CS 106A, Lecture 22
Interactors

suggested reading:

Java Ch. 10.5-10.6

Plan for today

- Recap: Inheritance
- Extending GCanvas
- *Example*: Aquarium
- Interactors
  - JButton
  - JLabel
  - JTextField
- Example: TipCalculator

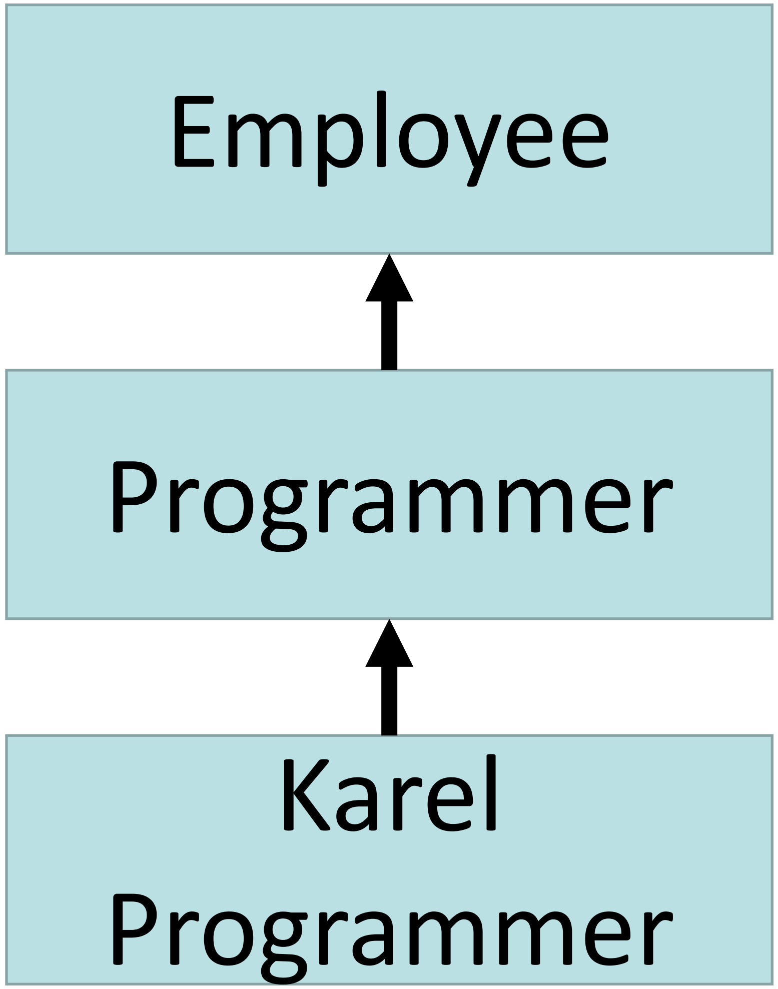
Plan for today

- Recap: Inheritance
- Extending GCanvas
- *Example*: Aquarium
- Interactors
  - JButton
  - JLabel
  - JTextField
- Example: TipCalculator

Inheritance

Inheritance lets us relate our variable types to one another.

Inheritance



Variable types can seem to “inherit” from each other. We don’t want to have to duplicate code for each one!

Using Inheritance

```
public class Name extends Superclass {
```

– Example:

```
public class Programmer extends Employee {
    ...
}
```

- By extending Employee, this tells Java that `Programmer` can do **everything an Employee can do, plus more**.
- Programmer automatically inherits all of the code from Employee!
- The **superclass** is Employee, the **subclass** is Programmer.

Example: Programmer

```
public class Programmer extends Employee {
      private int timeCoding;
      ...
      public void code() {
            timeCoding += 10;
      }
}

...

Programmer rishi = new Programmer(“Rishi”);
rishi.code();          // from Programmer
rishi.promote();       // from Employee!
```

Example: KarelProgrammer

```
public class KarelProgrammer extends Programmer {
      private int numBeepersPicked();
      ...
      public void pickBeepers() {
            numBeepersPicked += 2;
      }
}

...
KarelProgrammer nick = new KarelProgrammer("Nick");
nick.pickBeepers();            // from KarelProgrammer
nick.code();                   // from Programmer!
nick.promote();                // From Employee!
```

Plan for today

- Recap: Inheritance
- Extending GCanvas
- *Example*: Aquarium
- Interactors
  - JButton
  - JLabel
  - JTextField
- Example: TipCalculator

GCanvas

- A **GCanvas** is the canvas area that displays all graphical objects in a **GraphicsProgram**.
- When you create a **GraphicsProgram**, it automatically creates a **GCanvas** for itself, puts it on the screen, and uses it to add all graphical shapes.

- **GCanvas** is the one that contains methods like:
  - getElementAt
  - add
  - getWidth
  - getHeight
  - …

GCanvas

```
public class Graphics extends GraphicsProgram {
	public void run() {
		// A GCanvas has been created for us!
		GRect rect = new GRect(50, 50);
		add(rect); // adds to the GCanvas!


		...
		// Checks our GCanvas for elements!
		GObject obj = getElementAt(25, 25);
	}
}
```

Extending GCanvas

```
public class Graphics extends Program {
      public void run() {
            // We have to make our own GCanvas now
            MyCanvas canvas = new MyCanvas();
            add(canvas);

            // Operate on this canvas
            GObject obj = canvas.getElementAt(…);
      }
}
```

Extending GCanvas

```
public class MyCanvas extends GCanvas {
	public void addCenteredSquare(int size) {
		GRect rect = new GRect(size, size);
		int x = getWidth() / 2.0 -
			rect.getWidth() / 2.0;
		int y = getHeight() / 2.0 -
			rect.getHeight() / 2.0;
		add(rect, x, y);
	}
}
```

Extending GCanvas

```
public class Graphics extends Program {
      public void run() {
            // We have to make our own GCanvas now
            MyCanvas canvas = new MyCanvas();
            add(canvas);

            canvas.addCenteredSquare(20);
      }
}
```

Extending GCanvas

- Sometimes, we want to be able to have all of our graphics-related code in a separate file.
- To do this, instead of using the provided **GraphicsProgram** canvas, we **define our own subclass of GCanvas,** have our program **extend Program**, and add our own canvas ourselves.
- Then, all graphics-related code can go in our **GCanvas** subclass.

