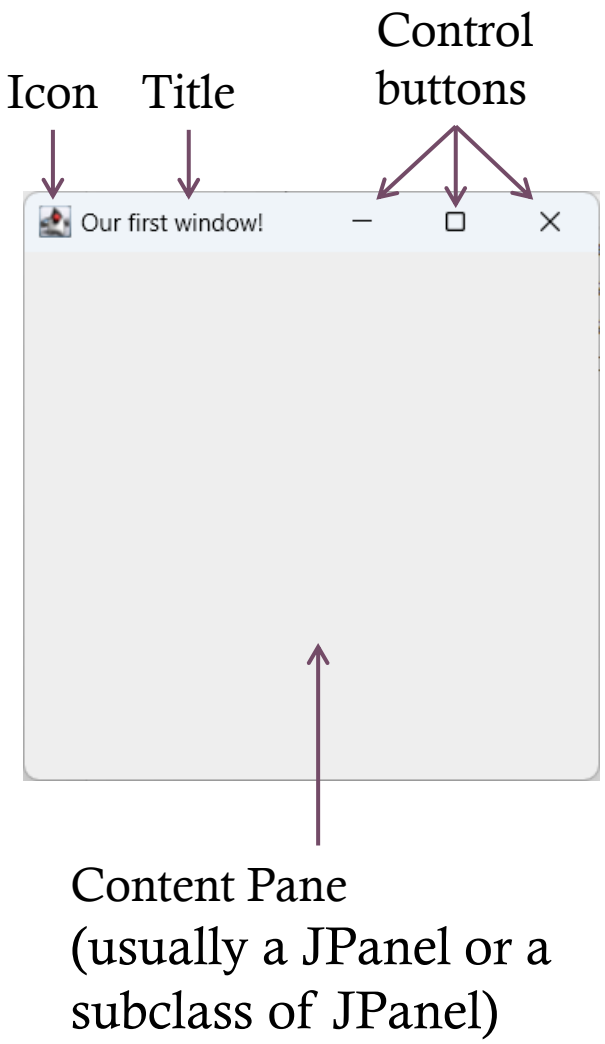
WEEK FOUR

Acknowledgements: Slides created based off material provided by Dr. Michael Raymer and Dr. Travis Doom

CREATING A WINDOW

```
public static void main(String args[]){
    JFrame theWindow = new JFrame("Our first window!");
    theWindow.setSize(300, 300);
    theWindow.setLocation(200, 400);
    theWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    theWindow.setVisible(true);
}
```



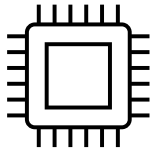
OUR FIRST EXECUTION THREAD

- Why is the program still running when main() is done?

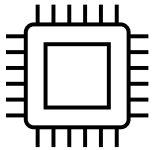
```
public static void main(String args[]){
     JFrame theWindow = new JFrame("Our first window!");
     theWindow.setSize(300, 300);
     theWindow.setLocation(200, 400);
     theWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     theWindow.setVisible(true);
     System.out.println("Done!");
 }
```

The Event Dispatch Thread (EDT) waits for clicks, drags, re-sizes, keyclicks, and other events and responds to them.

OUR FIRST EXECUTION THREAD



```java
public static void main(String args[]){
    JFrame theWindow = new JFrame("Our first window!");
    theWindow.setSize(300, 300);
    theWindow.setLocation(200, 400);
    theWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    theWindow.setVisible(true);
    System.out.println("Done!");
}
```



```java
while (window-open){
    if (button-clicked){
        do something;
    }
    else if (window-resized){
        do something;
    }
    else if …
}
```

KEEPING TRACK OF DATA

- If main() is going to exit, where do we keep all our variables and data?

- There are several approaches, but we'll usually create a **subclass of JFrame** and use **instance variables** for all our persistent data.
- This data will live for as long as our main application window is not closed.

