CS102

Introduction to
data structures, algorithms,
and object-oriented
programming

April 3rd , 2017

Homework 4

1. In homework 4, you will add a paddle to the bouncing ball scene.
2. In a sequence of steps, you draw the paddle, make it move, and then make sure the ball bounces off the paddle.
3. This is the first time you have used a Listener for a mouse event. The listener I ask you to use is a MouseMotionListener, which requires the implementing class to contain a mouseMoved and a mouseDragged method (but for the assignment, you only need a mouseMoved method). Like Action-Listener, the MouseMotionListener can either be an instance method or an anonymous inner class.

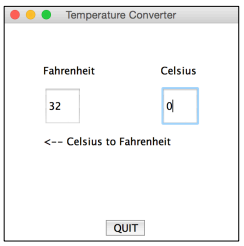
Homework 4

4. When a mouse event occurs, we get information in the event generated. In this case, the data we need is the mouse event getX(), which returns the x coordinate of the mouse on the scene.

5. We need an actionListener on the Timer to generate events that keep the ball moving.

Lab today

For lab 7, I will have you implement a simple TemperatureConverterGUI that converts degrees Fahrenheit to degrees Celsius.



Lab preview

In lab 7, you will use GUI component JTextField to get input from the user. The starter file extends JFrame instead of JPanel. If the class type is-a JFrame, you can get a drawing surface out of the JFrame by using the getContentPane method

Hopefully the instructions will allow you to complete this lab with little trouble.

The input should be a number. However, a try block should be started before every attempt to convert a String into a number (----------Exception)???

Reading JTextField Text

1. When an enter is generated by a JTextField (JTF), an ActionEvent is generated.

2. After source is known, use getText()

```
String name = "";
if (e.getSource().equals(getFahrInput)){
  try {
        name = getFahrInput.getText();
        fTemp = Double.parseDouble(name);
      }catch(NumberFormatException nfe) {
        intro.setText("Please enter #s in boxes.");
        getFahrInput.setText("");
        return;
     }
```

Reading from the CL

Sometimes, you may want to enter the file inputs from the command line (CL):

```
public static void main(String[] args) throws FileNotFoundException {
    String textFile  = args[0];
    Scanner scan = new Scanner(new FileReader(textFile));
    int vertices = Integer.parseInt(scan.nextLine());
    int edges = Integer.parseInt(scan.nextLine());
    String graph[] = new String[edges];
    for (int i = 0; i < edges; i++) {
       graph[i] = scan.next() + ", " + scan.next();
    }
    BFS(graph, source, vertices);
    printPaths();
}
```

7

Working from the CL

Suppose you are working from the terminal and you have a java file that reads from the CL and an input file in the same directory.

To compile the file, type

```
$ javac FileName.java
```

To run the file, type

```
$ java FileName data.txt
```

($ is the command line prompt)

8

Exception Review

Exceptions fall into 2 different hierarchies:

1. **Unchecked Exceptions** (subclasses of RunTimeException): You usually don't catch unchecked exceptions. Instead, you fix your program so it can't produce one of these. Example: ArrayIndexOutOfBoundsException, NullPointerException.

2. **Checked Exceptions** (subclasses of Exception): These are usually errors in the input data or pertaining to the input or output file. The programmer has no control over the input the user gives you for file names. If the user gives you a bad value, it may cause an exception at run time.

9

Exception Review

Checked exceptions are subject to the "Catch or Throw Requirement". This means that code that might throw checked exceptions must be enclosed by either of the following:

* A try statement that catches the exception. The try must provide a handler for the exception, as described below in Catching Exceptions.

* A method that specifies that it throws the exception. The method must provide a throws clause in the header line that lists the exception, as described below in Specifying the Exceptions Thrown by a Method.

10

File I/O with text

1. Use any text editor (DrJava will work too, just don't save file with .java extension).

2. Type in text.

3. Save the file as <name>.txt

11

Reading from text files

Using the Reader class BufferedReader:

1. Import java.io.* for I/O exceptions and readers.

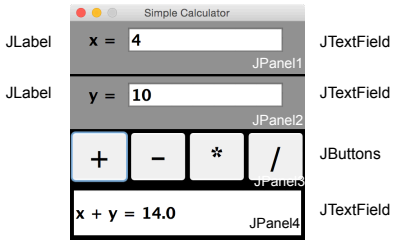
2. Declare a BufferedReader to be either a local or instance variable.

3. Instantiate the BufferedReader inside a while loop to keep trying in case an exception is thrown:

12

More Complex GUI

The GUI we'll look at next has a main JPanel with 4 nested JPanels.



