CS 106A, Lecture 24
Interactors and NameSurfer

suggested reading:
Java Ch. 10.5-10.6
Plan for today

• Recap: Extending GCanvas
• Interactors
  – JButton
  – JLabel
  – JTextField
• Example: TipCalculator
• NameSurfer
Plan for today

• Recap: Extending GCanvas

• Interactors
  – JButton
  – JLabel
  – JTextField

• Example: TipCalculator

• NameSurfer
public class Graphics extends Program {
    public void init() {
        // We can make our own GCanvas!
        MyCanvas canvas = new MyCanvas();
        add(canvas);
    }

    public void run() {
        // Operate on this canvas
        GObject obj = canvas.getElementAt(…);
    }
}
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);
    }
}
public class Graphics extends Program {

    public void init() {
        // We can make our own GCanvas!
        MyCanvas 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!

```java
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
    }
}
```
Example: Aquarium

- We used classes to make a graphical program called Aquarium that simulates fish swimming around.
public class Aquarium extends Program {
    private static final int NUM_FISH = 5;
    private FishTank tank;

    public void init() {
        tank = new FishTank();
        add(tank);
    }

    ...
}
public void run() {
    tank.initialize();
    for (int i = 0; i < NUM_FISH; i++) {
        tank.addFish();
    }
    while (true) {
        tank.moveFish();
        pause(30);
    }
}
public class FishTank extends GCanvas {
    private ArrayList<Fish> fish;

    public FishTank() {
        fish = new ArrayList<>();
    }

    public void initialize() {
        GImage background = new GImage("res/bkrnd.jpg");
        background.setSize(getWidth(), getHeight());
        add(background);
    }

    ...
}
... public void addFish() {
    Fish newFish = new Fish(getWidth(), getHeight());
    fish.add(newFish);
    add(newFish.getImage());
}

public void moveFish() {
    for (Fish currentFish : fish) {
        currentFish.swim(getWidth(), getHeight());
    }
}
Plan for today

• Recap: Extending GCanvas

• Interactors
  – JButton
  – JLabel
  – JTextField

• Example: TipCalculator

• NameSurfer
Interactors

<table>
<thead>
<tr>
<th>Interactuator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JButton</td>
<td>OK button</td>
</tr>
<tr>
<td>JCheckBox</td>
<td>Check box</td>
</tr>
<tr>
<td>JRadioBox</td>
<td>Radio button</td>
</tr>
<tr>
<td>JLabel</td>
<td>Image and Text, Text-Only Label</td>
</tr>
<tr>
<td>JTextField</td>
<td>Years: 30</td>
</tr>
<tr>
<td>JSlider</td>
<td>Frames Per Second</td>
</tr>
<tr>
<td>JToolBar</td>
<td>Toolbar</td>
</tr>
<tr>
<td>JComboBox</td>
<td>Pig, Bird, Cat, Dog, Rabbit, Pig</td>
</tr>
<tr>
<td>JList</td>
<td>January, February, March, April</td>
</tr>
<tr>
<td>JMenuBar, JMenu, JMenuItem</td>
<td>Menu bar, menu, menu item</td>
</tr>
<tr>
<td>JColorChooser</td>
<td>Swatches, HSB, RGB</td>
</tr>
<tr>
<td>JFileChooser</td>
<td>Open dialog for selecting files or folders</td>
</tr>
<tr>
<td>JTable</td>
<td>Table with columns: First Name, Last Name, Favorite F</td>
</tr>
<tr>
<td>JTree</td>
<td>Categories: Music, Classical, Beethoven, Brahms, Mozart, Jazz, Rock</td>
</tr>
</tbody>
</table>
Interactors

JComponent

JButton

JLabel

JTextField
Plan for today

• Recap: Extending GCanvas
• Interactors
  – JButton
  – JLabel
  – JTextField
• Example: TipCalculator
• NameSurfer
JButton
```java
import java.awt.event.*;
import javax.swing.*;

JButton button = new JButton("Press me");
add(button, SOUTH);
```
• 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.
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) {
        ... // ?
    }
}
The `ActionEvent` parameter contains useful event information.

- Use `getSource` or `getActionCommand` to figure out what button or component was interacted with.

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

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");
        }
    }
}
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) {
        println(event.getActionCommand());
    }
}

Plan for today

• Recap: Extending GCanvas

• Interactors
  – JButton
  – JLabel
  – JTextField

• Example: TipCalculator

• NameSurfer
JLabel label = new JLabel("Hello, world!");
add(label, SOUTH);
Plan for today

