CSE1341 - Lab 8 Assignment

Overview

In Lab 7 you added inheritance and File I/O to your BatterUp game. In this lab, you will make additional changes to the game by adding a graphical user interface.

Pre-Lab

No pre-lab this week. Just make sure Lab 7 is complete and working to use as a start for Lab 8.

Lab (100 Points)

Change your Lab 7 solution with the additions and changes described on the following pages. The behavior of your GUI should conform to the sample screen shots provided in these instructions.

Submit the *java* and *class* files via Canvas (as a single zip-file). Include a comment block at the top of each *Java* file that includes your name, student id number, and "Lab 8-Fall 2018".

NOTES:

Each program should include comments that explain what each block of code is doing. Additionally, the programs should compile without errors, and run with the results described in the exercise. The following deductions will be made from each exercise if any of the following is incorrect or missing:

Proper formatting [5 points]
Proper names for classes and variables [5 points]
Comments [5 points per class]
Program doesn't compile [5 points for each minor error up to 5 errors provided that after fixing the errors the program compiles. If the program does not compiler after the 5 errors are fixed, partial credit will be given not to exceed 50 points]
Source code (java file) missing [10 points]
Executable (class file) missing [10 points]
Missing array where an array was required [5 points each]
Missing loop where a loop was required [5 points each]
Missing class from the design provided [10 points each]
Missing method from the design provided [5 points each]
Missing GUI functionality [5 points each]

This Lab is due Saturday December 1 at 6:00am.

Code Changes

You will need to make the following code changes to allow the GUI to function as required

1. Replace the Launcher with the following class:

```
import javax.swing.JFrame;

public class GUILauncher {
    public static void main(String[] args)
    {
        BatterUpGUI theGame = new BatterUpGUI();
        theGame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        theGame.setSize(650,450);
        theGame.setVisible(true);

    }
}
```

Code Changes

You will need to make the following code changes to allow the GUI to function as required

2. Create **BatterUpGUI.java**. In this class, you will create an instance of **BatterUp** and call the *play()* method when the user clicks the Play button. Create the instance of **BatterUp** with the rest of the components you will be adding to the frame.

3. Add **RollResults.java**. We will use this class to in conjunction with the *roll()* method.

```
public class RollResult {
    private int[] vals;
    private String output;

    public RollResult(int[] vals, String output) {
        this.vals = vals;
        this.output = output;
    }

    //add getters and setters
}
```

Code Changes

You will need to make the following code changes to allow the GUI to function as required

4. Update all the roll methods to return **RollResult** instead of an **int[].** Here's one method as an example.

```
public RollResult roll() {
    int[] vals = new int[2];
    Random rand = new Random();
    vals[0] = rand.nextInt(6) + 1;
    vals[1] = rand.nextInt(6) + 1;
    String str = "  Rolled " + vals[0] + " " + vals[1];
    return new RollResult(vals, str);
}
```

5. Make the following changes to the Player class:
   - Add a **private String** data field (attribute) called **output**.
   - In the constructor, add **output = "";**
   - Create 2 methods: *getOutput()* which returns the **output** and *resetOutput()* which sets the **output = "";**
   - In *takeTurn()*, instead using a *System.out.println()* statement, use **String** concatenation to add to **output** (e.g. **output += " Strike out!!\n";**)
   - Similarly for *bat()* (e.g. **output += dice.getOutput() + " BALL!\n";**). *Remember* that *roll()* now returns **RollResult** instead of **int[]**. Update the code accordingly.

Code Changes

You will need to make the following code changes to allow the GUI to function as required

