Software Engineering and Testing

Session 5

Lecturer: Dr. Simon McLoughlin

Implementation

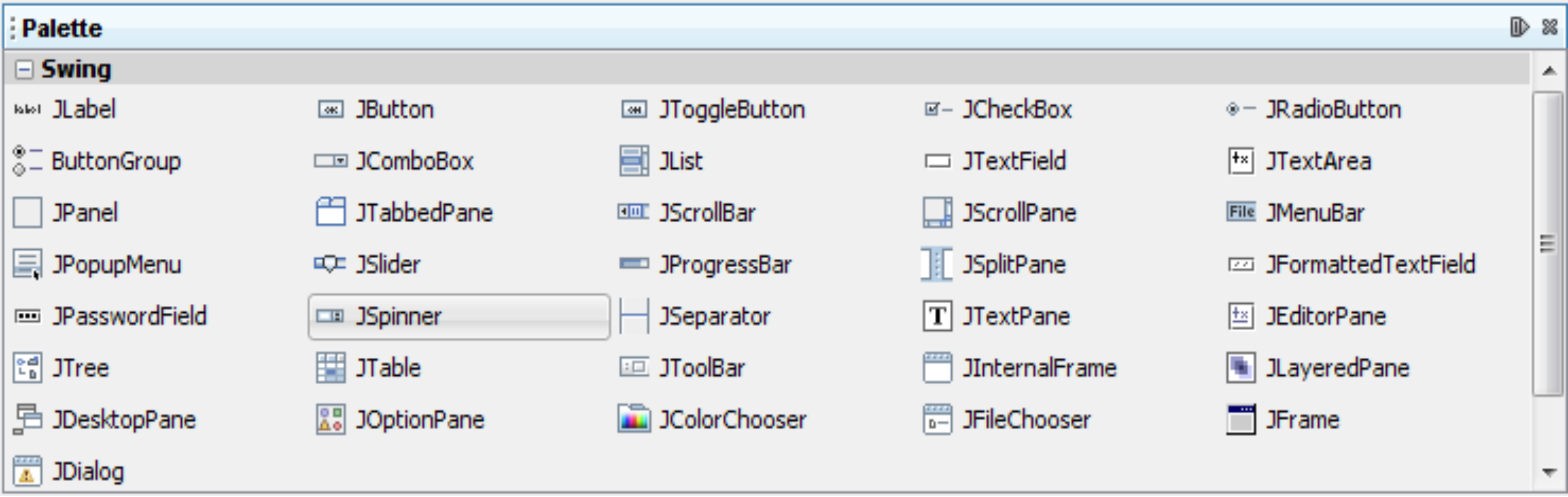
The next stage of your project is the implementation stage which will consist of:

- Modular well designed code (with comments)
- Executable files (e.g. project JAR)
- Associated Javadoc documentation
- Should be developed in Eclipse IDE
- Any other resource files (images etc)
- Installation Instructions (e.g. server configuration)
- Database (if your application uses one)

- Due first day back after Easter holidays (Friday 11th April, no extensions on this so please do not ask).

GUI Programming

- Recall from previous modules that Java provides comprehensive APIs for developing Graphical User Interfaces (GUIs)
- The API classes used for developing GUIs will normally come from the swing or awt packages and can be imported in your program
- Once imported you have access to a vast array of GUI components or widgets



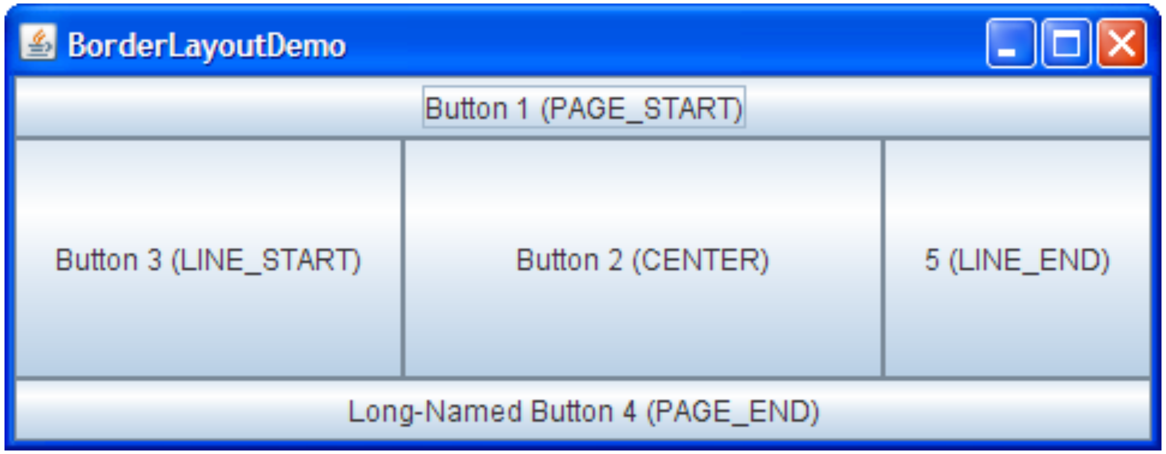
Which Components to choose?