The init method

- **init** is a special public method, like **run**, that is called when your program is being initialized.
- Unlike **run**, however, it is called *before* your program launches, letting you do any initialization you need.

```
public class MyProgram extends GraphicsProgram {
     public void init() {
          // executed before program launches
     }

     public void run() {
          // executed after program launches
     }
}
```

The init method

- **`init`** is typically used to initialize graphical components, such as adding a custom **GCanvas** to the screen.

```
public class MyProgram extends Program {
    private MyCanvas canvas;
    public void init() {
        canvas = new MyCanvas();
        add(canvas);
    }

    public void run() {
        canvas.addCenteredSquare(20);
    }
}
```

Common Bugs

- When you are using a custom canvas, make sure to not call **`getWidth`** or **`getHeight`** on the canvas until it is shown onscreen!

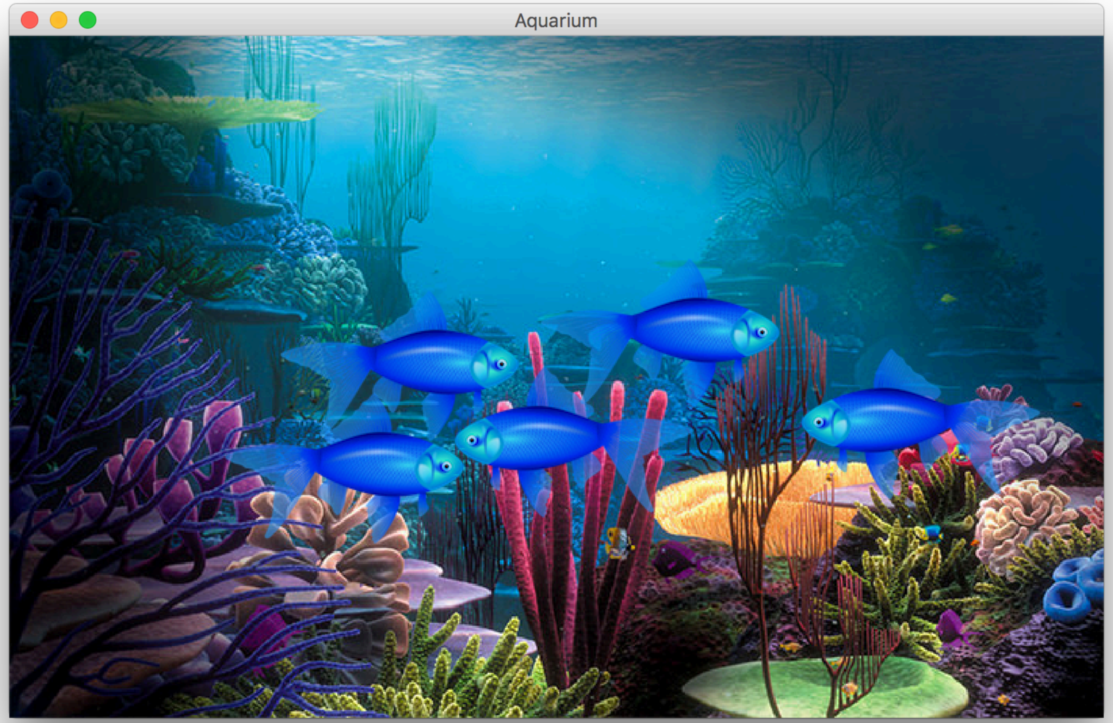
```
public class MyProgram extends Program {
	private MyCanvas canvas;
	public void init() {
		// canvas not created yet!
		canvas = new MyCanvas();
		// canvas not added yet!
		add(canvas);
		// window not showing yet!
	}
	public void run() {
		// good to go
	}
}
```

Plan for today

- Recap: Inheritance
- Extending GCanvas
- *Example*: Aquarium
- Interactors
  - JButton
  - JLabel
  - JTextField
- Example: TipCalculator

Example: Aquarium

- Let’s write a graphical program called **Aquarium** that simulates fish swimming around.
- To decompose our code, we can make our own **GCanvas** subclass.



Plan for today

- Recap: Inheritance
- Extending GCanvas
- *Example*: Aquarium
- Interactors
  - JButton
  - JLabel
  - JTextField
- Example: TipCalculator