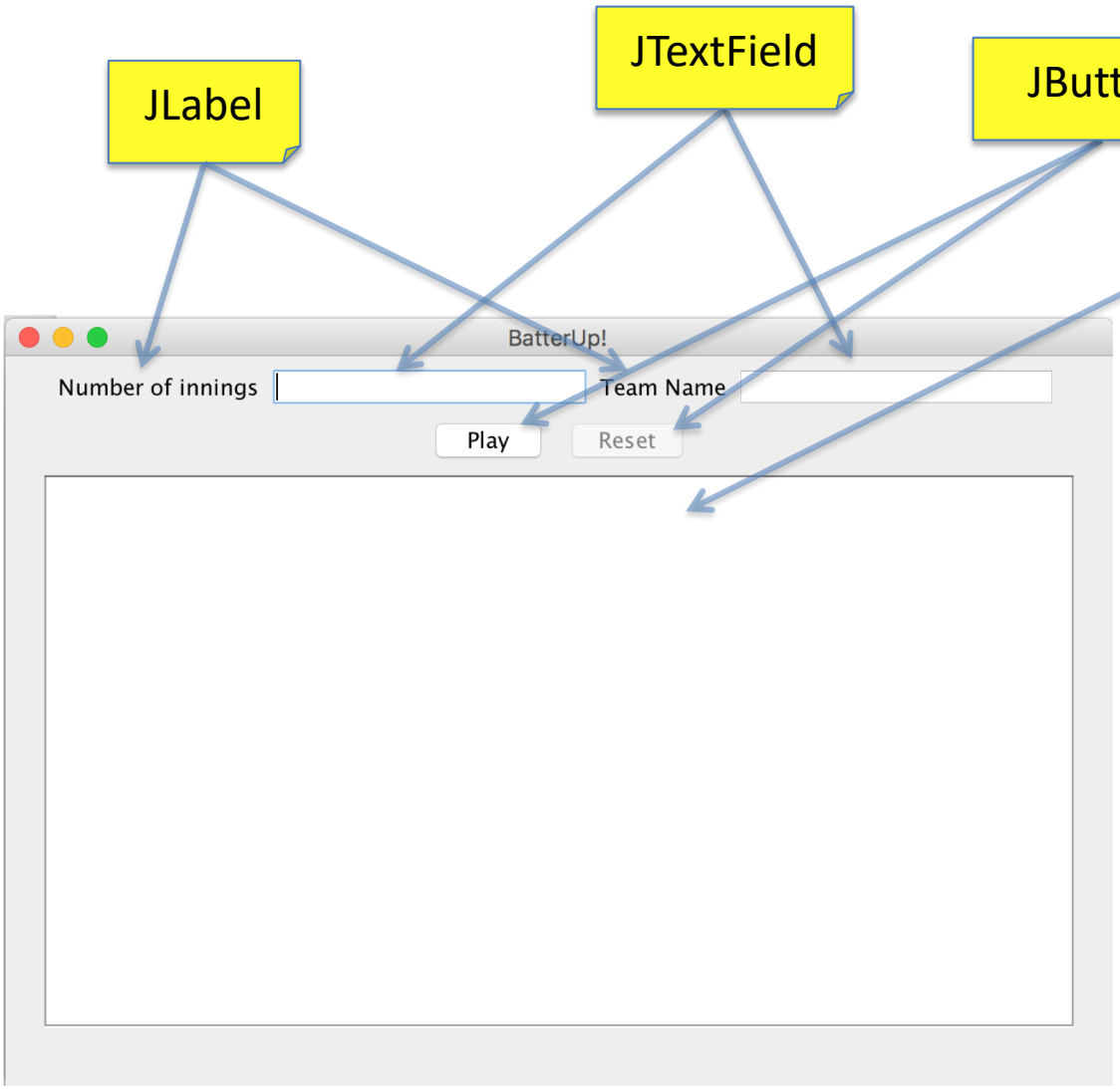
6. Update **BatterUp** as follows:
    - Change the *play()* method to return a **String** and accept an **int** called **numberOfInnings**. You will use this variable in the loop.
    - At the beginning of the method, create a **String** variable called **str = “”;**. You will add to the **String** any output the method produced using *System.out.println()* (e.g. **str += "Inning " + innings + "\n";**).
    - After you call *takeTurn()* add the following line: **str += p.getOutput();**
    - Before you end the **while** loop add these lines of code to reset the output:

```
if(outs >= 3) {
    p.resetOutput();
}
if(p.strikes == 3) {
    p.resetOutput();
}
if(p.balls == 4) {
    p.resetOutput();
}
```

    - After you call *printStats()* set **score = 0** and **return str;**
7. Change *movePlayers()* and *displayField()* to return Strings. At the beginning of the method, create a **String** variable called **str = “”;**. You will add to the String any output the method produced using *System.out.println()*. Return **str** at the end of the method.