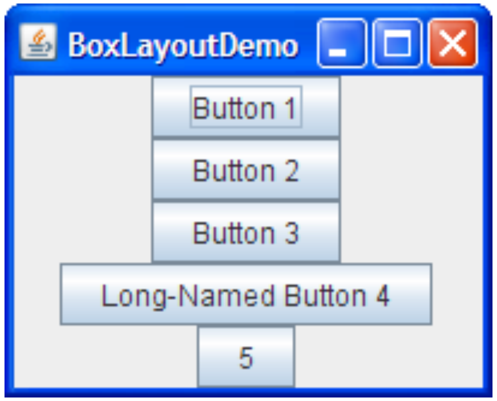
- One of the UI designe principles mentioned last week was to keep it simple and not to create “busy” GUIs
- Another one was to always understand the widgets you are using and use the correct one for the job
- With this in mind you should carefully go about deciding how many widgets you need, their placement and how they will be used.
- For example if you want to present the user with a long list of items and also give them the option of typing in the item they wish to select then use a JComboBox over a Jlist, the converse also applies
- You should not attempt to use as many widgets as you possibly can. You user interface will be judged based on appearance and functionality (ease of use) and not complexity.
- We will not go over all the different components available as you have seen most of these already but will have a look a flexible layout manager and look at the vending machine GUI

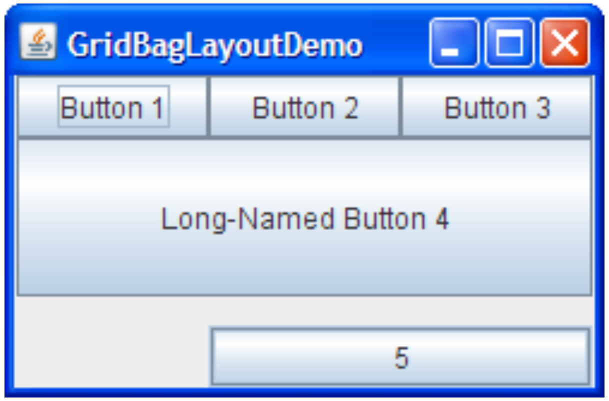
Layout Managers

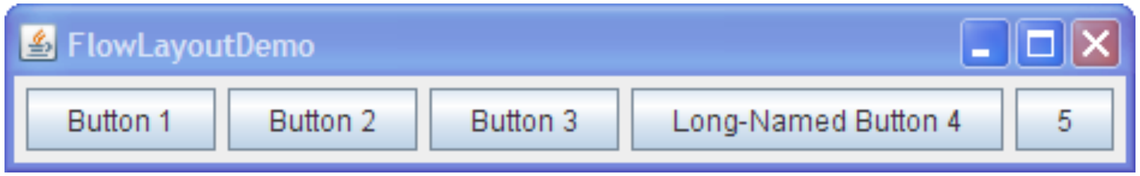
- LayoutManagers allow you to place components in different locations within the application's content pane.
- They provide scalable solutions that adjust automatically to resizes.
- There are a number of different layout managers at your disposal in Java
  - BorderLayout
  - BoxLayout
  - CardLayout
  - FlowLayout
  - GridBagLayout
  - GridLayout
  - GroupLayout
  - SpringLayout
- Depending on what you want to do and how you want to present your components, one or several (through JPanels) of the above layout managers may be applicable.
- For tips on choosing the right LayoutManager see:http://java.sun.com/docs/books/tutorial/uiswing/layout/using.html

Some Java Layout Managers



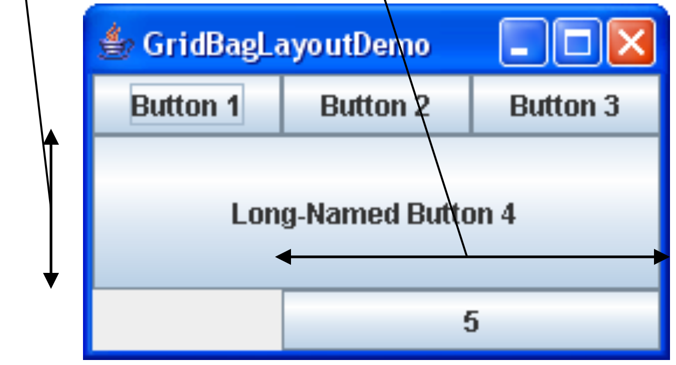






GridBagLayout

- The one I want to look at today is the GridBagLayout and is considered one of the most flexible but complex layout managers
- A GridBagLayout places components in a grid of rows and columns but where it differs from a GridLayout is that it allows signle components to span multiple rows and columns
- Not all rows need to have the same height either, nor do all columns need to have the same width. The row height/column width depends on the component's preferred size and padding.



