[redacted] – Development

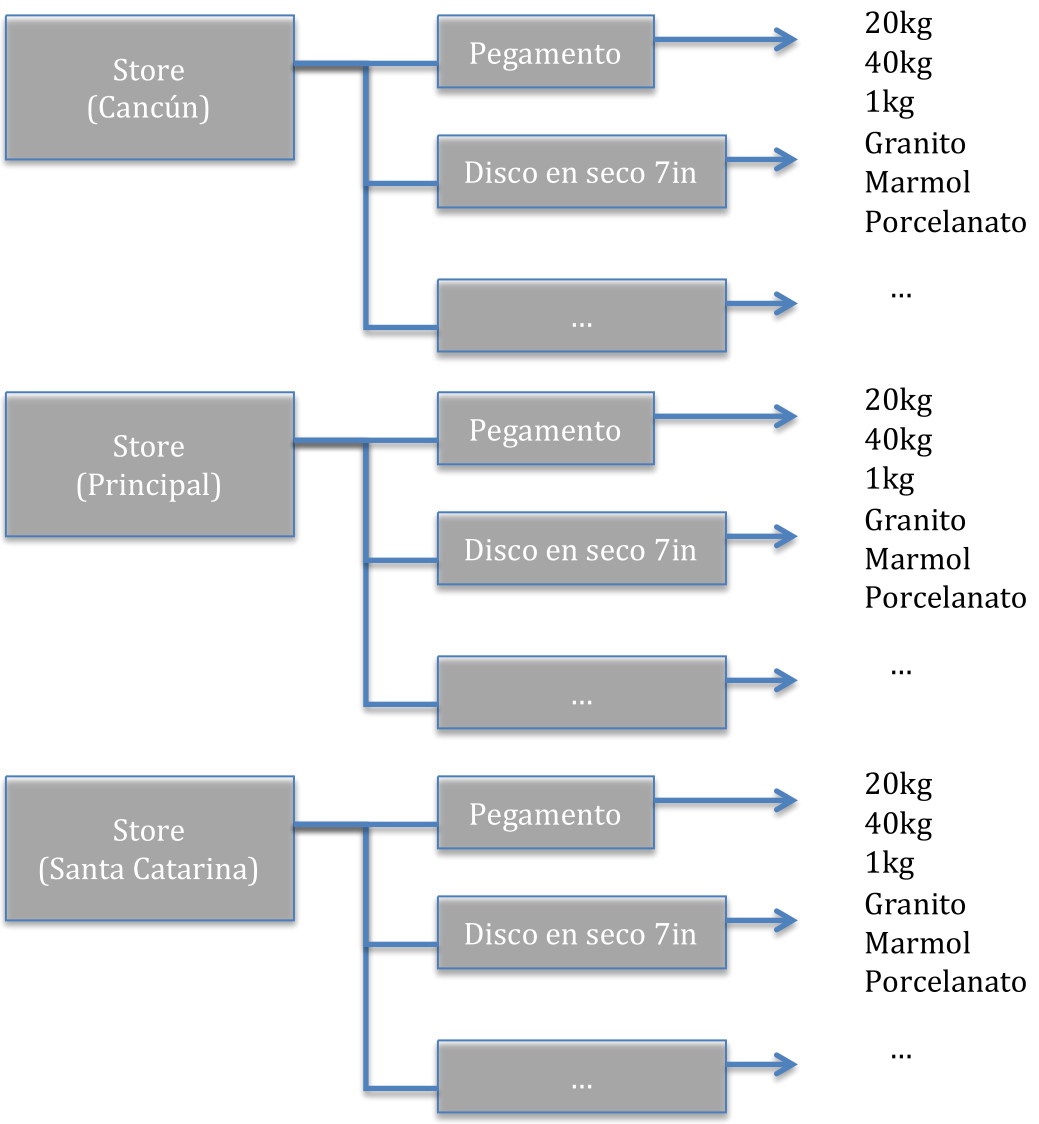
Techniques used

-Data structure
-Graphic interface (Java Swing)
-Excel database creation and overwriting (CSV)

