Week 5 - Friday

COMP 2000

Last time

- What did we talk about last time?
- GUIs
- **`JOptionPane`**

Questions?

Project 2

JFrame

JFrame

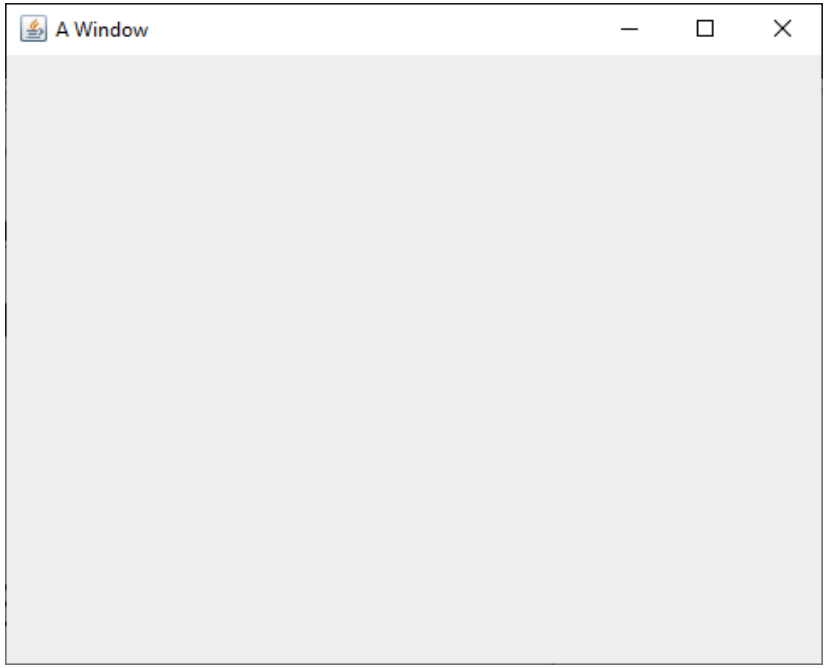
- **`JOptionPane`** was fine for creating a limited range of dialogs
- If we want to make a whole window, we use **`JFrame`**
- Java uses the term frame instead of window, probably because of concerns about lawsuits from Microsoft
- But when you hear **`JFrame`**, think "main window"

Creating or extending

- When designing a **`JFrame`**, there are two meaningful options:
  - Creating a **`JFrame`** object and adding stuff to it inside of some other class
  - Extending **`JFrame`** with your own class, making your class a **`JFrame`** plus more
- It doesn't really matter which one you pick
- To keep things simple, we'll create a **`JFrame`** object instead of extending the **`JFrame`** class

Creating a JFrame

- To create a **JFrame**, we will usually call its constructor that takes a **String**, giving it a title
- Then, we have to make it visible so that we can see it




```
JFrame frame = new JFrame("A Window");
frame.setVisible(true);
```

setSize()

- The code from the previous slide will make a **JFrame** and make it visible
- However, it will probably be so small that you won't even notice it
- To deal with this problem, you should set its size, ideally before you make it visible
  - Its **setSize()** method takes two **int** values: width and height in pixels
- Eventually, once we add widgets to a **JFrame**, we can simply call its **pack()** method, which will make it take up the amount of space it needs to fit everything

```
JFrame frame = new JFrame("A Window");
frame.setSize(500, 400);
frame.setVisible(true);
```

setDefaultCloseOperation()

- Next, you'll notice that closing the window doesn't end the program
  - The little red square on the Eclipse Console is still clickable, meaning that the program is running
- By default, closing the window by clicking its X only hides the window
- By calling the **`setDefaultCloseOperation()`**, we can make it so that the default operations is dispose (getting rid of the window)

```
JFrame frame = new JFrame("A Window");
frame.setSize(500, 400);
frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
frame.setVisible(true);
```

- Many books suggest passing in **`JFrame.EXIT_ON_CLOSE`**, but you **should not!**
- Doing so will kill the rest of your program like **`System.exit()`**

Recap

- To use a **`JFrame`** you must:
  - Create a **`JFrame`** object
  - Set its size (either directly or by putting widgets on it and then calling **`pack()`**)
  - Set its default close operation to dispose
  - Make it visible
- Now that we've got a window, we can put widgets on it!