Grid Bag Constraints

- The obvious question is then how do we specify the position and size of the components within the grid?
- The answer is through constraints. Each component will have a set of constraints associated with it which will determine its position, size and other appearance attributes.
- The constraints are created through a GridBagConstraints object and are associated with a component when adding it to the Container.
- For example:

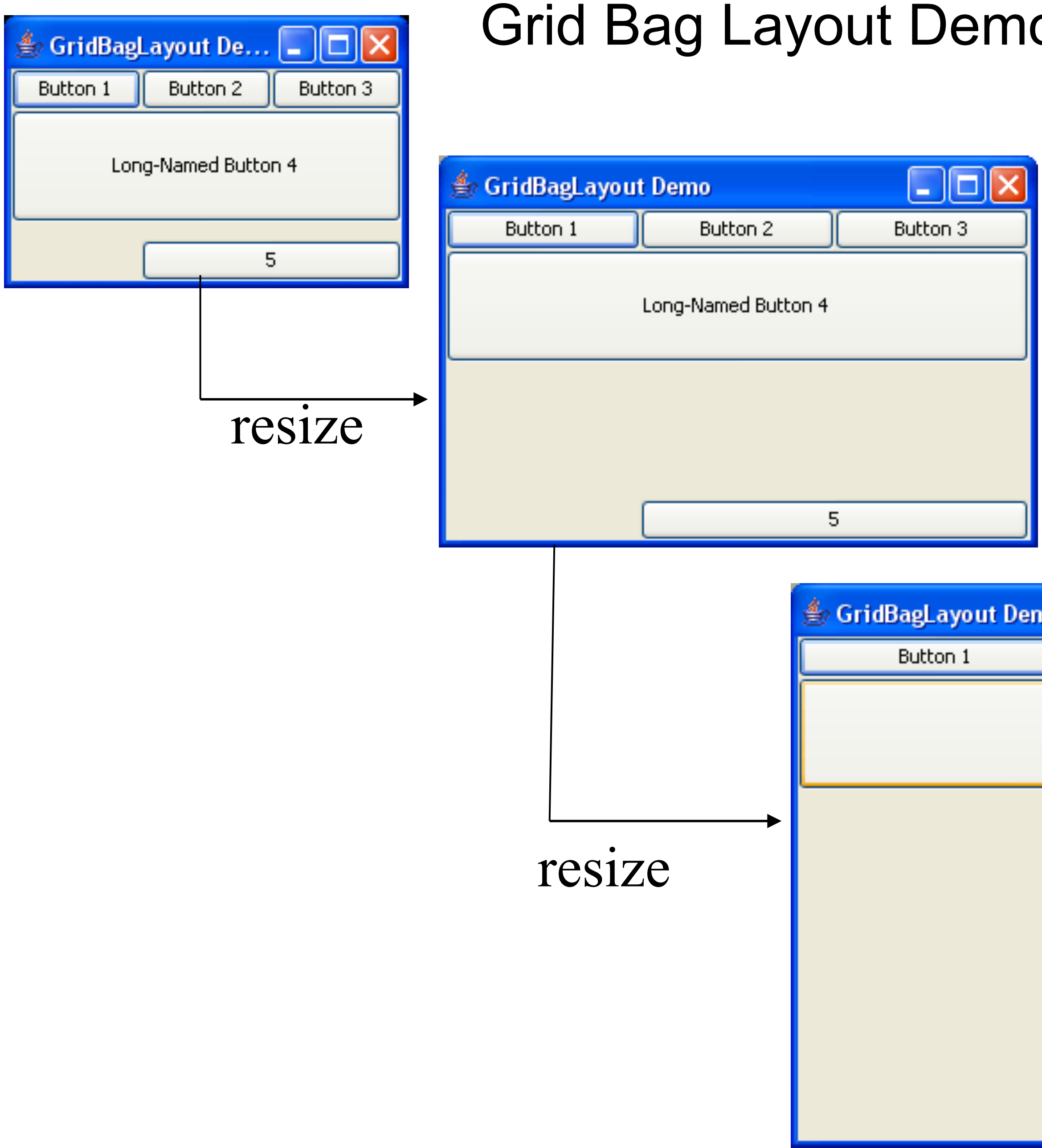
```
JPanel pane = new JPanel(new GridBagLayout());
GridBagConstraints c = new GridBagConstraints();
//For each component to be added to this container:
//...Create the component...
//...Set instance variables in the GridBagConstraints instance...
pane.add(theComponent, c);
```

Notice that the pane.add call takes in two arguments, the second being the grid constraints object that informs the LayoutManger how to display theComponent

Grid Bag Constraints

- So what are these instance variables that can be set within GridBagConstraints objects?
- Well there are quite a few of them and not all need to be changed.
- **gridx**, **gridy** – allow you to define the row and column position of the component
- **gridwidth**, **gridheight** – allows you to specify the number of rows and columns a particular component should span
- **fill** – determines if/how a component will grow when its display area is larger than its preferred size
- **ipadx**, **ipady** – specifies how much to pad the component (in pixels) in the x and y direction, i.e. to make it wider and taller
- **Insets** – specifies the minimum amount of space between a component and the edges of its display area
- **Anchor** – used to specify the alignment of a component within its own display area, e.g. CENTRE, FIRST_LINE_START, LAST_LINE_END
- **weightx**, **weighty** – values between 0 and 1 that inform the layout manager about space priority amongst components when resizing

