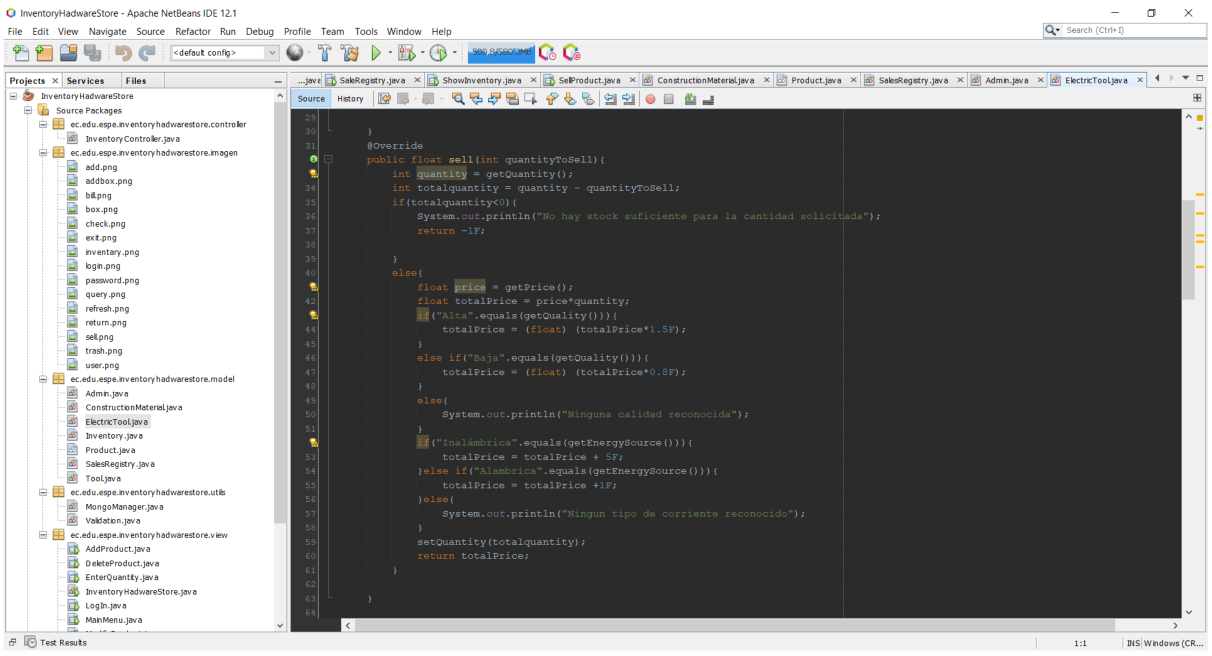
SOLID of team 5 - inventory Hardware Store

Single-responsibility principle

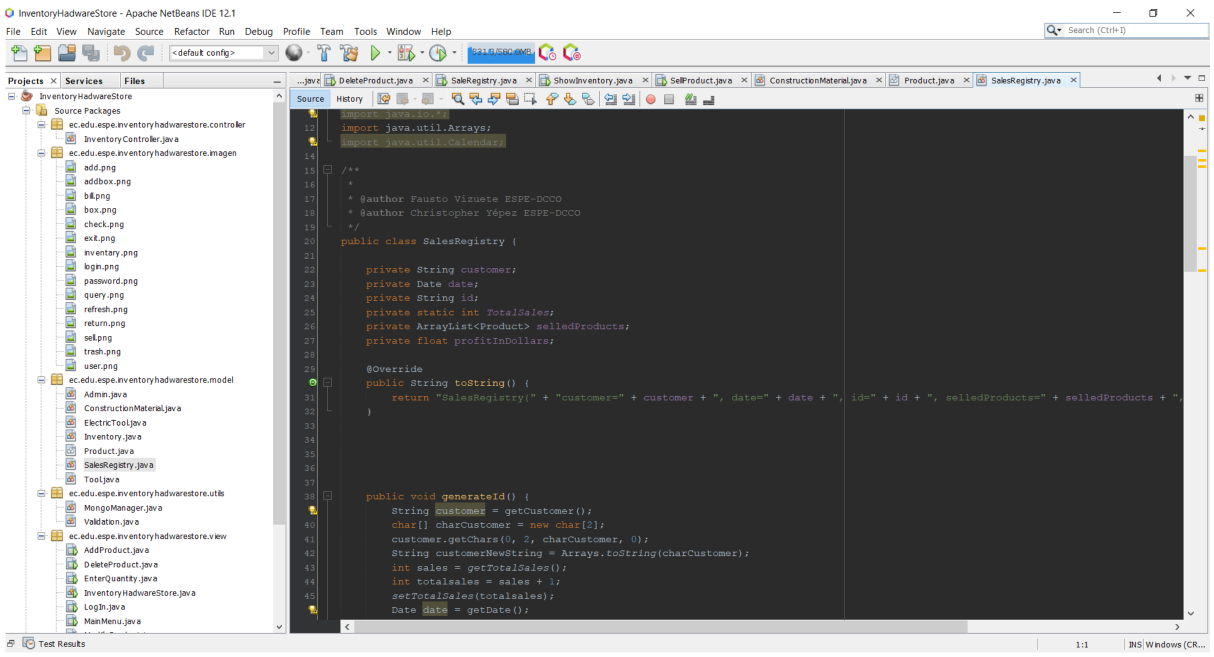
Model Package

- In the class electric tool there should only be a way to save the item, but there is also a validation about the quality of the product.



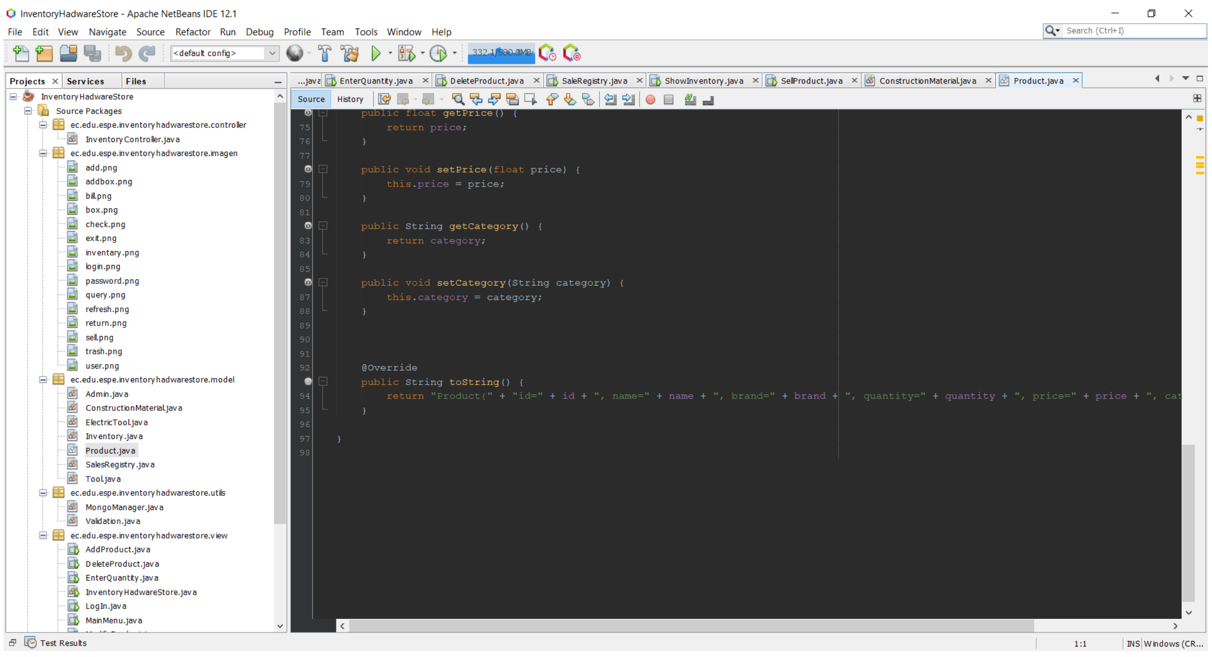
```java

    }
    @Override
    public float sell(int quantityToSell){
        int quantity = getQuantity();
        int totalquantity = quantity - quantityToSell;
        if(totalquantity<0){
            System.out.println("No hay stock suficiente para la cantidad solicitada");
            return -1F;

        }
        else{
            float price = getPrice();
            float totalPrice = price*quantity;
            if("Alta".equals(getQuality())){
                totalPrice = (float) (totalPrice*1.5F);
            }
            else if("Baja".equals(getQuality())){
                totalPrice = (float) (totalPrice*0.8F);
            }
            else{
                System.out.println("Ninguna calidad reconocida");
            }
            if("Inalámbrica".equals(getEnergySource())){
                totalPrice = totalPrice + 5F;
            }else if("Alambrica".equals(getEnergySource())){
                totalPrice = totalPrice +1F;
            }else{
                System.out.println("Ningun tipo de corriente reconocido");
            }
            setQuantity(totalquantity);
            return totalPrice;
        }

    }

