true},  
        {"Urano", 25559.0, 17, true},  
        {"Plutón", 1137.0, 1, false}};  
    //Agrega los datos al modelo  
    model.setDataVector(data, columnNames);  
  
    return model;  
}
```

Proyecto TableUI, cargar tabla al inicio de la ejecución

```
public class TableUI extends javax.swing.JFrame {  
  
    DefaultTableModel modelo1 = new DefaultTableModel();  
    public TableUI() {  
        initComponents();  
        Carga datos = new Carga();  
        modelo1=datos.cargaTablaPlanetas();  
        jTable1.setModel(modelo1);  
    }  
}
```

Crear una clase Planeta :

Constructor

Propiedades: String
nombre, String radio, String
lunas, boolean gaseoso

```
public class Planeta {  
    private String nombre;  
    private String radio;  
    private String lunas;  
    private boolean gaseoso;  
  
    public Planeta(String nombre,String radio,String lunas, boolean gaseoso){  
        this.nombre=nombre;  
        this.lunas=lunas;  
        this.radio=radio;  
        this.gaseoso=gaseoso;  
    }  
}
```


Botón agregar

```
private void btnAgregarActionPerformed(java.awt.event.ActionEvent evt) {  
    boolean gaseoso=false;  
    if(rbtnSi.isSelected()){  
        gaseoso=true;  
    }  
    Planeta nuevo = new Planeta(textNombre.getText(),textRadio.getText(),  
                                textLunas.getText(),gaseoso);  
  
    //agrega fila de datos  
    Object [] datos ={nuevo.getNombre(),nuevo.getRadio(),nuevo.getRadio(),nuevo.isGaseoso()};  
    modelo1.addRow(datos);  
}
```

Botón eliminar

```
private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {  
    //elimina la fila seleccionada  
    modelo1.removeRow(jTable1.getSelectedRow());  
}
```

Botón nuevo

```
private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {  
    //borra los campos y ubica el foco de atención al campo nombre  
    textNombre.setText("");  
    textRadio.setText("");  
    textLunas.setText("");  
    buttonGroup1.clearSelection();  
    //pone el cursor en el campo de nombre  
    textNombre.requestFocus();  
}
```

Botón modificar

```
private void btnModificarActionPerformed(java.awt.event.ActionEvent evt) {  
    int filaSelected = jTable1.getSelectedRow();  
    if(filaSelected >= 0){  
        //si selecciono una fila, envia los datos a los campos de texto y al rbtn  
        textNombre.setText(jTable1.getValueAt(filaSelected, 0).toString());  
        textRadio.setText(jTable1.getValueAt(filaSelected, 1).toString());  
        textLunas.setText(jTable1.getValueAt(filaSelected, 2).toString());  
        if(jTable1.getValueAt(filaSelected, 3).equals(true)){  
            rbtnSi.setSelected(true);  
        }else rbtnNo.setSelected(true);  
        //se elimina del modelo para ser ingresado nuevamente  
        modelo1.removeRow(filaSelected);  
    }else{  
        JOptionPane.showMessageDialog(this, "No hay fila seleccionada");  
    }  
}
```

Proyecto Farmacia


Busqueda

Cuenta

Busqueda producto

BusquedaAñadir

Nombre	Precio
panadol	0.50
dolofan	0.80
bismutol	1.20
Mejoral	1.50
Fortex	2.50
Antalgina	0.70
Calcibon	5.00



Busqueda

Cuenta

Nombre	Precio	Cantidad	Total
bismutol	1.20	3	3.6
Antalgina	0.70	2	1.4
dolofan	0.80	1	0.8

Calcular
Eliminar
Salir

Total
5.8

Propiedades requeridas

```
public class Principal extends javax.swing.JFrame {  
    //datos de la primera tabla  
    String[] columna = {"Nombre", "Precio"};  
    String[][] registros = {{"panadol", "0.50"}, {"dolofan", "0.80"}, {"bismutol", "1.20"},  
        {"Mejoral", "1.50"}, {"Fortex", "2.50"}, {"Antalgina", "0.70"}, {"Calcibon", "5.00"} };  
    //columna de la segunda tabla  
    String []col= {"Nombre", "Precio","Cantidad","Total"};  
  
    //modelo que contendra los datos de la primera tabla  
    DefaultTableModel modelo1 = new DefaultTableModel();  
    //ordenador de filas para la tabla  
    TableRowSorter sorter = new TableRowSorter(modelo1);  
  
    //datos de la segunda tabla  
    DefaultTableModel modelo2 = new DefaultTableModel(); //para el otra pestalla
```

Inicialización de componentes