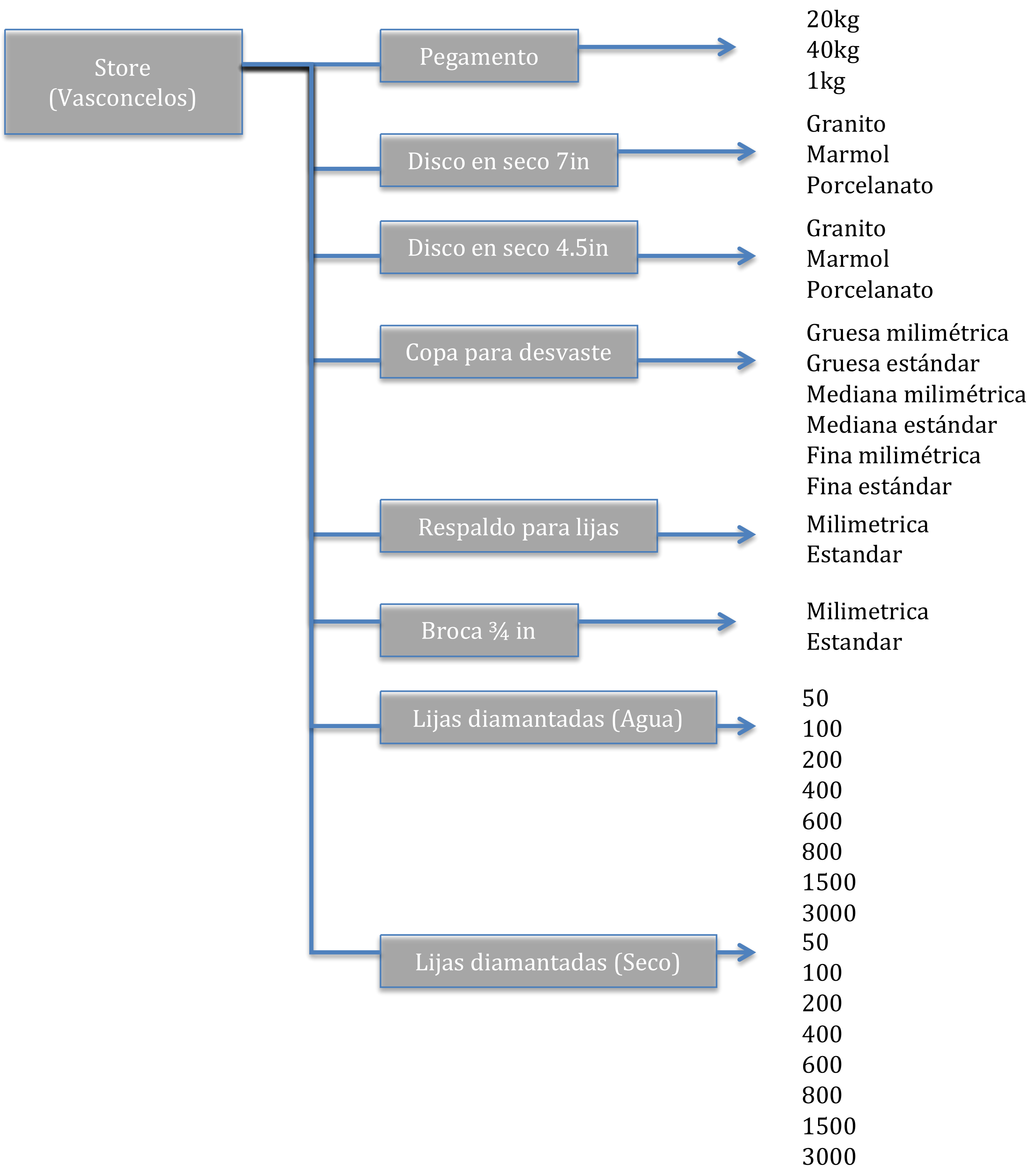
The objective of the program is to create a easy and fast way to organize sales and arrivals of products and modifying an excel document for their storage and access.

Data structure/Organization justification

The data structure was decided based on existence of multiple types and specifics of each product. Since each product might contain multiple variations and the information must be stored and managed per store and product type, the data was separated into multiple categories to facilitate its reading and usage as shown in the diagram.