```

- In class SalesRegistry the Constructor must go before the ToString



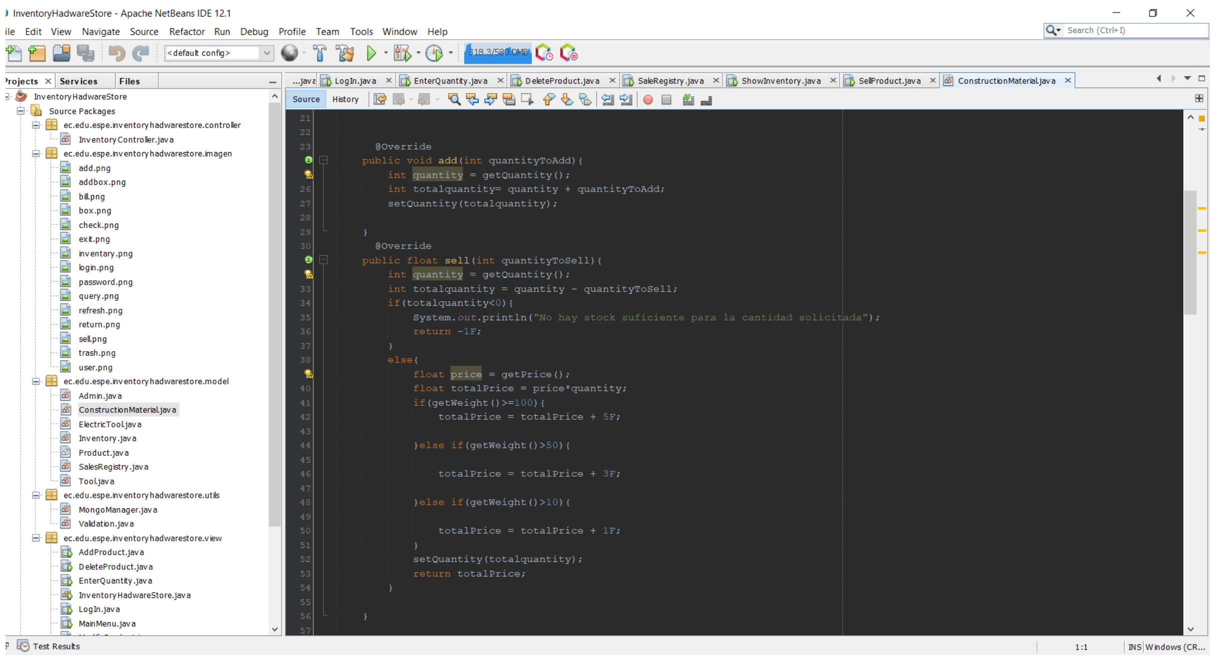
```java
import java.io.*;
import java.util.Arrays;
import java.util.Calendar;

/**
 *
 * @author Fausto Vizuete ESPE-DCCO
 * @author Christopher Yépez ESPE-DCCO
 */
public class SalesRegistry {

    private String customer;
    private Date date;
    private String id;
    private static int TotalSales;
    private ArrayList<Product> selledProducts;
    private float profitInDollars;

    @Override
    public String toString() {
        return "SalesRegistry{" + "customer=" + customer + ", date=" + date + ", id=" + id + ", selledProducts=" + selledProducts + ",
    }




    public void generateId() {
        String customer = getCustomer();
        char[] charCustomer = new char[2];
        customer.getChars(0, 2, charCustomer, 0);
        String customerNewString = Arrays.toString(charCustomer);
        int sales = getTotalSales();
        int totalsales = sales + 1;
        setTotalSales(totalsales);
        Date date = getDate();
```

- In class Product the ToString must go before the getters and setters



```java
    public float getPrice() {
        return price;
    }

    public void setPrice(float price) {
        this.price = price;
    }

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }



    @Override
    public String toString() {
        return "Product{" + "id=" + id + ", name=" + name + ", brand=" + brand + ", quantity=" + quantity + ", price=" + price + ", cat
    }

}
```

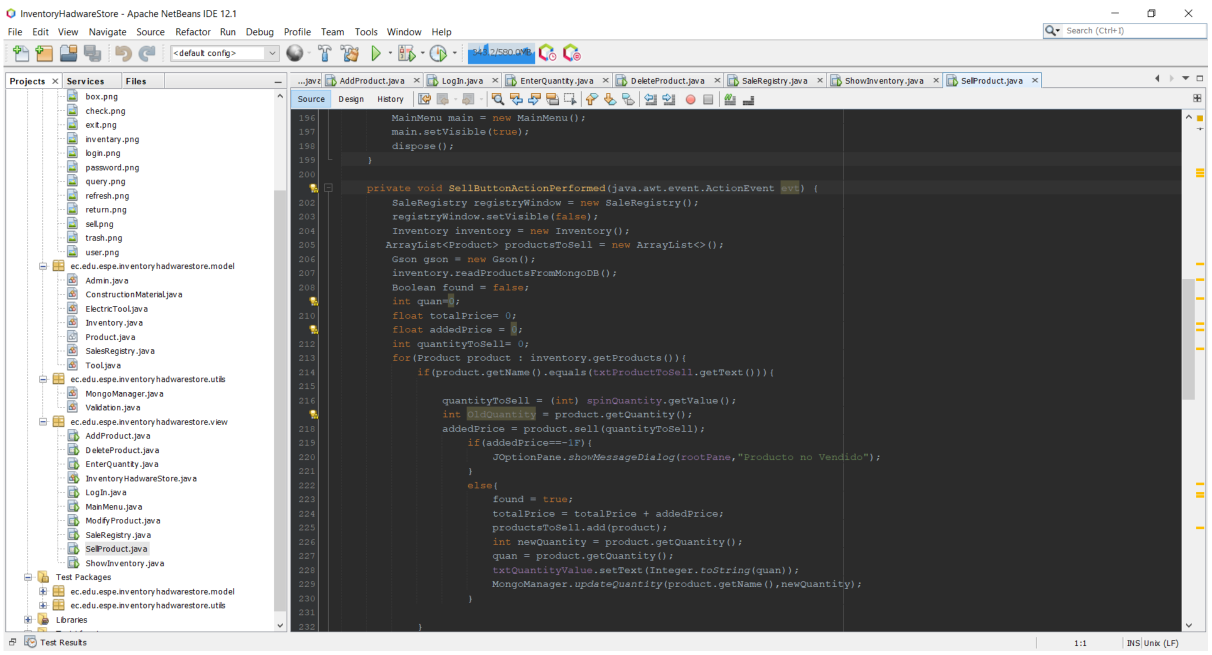
- The class ConstructionMaterial should not sell the products, the class should just create the information itself.



```java


     @Override
    public void add(int quantityToAdd){
        int quantity = getQuantity();
        int totalquantity= quantity + quantityToAdd;
        setQuantity(totalquantity);

    }
     @Override
    public float sell(int quantityToSell){
        int quantity = getQuantity();
        int totalquantity = quantity - quantityToSell;
        if(totalquantity<0){
            System.out.println("No hay stock suficiente para la cantidad solicitada");
            return -1F;
        }
        else{
            float price = getPrice();
            float totalPrice = price*quantity;
            if(getWeight()>=100){
                totalPrice = totalPrice + 5F;

            }else if(getWeight()>50){

                totalPrice = totalPrice + 3F;

            }else if(getWeight()>10){

                totalPrice = totalPrice + 1F;
            }
            setQuantity(totalquantity);
            return totalPrice;
        }

    }
```