```
public Principal() {  
    initComponents();  
    //agrega imagen a la pestaña  
    ImageIcon imagen = new ImageIcon(getClass().getResource("/imagenes/farmacy2.png"));  
    ImageIcon tamaño = new ImageIcon(imagen.getImage().getScaledInstance(5, 5, 1));  
    jTable1.setIconAt(0, imagen); //número de pestaña  
    jTable1.setIconAt(1, imagen);  
  
    //añade datos al modelo  
    modelo1.setDataVector(registros, columna);  
    //indicar que el modelo lo debe usar la tabla  
    jTable1.setModel(modelo1);  
    //indicar que el ordenador de filas debe ser usado en la tabla  
    jTable1.setRowSorter(sorter);  
  
    //asigna el modelo y las columnas a la tabla 2  
    jTable2.setModel(modelo2);  
    modelo2.setColumnIdentifiers(col);  
}
```

Evento mouseclicked - selección de la imagen

```
private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {  
    int i = jTable1.getSelectedRow();  
    String producto = jTable1.getValueAt(i,0).toString(); //nombre del producto (fila, columna)  
    ImageIcon icon = new ImageIcon (getClass().getResource("/imagenes/"+producto+".png"));  
    jLabel2.setIcon(icon);  
}
```

Botón búsqueda - filtro

```
private void btnBusquedaActionPerformed(java.awt.event.ActionEvent evt) {  
    //busqueda por la columna, nombre del medicamento  
    RowFilter filtro = RowFilter.regexFilter(textBusqueda.getText(), 0);  
    sorter.setRowFilter(filtro);  
}
```


Botón añadir - Añadir a la lista de compra

```
private void btnAñadirActionPerformed(java.awt.event.ActionEvent evt) {  
    //agrega elementos al carrito  
    //captura la fila seleccionada (producto y precio)  
    // fila, columna  
    String [] datos ={jTable1.getValueAt(jTable1.getSelectedRow(),0).toString(),  
jTable1.getValueAt(jTable1.getSelectedRow(),1).toString()  
};  
    //paso la fila seleccionada a la tabla 2  
    modelo2.addRow(datos);  
}
```

Botón eliminar - Eliminar a la lista de compra

```
private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {  
    modelo2.removeRow(jTable2.getSelectedRow());  
}
```


Botón Calcular

```
private void btnCalcularActionPerformed(java.awt.event.ActionEvent evt) {  
    //calcula lo que hay que pagar  
    double precio;  
    int cantidad;  
    double total=0, total1=0;  
    DecimalFormat df = new DecimalFormat("#.##");  
    for (int i = 0; i < jTable2.getRowCount(); i++) { //recorre la tabla2  
        //getValueAt(fila,columna)  
        precio=Double.parseDouble(jTable2.getValueAt(i,1).toString());  
        cantidad =Integer.parseInt(jTable2.getValueAt(i,2).toString());  
        total=cantidad*precio;  
        //manda el total a la celda de total  
        jTable2.setValueAt(df.format(total), i,3);  
  
        total1+=total;  
    }  
    labelTotal.setText(String.valueOf(df.format(total1)));  
}
```

Botón Salir

```
private void btnSalirActionPerformed(java.awt.event.ActionEvent evt) {  
    System.exit(0);  
}
```

- **Mismo proyecto pero con el menú del restaurante**

Hilos

Proyecto Cajera

```
public static void main(String[] args) {  
    Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2, 3 });  
    Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1 });  
  
    Cajera cajera1 = new Cajera("Cajera 1");  
    Cajera cajera2 = new Cajera("Cajera 2");  
  
    // Tiempo inicial de referencia  
    long initialTime = System.currentTimeMillis();  
  
    cajera1.procesarCompra(cliente1, initialTime);  
    cajera2.procesarCompra(cliente2, initialTime);  
}
```

Clase Cliente

Propiedades: String nombre, int [] carroCompra;

Constructor para inicializar los valores

```
public class Cliente {  
  
    private String nombre;  
    private int[] carroCompra;  
  
    // Constructor, getter y setter  
    public Cliente(String nombre, int [] carroCompra){  
        this.nombre=nombre;  
        this.carroCompra=carroCompra;  
    }  
}
```

Clase cajera