Widgets

Widgets

- **Widget** is a generic term for a wide range of GUI controls
  - Buttons
  - Labels (allowing us to put text or images on a GUI)
  - Text fields
  - Text areas (like text fields but larger)
  - Menus
  - Checkboxes
  - Radio buttons
  - Lists
  - Combo boxes
  - Sliders

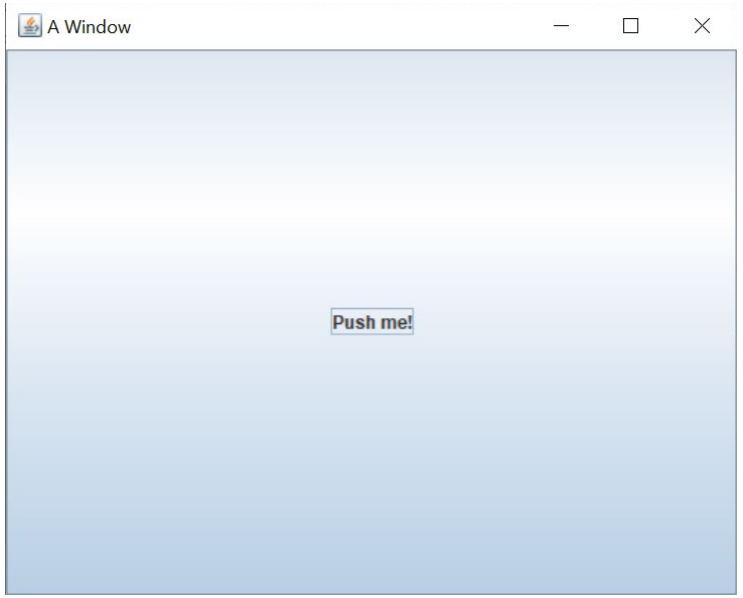
JButton

- A button you can click on is provided by the **JButton** class
- A **JButton** is usually created with text or an image
  - You'll need to make **JButton**s with images for Project 2
- Just creating the **JButton** doesn't do anything
- You have to add it to a **JFrame** (or other container) to see it
- Right now, we're just creating the buttons
- Next week, we'll learn how to add actions to them

```
JButton button = new JButton("Push me!");
```

Adding a JButton to a JFrame

- Once you've created a **JButton**, you can add it to a **JFrame** by calling the **add()** method on the **JFrame**
- All GUI containers have an **add()** method that allows us to add a widget to it