This is a detailed view of the data structure as it would be seen in each store.



The data is managed individually for each of the end specific products, and store. Although this would create a lot of similar variables, it is necessary for the objective for the product, which is why the decision of organizing it as shown above was taken. Also, the storage method asked by the client (an excel document) was taken into consideration. Since it is possible for the data to be stored on tables, this data management is optimal for displaying individual values and characteristics. This will be achieved by the reading of the database in excel which will be explained later on.

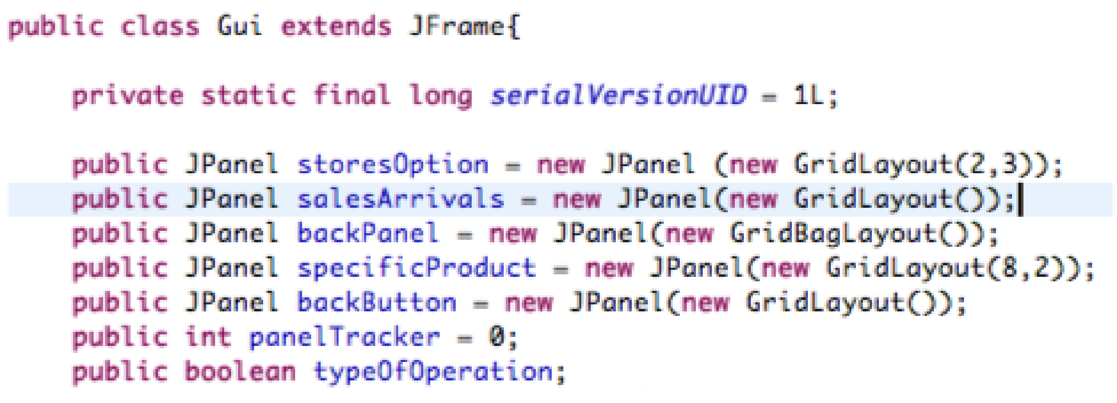
Graphic interface

As the product seeks to facilitate inventory management, a sales/arrivals interface for a user is needed. Since every product arrival arrives at the main sore to then be distributed when needed, the interface was designed to be operated from one location, with the options to change product quantity in each location.

Taking into consideration the same problem as for the data structure, the interface was divided into different categories for it to be easy to use, following a similar style as proposed in criterion B.

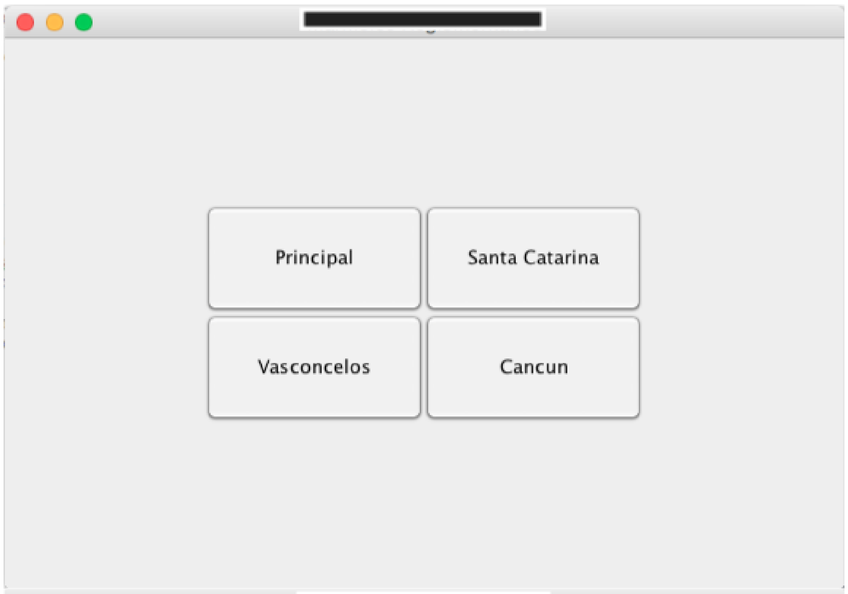
The interface was created using JSwing because of its simplicity and not requiring to have a specific design or images. CSS and Java-fx with Scene Builder were taken into consideration but it was decided that the product required simplicity to achieve its objective and JSwing was good for this purpose.

