# CS 106A, Lecture 24
# GCanvas and BiasBars

suggested reading:

*Java Ch. 10.5-10.6*

# Plan for today

- Review: Interactors and GCanvas
- Practice: Aquarium
- BiasBars

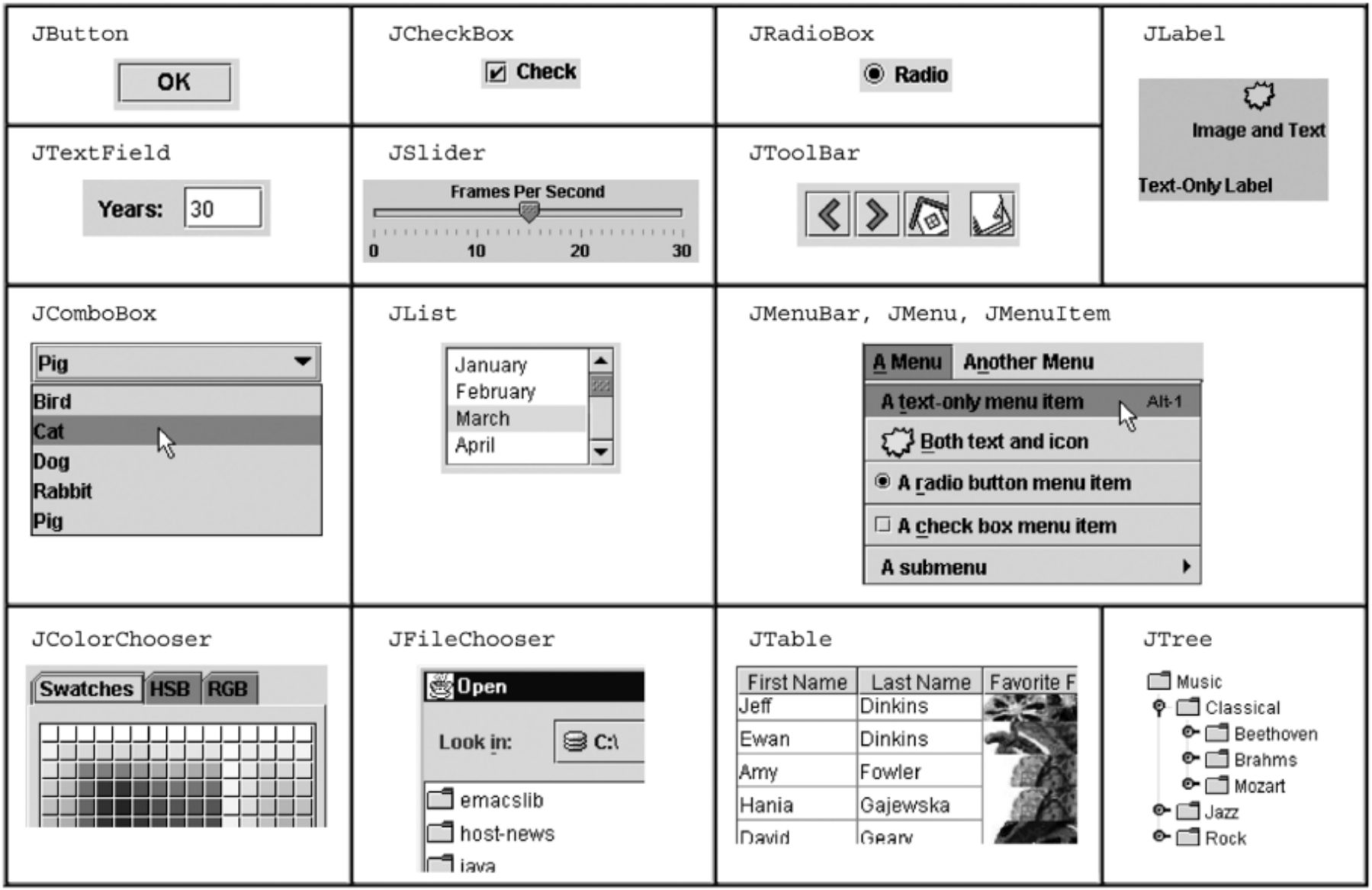
# Learning Goals

- Feel comfortable writing graphical and/or animated programs with multiple classes