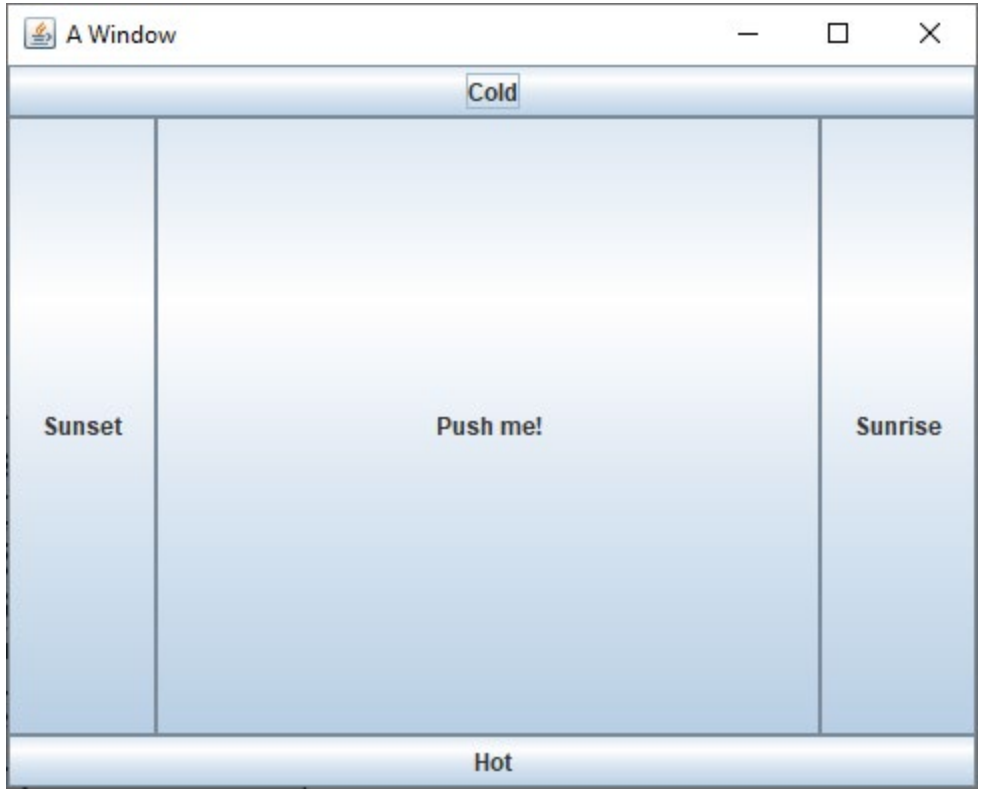



```
JFrame frame = new JFrame("A Window");
frame.setSize(500, 400);
frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
JButton button = new JButton("Push me!");
frame.add(button);
frame.setVisible(true);
```

Adding to different parts of a `JFrame`

- By default, a **`JFrame`** uses a layout manager called the **`BorderLayout`** that has five region
- Calling the simplest **`add()`** method adds a widget to the center, which stretches to take up all available space
- You can specify that you're adding to:
  - **`BorderLayout.CENTER`**
  - **`BorderLayout.NORTH`**
  - **`BorderLayout.SOUTH`**
  - **`BorderLayout.EAST`**
  - **`BorderLayout.WEST`**

```
JButton centerButton = new JButton("Push me!");
frame.add(centerButton, BorderLayout.CENTER);
JButton northButton = new JButton("Cold");
frame.add(northButton, BorderLayout.NORTH);
JButton southButton = new JButton("Hot");
frame.add(southButton, BorderLayout.SOUTH);
JButton eastButton = new JButton("Sunrise");
frame.add(eastButton, BorderLayout.EAST);
JButton westButton = new JButton("Sunset");
frame.add(westButton, BorderLayout.WEST);
```



Displaying an icon on a JButton

- You can also make a **JButton** with an image instead of text
- To do so, you create an **ImageIcon** and pass that to the constructor of the **JButton**
- You'll need the path to an image


```
JButton bowieButton = new JButton(new ImageIcon("bowie.jpg"));
frame.add(bowieButton, BorderLayout.CENTER);
```