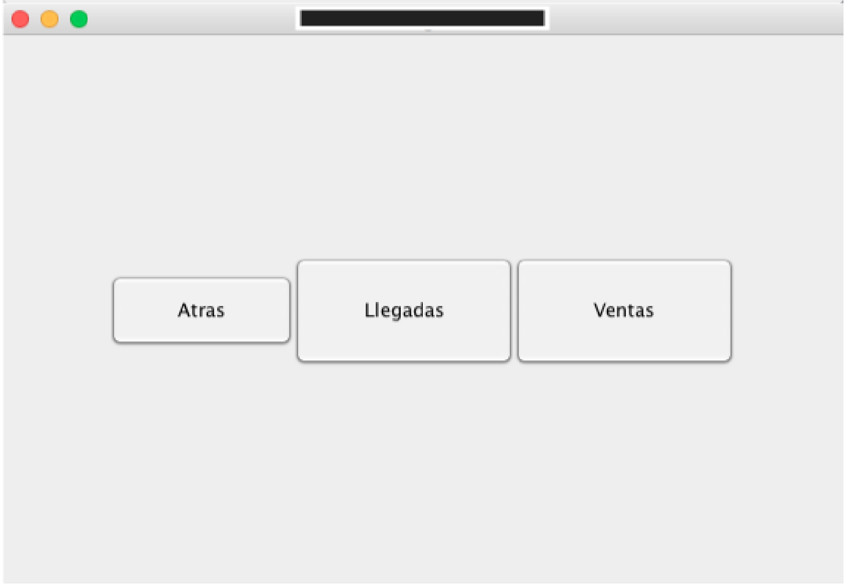
A single JFrame is used to guide the user throughout the product categories and the sales or arrival option in order to simplify the selection process. GridBagLayout is used to maintain the shape of the contents from filling the entire interface, while multiple GridLayouts are used to store the buttons for the different windows inside the JFrame(GridBagLayout) and easily organize them by stating the number of columns and rows.



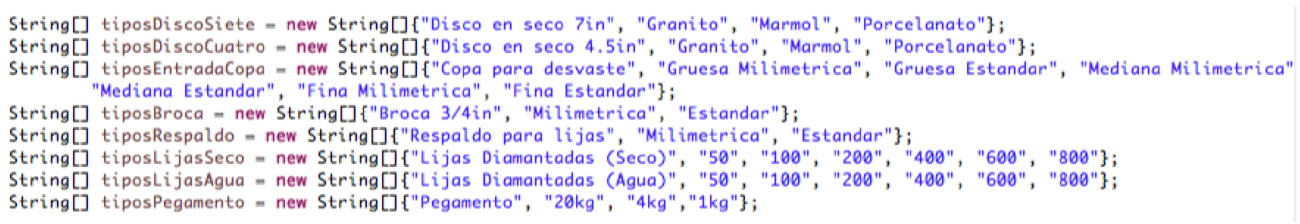
```
public class Gui extends JFrame{

    private static final long serialVersionUID = 1L;

    public JPanel storesOption = new JPanel (new GridLayout(2,3));
    public JPanel salesArrivals = new JPanel(new GridLayout());
    public JPanel backPanel = new JPanel(new GridBagLayout());
    public JPanel specificProduct = new JPanel(new GridLayout(8,2));
    public JPanel backButton = new JPanel(new GridLayout());
    public int panelTracker = 0;
    public boolean typeOfOperation;
```

In addition, an int storeTracker is used to save the store selection when activated the specific stores action listener and a boolean typeOfOperation is used to determine if the action performed will be of arrivals or sales.