```
public class Cajera {  
    private String nombre;  
    public Cajera(String nombre){  
        this.nombre=nombre;  
    }  
    public void procesarCompra(Cliente cliente, long timeStamp) {  
        System.out.println("La cajera " + this.nombre +  
            " COMIENZA A PROCESAR LA COMPRA DEL CLIENTE " + cliente.getNombre() +  
            " EN EL TIEMPO: " + (System.currentTimeMillis() - timeStamp) / 1000 +  
            "seg");  
  
        for (int i = 0; i < cliente.getCarroCompra().length; i++) {  
            this.esperarXsegundos(cliente.getCarroCompra()[i]);  
            System.out.println("Procesado el producto " + (i + 1) +  
                " ->Tiempo: " + (System.currentTimeMillis() - timeStamp) / 1000 +  
                "seg");  
        }  
  
        System.out.println("La cajera " + this.nombre + " HA TERMINADO DE PROCESAR " +  
            cliente.getNombre() + " EN EL TIEMPO: " +  
            (System.currentTimeMillis() - timeStamp) / 1000 + "seg");  
    }  
}
```

Método esperarXSegundos en la clase Cajera

```
private void esperarXsegundos(int segundos) {  
    try {  
        Thread.sleep(segundos * 1000);    //espera los segundos para  
    } catch (InterruptedException ex) {  
        Thread.currentThread().interrupt();  
    }  
}
```


main del proyecto

```
public class Hilos {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2, 3 });  
        Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1 });  
  
        Cajera cajera1 = new Cajera("Cajera 1");  
        Cajera cajera2 = new Cajera("Cajera 2");  
  
        // Tiempo inicial de referencia  
        long initialTime = System.currentTimeMillis();  
  
        cajera1.procesarCompra(cliente1, initialTime);  
        cajera2.procesarCompra(cliente2, initialTime);  
    }  
}
```

Con hilos

Clase principal

```
public class MainThreads {  
    public static void main(String[] args) {  
        Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2, 3 });  
        Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1 });  
  
        // Tiempo inicial de referencia  
        long initialTime = System.currentTimeMillis();  
        CajeraHilos cajera1 = new CajeraHilos("Cajera 1", cliente1, initialTime);  
        CajeraHilos cajera2 = new CajeraHilos("Cajera 2", cliente2, initialTime);  
  
        cajera1.start();  
        cajera2.start();  
    }  
}
```

Cajera con hilos

```
public class CajeraHilos extends Thread {  
    private String nombre;  
    private Cliente cliente;  
    private long initialTime;  
  
    // Constructor, getter & setter  
  
    public CajeraHilos(String nombre, Cliente cliente, long initialTime){  
        this.nombre=nombre;  
        this.cliente=cliente;  
        this.initialTime=initialTime;  
    }  
}
```

Método run

```
@Override
public void run() {

    System.out.println("La cajera " + this.nombre + " COMIENZA A PROCESAR LA COMPRA DEL CLIENTE "
        + this.cliente.getNombre() + " EN EL TIEMPO: "
        + (System.currentTimeMillis() - this.initialTime) / 1000
        + "seg");

    for (int i = 0; i < this.cliente.getCarroCompra().length; i++) {
        this.esperarXsegundos(cliente.getCarroCompra()[i]);
        System.out.println("Procesado el producto " + (i + 1)
            + " del cliente " + this.cliente.getNombre() + "->Tiempo: "
            + (System.currentTimeMillis() - this.initialTime) / 1000
            + "seg");
    }

    System.out.println("La cajera " + this.nombre + " HA TERMINADO DE PROCESAR "
        + this.cliente.getNombre() + " EN EL TIEMPO: "
        + (System.currentTimeMillis() - this.initialTime) / 1000
        + "seg");
}
```

Con hilos 2da Versión

```
public class MainThreads2 implements Runnable{  
  
    private Cliente cliente;  
    private Cajera cajera;  
    private long initialTime;  
  
    public MainThreads2 (Cliente cliente, Cajera cajera, long initialTime){  
        this.cajera = cajera;  
        this.cliente = cliente;  
        this.initialTime = initialTime;  
    }  
}
```

Main

```
public static void main(String[] args) {

    Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2, 3 });
    Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1 });

    Cajera cajera1 = new Cajera("Cajera 1");
    Cajera cajera2 = new Cajera("Cajera 2");

    // Tiempo inicial de referencia
    long initialTime = System.currentTimeMillis();

