

Punto de venta



Universidad de la Sierra Sur

Docendo Discimus

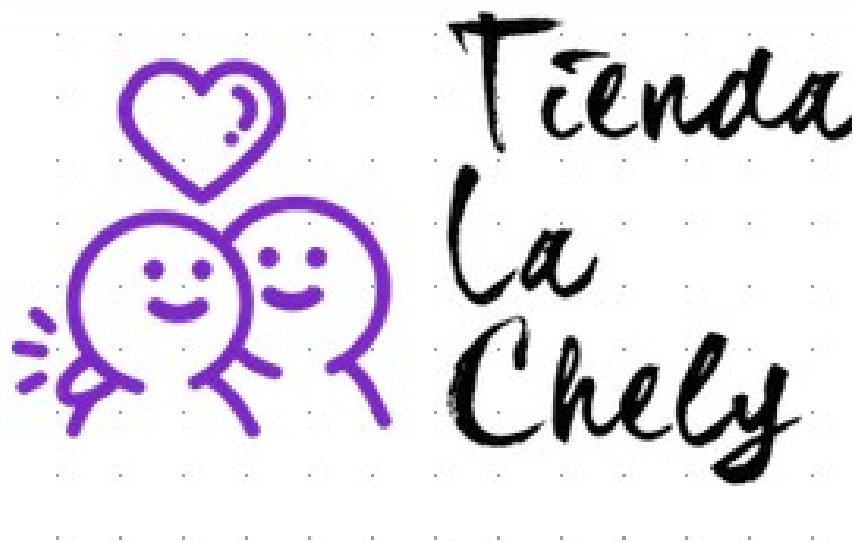
Paradigmas de programación

licenciatura en Informática

América Yaridsaida Villalobos Rodríguez

Trabajo final

Punto de venta



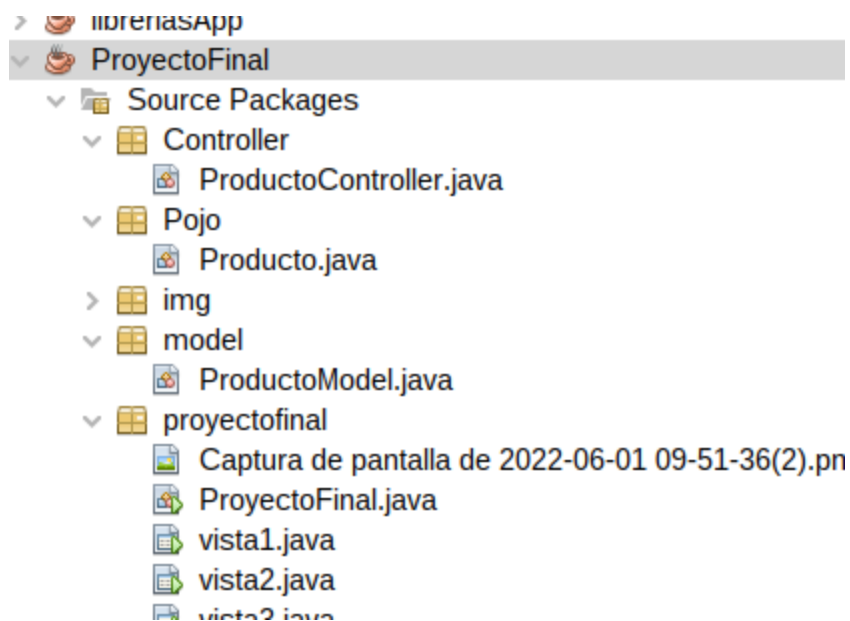
Punto de venta

objetivo realizar un programa que realice un punto de ventas, en este caso será una tienda de abarrotes.

Primero se identifican las clases, en esta programa solo será una clase la cual es productos mejor conocido como uml

Producto
-id:String -nombre:String -precio:String
+set id (id String): void +get id();String +set nombre (nombre String): void +get nombre();String +set precio (precio String): void +get precio();String

Lo que se hizo fue crear una lista de productos para que sea más fácil su elaboración las operaciones del programa se dividieron con el método Modelo-Vista-Controlador



Para poder separarlos hace primero se tiene que hacer un proyecto donde se introduzcan los paquetes con sus repetidas clases, la finalidad del modelo es que sé a más fácil programar

Punto de venta

para tener mejores resultados también hay un paquete que se llama proyecto final que es donde está la vista del programa