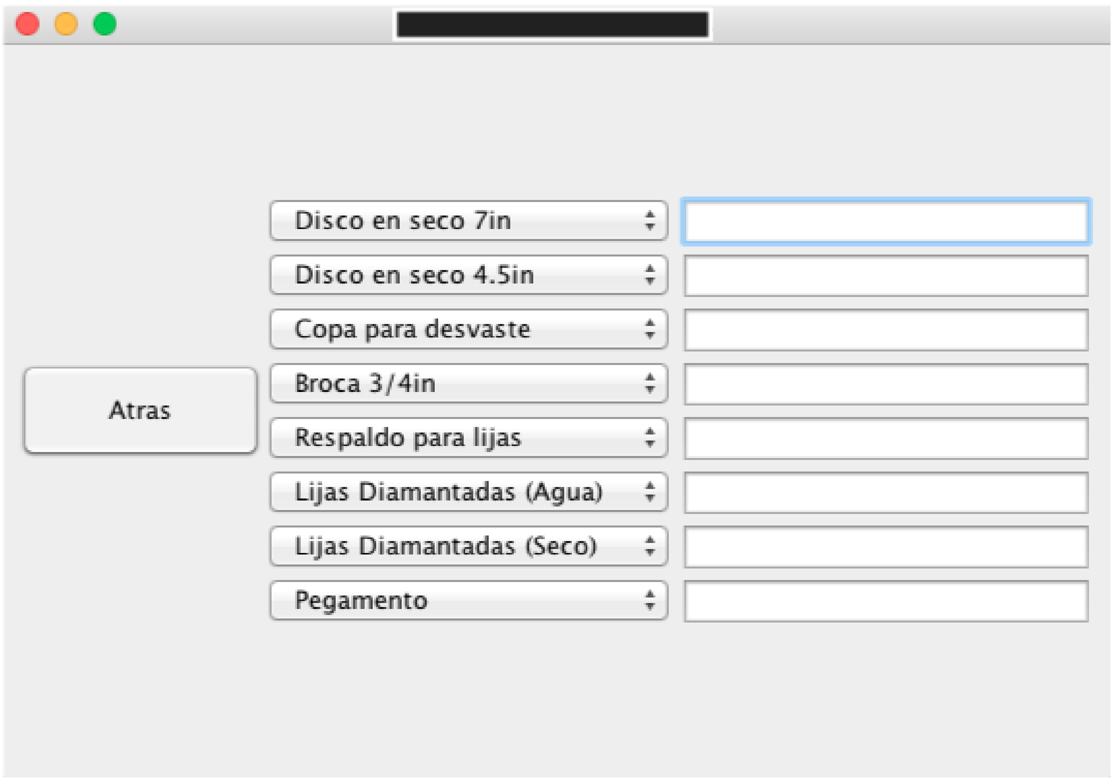

[redacted]


[redacted]

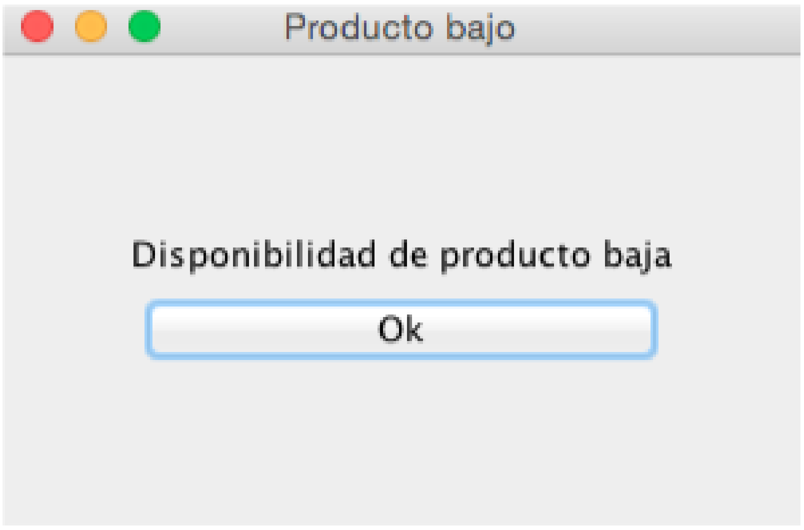
To complete the data structure process, it was necessary to display all options of each product and its type. Considering the data needed to be displayed, a JComoBox button type was chosen, as well as JTextFields for the input of sales/arrivals. Text Fields are cleared after each input for ease of use and Combo Boxes are reset to their default values after exiting the window.



```
String[] tiposDiscoSiete = new String[]{"Disco en seco 7in", "Granito", "Marmol", "Porcelanato"};
String[] tiposDiscoCuatro = new String[]{"Disco en seco 4.5in", "Granito", "Marmol", "Porcelanato"};
String[] tiposEntradaCopa = new String[]{"Copa para desvaste", "Gruesa Milimetrica", "Gruesa Estandar", "Mediana Milimetrica",
        "Mediana Estandar", "Fina Milimetrica", "Fina Estandar"};
String[] tiposBroca = new String[]{"Broca 3/4in", "Milimetrica", "Estandar"};
String[] tiposRespaldo = new String[]{"Respaldo para lijas", "Milimetrica", "Estandar"};
String[] tiposLijasSeco = new String[]{"Lijas Diamantadas (Seco)", "50", "100", "200", "400", "600", "800"};
String[] tiposLijasAgua = new String[]{"Lijas Diamantadas (Agua)", "50", "100", "200", "400", "600", "800"};
String[] tiposPegamento = new String[]{"Pegamento", "20kg", "4kg","1kg"};
```