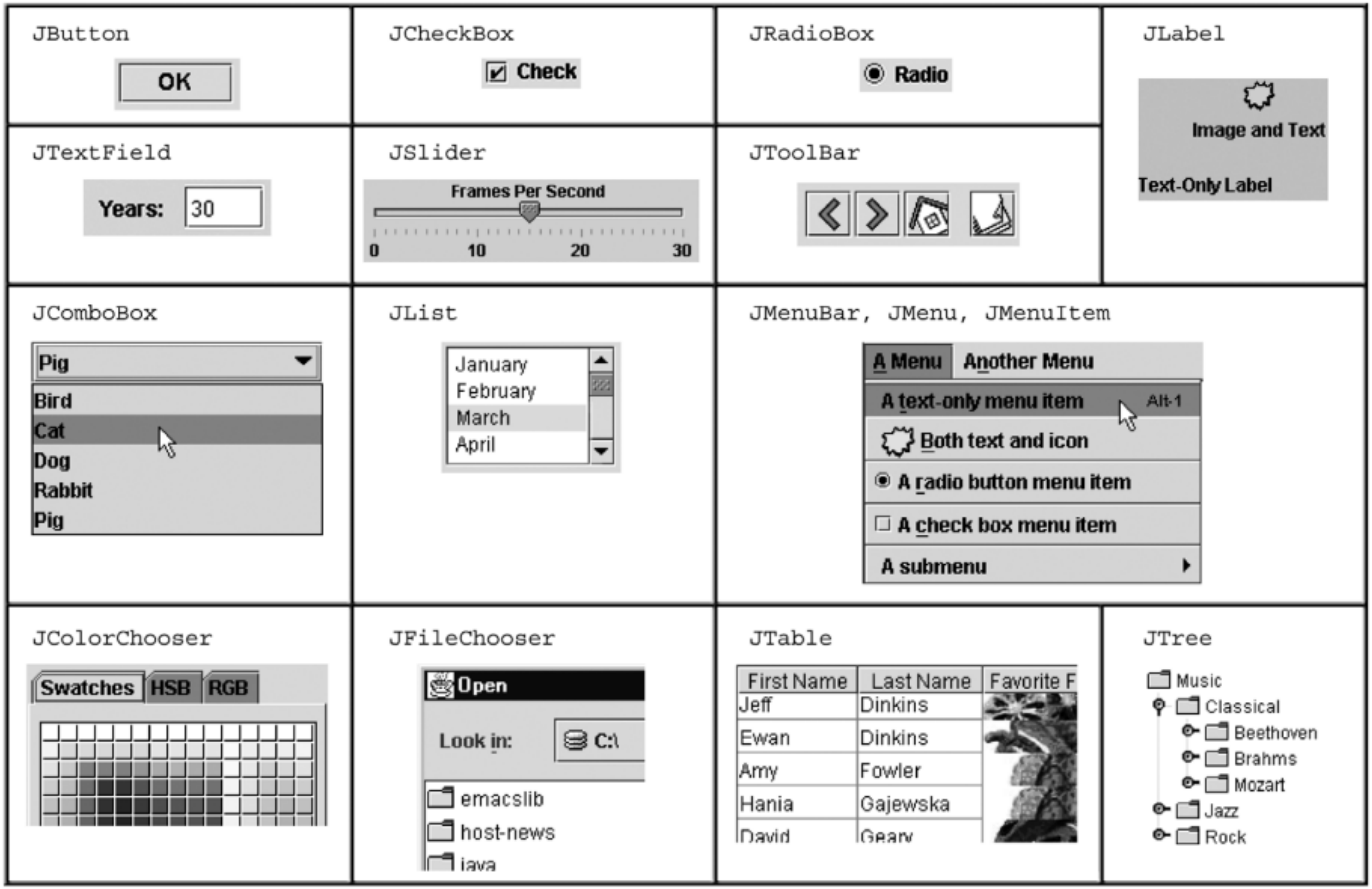
Interactive Programs

So far, we have learned about two ways of making interactive programs:

- Reading user input in ConsolePrograms
- Detecting mouse events in GraphicsPrograms

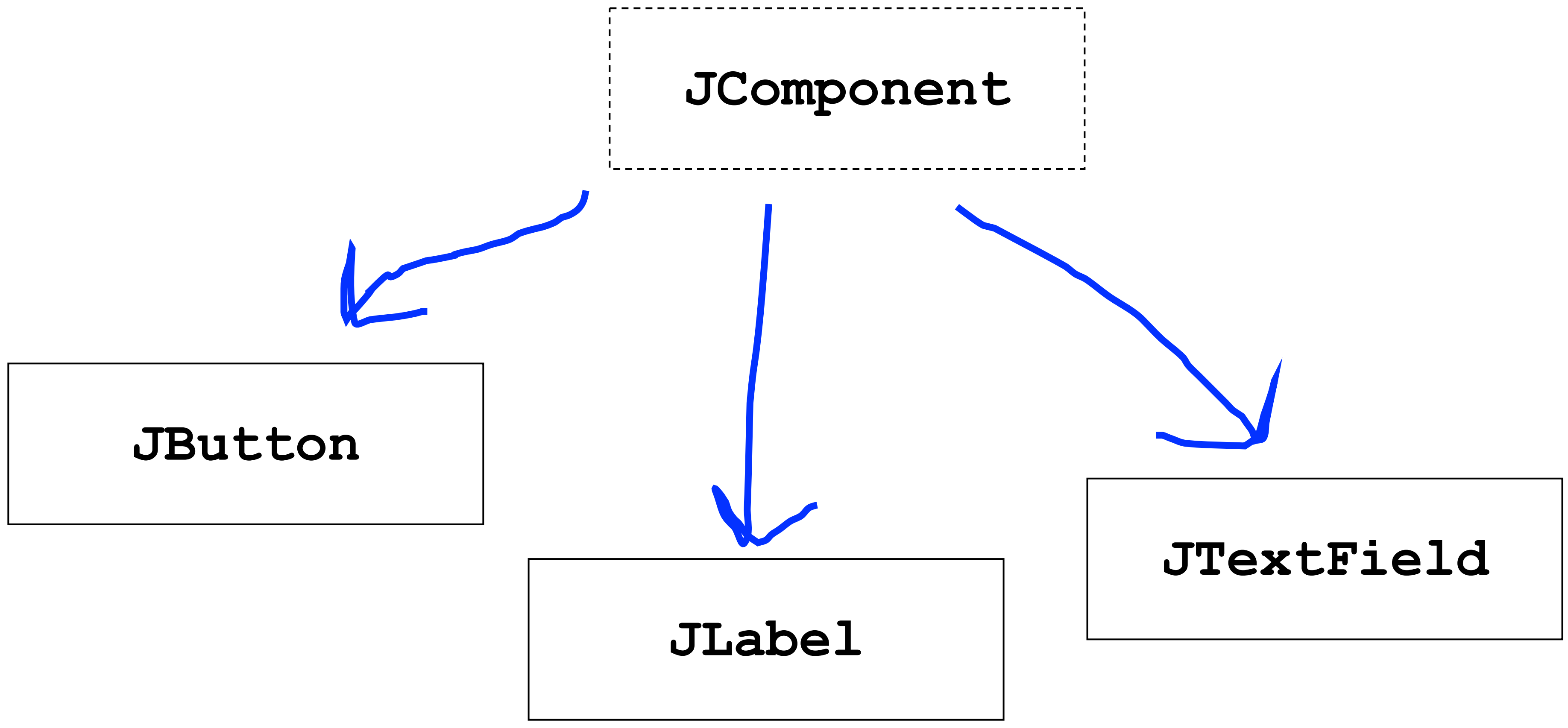
Today, we're going to learn about a third: *interactors*.