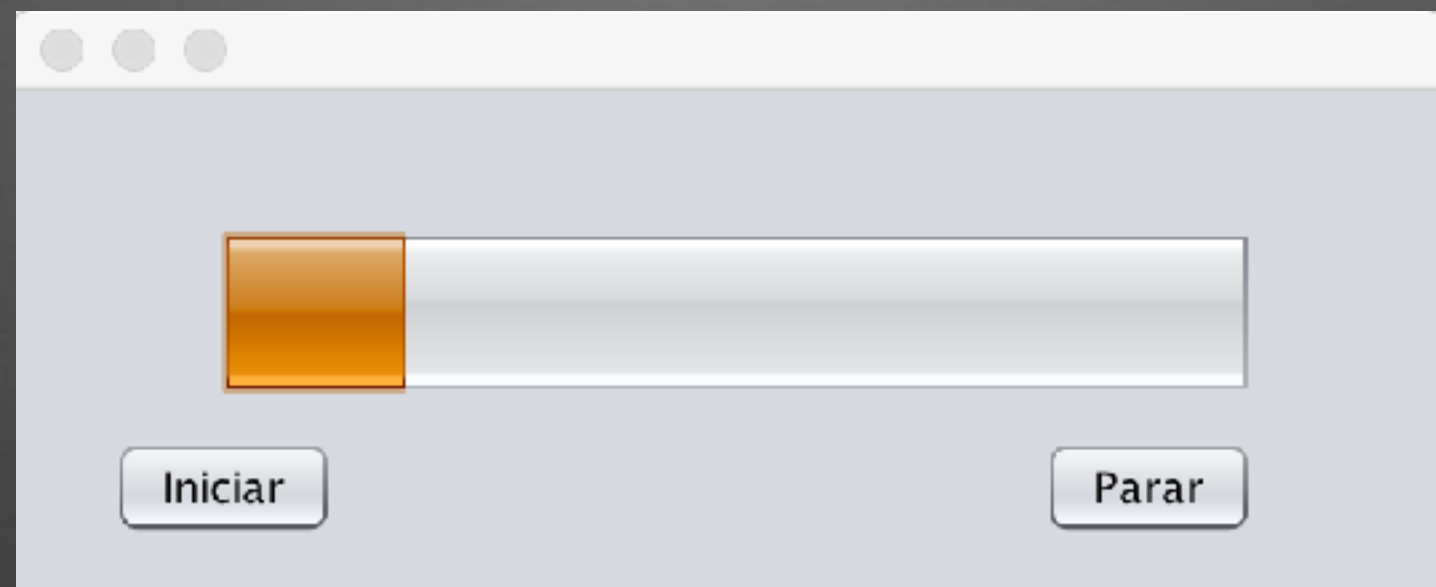
    Runnable proceso1 = new MainThreads2(cliente1, cajera1, initialTime);
    Runnable proceso2 = new MainThreads2(cliente2, cajera2, initialTime);

    new Thread(proceso1).start();
    new Thread(proceso2).start();

}

@Override
public void run() {
    this.cajera.procesarCompra(this.cliente, this.initialTime);
}
```


Progress Bar



Manejador del progressBar

```
*/
public class pbHandler extends Thread{
    //propiedades
    private boolean life;
    private int order;
    private int prog;
    private javax.swing.JProgressBar progBar;

    //constructor
    public pbHandler(Object in){
        this.progBar = (javax.swing.JProgressBar) in;
        this.life=true;
        this.prog=0;
    }

    public void kill(){
        this.life=false;
    }

    public void cmd (int in){ //setter asigna la orden
        this.setOrder(in);
    }
}
```

Método run()

```
public void run(){
    while(life){
        if(order==1){
            System.out.println("Hilo esta corriendo .." + this.getProg());
            this.setProg(this.getProg() + 1);
            this.progBar.setValue(this.getProg());
            try{
                Thread.sleep(125);
            }catch (Exception e){
                e.printStackTrace();
                System.out.println("Error");
            }
        }
        if(this.getProg()==100){
            this.life=false;
            System.out.println("Hilo muerto");
        }
    }
}
```

Bóton iniciar

```
private void btnIniciarMouseClicked(java.awt.event.MouseEvent evt) {  
  
    try{  
        this.pbHd.start();  
  
    }catch (Exception e){  
        System.out.println("Error al iniciar el hilo");  
    }  
    try{  
        this.pbHd.cmd(1);  
    }catch (Exception e){  
        e.printStackTrace();  
        System.out.println("Error al iniciar el comando");  
    }  
}
```

Botón parar, evento al iniciar la ventana y obtener el elemento de barra

```
private void btnPararMouseClicked(java.awt.event.MouseEvent evt) {  
    try{  
        this.pbHd.cmd(0);  
    }catch(Exception e){  
        e.printStackTrace();  
        System.out.println("Error al parar el hilo");  
    }  
}  
  
private void formWindowOpened(java.awt.event.WindowEvent evt) {  
    this.pbHd = new pbHandler(this.jProgressBar1);  
    System.out.println("main Windows: todos los componentes cargados");  
}  
  
public Object getjProgressBar1(){  
    return this.jProgressBar1;  
}
```