KEEPING TRACK OF DATA

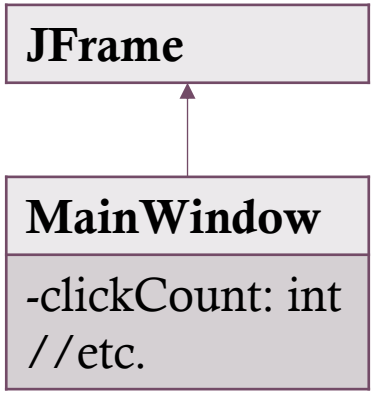
```
class MainWindow extends JFrame{
    private int clickCount;

    public MainWindow(String title){
        super(title);
        clickCount = 0;
    }
    public static void main(String args[]){
        JFrame theWindow = new MainWindow("Our first window!");
        theWindow.setSize(300, 300);
        theWindow.setLocation(200, 400);
        theWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        theWindow.setVisible(true);
        System.out.println("Done!");
    }
}
```

Data I want to keep around until the main window is closed.

Utilizing inheritance in this way is extremely useful!

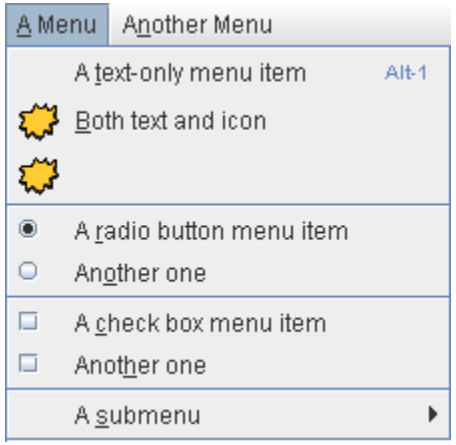
KEEPING TRACK OF DATA

```
class MainWindow extends JFrame{
    private int clickCount;

    public MainWindow(String title){
        super(title);
        clickCount = 0;
    }
    public static void main(String args[]){
        JFrame theWindow = new MainWindow("Our first window!");
        theWindow.setSize(300, 300);
        theWindow.setLocation(200, 400);
        theWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        theWindow.setVisible(true);
        System.out.println("Done!");
    }
}
```



- I can store any data I want to be persistent here
- It is accessible to any method of MainWindow

GUI ELEMENTS IN SWING



- `import javax.swing.*;`
- Includes many useful GUI elements:
  - JButton
  - JLabel
  - JCheckBox
  - JRadioButton and ButtonGroup
  - JList
  - JMenuBar, Jmenu, and JMenuItem
  - JComboBox
  - JSlider, JScrollBar

Image source: https://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

JBUTTONS

- Clickable buttons
- Can be labeled, arranged, etc.
- We can listen for the button to be pressed

```
public MainWindow(String title){
    super(title);
    clickCount = 0;

    JButton aButton = new JButton("Click me!");
    this.add(aButton);
}
```

ADDING GUI ELEMENTS

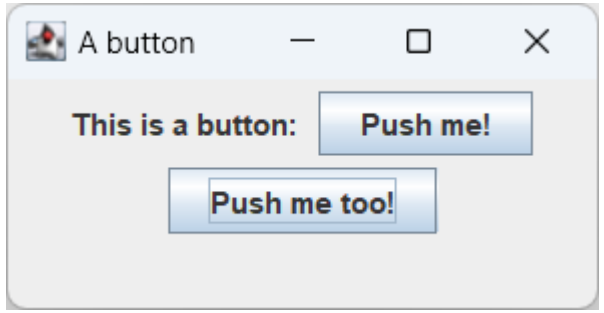
- However, the frame will only show the most recently added item

```
public MainWindow(String title){
    super(title);
    clickCount = 0;

    JButton aButton = new JButton("Click me!");
    this.add(aButton);
    JButton otherButton = new JButton("Click me too!");
    this.add(otherButton);

}
```

- How can we display more than one element?

JPANEL (AND LAYOUT MANAGERS) TO THE RESCUE!