Grid Bag Layout Demo



Grid Bag Layout Demo

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.LineBorder;

public class GridBagEg extends JFrame{


    public GridBagEg (String title)
    {
        super(title);
        createComponents();
    }

    public void createComponents() {


        this.getContentPane().setLayout(new GridBagLayout());
        GridBagConstraints c = new GridBagConstraints();

        JButton button;
        button = new JButton("Button 1");
        c.weightx = 0.5;
        c.fill = GridBagConstraints.HORIZONTAL;
        c.gridx = 0;
        c.gridy = 0;
        this.getContentPane().add(button, c);

        button = new JButton("Button 2");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.5;
        c.gridx = 1;
        c.gridy = 0;
        this.getContentPane().add(button, c);
```

Weight of 0.5, effect of this will depend on other component weights

Fill display space horizontally but not vertically

Location for this component in the grid

Add this component to the frames content pane using the constraints in c

Software Engineering and Testing – COMP H2027

```
        button = new JButton("Button 3");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.5;
        c.gridx = 2;
        c.gridy = 0;
        this.getContentPane().add(button, c);

        button = new JButton("Long-Named Button 4");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.ipady = 40;      //make this component tall
        c.weightx = 0.0;
        c.gridwidth = 3;
        c.gridx = 0;
        c.gridy = 1;
        this.getContentPane().add(button, c);

        button = new JButton("5");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.ipady = 0;       //reset to default
        c.weighty = 1.0;   //request any extra vertical space
        c.anchor = GridBagConstraints.PAGE_END; //bottom of space
        c.insets = new Insets(10,0,0,0);  //top padding
        c.gridx = 1;       //aligned with button 2
        c.gridwidth = 2;   //2 columns wide
        c.gridy = 2;       //third row
        this.getContentPane().add(button, c);
    }