Pojo Producto: es la clase más simple

```
public class Producto {  
  
    private String id;  
    private String nombre;  
    private String precio;  
  
    public Producto() {  
    }  
  
    public Producto(String id, String nombre, String precio) {  
        super();  
        this.id = id;  
        this.nombre = nombre;  
        this.precio = precio;  
    }  
  
    public String getId() {  
        return id;  
    }  
  
    public void setId(String id) {  
        this.id = id;  
    }  
  
    public String getNombre() {  
        return nombre;  
    }  
  
    public void setNombre(String nombre) {  
        this.nombre = nombre;  
    }  
}
```

Punto de venta

```
public String getPrecio() {  
    return precio;  
}
```

```
public void setPrecio(String precio) {  
    this.precio = precio;  
}
```

clase ModelProducto

```
public class ProductoModel {
```

//5 operaciones CRUD

se crea la lista de productoss

```
public void crearProducto(List<Producto> lista, Producto producto){  
    lista.add(producto);
```

```
}
```

en este codigo se puede eliminar los productos

```
public void eliminarProducto(List<Producto> lista, String id){
```

```
    for (int i = 0; i < lista.size(); i++){
```

```
        if(lista.get(i).getId().compareTo(id)==0){
```

```
            lista.remove(i);
```

```
            break;
```

```
        }
```

```
    }
```

```
}
```

```
public void actualizarProductos(List<Producto> lista, Producto producto){
```

```
    for (int i =0; i < lista.size(); i++){
```

```
        if(lista.get(i).getId().compareTo(producto.getId())==0){
```

```
            lista.set(i, producto);
```

```
            break;
```

```
        }
```

```
    }
```

```
}
```

```
public void mostrarProductos (List <Producto> lista, DefaultTableModel modelo){
```

```
    modelo.setRowCount(0);
```

Punto de venta

```
for(int i =0; i < lista.size(); i++) {  
    Object []fila = new Object[2];  
  
    fila[0] = lista.get(i) .getId();  
    fila[1] = lista.get(i) .getNombre();  
    fila[2] = lista.get(i) .getPrecio();  
  
    modelo.addRow(fila);  
}  
}
```

Clase ControllerProducto

// se instancia al Producto model

```
ProductoModel model = new ProductoModel();  
  
public void crearProducto(List<Producto> lista, Producto producto){  
    model.crearProducto(lista, producto);  
}  
  
public void eliminarProducto(List<Producto> lista, String id){  
    model.eliminarProducto(lista, id);  
}  
  
public void actualizarProductos(List<Producto> lista, Producto producto){  
    model.actualizarProductos(lista, producto);  
}  
public void mostrarProductos (List <Producto> lista, DefaultTableModel modelo){  
    model.mostrarProductos(lista, modelo);  
}
```

Ya que se terminó de ejecutar el código, lo siguiente que se realizó fue hacer los modelos de baja fidelidad para la aplicación



Punto de venta

Ya que se termino de ejecutar el codigo lo siguiente que se realizo fue hacer los modelos de baja fidelidad para la aplicación
La vista principal del programa que consta de una imagen que el logo de la tienda con dos botones el primero es el de productos y el segundo de compras

Compras

ID	Nombre	Precio

Añadir al carrito

Total

Pagar

Cambio

100 x 50

presionado el boton producto nos debe de llevar a la vista 2

Compras

ID	Nombre	Precio

Añadir al carrito

Total

Pagar

Cambio

100 x 50

Punto de venta

La vista 3 es donde se puede realizar la compra, tendrá una tabla y contendrá dos botones, uno para seleccionar los productos para comprar y el otro para pagar el código para hacer las compras, es este:

Código para añadir al carrito

```
if (jTable1.getSelectedRow() > -1) {  
    double precio = Double.parseDouble(jTable1.getValueAt(jTable1.getSelectedRow(),  
2).toString());  
    this.suma = suma + precio;  
    jTextField1.setText(String.valueOf(suma));  
}else{
```

```
    JOptionPane.showMessageDialog(null, "Selecciona un registro de la tabla.");  
}con este código se puede seleccionar el producto seleccionado y se hace la suma de  
los que seleccionados
```

código para dar cambio

```
double pagar = Double.parseDouble(this.pagar.getText());
```

```
    cambio.setText(String.valueOf(pagar – suma));
```

código para dar cambio resta la suma del producto con la cantidad ingresada por el usuario

Para finalizar la vista del trabajo terminaría así:



Compras

Id	Nombre	Precio
----	--------	--------

Añadir al carrito

total

Pagar

cambio

SourceDesignHistory


Productos


Id

Nombre

Precio

Id	Nombre	Precio
----	--------	--------

Modificar

Eliminar

Agregar