• Recap: Extending GCanvas
• Interactors
  – JButton
  – JLabel
  – JTextField
• Example: TipCalculator
• NameSurfer
JTextField field = new JTextField(10);
add(field, SOUTH);
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
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 the ENTER key pressed in a JTextField requires extra work.

```java
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.setActionActionCommand("Go");

add(field, SOUTH);
```
Detecting the ENTER key pressed in a JTextField requires extra work.

```java
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!
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")) {
        ...
    }
}
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: Extending GCanvas
• Interactors
  – JButton
  – JLabel
  – JTextField
• Example: TipCalculator
• NameSurfer
Let’s write a program called **TipCalculator** that uses interactors to calculate the tip for a bill.
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);`
Plan for today

• Recap: Extending GCanvas
• Interactors
  – JButton
  – JLabel
  – JTextField
• Example: TipCalculator
• NameSurfer
NameSurfer
NameSurfer Structure

- **NameSurfer.java** – handles the interactors and overall program
- **NameSurferEntry** – handles information about a single name and its ranks
- **NameSurferDatabase** – handles information about all names and their ranks, and looks up info by name
- **NameSurferGraph** – a GCanvas subclass that displays the name plots
NameSurfer Structure

NameSurferDatabase

Loads and manages NameSurferEntries

NameSurfer

Main program. Receives user input, reads from the database, and tells the graph what to display.

NameSurferGraph

Graphs NameSurferEntries and keeps track of which entries need to be displayed.

NameData.txt

... Sam 58 69 99 131 168 236 278 380 467 408 466
Samara 0 0 0 0 0 0 0 0 0 0 0 0 0
Samir 0 0 0 0 0 0 0 0 0 0 0 0 0
Sammie 537 545 351 325 333 396 565 772 930 0 0
Sammy 0 887 544 299 202 262 321 395 575 639 755
Samson 0 0 0 0 0 0 0 0 0 0 0 0 0
Samuel 31 41 46 60 61 71 83 61 52 35 28
Sandi 0 0 0 0 704 864 621 695 0 0 0
Sandra 0 942 606 50 6 12 11 39 94 168 257
...

Asks for NameSurferEntries

Returns NameSurferEntries

Hands over NameSurferEntries that need to be displayed.
NameSurfer Structure

- **NameSurfer.java** – handles the interactors and overall program
- **NameSurferEntry** – handles information about a single name and its ranks
- **NameSurferDatabase** – handles information about all names and their ranks, and looks up info by name
- **NameSurferGraph** – a GCanvas subclass that displays the name plots
NameSurfer Structure

- **NameSurfer.java** – handles the interactors and overall program

- **NameSurferEntry** – handles information about a single name and its ranks

- **NameSurferDatabase** – handles information about all names and their ranks, and looks up info by name

- **NameSurferGraph** – a GCanvas subclass that displays the name plots
NameSurferEntry

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Rank1</th>
<th>Rank2</th>
<th>Rank3</th>
<th>Rank4</th>
<th>Rank5</th>
<th>Rank6</th>
<th>Rank7</th>
<th>Rank8</th>
<th>Rank9</th>
<th>Rank10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>58</td>
<td>69</td>
<td>99</td>
<td>131</td>
<td>168</td>
<td>236</td>
<td>278</td>
<td>380</td>
<td>467</td>
<td>408</td>
</tr>
</tbody>
</table>

• What instance variables does a NameSurferEntry need?

• Implement the following methods:
  - `public NameSurferEntry(String dataLine)`
  - `public String getName()`
  - `public int getRank(int decadesSince1900)`
  - `public String toString()`
NameSurfer Structure

- **NameSurfer.java** – handles the interactors and overall program
- **NameSurferEntry** – handles information about a single name and its ranks
- **NameSurferDatabase** – handles information about all names and their ranks, and looks up info by name
- **NameSurferGraph** – a GCanvas subclass that displays the name plots
NameSurferDatabase

• Responsible for reading in the text file and creating/storing NameSurferEntry objects.

• Needs to be able to find entries **given their name** (case insensitive!). What data structure might be useful here?
// TODO: comment this file
import java.io.*;
import java.util.*;
public class NameSurferDatabase implements NameSurferConstants {

    // TODO: comment this constructor
    public NameSurferDatabase(String filename) {
        // TODO: fill this in
    }

    // TODO: comment this method
    public NameSurferEntry findEntry(String name) {
        // TODO: implement this method
        return null; // remove this line
    }
}
NameSurfer Structure

- **NameSurfer.java** – handles the interactors and overall program
- **NameSurferEntry** – handles information about a single name and its ranks
- **NameSurferDatabase** – handles information about all names and their ranks, and looks up info by name
- **NameSurferGraph** – a GCanvas subclass that displays the name plots
NameSurferGraph

- A subclass of `GCanvas` that handles all the graph drawing (similar to `FishTank.java` in our `Aquarium` program)
- Different, cycling colors for each plot line
- Ranks range from 1 to 1000, with rank 0 specially marked at the bottom
- *Tip*: use the output comparison tool!
NameSurferGraph

Rank 1

Rank 1000 AND 0
// TODO: comment this method
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) {}
NameSurferGraph: Resizing

• Every time the window resizes, update() is called.
• Therefore, update() must clear and redraw the whole graph.
• This means the graph must store the entries being graphed so it can redraw them whenever it needs to. What might be appropriate to help us store this?
• Other required methods:
  – clear()
  – addEntry(NameSurferEntry entry)
• These methods do NOT actually alter the graphics. You must call update() to do that, since update() must do all the drawing.
Make sure to *always* use the provided constants! You may add more, but add them in *other* files, not this provided one.

```java
public interface NameSurferConstants {

    /** The name of the file containing the data */
    public static final String NAMES_DATA_FILE = "res/names-data.txt";

    /** The width of the text field in the NORTH of the window */
    public static final int TEXT_FIELD_WIDTH = 16;

    /** The first decade in the database */
    public static final int START_DECADE = 1900;

    /** The number of decades */
    public static final int NUM_DECASES = 11;

    /** The maximum rank in the database */
    public static final int MAX_RANK = 1000;

    /** The number of pixels to reserve at the top and bottom */
    public static final int GRAPH_MARGIN_SIZE = 20;

    /** The number of pixels between the baseline of the decade labels and the bottom of the window */
    public static final int DECADE_LABEL_MARGIN_SIZE = GRAPH_MARGIN_SIZE / 4;
}
```
Recap

• Recap: Extending GCanvas
• Interactors: JButton, JLabel, JTextField
• Example: TipCalculator
• NameSurfer

Next time: Life after CS 106A, Part 1