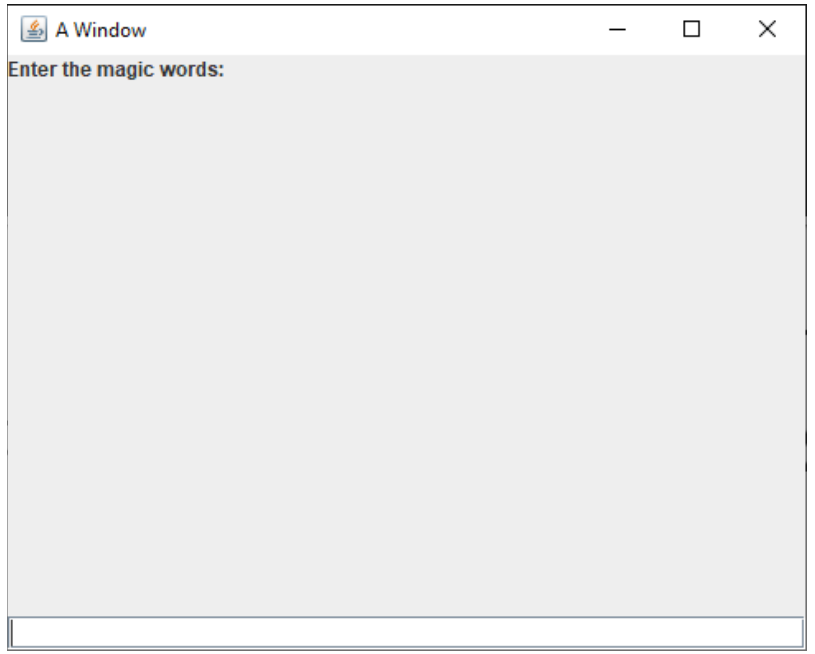
JLabel

- A **JLabel** is like a button you can't click
- Its constructors work just like the **JButton** ones
- It allows you to display text or an image


```
JLabel nameLabel = new JLabel("David Bowie");
JLabel bowieLabel = new JLabel(new ImageIcon("bowie.jpg"));
frame.add(nameLabel, BorderLayout.NORTH);
frame.add(bowieLabel, BorderLayout.CENTER);
```

JTextField

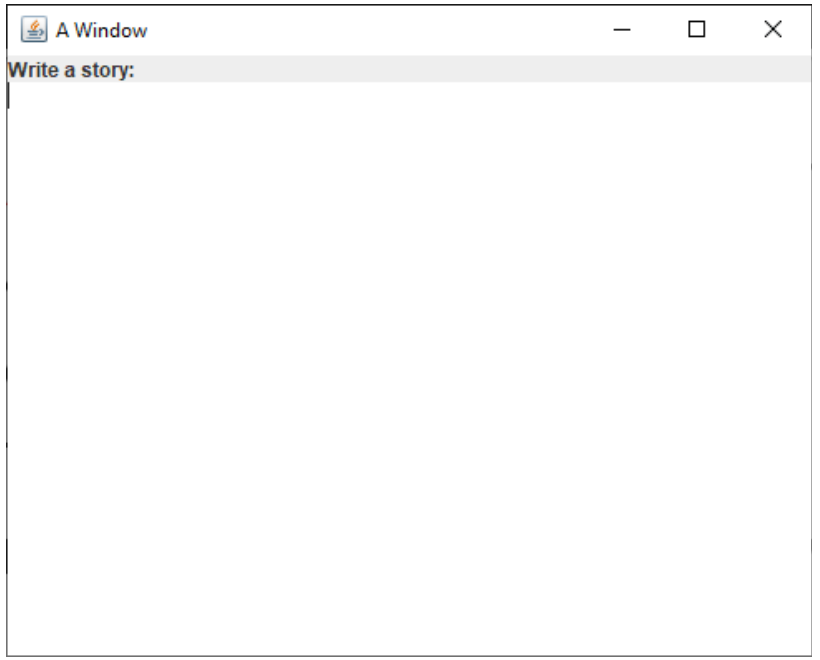
- A **JTextField** allows a user to enter a (short) amount of text
- Usually, you'll need a **JLabel** to tell the person what they should enter
- The example is ugly because the **JLabel** and the **JTextField** don't fill the 500 x 400 **JFrame**




```
JLabel messageLabel = new JLabel("Enter the magic words:");
JTextField magicField = new JTextField();
frame.add(messageLabel, BorderLayout.NORTH);
frame.add(magicField, BorderLayout.SOUTH);
```

JTextArea

- A **JTextField** is for entering small pieces of information
  - Name
  - Address
  - Telephone number
- For larger texts, we can use a **JTextArea**




```
JLabel storyLabel = new JLabel("Write a story:");
JTextArea storyArea = new JTextArea();
frame.add(storyLabel, BorderLayout.NORTH);
frame.add(storyArea, BorderLayout.CENTER);
```