GUI

Create a **JFrame** subclass named **BatterUpGUI**. Use a **FlowLayout** with recommended sizing in the instructions so your Frame matches the diagram. The **JComponents** required are depicted in the annotations below:

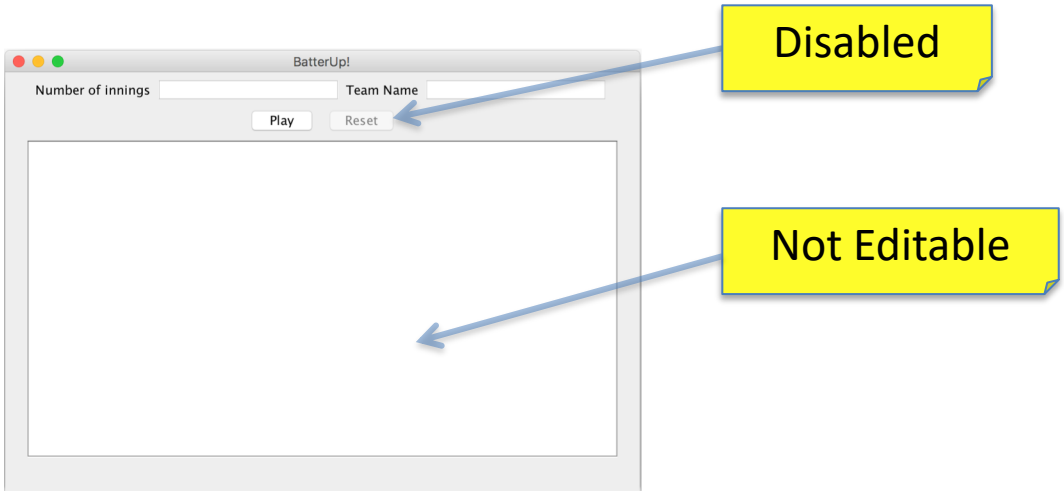


JTextArea Help

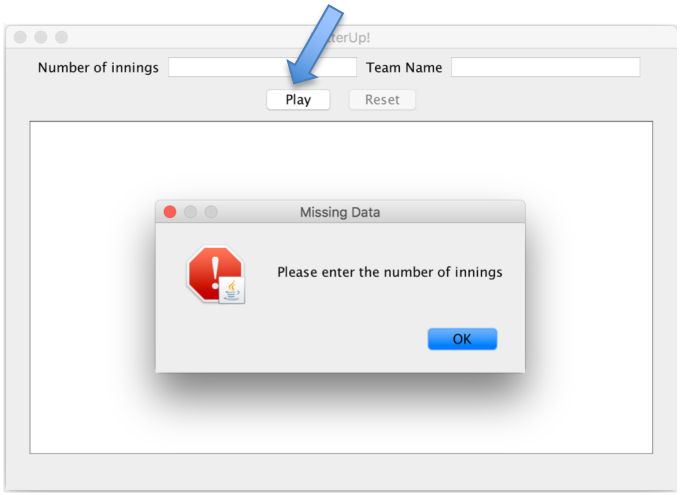
```
//Declare JTextArea and JScrollPane
//above the Constructor
JTextArea output;
JScrollPane scrollPane;

//Create and add widgets in the Constructor
output= new JTextArea(20,50);
scrollPane = new JScrollPane(output);
add(scrollPane);

//Disable editing
output.setEditable(false);

//Elsewhere – as needed
//Erase contents and replace with text
output.setText(”text");
//Append contents with additional text
output.append(“text”);
```

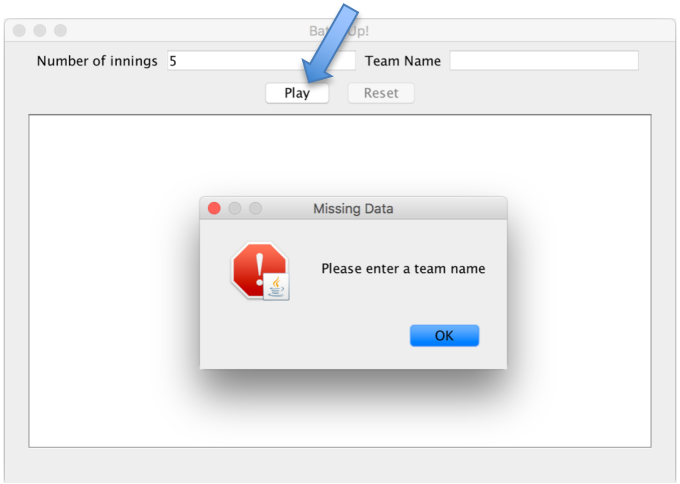
GUI Behavior

On startup, the **Reset** button and the **JTextArea** should be disabled, the other widgets should be enabled.