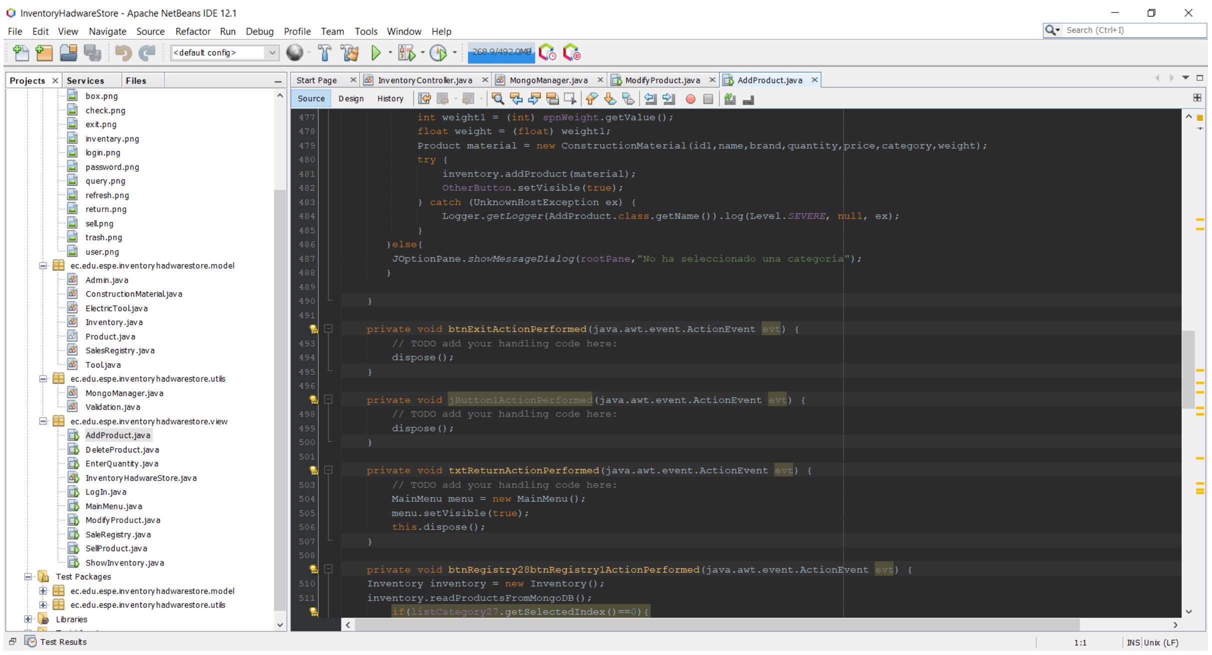
View Package

- The Gson object is found in some classes and do not use methods in each of its buttons



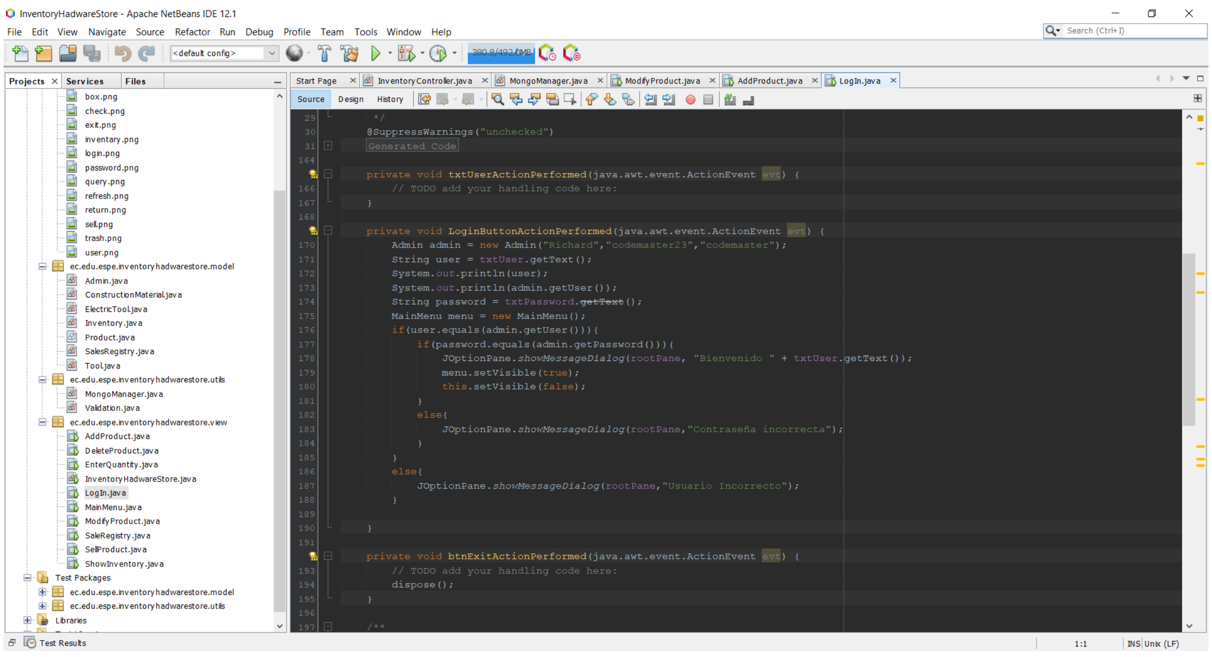
```java
            MainMenu main = new MainMenu();
            main.setVisible(true);
            dispose();
        }

    private void SellButtonActionPerformed(java.awt.event.ActionEvent evt) {
            SaleRegistry registryWindow = new SaleRegistry();
            registryWindow.setVisible(false);
            Inventory inventory = new Inventory();
           ArrayList<Product> productsToSell = new ArrayList<>();
            Gson gson = new Gson();
            inventory.readProductsFromMongoDB();
            Boolean found = false;
            int quan=0;
            float totalPrice= 0;
            float addedPrice = 0;
            int quantityToSell= 0;
            for(Product product : inventory.getProducts()){
                if(product.getName().equals(txtProductToSell.getText())){

                    quantityToSell = (int) spinQuantity.getValue();
                    int OldQuantity = product.getQuantity();
                    addedPrice = product.sell(quantityToSell);
                        if(addedPrice==-1F){
                            JOptionPane.showMessageDialog(rootPane,"Producto no Vendido");
                        }
                        else{
                            found = true;
                            totalPrice = totalPrice + addedPrice;
                            productsToSell.add(product);
                            int newQuantity = product.getQuantity();
                            quan = product.getQuantity();
                            txtQuantityValue.setText(Integer.toString(quan));
                            MongoManager.updateQuantity(product.getName(),newQuantity);
                        }

                }
```

- AddProduct.In this class you have programmed in the buttons do not work with methods, in the design of the FRM you have a double screen that is not functional.