Interactors



First Name	Last Name	Favorite F
Jeff	Dinkins	
Ewan	Dinkins	
Amy	Fowler	
Hania	Gajewska	
David	Geary	

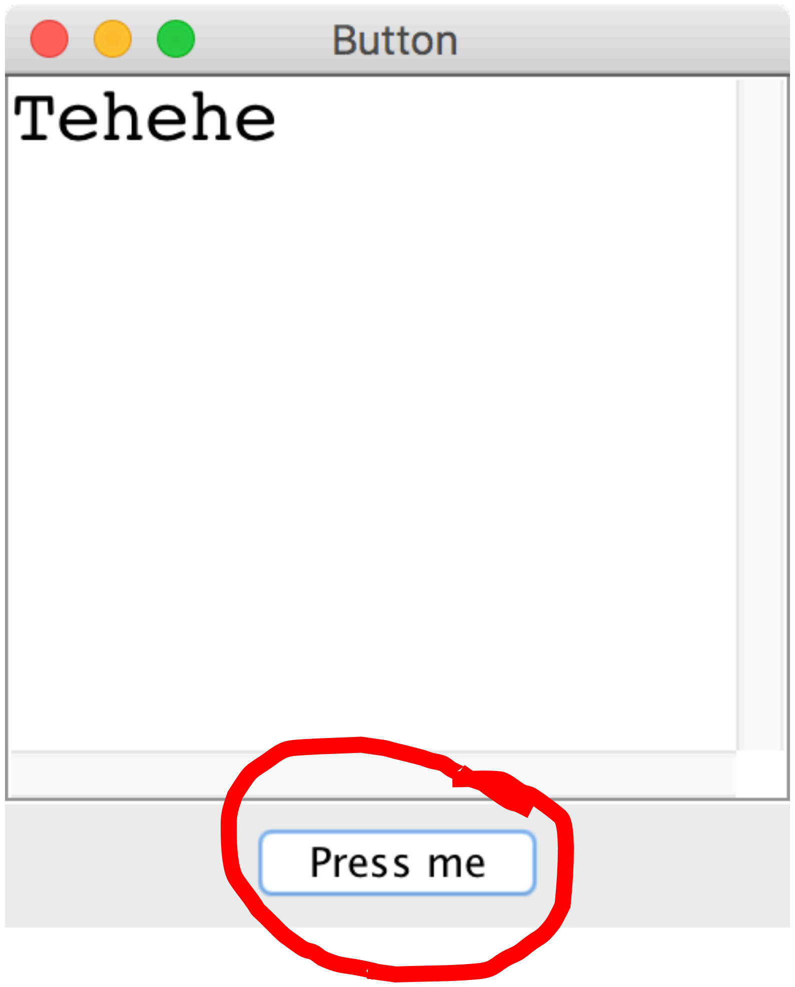
Interactors



Plan for today

- Recap: Inheritance
- Extending GCanvas
- *Example*: Aquarium
- Interactors
  - JButton
  - JLabel
  - JTextField
- Example: TipCalculator

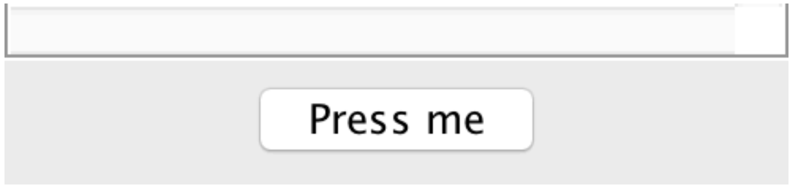
JButton



Tehehe

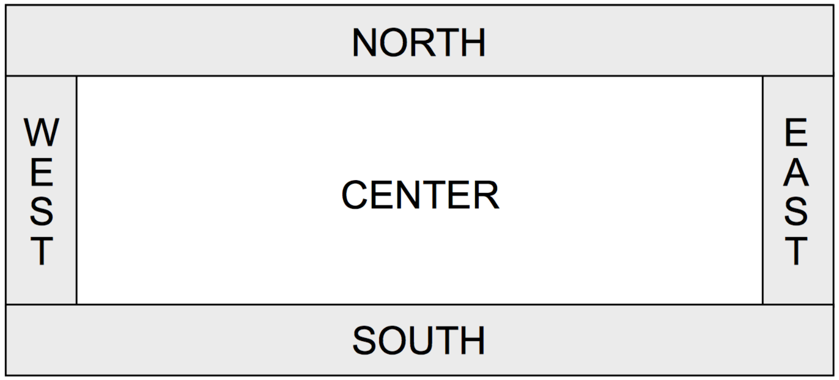
JButton

```
import java.awt.event.*;
import javax.swing.*;



JButton button = new JButton(“Press me”);
add(button, SOUTH);
```