    public static void main(String[] args) {

        try {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InstantiationException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (UnsupportedLookAndFeelException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        GridBagEg gbeg = new GridBagEg("GridBagLayout Demo");
        gbeg.pack();
        gbeg.setVisible(true);
    }
}
```

Make this component taller, by 40 pixels

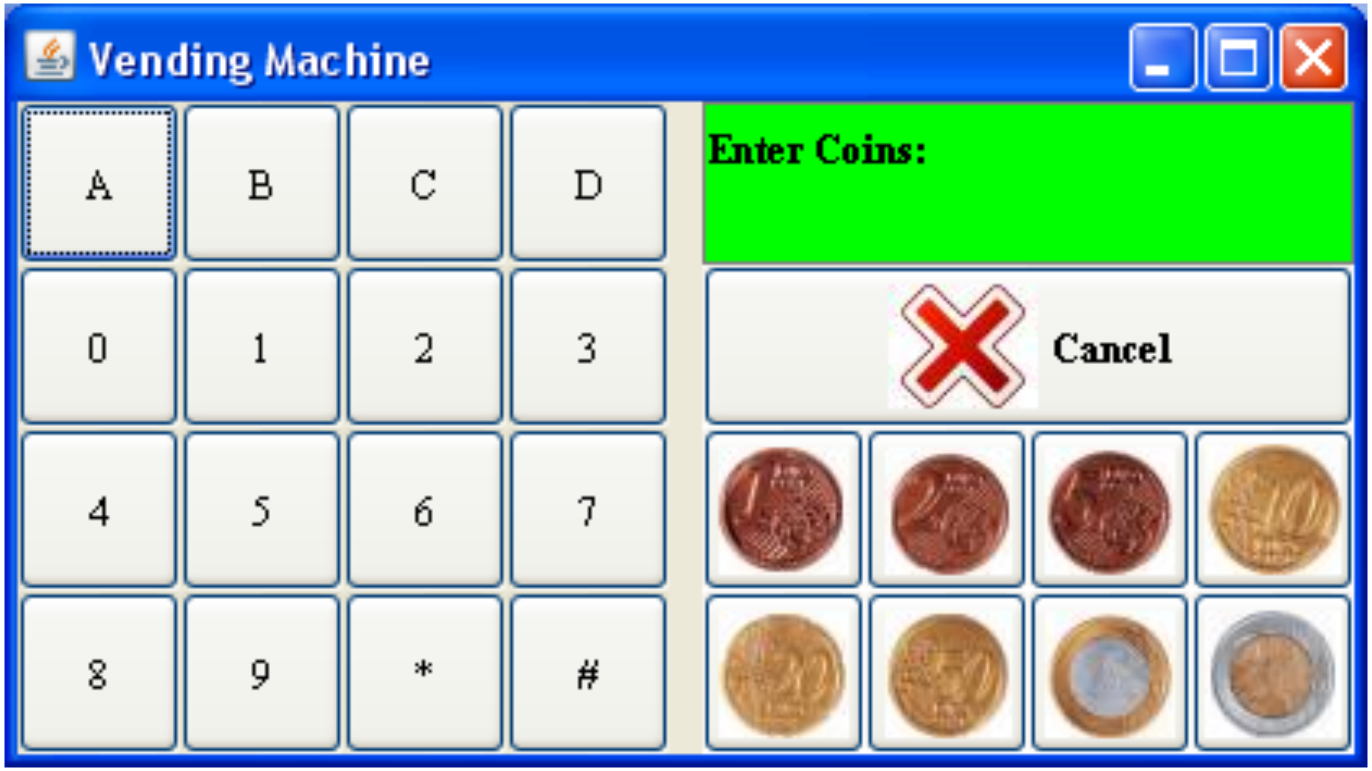
Notice grid width is 3, meaning it will span 3 grid cells horizontally