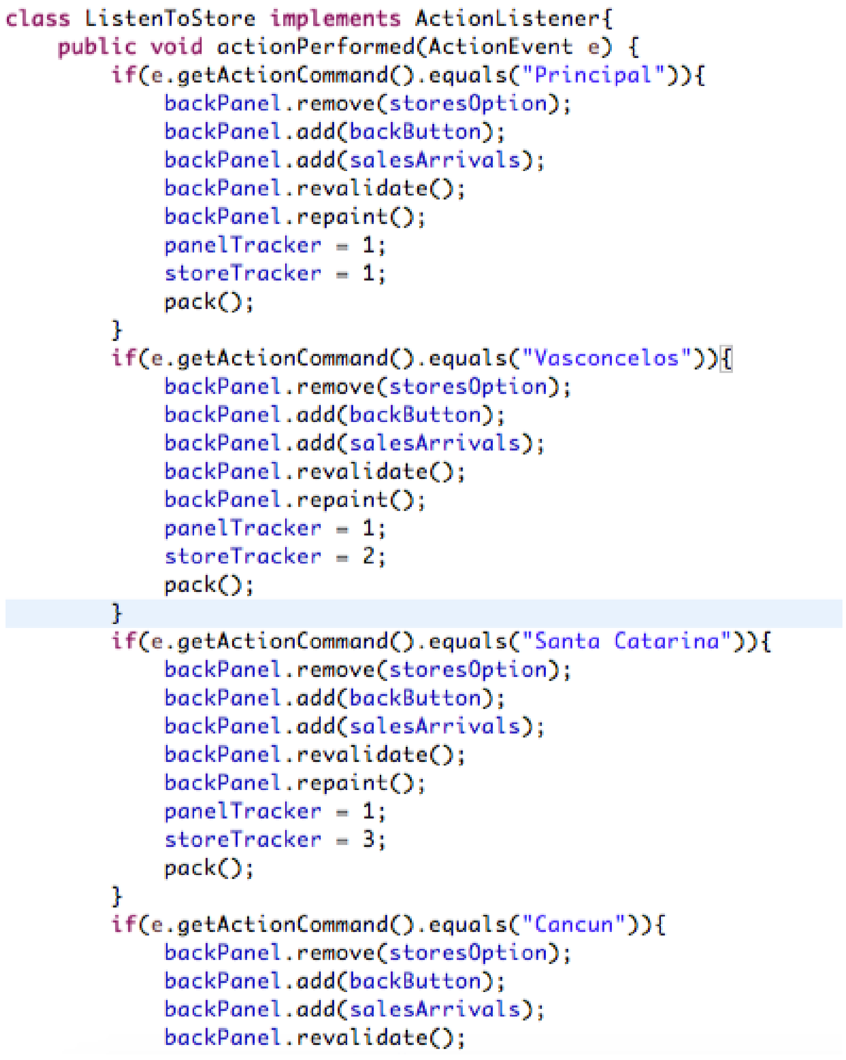

[redacted]

When entering a number on the JTextField, it is verified and retrieves the amount of product, displaying it in a new JFrame with a warning in the case of stocks being low. In the case of an invalid input (product availability is below 0) an appropriate warning is displayed as well.