Window Regions

- In graphics or console programs, the window is divided into five regions:



- The **CENTER** region is typically where the action happens.
  - **ConsoleProgram** adds a console there
  - **GraphicsProgram** puts a **GCanvas** there
- Other regions are visible only if you add an interactor to them using `add(`***`component, REGION`***`);`
- Interactors are automatically centered within each region.

Responding To Button Clicks

To respond to events from interactors, we must do the following:

1. Call **`addActionListeners()`** at the end of init, *once we are done adding buttons*. This tells Java to let us know if any of the previous buttons were clicked.
2. Implement the public **`actionPerformed`** method. This method is called whenever a button is clicked.

JButton Example

```
public class Interactors extends ConsoleProgram {
     public void init() {
          JButton yayButton = new JButton("Yay");
          add(yayButton, SOUTH);
          JButton nayButton = new JButton("Nay");
          add(nayButton, SOUTH);
          addActionListeners();
     }

     public void actionPerformed(ActionEvent event) {
          ... // ?
     }
}
```

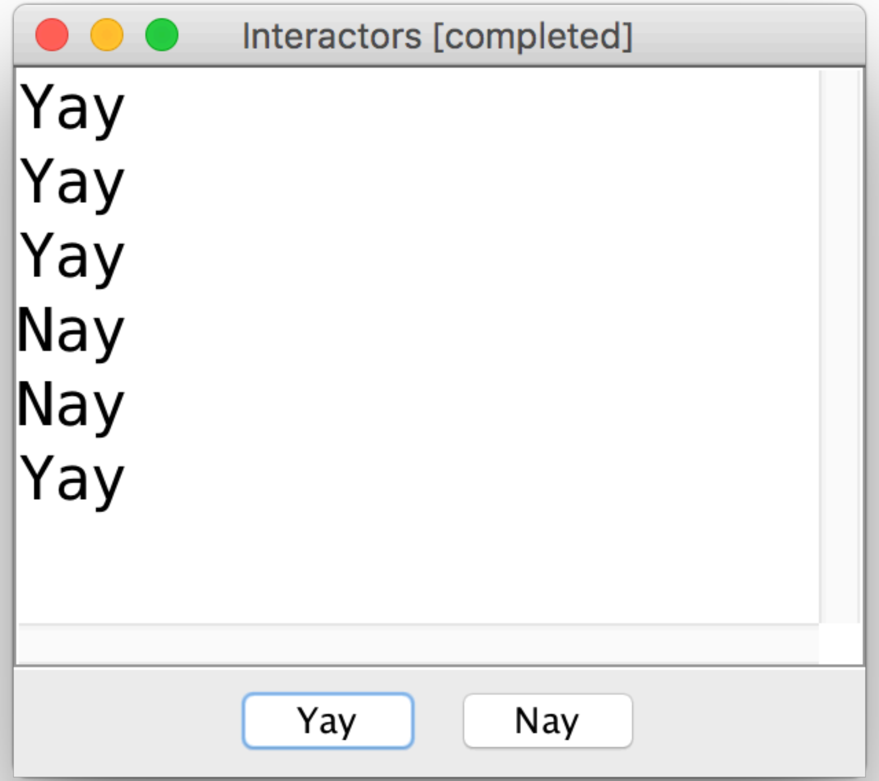
ActionEvent

- The **`ActionEvent`** parameter contains useful event information.
  - Use `getSource` or `getActionCommand` to figure out what button or component was interacted with.

Method	Description
e`.getActionCommand()`	a text description of the event *(e.g. the text of the button clicked)*
e`.getSource()`	the interactor that generated the event

```
public void actionPerformed(ActionEvent event) {
    String command = event.getActionCommand();
    if (command.equals("Save File")) {
        // user clicked the Save File button
        ...
    }
}
```

JBytton Example



```
Yay
Yay
Yay
Nay
Nay
Yay
```

JBillion Example

```
public class Interactors extends ConsoleProgram {
	private JButton yayButton;
	private JButton nayButton;
	public void init() {
		yayButton = new JButton("Yay");
		add(yayButton, SOUTH);
		nayButton = new JButton("Nay");
		add(nayButton, SOUTH);
		addActionListeners();
	}

	public void actionPerformed(ActionEvent event) {
		if (event.getSource() == yayButton) {
			println("Yay");
		} else if (event.getSource() == nayButton) {
			println("Nay");
		}
	}
}
```

JButton Example

```
public class Interactors extends ConsoleProgram {
        private JButton yayButton;
        private JButton nayButton;
        public void init() {
                yayButton = new JButton("Yay");
                add(yayButton, SOUTH);
                nayButton = new JButton("Nay");
                add(nayButton, SOUTH);
                addActionListeners();
        }

        public void actionPerformed(ActionEvent event) {
                if (event.getSource() == yayButton) {
                        println("Yay");
                } else if (event.getSource() == nayButton) {
                        println("Nay");
                }
        }
}
```