Key point: notice that its weighty=1 which means that it takes priority over the vertical space available when resizing, other components have weighty=0

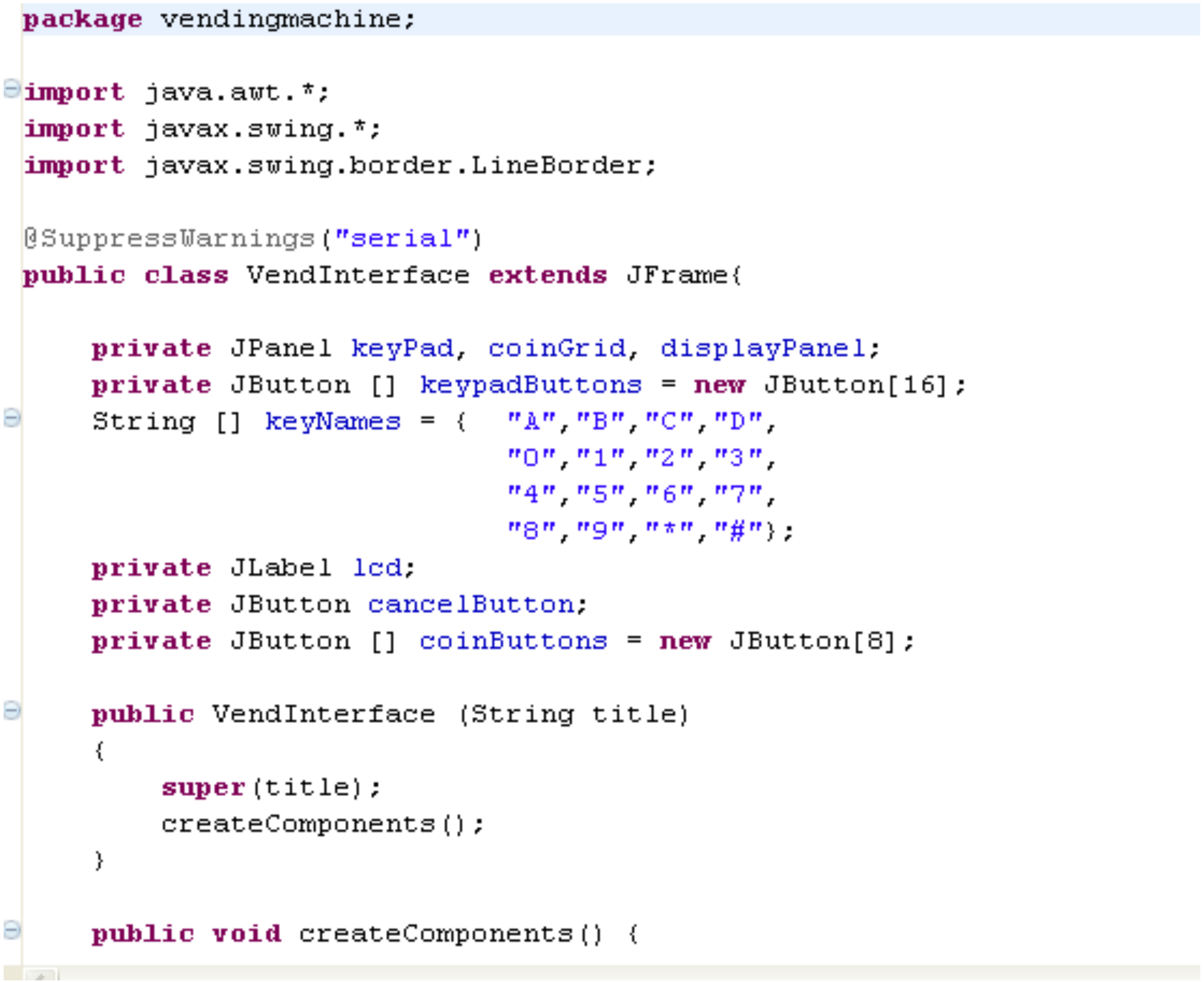
Notice how this component has its anchor constraint set to PAGE_END which is at the bottom of its display space

Notice that this component has padding between it and the top edge of its display area.

Vending Machine GUI

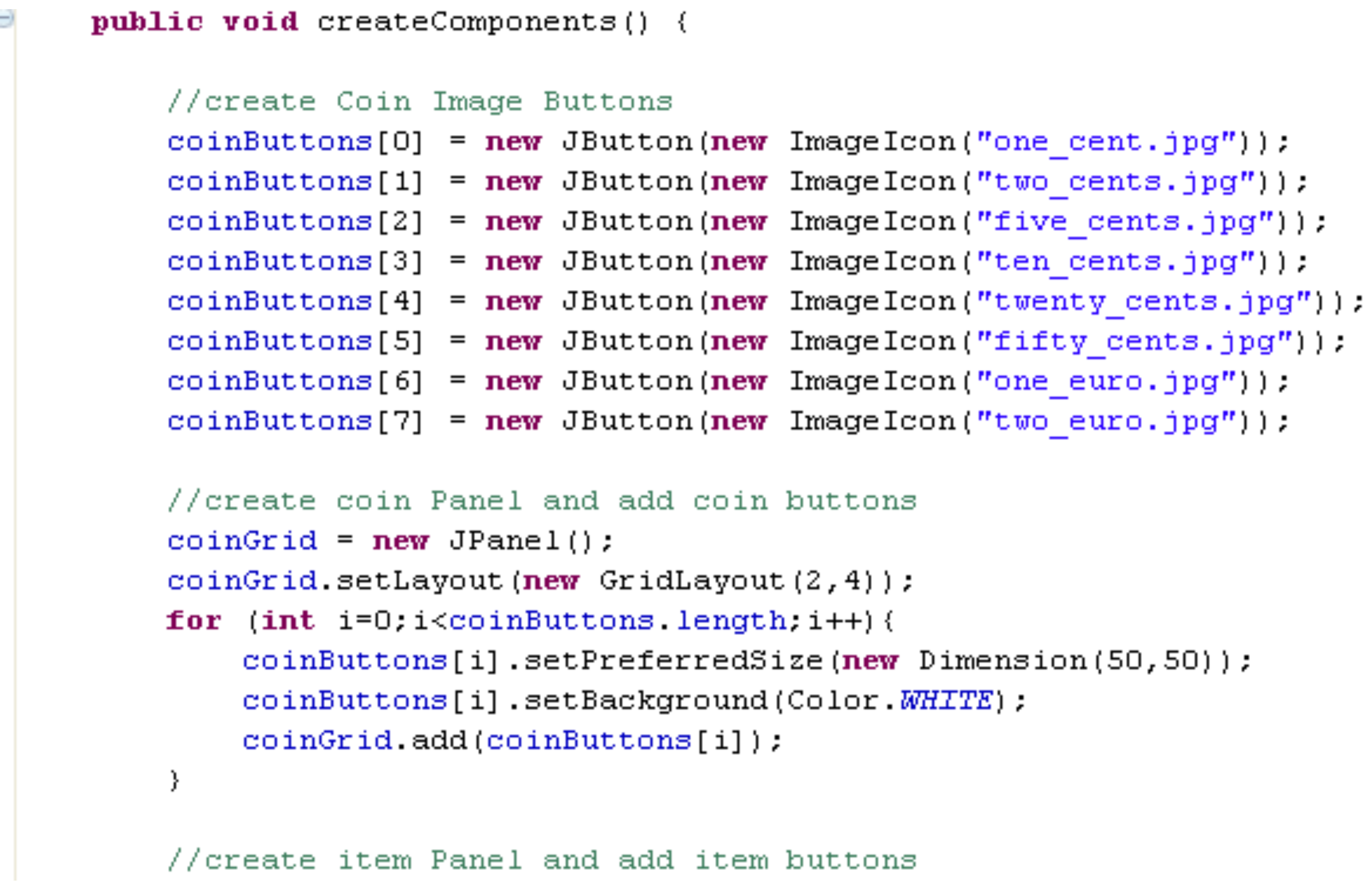


Vending Machine GUI