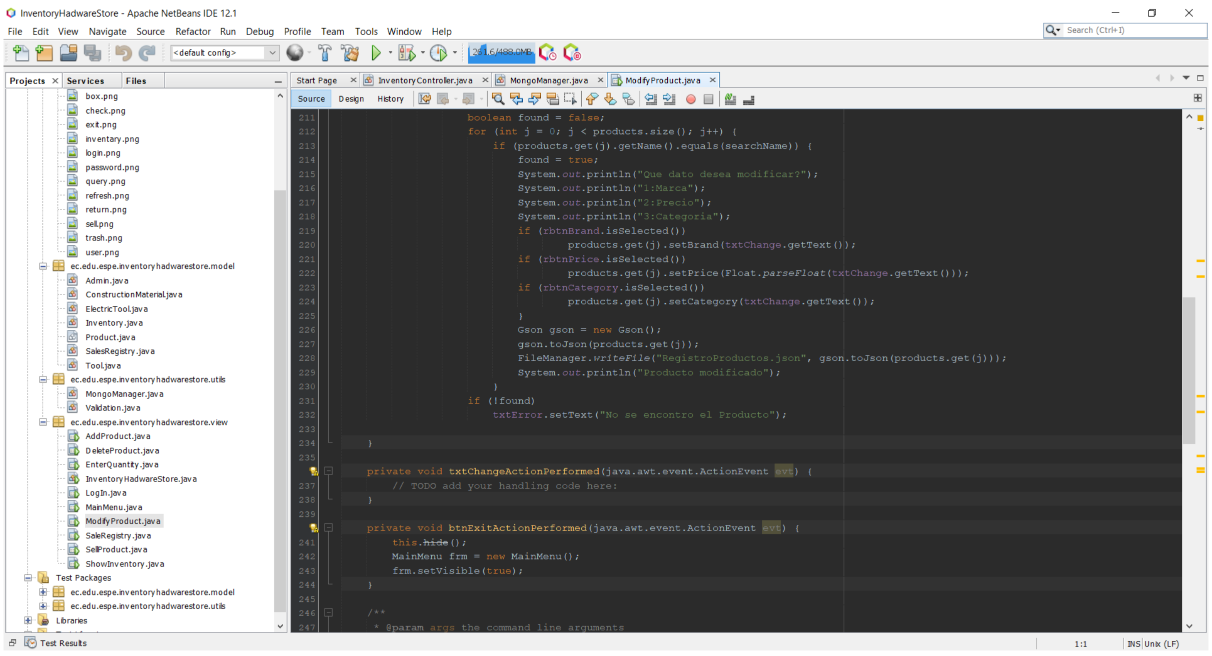
```java
                int weight1 = (int) spnWeight.getValue();
                float weight = (float) weight1;
                Product material = new ConstructionMaterial(id1,name,brand,quantity,price,category,weight);
                try {
                    inventory.addProduct(material);
                    OtherButton.setVisible(true);
                } catch (UnknownHostException ex) {
                    Logger.getLogger(AddProduct.class.getName()).log(Level.SEVERE, null, ex);
                }
            }else{
             JOptionPane.showMessageDialog(rootPane,"No ha seleccionado una categoría");
            }

        }

    private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        dispose();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        dispose();
    }

    private void txtReturnActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        MainMenu menu = new MainMenu();
        menu.setVisible(true);
        this.dispose();
    }

    private void btnRegistry28btnRegistry1ActionPerformed(java.awt.event.ActionEvent evt) {
    Inventory inventory = new Inventory();
    inventory.readProductsFromMongoDB();
        if(listCategory27.getSelectedIndex()==0){
```

- Logging in this class the validation of the user and the password is being done locally.



```java
     */
    @SuppressWarnings("unchecked")
    Generated Code

    private void txtUserActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void LoginButtonActionPerformed(java.awt.event.ActionEvent evt) {
        Admin admin = new Admin("Richard","codemaster23","codemaster");
        String user = txtUser.getText();
        System.out.println(user);
        System.out.println(admin.getUser());
        String password = txtPassword.~~getText~~();
        MainMenu menu = new MainMenu();
        if(user.equals(admin.getUser())){
            if(password.equals(admin.getPassword())){
                JOptionPane.showMessageDialog(rootPane, "Bienvenido " + txtUser.getText());
                menu.setVisible(true);
                this.setVisible(false);
            }
            else{
                JOptionPane.showMessageDialog(rootPane,"Contraseña incorrecta");
            }
        }
        else{
            JOptionPane.showMessageDialog(rootPane,"Usuario Incorrecto");
        }

    }

    private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        dispose();
    }

    /**
```

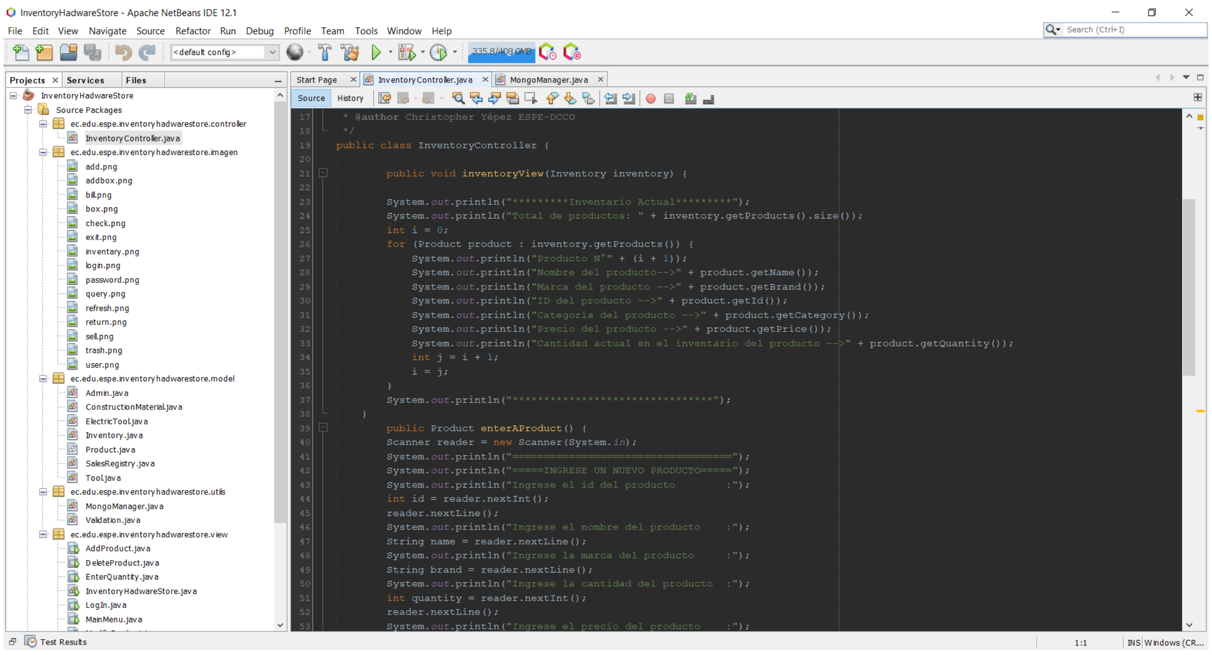
- ModifyProduct In this class you have programmed in the buttons do not work with methods



```java
                    boolean found = false;
                    for (int j = 0; j < products.size(); j++) {
                        if (products.get(j).getName().equals(searchName)) {
                            found = true;
                            System.out.println("Que dato desea modificar?");
                            System.out.println("1:Marca");
                            System.out.println("2:Precio");
                            System.out.println("3:Categoria");
                            if (rbtnBrand.isSelected())
                                    products.get(j).setBrand(txtChange.getText());
                            if (rbtnPrice.isSelected())
                                    products.get(j).setPrice(Float.parseFloat(txtChange.getText()));
                            if (rbtnCategory.isSelected())
                                    products.get(j).setCategory(txtChange.getText());
                            }
                            Gson gson = new Gson();
                            gson.toJson(products.get(j));
                            FileManager.writeFile("RegistroProductos.json", gson.toJson(products.get(j)));
                            System.out.println("Producto modificado");
                        }
                    if (!found)
                        txtError.setText("No se encontro el Producto");

    }

    private void txtChangeActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
        this.~~hide~~();
        MainMenu frm = new MainMenu();
        frm.setVisible(true);
    }

    /**
     * @param args the command line arguments
```