# Plan for today

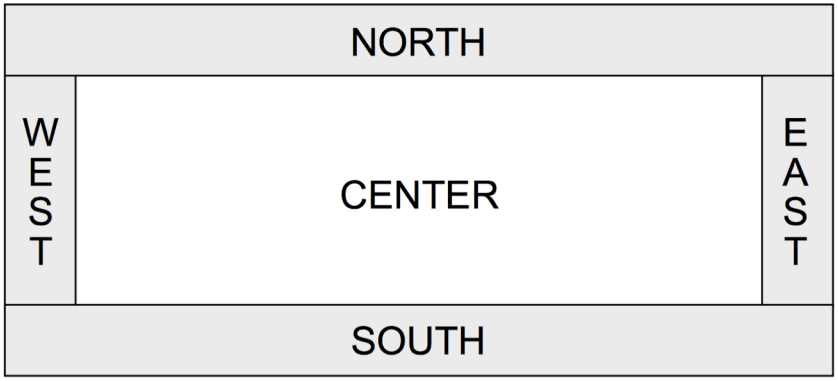
- Review: Interactors and GCanvas
- Practice: Aquarium
- BiasBars

# Interactors



# Window Regions

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

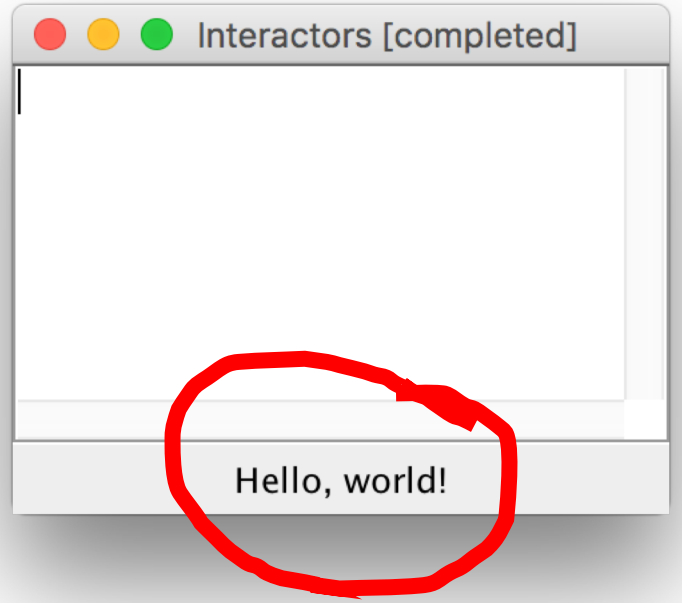



- The **CENTER** region is typically where the action happens.
  - **ConsoleProgram** adds a console there
  - **GraphicsProgram** puts a **GCanvas** there
- Other regions are visible only if you add an interactor to them using add(***component***, ***REGION***);
- Interactors are automatically centered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

# JButton

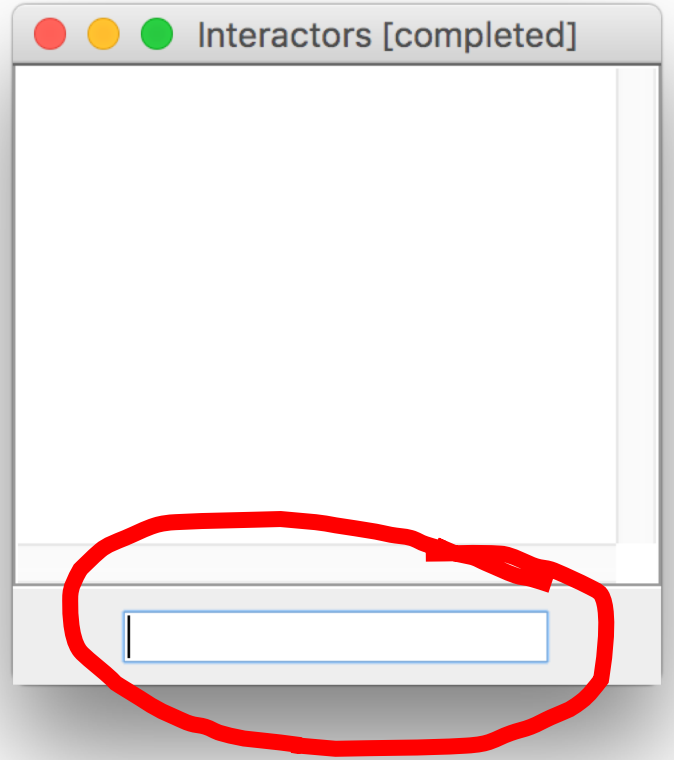
# JLabel

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



# JTextField

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



# Responding To User Inputs

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.
3. Call **`addActionListener(this)`** and optionally **`setActionCommand("some command")`** on JTextFields that should respond when ENTER is pressed.