Putting a GUI class together

Writing a GUI can be done in many different ways. I will cover the setup for the simple calculator on the last slide:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/**
 * This class uses nested JPanels to create a simple calculator.
 */
public class SimpleCalculator extends JPanel implements ActionListener {
```

The SimpleCalculator class is-a JPanel and is-of ActionListener type.

After the class signature, declare all the class instance variables (you will often add these as you discover they are needed in the code).

```
    JPanel calcPanel, panelX, panelY, buttonPanel, resultPanel;
    JButton plus, minus, mult, div;
    JLabel xEqual, yEqual;
    JTextField enterX, enterY, answer;

    public static void main(String[] args) {
        SimpleCalculator sgs = new SimpleCalculator();
    } // end main

    public SimpleCalculator() {
        JFrame bigPane = new JFrame("Simple Calculator");
        bigPane.setLayout(null);
        bigPane.setBackground(Color.BLACK);
        bigPane.setLocation(100,50);

        calcPanel = new JPanel(); // panel to hold all others
        calcPanel.setLayout(new GridLayout(4,1,3,3));
        bigPane.setContentPane(calcPanel);
```


Each JPanel is instantiated in the constructor, the layout manager is set up for each one, and other properties are set. Here, panelX and panelY are instantiated.

```
        panelX = new JPanel();
        panelX.setBackground(Color.GRAY);
        panelX.setLayout(new FlowLayout());
        enterX = new JTextField("0", 10);
        Font bigText = new Font("SansSerif",Font.BOLD,20);
        enterX.setFont(bigText);

        panelY = new JPanel();
        panelY.setBackground(Color.GRAY);
        panelY.setLayout(new FlowLayout());
        enterY = new JTextField("0", 10);
        enterY.setFont(bigText);
```


panelX and panelY each contain a JLabel and a JTextField that are instantiated and added to each panel. Then each panel is added to calcPanel.

```
        xEqual = new JLabel("x = ");
        xEqual.setFont(bigText);
        yEqual = new JLabel("y = ");
        yEqual.setFont(bigText);
        panelX.add(xEqual);
        panelX.add(enterX);
        panelY.add(yEqual);
        panelY.add(enterY);

        calcPanel.add(panelX);
        calcPanel.add(panelY);
```


Instantiate each JButton and add "this" as the action listener

```
        Font biggerText = new Font("SansSerif",Font.BOLD,36);
        plus = new JButton("+");
        plus.setFont(biggerText);
        plus.addActionListener(this);
        minus = new JButton("-");
        minus.setFont(biggerText);
        minus.addActionListener(this);
        mult = new JButton("*");
        mult.setFont(biggerText);
        mult.addActionListener(this);
        div = new JButton("/");
        div.setFont(biggerText);
        div.addActionListener(this);));
```

Finish the constructor by adding buttons to buttonPanel (after the layout manager is specified. Add all JPanels to calcPanel and finish setting up JFrame.

```
        buttonPanel = new JPanel();
        buttonPanel.setLayout(new GridLayout(1,1));
        buttonPanel.add(plus);
        buttonPanel.add(minus);
        buttonPanel.add(mult);
        buttonPanel.add(div);

        bigPane.setPreferredSize(new Dimension(300,300));
        calcPanel.add(panelX);
        calcPanel.add(panelY);
        calcPanel.add(buttonPanel);
        calcPanel.add(resultPanel);

        bigPane.pack();
        bigPane.setLocation(100,50);
        bigPane.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        bigPane.setResizable(false);
        bigPane.setVisible(true);
    } // end constructor
```

19

```
public void actionPerformed(ActionEvent evt){
     double x, y;
     String xStr, yStr;
     // first, get the text from the JTextFields
     try {
         xStr = enterX.getText();
         x = Double.parseDouble(xStr);
     }catch(NumberFormatException nfe) {
         answer.setText("Illegal data for x.");
         enterX.requestFocusInWindow();
         return;
     }
     try {
         yStr = enterY.getText();
         y = Double.parseDouble(yStr);
     }catch(NumberFormatException nfe) {
         answer.setText("Illegal data for y.");
         enterY.requestFocusInWindow();
         return;
     }
```

20

```
        String op = evt.getActionCommand();
        if (op.equals("+"))
            answer.setText("x + y = " + (x+y));
        else if (op.equals("-"))
            answer.setText("x - y = " + (x-y));
        else if (op.equals("*"))
            answer.setText("x * y = " + (x*y));
        else if (op.equals("/")) {
            if (y == 0)
                answer.setText("Can't divide by zero.");
            else
                answer.setText("x / y = " + (x/y));
        } // end if
    } // end actionPerformed
} // end class
```

21