Assignment Notes

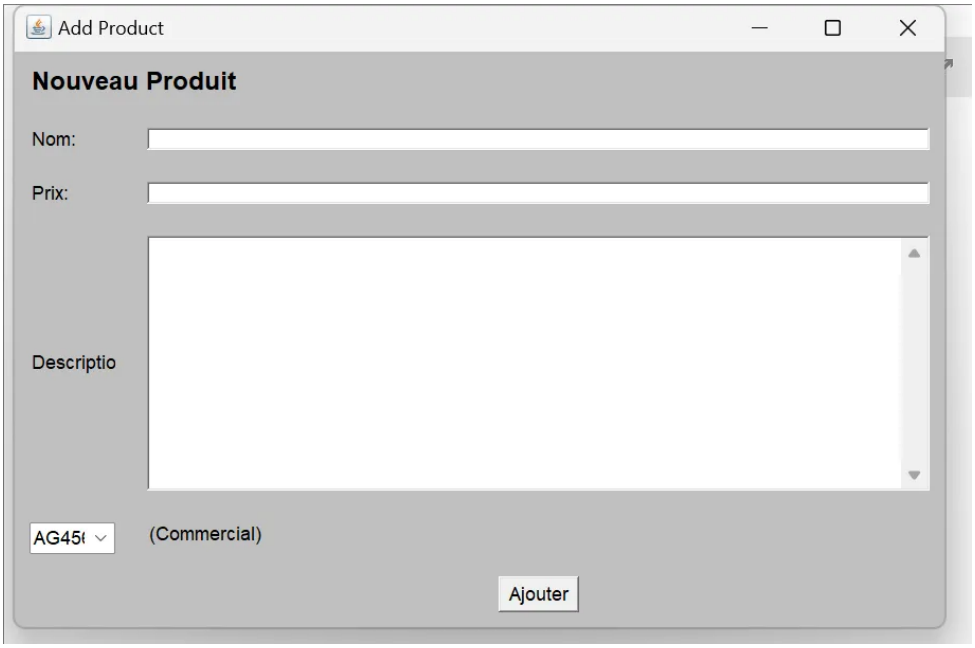

Figure 1 : Interface of our Assignment (Add Product)

I chose to use `Java AWT`, an older Java library, for this interface. While it has certain limitations compared to `javax.swing` or `JavaFX`, the goal was to gain familiarity with the basics of Java interface design. Since `AWT` serves as the foundation for both `Swing` and `JavaFX`, having a basic understanding of it is essential. Additionally, I used `GridBagLayout`, a more complex layout manager that is available in both `Swing` and `JavaFX`.

GridBagLayout :

Compared to other layouts, `GridBagLayout` is the most flexible layout in Java AWT. It is similar to `GridLayout` but offers more flexibility. For example, in `GridLayout`, all components must be of equal size, whereas in `GridBagLayout`, components can have different sizes. `GridBagLayout` uses an object called `GridBagConstraints`, which provides features to control the positioning and dimensions of specific components within the layout.

Positionning The components :

GridX , GridY of GridBagConstraints :

`GridX` and `GridY` are used to position elements or components at specific locations within a grid layout. To better understand the use of `GridX` and `GridY` in the code, here is a representation that may be helpful.

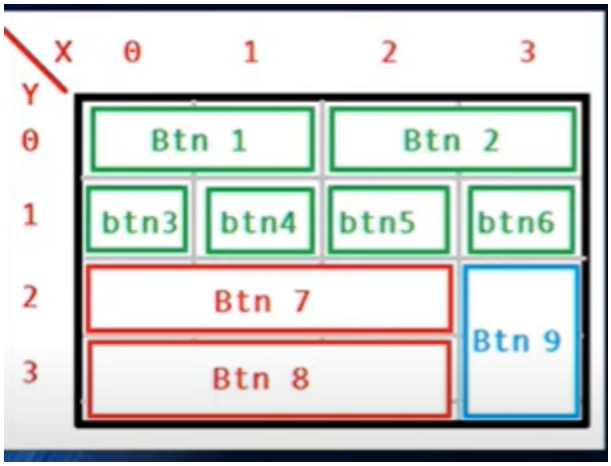

Figure 2 : Illustration of gridx and gridy in GridBagLayout

```
GridBagLayout gb = new GridBagLayout();
GridBagConstraints gbc = new GridBagConstraints();
Button btn1 = new Button("Btn 1");
gbc.gridx = 0; //Column
gbc.gridy = 0; //Row
gb.setConstraints(btn1 , gbc); //Or directly adding it to the
frame using frame.add(btn1 , gbc)
```