- JPanel can hold many GUI elements and allows you to set a Layout Manager to keep them neatly arranged

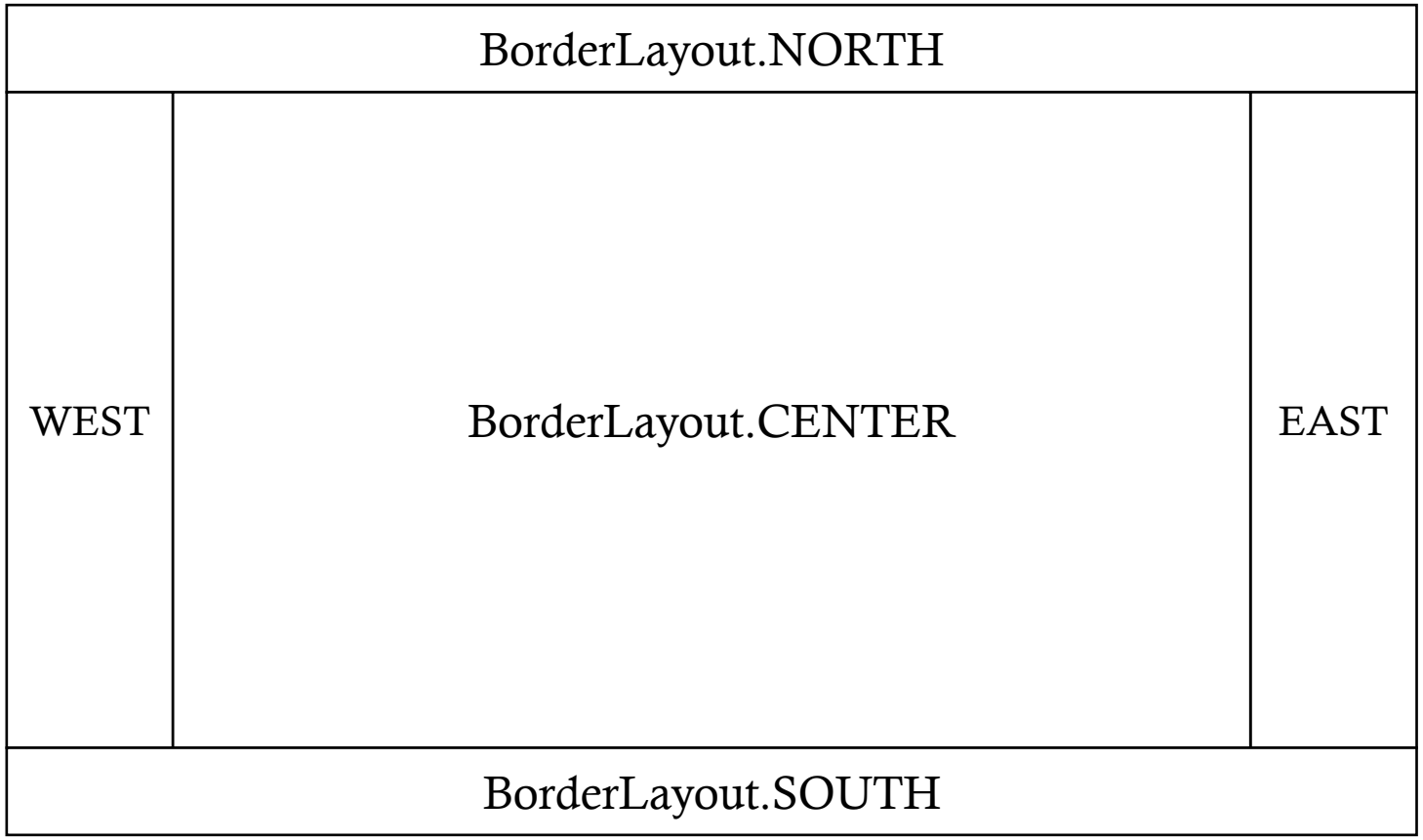


JPANEL (AND LAYOUT MANAGERS) TO THE RESCUE!