# 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<br>*(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
        ...
    }
}
```

# Extending GCanvas

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

            // Can't do this anymore, because we are
            // not using GraphicsProgram's canvas
            // GObject obj = getElementAt(…);
            // Do stuff with obj
      }
}
```

# 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(…);
            // Do stuff with obj
      }
}
```

# 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
            canvas.doStuffWithObj(…);
            // Best: let canvas handle graphics!
      }
}
```

# 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
	}
}
```

# Plan for today

- Review: Interactors and GCanvas
- Practice: Aquarium
- BiasBars

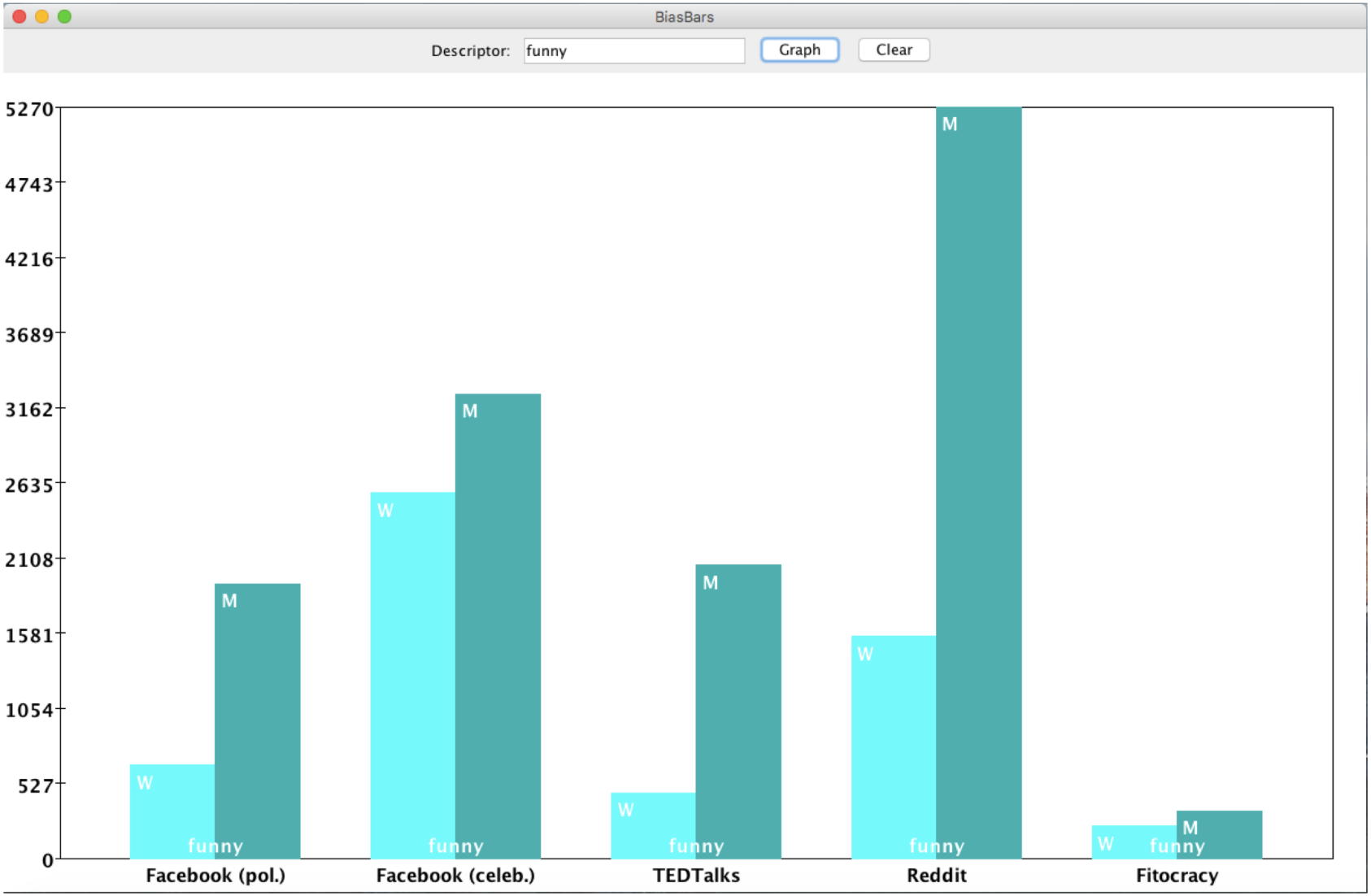
# Practice: Aquarium

- 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

- Review: Interactors and GCanvas
- Practice: Aquarium
- BiasBars

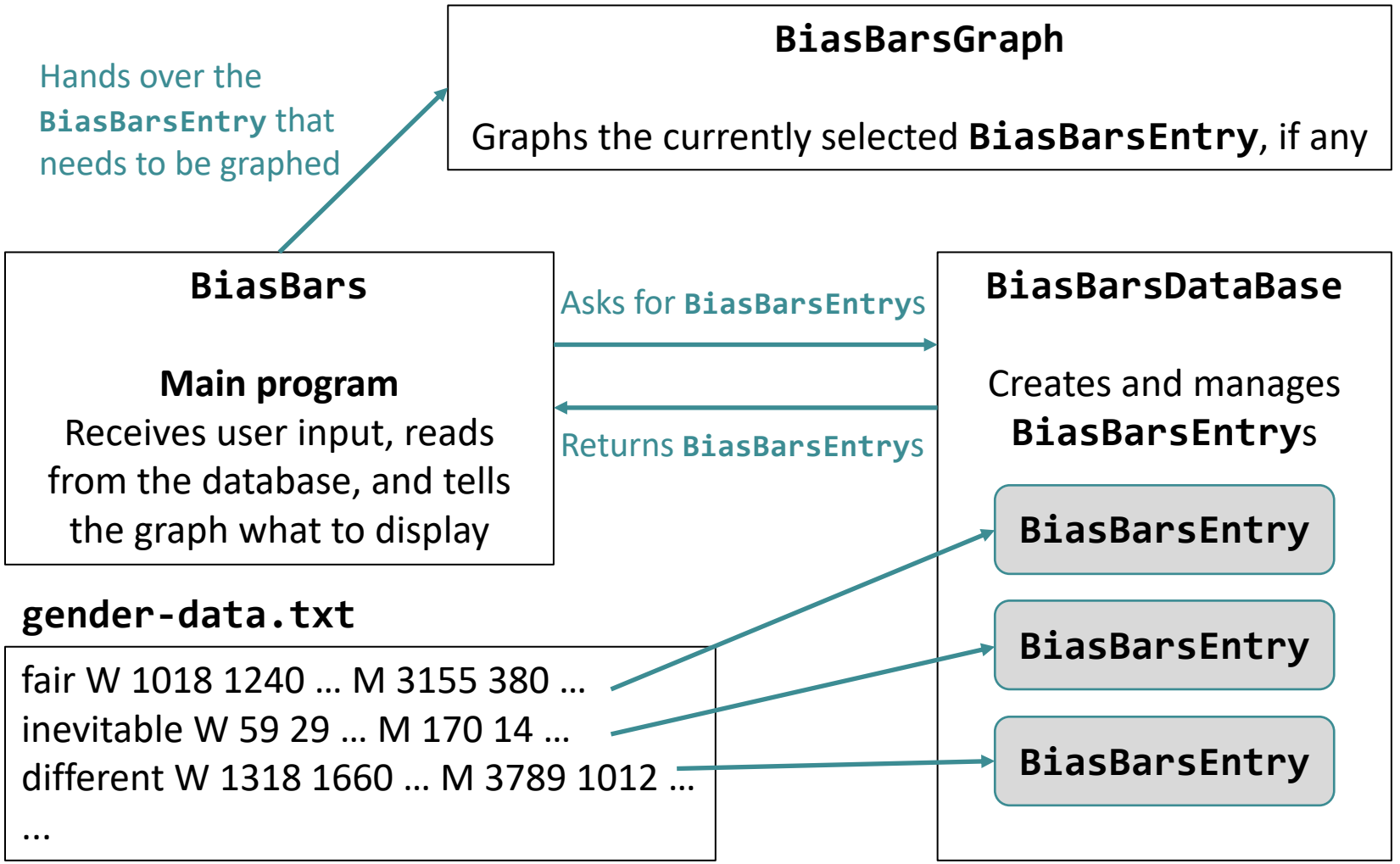
# BiasBars


BiasBars
Descriptor:
funny
Graph
Clear
5270
4743
4216
3689
3162
2635
2108
1581
1054
527
0
W
M
funny
Facebook (pol.)
Facebook (celeb.)
TEDTalks
Reddit
Fitocracy

# BiasBars Structure

- **BiasBars.java** – handles the interactors and overall program
- **BiasBarsEntry** – handles information about a single descriptor and its frequencies for each gender
- **BiasBarsDatabase** – keeps track of all entries and looks up info by descriptor
- **BiasBarsGraph** – a GCanvas subclass that displays bar graphs for descriptors specified by the user

# BiasBars Structure



# BiasBars Structure

- **BiasBars.java** – handles the interactors and overall program
- **BiasBarsEntry** – handles information about a single descriptor and its frequencies for each gender
- **BiasBarsDatabase** – keeps track of all entries and looks up info by descriptor
- **BiasBarsGraph** – a GCanvas subclass that displays bar graphs for descriptors specified by the user

# BiasBars Structure

- **BiasBars.java** – handles the interactors and overall program
- **BiasBarsEntry** – handles information about a single descriptor and its frequencies for each gender
- **BiasBarsDatabase** – keeps track of all entries and looks up info by descriptor
- **BiasBarsGraph** – a GCanvas subclass that displays bar graphs for descriptors specified by the user

# BiasBarsEntry

- Responsible for storing the data about **one name/line** in the text file -> name and ranks. (Hint: use a Scanner!)