JButton Example

```
public class Interactors extends ConsoleProgram {
        private JButton yayButton;
        private JButton nayButton;
        public void init() {
                yayButton = new JButton("Yay");
                add(yayButton, SOUTH);
                nayButton = new JButton("Nay");
                add(nayButton, SOUTH);
                addActionListeners();
        }

        public void actionPerformed(ActionEvent event) {
                if (event.getSource() == yayButton) {
                        println("Yay");
                } else if (event.getSource() == nayButton) {
                        println("Nay");
                }
        }
}
```

JButton Example

```
public class Interactors extends ConsoleProgram {
        private JButton yayButton;
        private JButton nayButton;
        public void init() {
                yayButton = new JButton("Yay");
                add(yayButton, SOUTH);
                nayButton = new JButton("Nay");
                add(nayButton, SOUTH);
                addActionListeners();
        }

        public void actionPerformed(ActionEvent event) {
                if (event.getSource() == yayButton) {
                        println("Yay");
                } else if (event.getSource() == nayButton) {
                        println("Nay");
                }
        }
}
```

JButton Example

```
public class Interactors extends ConsoleProgram {
        private JButton yayButton;
        private JButton nayButton;
        public void init() {
                yayButton = new JButton("Yay");
                add(yayButton, SOUTH);
                nayButton = new JButton("Nay");
                add(nayButton, SOUTH);
                addActionListeners();
        }

        public void actionPerformed(ActionEvent event) {
                if (event.getSource() == yayButton) {
                        println("Yay");
                } else if (event.getSource() == nayButton) {
                        println("Nay");
                }
        }
}
```

JButton Example

```
public class Interactors extends ConsoleProgram {
        private JButton yayButton;
        private JButton nayButton;
        public void init() {
                yayButton = new JButton("Yay");
                add(yayButton, SOUTH);
                nayButton = new JButton("Nay");
                add(nayButton, SOUTH);
                addActionListeners();
        }

        public void actionPerformed(ActionEvent event) {
                if (event.getSource() == yayButton) {
                        println("Yay");
                } else if (event.getSource() == nayButton) {
                        println("Nay");
                }
        }
}
```

JButton Example #2

```
public class Interactors extends ConsoleProgram {
	~~private JButton yayButton;~~
	~~private JButton nayButton;~~
	public void init() {
		JButton yayButton = new JButton("Yay");
		add(yayButton, SOUTH);
		JButton nayButton = new JButton("Nay");
		add(nayButton, SOUTH);
		addActionListeners();
	}

	public void actionPerformed(ActionEvent event) {
		if (event.getActionCommand().equals("Yay")) {
			println("Yay");
		} else if (event.getActionCommand().equals("Nay")) {
			println("Nay");
		}
	}
}
```

JButton Example #2

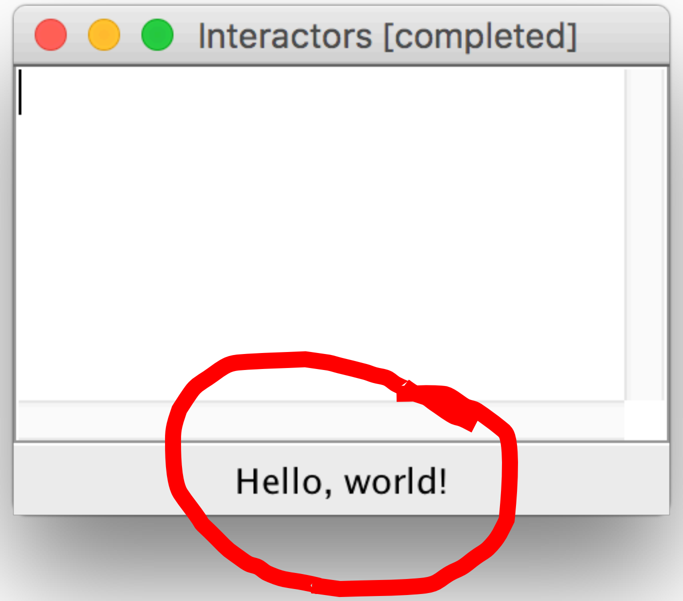
```
public class Interactors extends ConsoleProgram {
        ~~private JButton yayButton;~~
        ~~private JButton nayButton;~~
        public void init() {
                JButton yayButton = new JButton("Yay");
                add(yayButton, SOUTH);
                JButton nayButton = new JButton("Nay");
                add(nayButton, SOUTH);
                addActionListeners();
        }

        public void actionPerformed(ActionEvent event) {
                println(event.getActionCommand());
        }
}
```

Plan for today

- Recap: Inheritance
- Extending GCanvas
- *Example*: Aquarium
- Interactors
  - JButton
  - JLabel
  - JTextField
- Example: TipCalculator