- The default layout manager is called FlowLayout
- Elements are added to a row until there is no more space, then starts another row

```
public MainWindow(String title){
    super(title);
    clickCount = 0;

    JPanel content = new JPanel();
    this.setContentPane(content);

    JButton aButton = new JButton("Click me!");
    content.add(aButton);
    JButton otherButton = new JButton("Click me too!");
    content.add(otherButton);

}
```

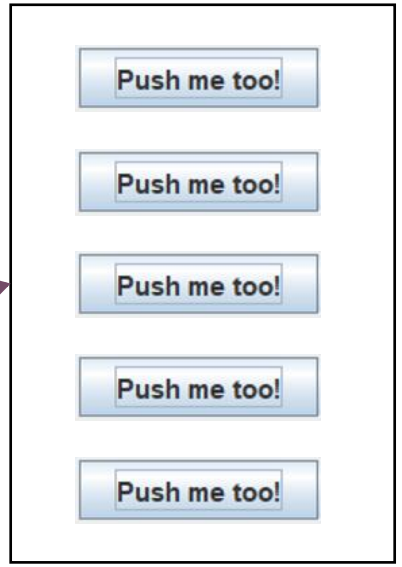
BORDERLAYOUT



BOXLAYOUT

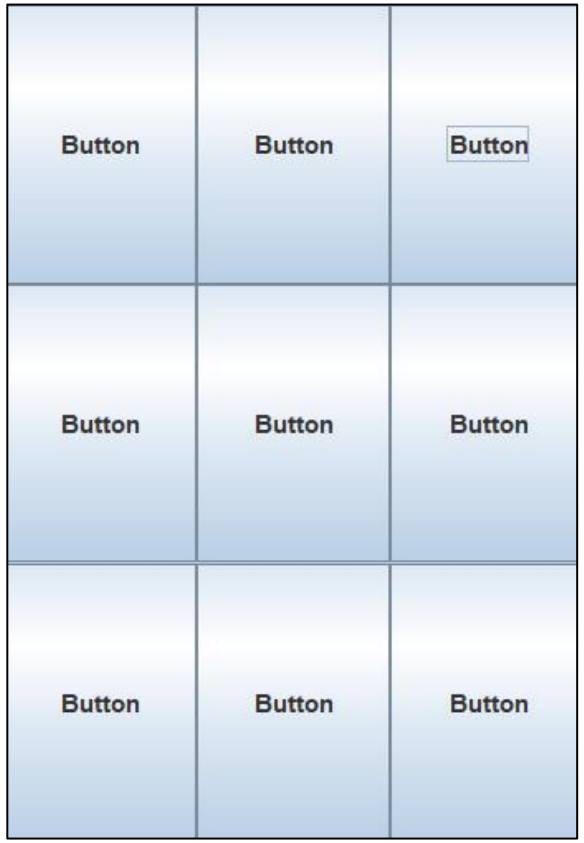
```
content.setLayout(new BoxLayout(content, BoxLayout.X_AXIS));
```


```
content.setLayout(new BoxLayout(content, BoxLayout.Y_AXIS));
```



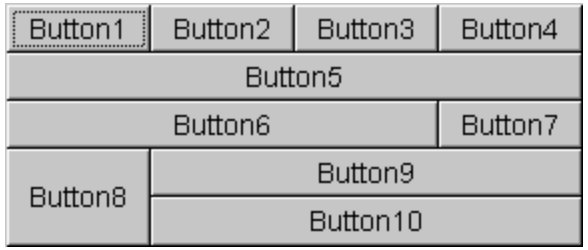
GRIDLAYOUT

```
content.setLayout(new GridLayout(3, 3));
```



GRIDBAGLAYOUT

- Dynamic Grid – created as you add elements
- Set Sizes and insets (padding) as you go
- You can span rows and columns
- Complex, but powerful
- See examples in ZyBooks



https://docs.oracle.com/javase/8/docs/api/java/awt/GridBagLayout.html