```
nice W 2031 10179 1077 3338 2311 M 4606 2926 3274 9603 2682
```

- What instance variables does a BiasBarsEntry need?
- Implement the following methods:
  - `public BiasBarsEntry(String dataLine)`
  - `public String getDescriptor()`
  - `public ArrayList<Integer> getFrequencies(char gender)`
  - `public int getMaxFrequency()`
  - `public String toString()`

# BiasBars Structure

- **BiasBars.java** – handles the interactors and overall program
- **BiasBarsEntry** – handles information about a single descriptor and its frequencies for each gender
- **BiasBarsDatabase** – keeps track of all entries and looks up info by descriptor
- **BiasBarsGraph** – a GCanvas subclass that displays bar graphs for descriptors specified by the user

# BiasBarsDatabase

- Responsible for reading in the text file and creating/storing BiasBarsEntry objects.
- Needs to be able to find entries **given their descriptor** (case insensitive!). What data structure might be useful here?

# BiasBarsDatabase

```
import java.io.*;
import java.util.*;
public class BiasBarsDatabase implements BiasBarsConstants {

  public BiasBarsDatabase(String filename) {
    // TODO: fill this in
  }

  public BiasBarsEntry findEntry(String descriptor) {
    // TODO: implement this method
    return null;
  }
}
```

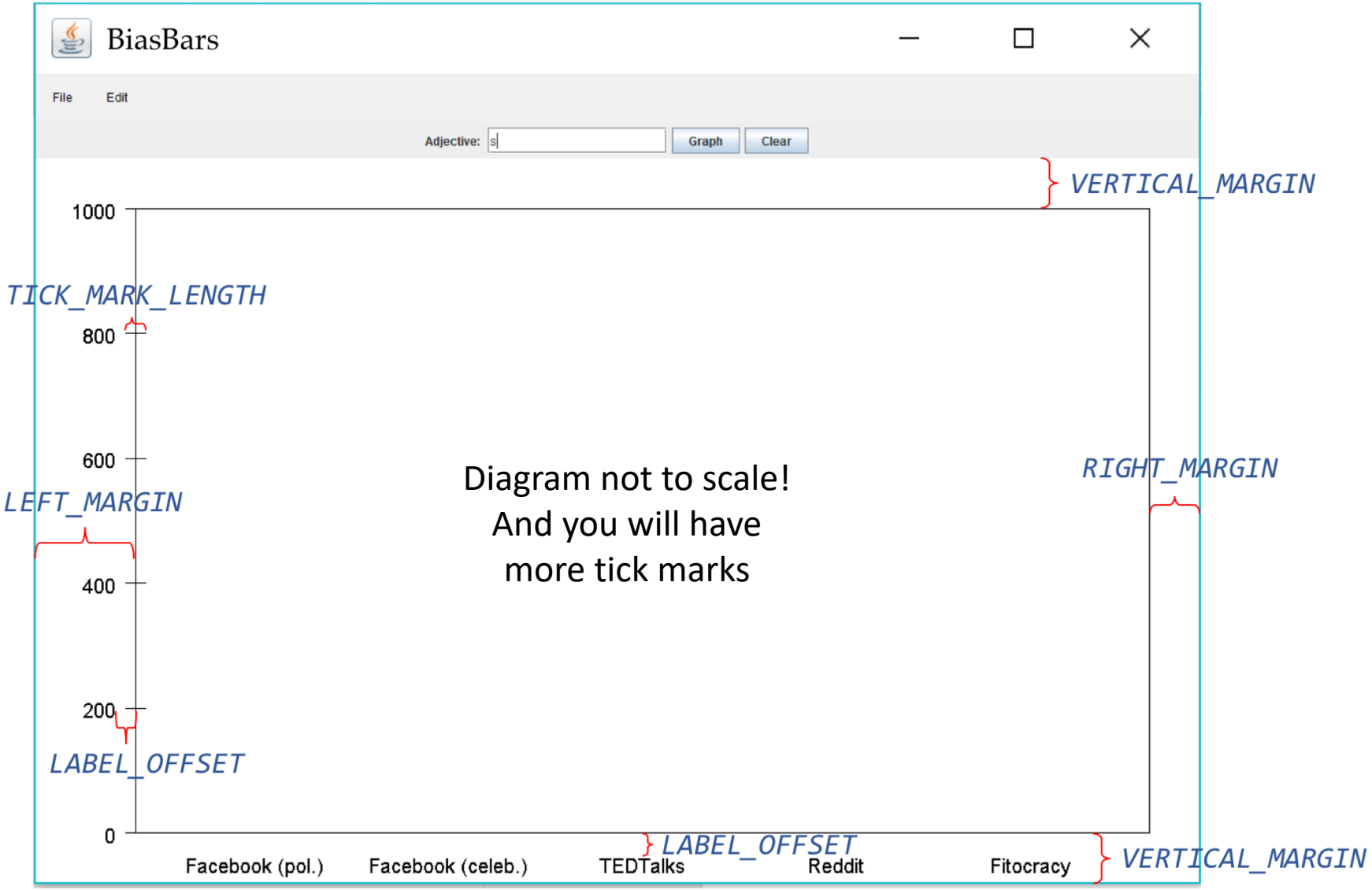
# BiasBars Structure

- **BiasBars.java** – handles the interactors and overall program
- **BiasBarsEntry** – handles information about a single descriptor and its frequencies for each gender