Controller Package

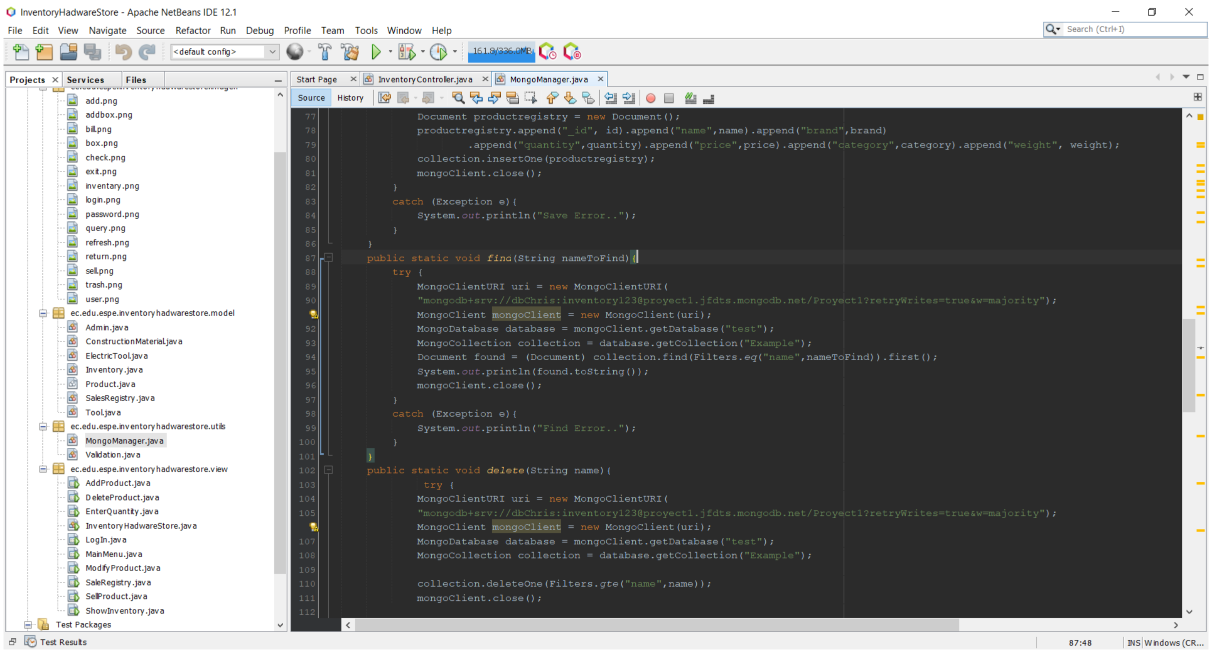
- In this class you have programmed menus, they should be called by methods.


InventoryHadwareStore - Apache NetBeans IDE 12.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Start Page
InventoryController.java
MongoManager.java
Source History

```java
 * @author Christopher Yépez ESPE-DCCO
 */
public class InventoryController {

        public void inventoryView(Inventory inventory) {

        System.out.println("*********Inventario Actual*********");
        System.out.println("Total de productos: " + inventory.getProducts().size());
        int i = 0;
        for (Product product : inventory.getProducts()) {
            System.out.println("Producto N°" + (i + 1));
            System.out.println("Nombre del producto-->" + product.getName());
            System.out.println("Marca del producto -->" + product.getBrand());
            System.out.println("ID del producto -->" + product.getId());
            System.out.println("Categoría del producto -->" + product.getCategory());
            System.out.println("Precio del producto -->" + product.getPrice());
            System.out.println("Cantidad actual en el inventario del producto -->" + product.getQuantity());
            int j = i + 1;
            i = j;
        }
        System.out.println("*******************************");
    }
        public Product enterAProduct() {
        Scanner reader = new Scanner(System.in);
        System.out.println("===================================");
        System.out.println("=====INGRESE UN NUEVO PRODUCTO=====");
        System.out.println("Ingrese el id del producto        :");
        int id = reader.nextInt();
        reader.nextLine();
        System.out.println("Ingrese el nombre del producto    :");
        String name = reader.nextLine();
        System.out.println("Ingrese la marca del producto     :");
        String brand = reader.nextLine();
        System.out.println("Ingrese la cantidad del producto  :");
        int quantity = reader.nextInt();
        reader.nextLine();
        System.out.println("Ingrese el precio del producto    :");
```

Utils Package

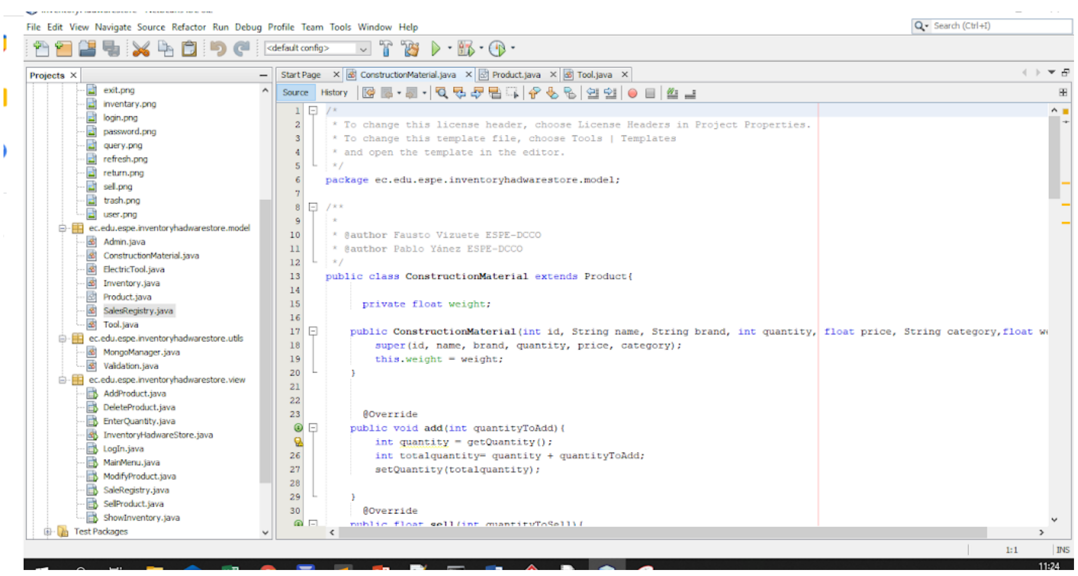
- To create save search, in each method call back to mongo URI, creating excess and repeated code