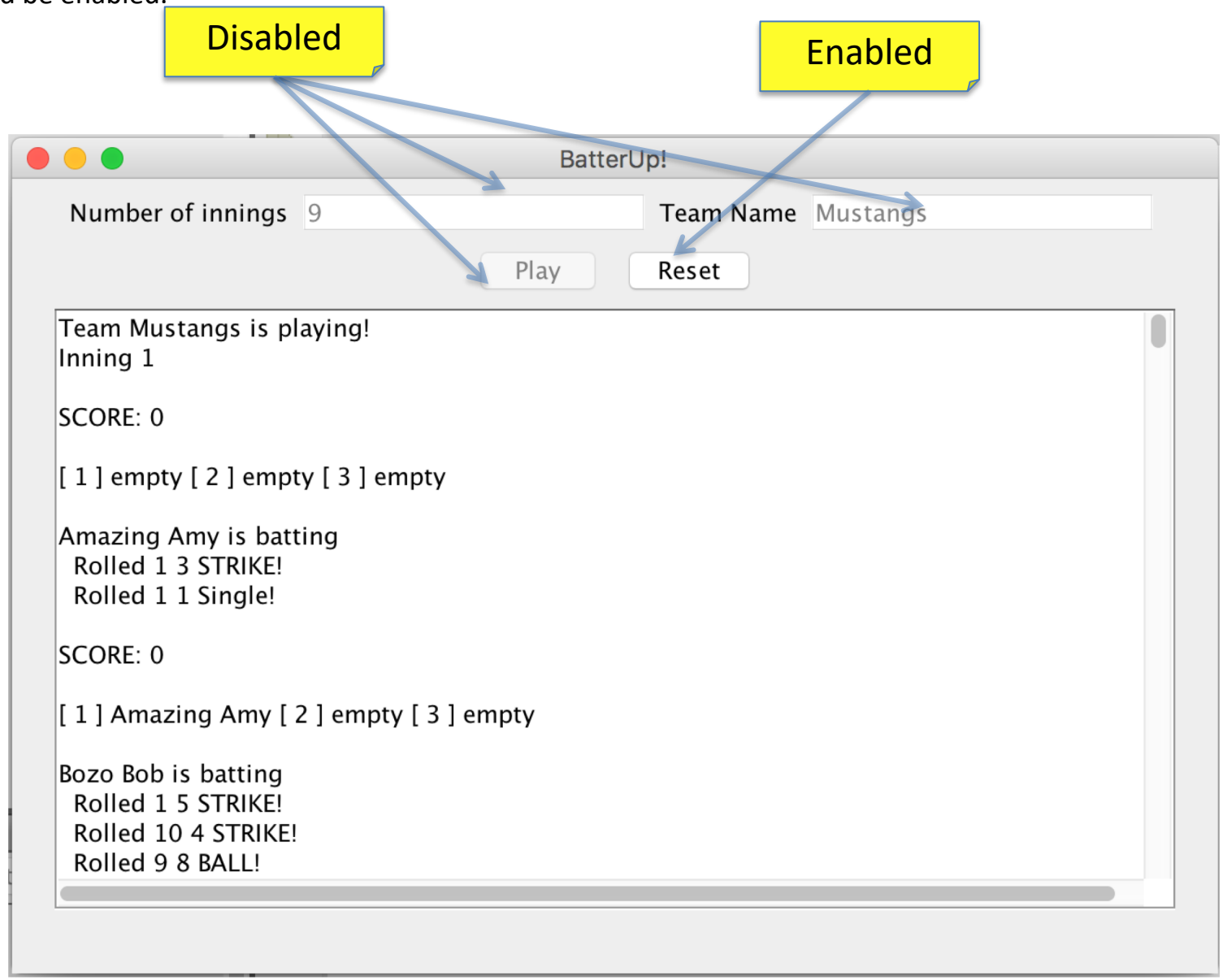
The user should enter a value for **Number of Innings and Team Name**, then click the "**Play**" button. If any of these values is missing, pop up a error message dialog when the **Play** button is cliched, and don't start the game until all fields contain values.





GUI Behavior (continued)

When all values are entered and **Play** is clicked, the text fields and the **Play** button should be disabled, and the **Reset** button should be enabled.



GUI Behavior (continued)

When the player clicks the **Play** button, append the name of the team to the textArea and call the *play()* method passing it the number of innings. Use **Integer.parseInt(innings.getText())** to parse the String number into an integer. The play method will return a String. Append the textArea with the String returned by *play()*.

When the player clicks the **Reset** button, the GUI goes back to its original with the text fields and **Play** button enabled and the **Reset** button disabled.