- **BiasBarsDatabase** – keeps track of all entries and looks up info by descriptor
- **BiasBarsGraph** – a GCanvas subclass that displays bar graphs for descriptors specified by the user

# BiasBarsGraph

- A subclass of **GCanvas** that handles all the graph drawing (similar to **FishTank.java** in our **Aquarium** program)
- Two bars for each comment source: one for women (left) and one for men (right)
- Setting up axes requires lots of math, but don't stress about details until basic functionality is done
- *Tip*: use the output comparison tool for the finishing touches!

# BiasBarsGraph



# BiasBarsGraph: Resizing

```
public void update() {
    // TODO: implement this method
}


/* Implementation of the ComponentListener interface for updating when the window is resized */
public void componentHidden(ComponentEvent e) { }
public void componentMoved(ComponentEvent e) { }
public void componentResized(ComponentEvent e) { update(); }
public void componentShown(ComponentEvent e) { }
```

# BiasBarsGraph: Resizing

- Every time the window resizes, update() is called.
- Therefore, update() *must* clear and redraw the whole graph.
- This means the graph must keep track of the entry currently graphed so it can redraw the bars whenever it needs to.
- Other required methods:
  - `clear()`
  - `addEntry(BiasBarsEntry entry)`
- These methods do NOT actually alter the graphics. You must call `update()` to do that, since `update()` must do all the drawing.

# Recap

- Review: Interactors and GCanvas
- Practice: Aquarium
- BiasBars

**Next time:** Life After CS106A, Part 1