InventoryHadwareStore - Apache NetBeans IDE 12.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Start Page
InventoryController.java
MongoManager.java
Source History

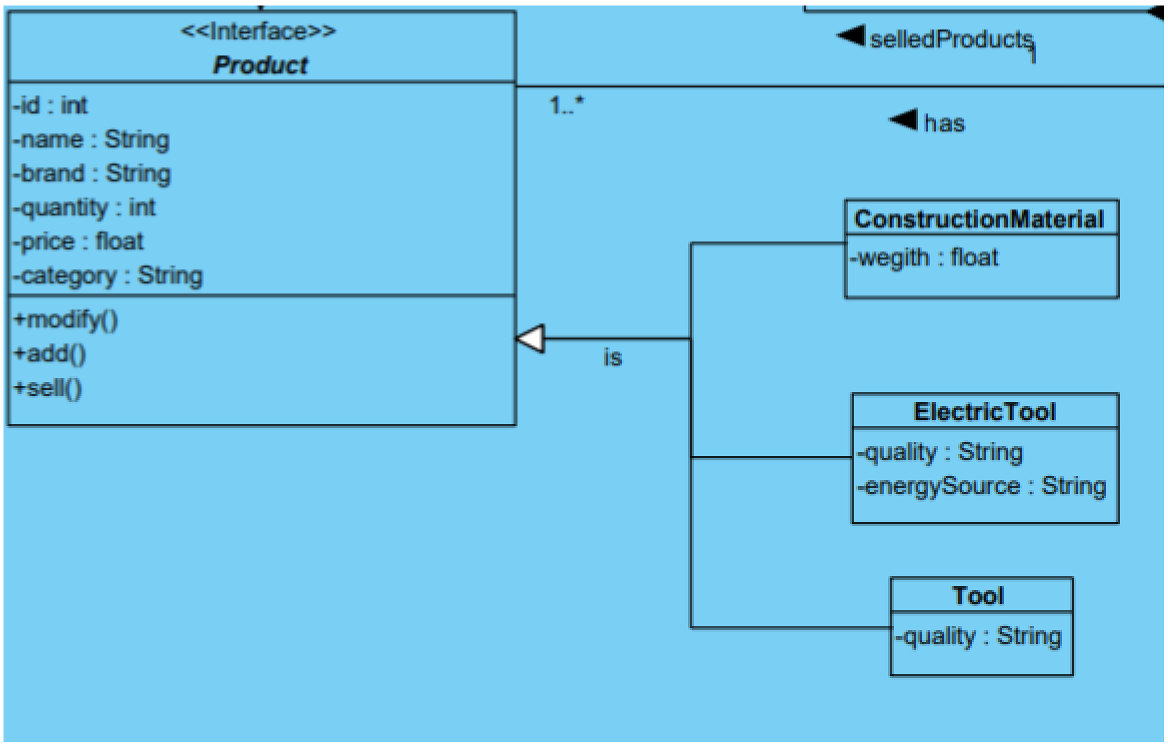
```java
            Document productregistry = new Document();
            productregistry.append("_id", id).append("name",name).append("brand",brand)
                    .append("quantity",quantity).append("price",price).append("category",category).append("weight", weight);
            collection.insertOne(productregistry);
            mongoClient.close();
        }
        catch (Exception e){
            System.out.println("Save Error..");
        }
    }
    public static void find(String nameToFind){
        try {
            MongoClientURI uri = new MongoClientURI(
            "mongodb+srv://dbChris:inventory123@proyect1.jfdts.mongodb.net/Proyect1?retryWrites=true&w=majority");
            MongoClient mongoClient = new MongoClient(uri);
            MongoDatabase database = mongoClient.getDatabase("test");
            MongoCollection collection = database.getCollection("Example");
            Document found = (Document) collection.find(Filters.eq("name",nameToFind)).first();
            System.out.println(found.toString());
            mongoClient.close();
        }
        catch (Exception e){
            System.out.println("Find Error..");
        }
    }
    public static void delete(String name){
            try {
            MongoClientURI uri = new MongoClientURI(
            "mongodb+srv://dbChris:inventory123@proyect1.jfdts.mongodb.net/Proyect1?retryWrites=true&w=majority");
            MongoClient mongoClient = new MongoClient(uri);
            MongoDatabase database = mongoClient.getDatabase("test");
            MongoCollection collection = database.getCollection("Example");

            collection.deleteOne(Filters.gte("name",name));
            mongoClient.close();

```

Open/Closed

Model Package

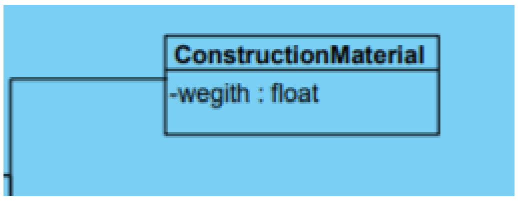
- 

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.edu.espe.inventoryhadwarestore.model;

/**
 *
 * @author Fausto Vizuete ESPE-DCCO
 * @author Pablo Yánez ESPE-DCCO
 */
public class ConstructionMaterial extends Product{

    private float weight;

    public ConstructionMaterial(int id, String name, String brand, int quantity, float price, String category,float we
        super(id, name, brand, quantity, price, category);
        this.weight = weight;
    }


    @Override
    public void add(int quantityToAdd){
        int quantity = getQuantity();
        int totalquantity= quantity + quantityToAdd;
        setQuantity(totalquantity);