```java
package vendingmachine;

import java.awt.*;
import javax.swing.*;
import javax.swing.border.LineBorder;

@SuppressWarnings("serial")
public class VendInterface extends JFrame{

    private JPanel keyPad, coinGrid, displayPanel;
    private JButton [] keypadButtons = new JButton[16];
    String [] keyNames = {   "A","B","C","D",
                            "0","1","2","3",
                            "4","5","6","7",
                            "8","9","*","#"};
    private JLabel lcd;
    private JButton cancelButton;
    private JButton [] coinButtons = new JButton[8];

    public VendInterface (String title)
    {
        super(title);
        createComponents();
    }

    public void createComponents() {
```

Vending Machine GUI

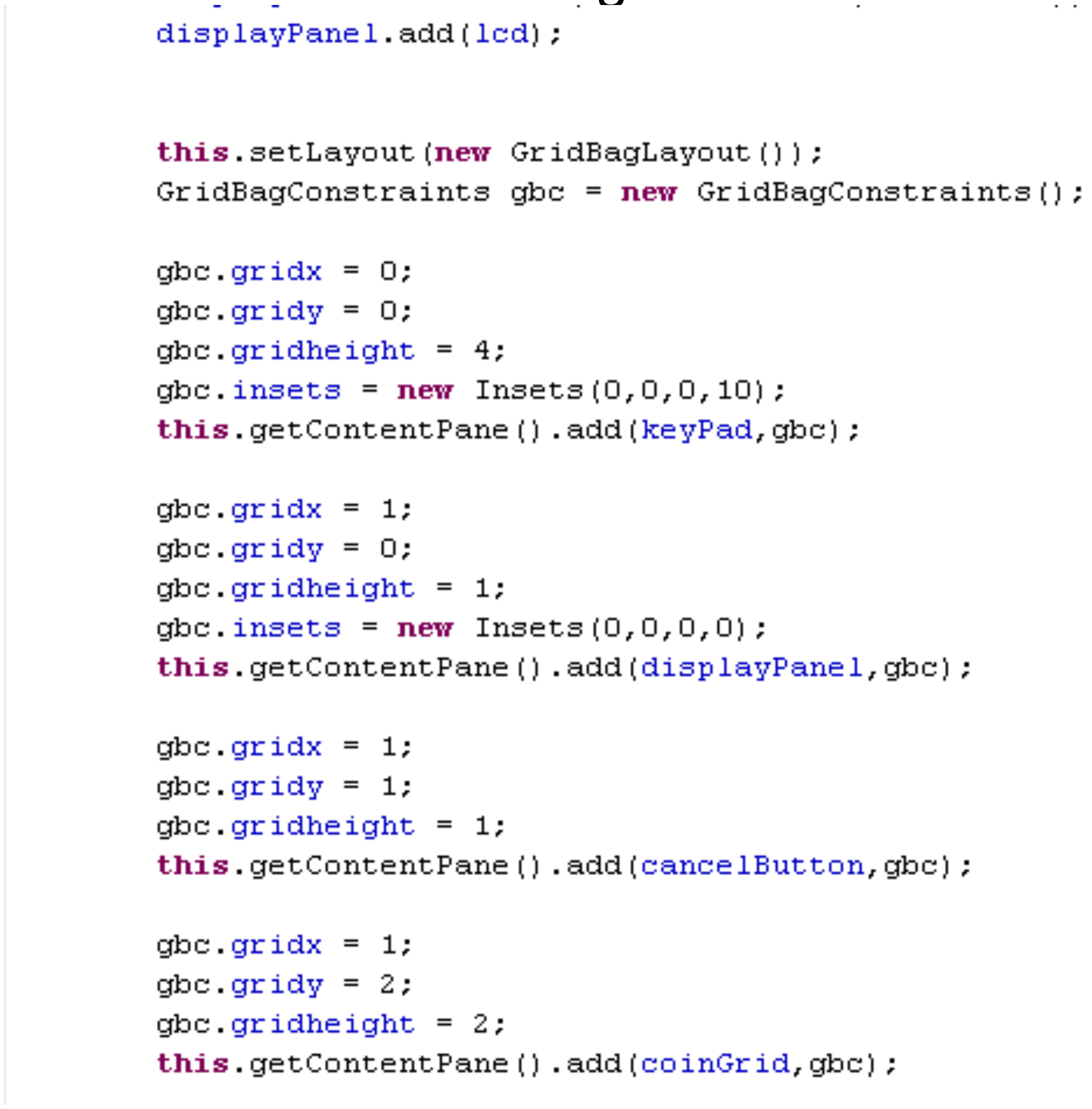


```
public void createComponents() {

    //create Coin Image Buttons
    coinButtons[0] = new JButton(new ImageIcon("one_cent.jpg"));
    coinButtons[1] = new JButton(new ImageIcon("two_cents.jpg"));
    coinButtons[2] = new JButton(new ImageIcon("five_cents.jpg"));
    coinButtons[3] = new JButton(new ImageIcon("ten_cents.jpg"));
    coinButtons[4] = new JButton(new ImageIcon("twenty_cents.jpg"));
    coinButtons[5] = new JButton(new ImageIcon("fifty_cents.jpg"));
    coinButtons[6] = new JButton(new ImageIcon("one_euro.jpg"));
    coinButtons[7] = new JButton(new ImageIcon("two_euro.jpg"));

    //create coin Panel and add coin buttons
    coinGrid = new JPanel();
    coinGrid.setLayout(new GridLayout(2,4));
    for (int i=0;i<coinButtons.length;i++){
        coinButtons[i].setPreferredSize(new Dimension(50,50));
        coinButtons[i].setBackground(Color.WHITE);
        coinGrid.add(coinButtons[i]);
    }