JLabel

```
JLabel label = new JLabel("Hello, world!");
add(label, SOUTH);
```

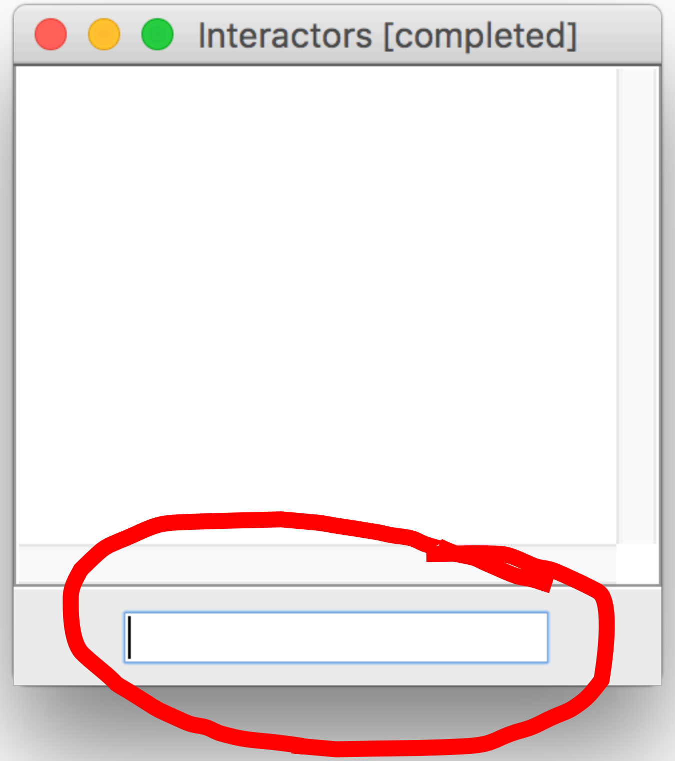


Plan for today

- Recap: Inheritance
- Extending GCanvas
- *Example*: Aquarium
- Interactors
  - JButton
  - JLabel
  - JTextField
- Example: TipCalculator

JTextField

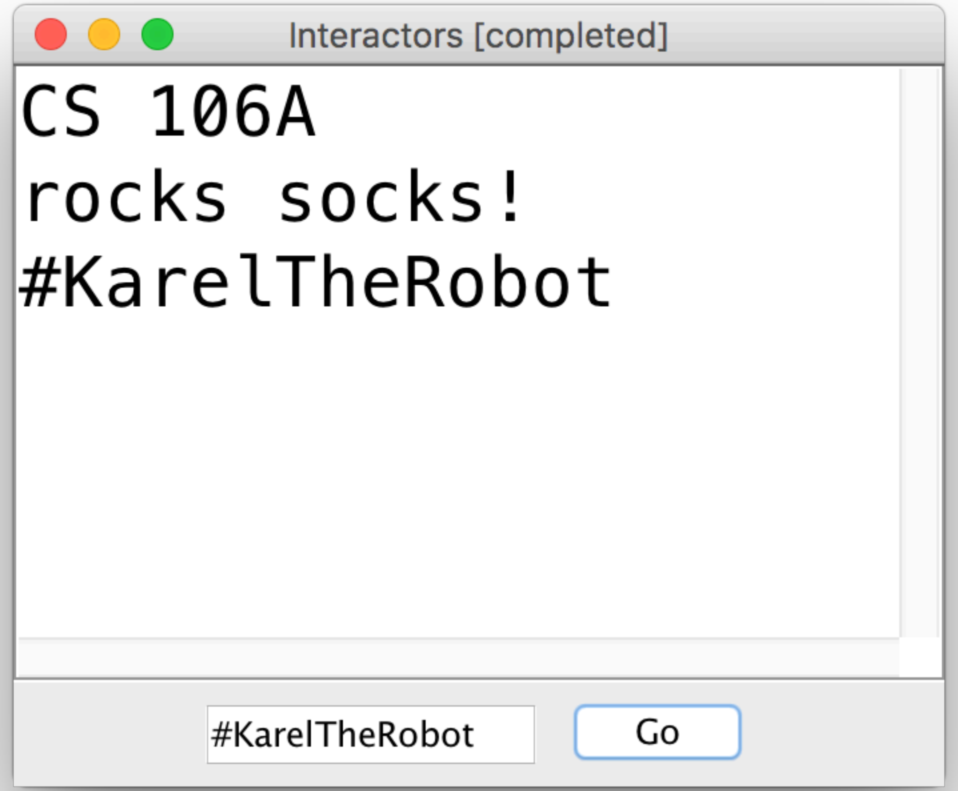
```
JTextField field = new JTextField(10);
add(field, SOUTH);
```



JTextField

```
JTextField field = new JTextField(10);
add(field, SOUTH);

// Set the text in the text field
field.setText("Hello!");

// Get the text currently in the text field
String text = field.getText();
```

JTextField Example



```
CS 106A
rocks socks!
#KarelTheRobot
```

JTextField Example

```
public class Interactors extends ConsoleProgram {
      private JTextField textField;
      public void init() {
            textField = new JTextField(10);
            add(textField, SOUTH);
            JButton goButton = new JButton("Go");
            add(goButton, SOUTH);
            addActionListeners();
      }

      public void actionPerformed(ActionEvent event) {
            println(textField.getText());
      }
}
```

Detecting ENTER Pressed

Detecting the ENTER key pressed in a JTextField requires extra work.

```
JTextField field = new JTextField(10);
// Tells Java to listen for ENTER on the text field
field.addActionListener(this);
// Sets the action command (like JButtons) to “Go”
field.setActionCommand("Go");
add(field, SOUTH);
```

Detecting ENTER Pressed

Detecting the ENTER key pressed in a JTextField requires extra work.

```
JTextField field = new JTextField(10);
field.addActionListener(this);
field.setActionCommand("Go");
add(field, SOUTH);


...
public void actionPerformed(ActionEvent event) {
     if (event.getActionCommand().equals(“Go”)) {
          ...
     }
}
```

getActionCommand

Oftentimes, a text field has a "corresponding" button that takes action with the entered text. If we set the text field's action command to be the *same* as its corresponding button, we can check for both a click and ENTER at once!