    }
    @Override
```

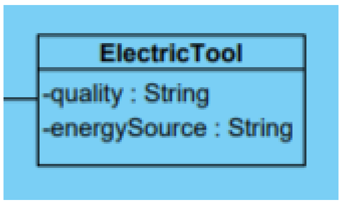


- encontramos que en las tres clases heredadas, no hay polimorfismo, por tanto de acuerdo al principio open closed, estos cambian ciertos aspectos y además añaden una función.

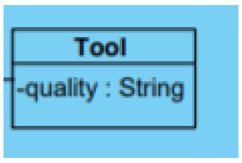
Principio de liskov de sustitución inversa.



1. no puede usar las mismas aplicaciones que la super clase padre.



2. Esta colinda con la utilización de otra clase que hacen algo parecido, además no puedo tener las mismas funciones que la superclase sin dañar la aplicación.

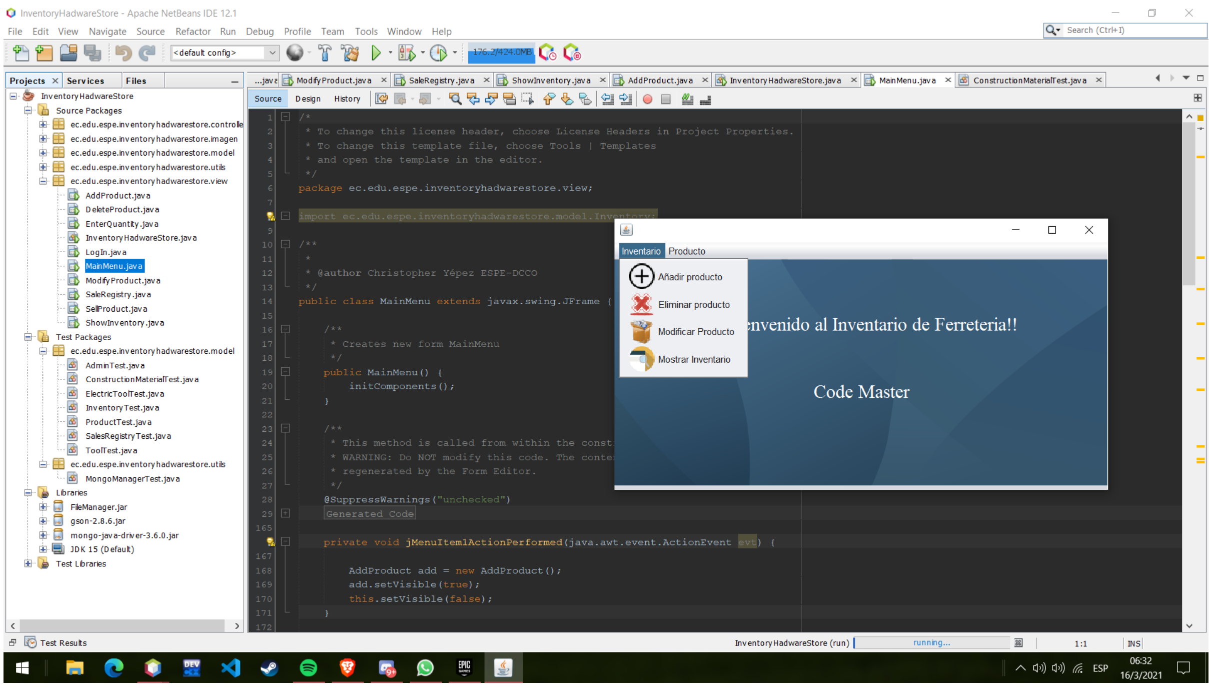


3. Esta es la clase que colinda en funciones con electric tool, ademas no hace lo que la super clase hace.

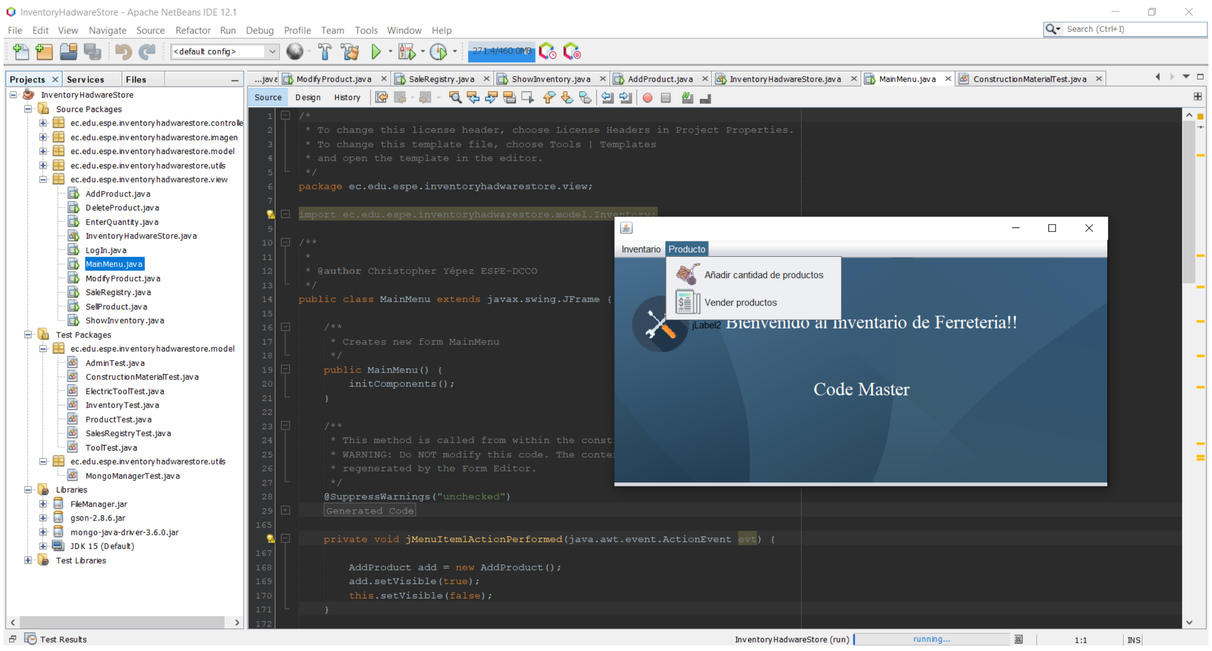
Por tanto estas tres co cumplen el principio liskov.

Interface segregation principle

View Package


InventoryHadwareStore - Apache NetBeans IDE 12.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Services Files
InventoryHadwareStore
Source Packages
ec.edu.espe.inventoryhadwarestore.controller
ec.edu.espe.inventoryhadwarestore.imagen
ec.edu.espe.inventoryhadwarestore.model
ec.edu.espe.inventoryhadwarestore.utils
ec.edu.espe.inventoryhadwarestore.view
AddProduct.java
DeleteProduct.java
EnterQuantity.java
InventoryHadwareStore.java
LogIn.java
MainMenu.java
ModifyProduct.java
SaleRegistry.java
SellProduct.java
ShowInventory.java
Test Packages
ec.edu.espe.inventoryhadwarestore.model
AdminTest.java
ConstructionMaterialTest.java
ElectricToolTest.java
InventoryTest.java
ProductTest.java
SalesRegistryTest.java
ToolTest.java
ec.edu.espe.inventoryhadwarestore.utils
MongoManagerTest.java
Libraries
FileManager.jar
gson-2.8.6.jar
mongo-java-driver-3.6.0.jar
JDK 15 (Default)
Test Libraries
Source Design History
Inventario Producto
Añadir producto
Eliminar producto
Modificar Producto
Mostrar Inventario
envenido al Inventario de Ferreteria!!
Code Master
InventoryHadwareStore (run)
running...

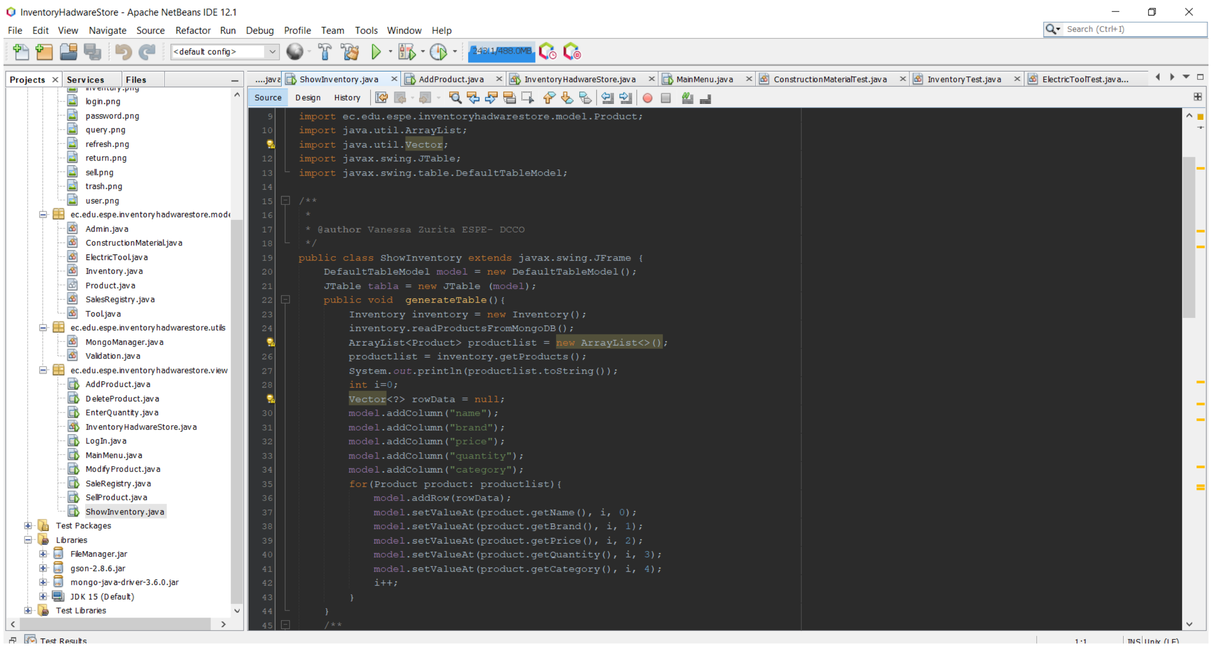
```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.edu.espe.inventoryhadwarestore.view;

import ec.edu.espe.inventoryhadwarestore.model.Inventory;