    //create item Panel and add item buttons
```

Vending Machine GUI

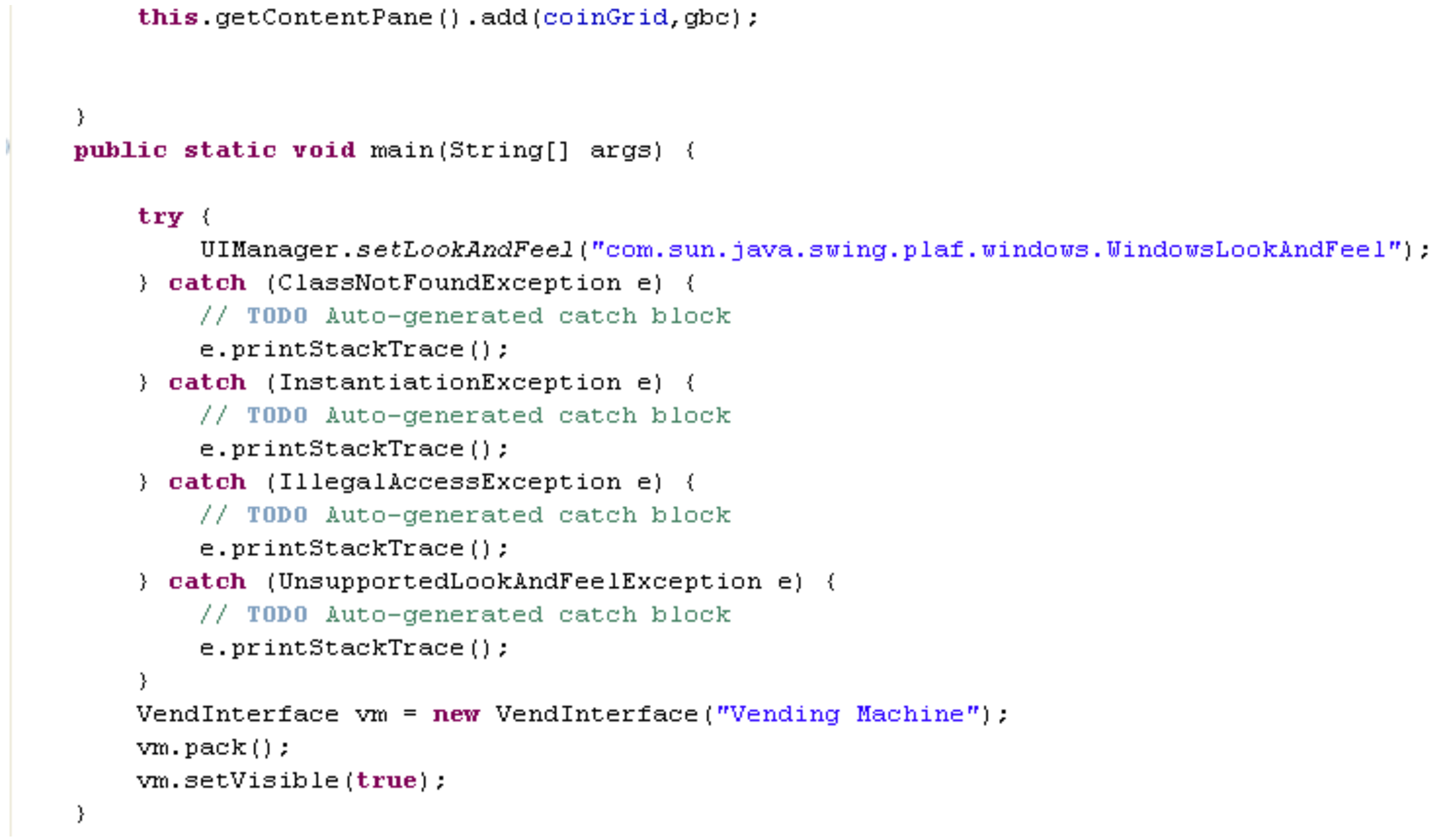
```
//create item Panel and add item buttons
keyPad = new JPanel();
keyPad.setLayout(new GridLayout(4,4));
for (int i=0;i<keypadButtons.length;i++) {
    keypadButtons[i] = new JButton(keyNames[i]);
    keypadButtons[i].setPreferredSize(new Dimension(50,50));
    keypadButtons[i].setFont(new Font("Serif",Font.PLAIN,12));
    keyPad.add(keypadButtons[i]);
}

//create cancel button
cancelButton = new JButton("Cancel", new ImageIcon("cancel.jpg"));
cancelButton.setFont(new Font("Serif",Font.BOLD,12));
cancelButton.setBackground(Color.WHITE);
cancelButton.setPreferredSize(new Dimension(200,50));

//create display label
lcd = new JLabel("Enter Coins:");
lcd.setHorizontalAlignment(SwingConstants.LEFT);
lcd.setVerticalAlignment(SwingConstants.TOP);
lcd.setFont(new Font("Serif",Font.BOLD,12));
lcd.setPreferredSize(new Dimension(196,48));
displayPanel = new JPanel();
displayPanel.setBackground(Color.GREEN);
displayPanel.setPreferredSize(new Dimension(200,50));
displayPanel.setBorder(new LineBorder(Color.GRAY));
displayPanel.add(lcd);
```

Vending Machine GUI



```
displayPanel.add(lcd);


this.setLayout(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridheight = 4;
gbc.insets = new Insets(0,0,0,10);
this.getContentPane().add(keyPad,gbc);

gbc.gridx = 1;
gbc.gridy = 0;
gbc.gridheight = 1;
gbc.insets = new Insets(0,0,0,0);
this.getContentPane().add(displayPanel,gbc);

gbc.gridx = 1;
gbc.gridy = 1;
gbc.gridheight = 1;
this.getContentPane().add(cancelButton,gbc);

gbc.gridx = 1;
gbc.gridy = 2;
gbc.gridheight = 2;
this.getContentPane().add(coinGrid,gbc);
```

Vending Machine GUI



```
        this.getContentPane().add(coinGrid,gbc);


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

        try {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InstantiationException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (UnsupportedLookAndFeelException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        VendInterface vm = new VendInterface("Vending Machine");
        vm.pack();
        vm.setVisible(true);
    }
```