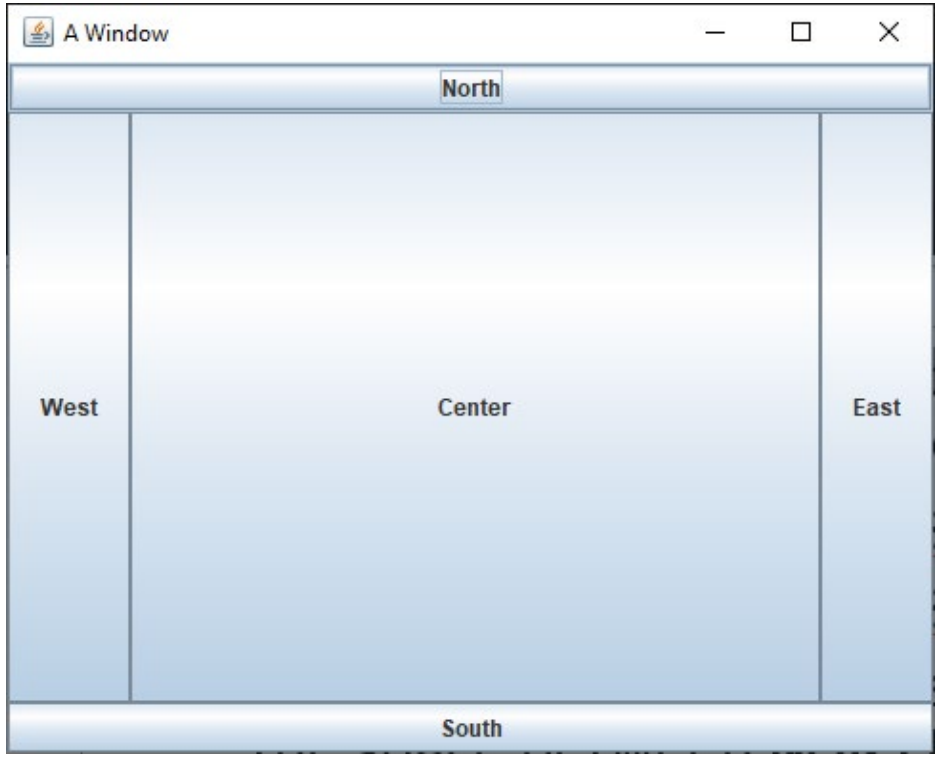
Layout Managers

Layout managers

- When you add a widget to a **`JFrame`** (or to a **`JPanel`**), its layout manager determines how it will be arranged
- There are lots of layout managers, but it's worth mentioning four:
  - **`BorderLayout`**
  - **`GridLayout`**
  - **`FlowLayout`**
  - **`BoxLayout`**
- Note that we won't talk about **`BoxLayout`**, but you should look it up if you get serious about Swing GUIs
- **`BoxLayout`** makes it easy to arrange widgets in a horizontal or vertical line, with different amount of spacing between widgets

BorderLayout

- **`BorderLayout`** is the default layout for **`JFrame`**
- When you add widgets, you can specify the location as one of five regions:
  - **`BorderLayout.NORTH`** stretches the width of the container on the top
  - **`BorderLayout.SOUTH`** stretches the width of the container on the bottom
  - **`BorderLayout.EAST`** sits on the right of the container, stretching to fill all the space between **`NORTH`** and **`SOUTH`**
  - **`BorderLayout.WEST`** sits on the left of the container, stretching to fill all the space between **`NORTH`** and **`SOUTH`**
  - **`BorderLayout.CENTER`** sits in the middle of the container and stretches to fill all available space
- If you don't specify where you're adding a widget, it adds to **`CENTER`**
- If you add more than one widget to a region, the new one **replaces** the old
- Unused regions disappear



GridLayout

- **GridLayout** allows you to create a grid with a specific number of rows and columns
- All the cells in the grid are the same size
- As you add widgets, they fill each row


```
frame.setLayout(new GridLayout(4, 5));
for(int row = 0; row < 4; ++row)
  for(int column = 0; column < 5; ++column)
     frame.add(new JButton("" + (row * 5 + column + 1)));
```

Quiz

Upcoming

Next time…

- More on layout
- Action listeners

Reminders

- Keep reading Chapter 15
- Keep working on Project 2