/**
 *
 * @author Christopher Yépez ESPE-DCCO
 */
public class MainMenu extends javax.swing.JFrame {

    /**
     * Creates new form MainMenu
     */
    public MainMenu() {
        initComponents();
    }

    /**
     * This method is called from within the const
     * WARNING: Do NOT modify this code. The conte
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {

        AddProduct add = new AddProduct();
        add.setVisible(true);
        this.setVisible(false);
    }
```


Inventario
Producto
Añadir cantidad de productos
Vender productos
jLabel2 Bienvenido al Inventario de Ferreteria!!
Code Master

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.edu.espe.inventoryhadwarestore.view;

import ec.edu.espe.inventoryhadwarestore.model.Inventory;

/**
 *
 * @author Christopher Yépez ESPE-DCCO
 */
public class MainMenu extends javax.swing.JFrame {

    /**
     * Creates new form MainMenu
     */
    public MainMenu() {
        initComponents();
    }

    /**
     * This method is called from within the const
     * WARNING: Do NOT modify this code. The conte
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {

        AddProduct add = new AddProduct();
        add.setVisible(true);
        this.setVisible(false);
    }
```

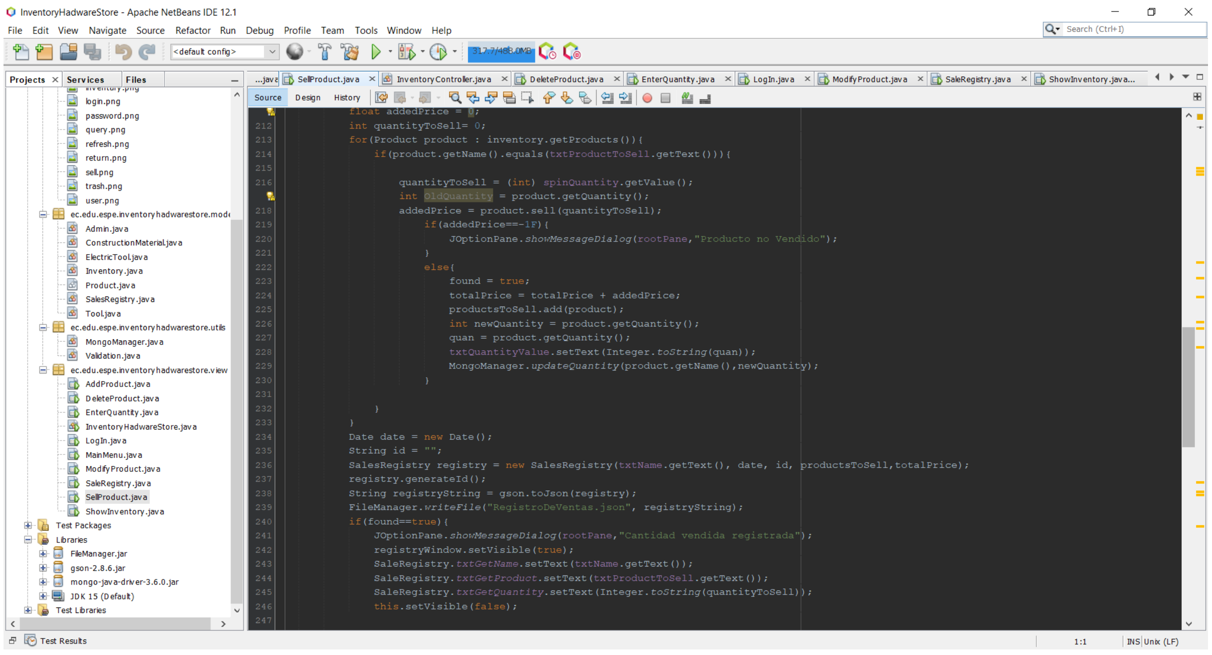
- In this case the interfaces are well displayed because they are not using general-purpose interfaces, instead they have specific interfaces with specific methods, the error is that some interfaces are not deployed when buttons are clicked.

Dependency inversion principle



```java
import ec.edu.espe.inventoryhadwarestore.model.Product;
import java.util.ArrayList;
import java.util.Vector;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author Vanessa Zurita ESPE- DCCO
 */
public class ShowInventory extends javax.swing.JFrame {
    DefaultTableModel model = new DefaultTableModel();
    JTable tabla = new JTable (model);
    public void  generateTable(){
        Inventory inventory = new Inventory();
        inventory.readProductsFromMongoDB();
        ArrayList<Product> productlist = new ArrayList<>();
        productlist = inventory.getProducts();
        System.out.println(productlist.toString());
        int i=0;
        Vector<?> rowData = null;
        model.addColumn("name");
        model.addColumn("brand");
        model.addColumn("price");
        model.addColumn("quantity");
        model.addColumn("category");
        for(Product product: productlist){
            model.addRow(rowData);
            model.setValueAt(product.getName(), i, 0);
            model.setValueAt(product.getBrand(), i, 1);
            model.setValueAt(product.getPrice(), i, 2);
            model.setValueAt(product.getQuantity(), i, 3);
            model.setValueAt(product.getCategory(), i, 4);
            i++;
        }
    }
    /**
```

- A higher-level class, such as the get inventory, is depending on other higher-level ones, such as show inventory method. It is responsible for creating instances of those objects and then using them.



```java
        float addedPrice = 0;
        int quantityToSell= 0;
        for(Product product : inventory.getProducts()){
            if(product.getName().equals(txtProductToSell.getText())){

                quantityToSell = (int) spinQuantity.getValue();
                int OldQuantity = product.getQuantity();
                addedPrice = product.sell(quantityToSell);
                    if(addedPrice==-1F){
                        JOptionPane.showMessageDialog(rootPane,"Producto no Vendido");
                    }
                    else{
                        found = true;
                        totalPrice = totalPrice + addedPrice;
                        productsToSell.add(product);
                        int newQuantity = product.getQuantity();
                        quan = product.getQuantity();
                        txtQuantityValue.setText(Integer.toString(quan));
                        MongoManager.updateQuantity(product.getName(),newQuantity);
                    }

            }
        }
        Date date = new Date();
        String id = "";
        SalesRegistry registry = new SalesRegistry(txtName.getText(), date, id, productsToSell,totalPrice);
        registry.generateId();
        String registryString = gson.toJson(registry);
        FileManager.writeFile("RegistroDeVentas.json", registryString);
        if(found==true){
            JOptionPane.showMessageDialog(rootPane,"Cantidad vendida registrada");
            registryWindow.setVisible(true);
            SaleRegistry.txtGetName.setText(txtName.getText());
            SaleRegistry.txtGetProduct.setText(txtProductToSell.getText());
            SaleRegistry.txtGetQuantity.setText(Integer.toString(quantityToSell));
            this.setVisible(false);

```

- A higher-level class, such as add product, depends on higher-level ones, such as sell product method. It is responsible for creating instances of those objects and then using them.