Producto bajo

Disponibilidad de producto baja

To navigate through windows, layouts are added and removed, while registering the necessary data for modifications on the database.

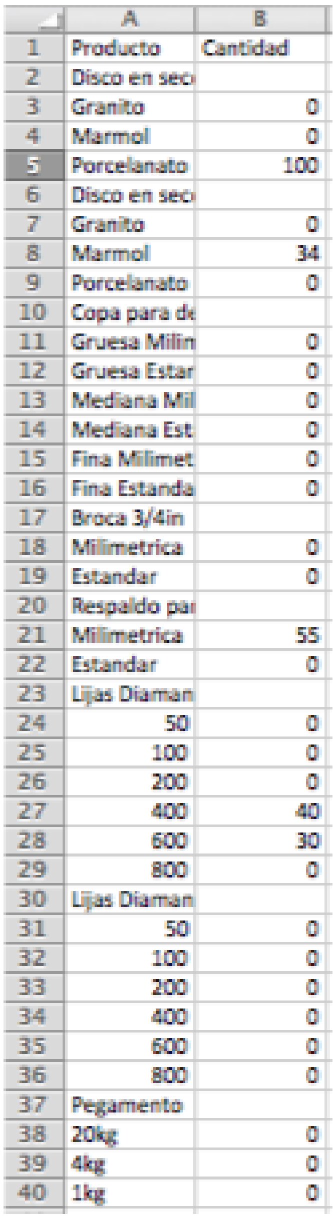


```
class ListenToStore implements ActionListener{
    public void actionPerformed(ActionEvent e) {
        if(e.getActionCommand().equals("Principal")){
            backPanel.remove(storesOption);
            backPanel.add(backButton);
            backPanel.add(salesArrivals);
            backPanel.revalidate();
            backPanel.repaint();
            panelTracker = 1;
            storeTracker = 1;
            pack();
        }
        if(e.getActionCommand().equals("Vasconcelos")){
            backPanel.remove(storesOption);
            backPanel.add(backButton);
            backPanel.add(salesArrivals);
            backPanel.revalidate();
            backPanel.repaint();
            panelTracker = 1;
            storeTracker = 2;
            pack();
        }
        if(e.getActionCommand().equals("Santa Catarina")){
            backPanel.remove(storesOption);
            backPanel.add(backButton);
            backPanel.add(salesArrivals);
            backPanel.revalidate();
            backPanel.repaint();
            panelTracker = 1;
            storeTracker = 3;
            pack();
        }
        if(e.getActionCommand().equals("Cancun")){
            backPanel.remove(storesOption);
            backPanel.add(backButton);
            backPanel.add(salesArrivals);
            backPanel.revalidate();
```

Excel database creation and overwriting

The database is managed using CSV files, which allows for overwriting in the form of a document, which can be viewed in excel as the client asked. A CSV document will be created for each store, each managing their products. It will be separated in the same way as the GUI in the sense that its specific subtypes and their availability will then follow each product. The CVS reader utilizes a list to save the current values, search through the document and overwrites the wanted value with the new changes. The CSV document will be created on the location of the .jar file for the application. In the case there is no present document, a default empty format will be created automatically.

Example of the created CSV file:



	A	B
1	Producto	Cantidad
2	Disco en seco	
3	Granito	0
4	Marmol	0
5	Porcelanato	100
6	Disco en seco	
7	Granito	0
8	Marmol	34
9	Porcelanato	0
10	Copa para de	
11	Gruesa Milin	0
12	Gruesa Estar	0
13	Mediana Mil	0
14	Mediana Est	0
15	Fina Milimet	0
16	Fina Estanda	0
17	Broca 3/4in	
18	Milimetrica	0
19	Estandar	0
20	Respaldo par	
21	Milimetrica	55
22	Estandar	0
23	Lijas Diaman	
24	50	0
25	100	0
26	200	0
27	400	40
28	600	30
29	800	0
30	Lijas Diaman	
31	50	0
32	100	0
33	200	0
34	400	0
35	600	0
36	800	0
37	Pegamento	
38	20kg	0
39	4kg	0
40	1kg	0