If we don’t specify `gridy` and we already specify `gridx`, the `GridBagLayout` will, by default, place each element in a new line within the same column defined by `gridx`. In this case, `gridy` is automatically incremented for each element or component added to the layout.
However, if we don’t specify both
`gridx` and `gridy`, then `GridBagLayout` manages the elements similarly to `FlowLayout`, placing them sequentially in rows. Once a row is filled, it moves to the next row .

Controlling and Managing Dimensions :

Gridwidth and GridHeight :

`gridwidth` and `gridheight` are used to control the dimensions of components in a `GridBagLayout`. They determine how much span a component should occupy. For instance, `gridwidth` is used to specify how many columns a component should span, while `gridheight` is used to specify how many rows a component should span .

```
// THIS WILL ADD THE ELEMENT AT THE END OF A LINE
// THE COMPONENT WILL OCCUPY ALL THE REMAINING COLUMNS IN THE
CURRENT ROW
gbc.gridwidth = GridBagConstraints.REMAINDER;
// THIS WILL ADD THE ELEMENT AT THE END OF A COLUMN
// THE COMPONENT WILL OCCUPY ALL THE REMAINING ROWS IN THE CU
RRENT COLUMN
gbc.gridheight = GridBagConstraints.REMAINDER;
```

weightx and weighty :

These properties determine how much space a component should occupy when the container is resized. For instance:

```
GridBagLayout gb = new GridBagLayout();
GridBagConstraints gbc = new GridBagConstraints();
Button btn1 = new Button("Btn 1");
gbc.weightx = 1;
gb.setConstraints(btn1 , gbc);
Button btn2 = new Button("Btn 2");
gbc.weightx = 2;
gb.setConstraints(btn1 , gbc);
```

In this example if the frame’s original size is 100px then btn1 will occupy 33.33px of the space while btn2 will occupy 66.67px , if the frame’s changed after being extended then the elements size will also change and btn2 will always occupy more space that btn1 .

Detailed Explanation :

btn1+2btn2 =3 , btn1 will occupy 1/3 of the original size frame and btn1 will occupy 2/3 of the original frame’s size.

Fill :

```
gridBagConstraints gbc = new gridBagConstraints();
 //USED TO FILL THE COMPONET HORIZONTALLY WITHIN ITS ALLOCATE
D CELL.
gbc.fill = GridBagConstraints.HORIZONTAL; //(Figure 2)
 //USED TO FILL THE COMPONET VERTICALLY WITHIN ITS ALLOCATED
CELL.
gbc.fill = GridBagConstraints.VERTICAL;  //(Figure 3)
 //USED TO FILL THE COMPONET THE ENTIRE WIDTH AND HIGHT WITHI
N ITS ALLOCATED CELL.
gbc.fill = GridBagConstraints.VERTICAL; //(Figure 4)
```

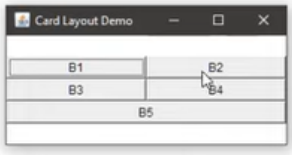

Figure 1 : Horizontal Fill

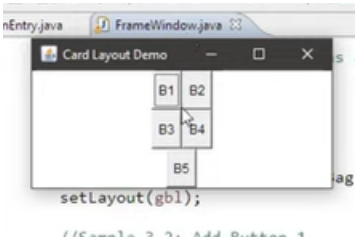

Figure 2 : Vertical Fill

Figure 3 : BOTH fill

References :

- 12 Java AWT GridBagLayout & GrigBagConstraints - Fill, WeightX, WeightY - [***Coding Examples Youtube Channel***]
- 13 Java AWT GridbagLayout & GridBagConstraints GridX, GridY, GridWidth & GridHeight - [***Coding Examples Youtube Channel***]
- https://www.jmdoudoux.fr/java/dej/index.htm - [***Développons en Java Website***]