getActionCommand

```
public void init() {
      JButton button = new JButton("Go");
      add(button, SOUTH);
      JTextField field = new JTextField(10);
      field.addActionListener(this);
      field.setActionCommand("Go");
      add(field, SOUTH);
      addActionListeners();
}

public void actionPerformed(ActionEvent event) {
      if (event.getActionCommand().equals(“Go”)) {
            ...
      }
}
```

getActionCommand

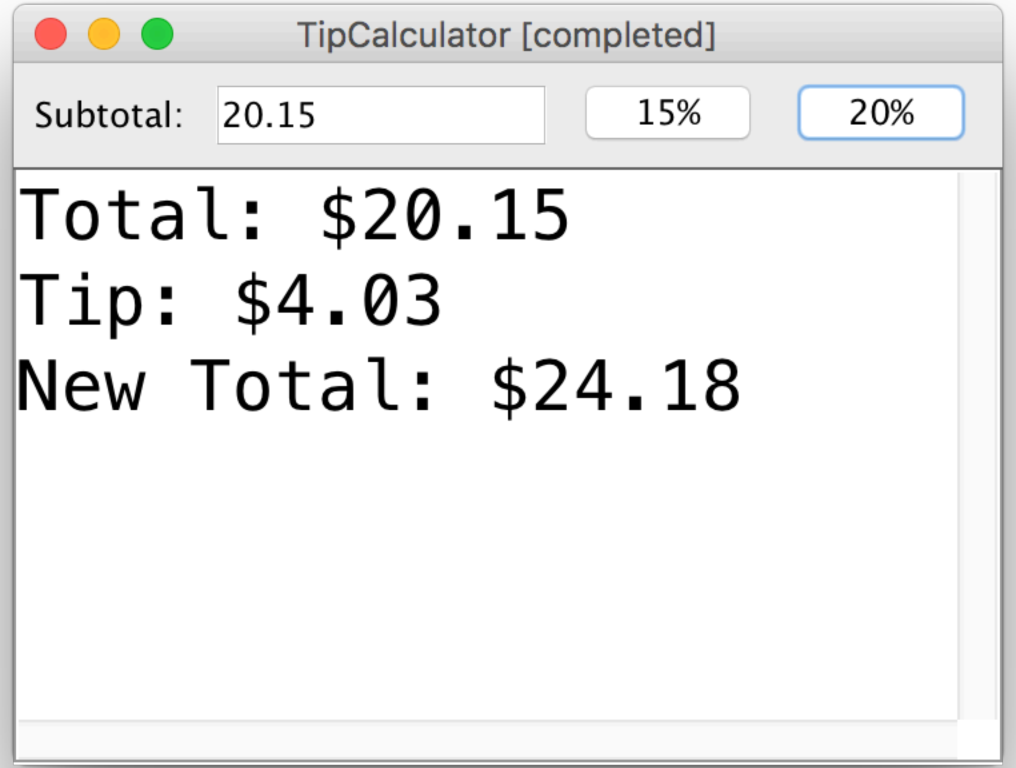
```
public void init() {
        JButton button = new JButton("Go");
        add(button, SOUTH);
        JTextField field = new JTextField(10);
        field.addActionListener(this);
        field.setActionCommand("Go");
        add(field, SOUTH);
        addActionListeners();
}

public void actionPerformed(ActionEvent event) {
        if (event.getActionCommand().equals(“Go”)) {
                ...
        }
}
```

Plan for today

- Recap: Inheritance
- Extending GCanvas
- *Example*: Aquarium
- Interactors
  - JButton
  - JLabel
  - JTextField
- Example: TipCalculator

Practice: TipCalculator

Let's write a program called **TipCalculator** that uses interactors to calculate the tip for a bill.

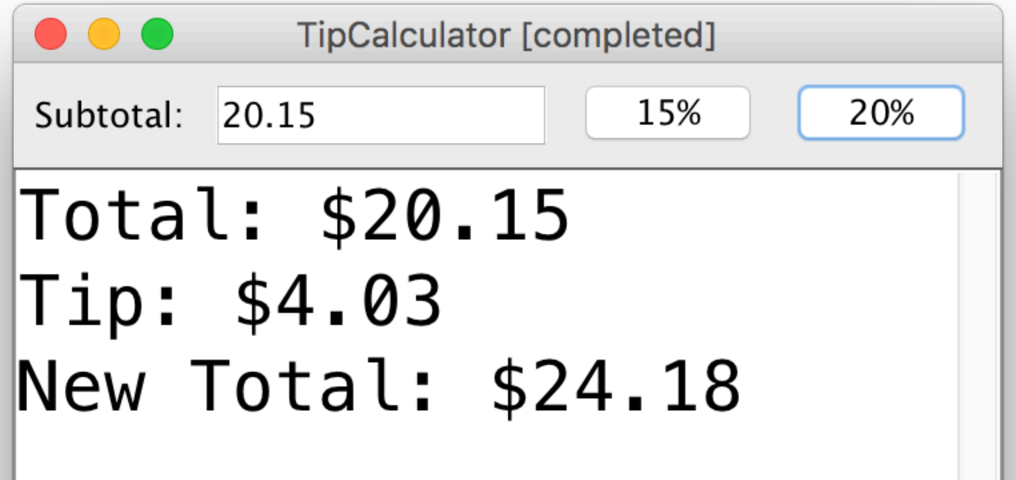


```
Total: $20.15
Tip: $4.03
New Total: $24.18
```

Practice: TipCalculator

Let's write a program called **TipCalculator** that uses interactors to calculate the tip for a bill.

- The program should calculate the appropriate tip depending on the button the user clicks on
- The console should clear when a new tip is calculated (hint: use **`clearConsole()`**).
- Convert a string into a double using **`Double.parseDouble(str);`**



```
Total: $20.15
Tip: $4.03
New Total: $24.18
```

Recap

- Recap: Inheritance
- Extending GCanvas
- *Example*: Aquarium
- Interactors: JButton, JLabel, JTextField
- Example: TipCalculator

Next time: NameSurfer overview