COMP 322: Fundamentals of Parallel Programming

Lecture 11: GUI Programming

Mack Joyner and Zoran Budimlić
{mjoyner, zoran}@rice.edu

http://comp322.rice.edu

Announcements & Reminders

- Regular office hour schedule can be found at <u>Office Hours</u> link on course web site
- Hw #1 is due Friday, Feb. 4th by 11:59pm
- Quiz #2 is due Sunday, Feb. 6th by 11:59pm
- Midterm exam will be Thursday, Feb. 24th from 7-10pm in Canvas

Login/Logout registered users with Futures

```
var username = …
var password = …
…
var regUser = future(() -> registerNewUser(username, password)); // { username: user, result: "success" or "failure"}
…
var logUser = future(() -> { if (regUser.get().result.equals("success"))
                                        return loginUser(username, password); // {userId: id, result: "success" or "failure"}
                                   return {result: "failure" };
                                  });
…
var loggedIn = future(() -> { if (logUser.get().result.equals("success"))
                                        return isLoggedIn(logUser.get().userId); // {userId: id,  result: "success" or "failure" }
                                  return {result: "failure" };
…
var logOut = future(() -> { if (loggedIn.get().result.equals("success"))
                                     return logoutUser(loggedIn.get().userId)); // { result: "success" or "failure" }
                                return {result: "failure" };
…
```

Login/Logout registered users with DDTs

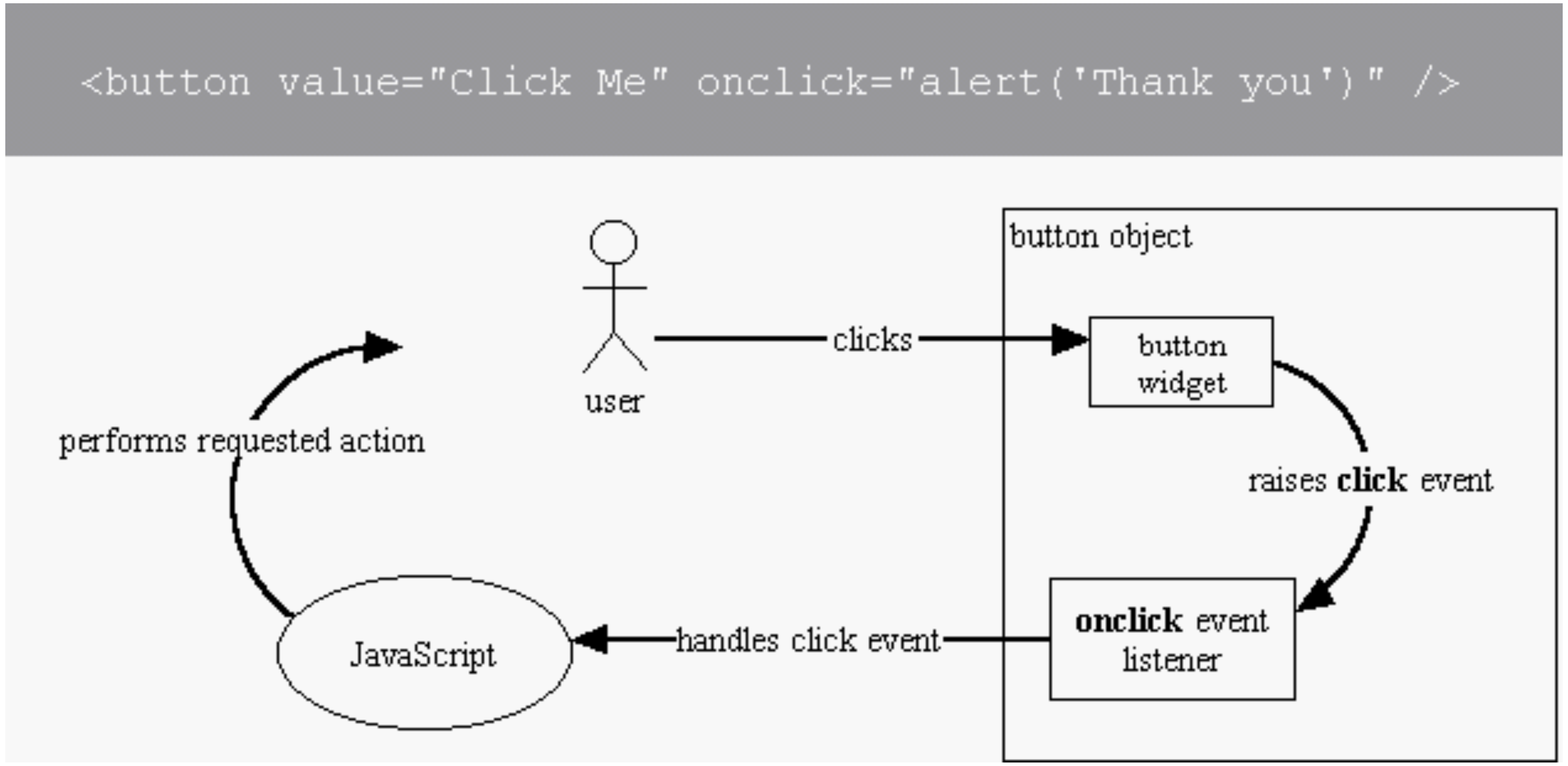
```
var username = …
var password = …
…
var regUser = newDataDrivenFuture();
var logUser = newDataDrivenFuture();
var loggedIn = newDataDrivenFuture();
var logOut = newDataDrivenFuture();
…
async(() -> regUser.put(registerNewUser(username, password))); // { username: user, result: “success” or “failure”}
…
asyncAwait(regUser, () -> { if (regUser.safeGet().result.equals(“success”))
                                    logUser.put(loginUser(username, password)); // {userId: id, result: “success” or “failure”}
                                 else
                                    logUser.put({result: “failure” });      });
                               …
asyncAwait(logUser, () -> { if (logUser.safeGet().result.equals(“success”))
                                    loggedIn.put(isLoggedIn(logUser.safeGet().userId)); // {userId: id,  result: “success” or “failure” }
                               else
                                    loggedIn.put({result: “failure” });    });
…
asyncAwait(loggedIn, () -> { if (loggedIn.safeGet().result.equals(“success”))
                                  logOut.put(logoutUser(loggedIn.safeGet().userId)); // { result: “success” or “failure” }
                                else
                                  logOut.put({result: “failure” });     });
…
```

GUI Programming

- Events are often triggered by a user within a GUI framework
- Events include:
  - Mouse events (clicks, mouse over)
  - Timeouts, Intervals
  - Keyboard events (key press down/up)

See: https://en.wikipedia.org/wiki/Event-driven_programming

GUI Event Handling



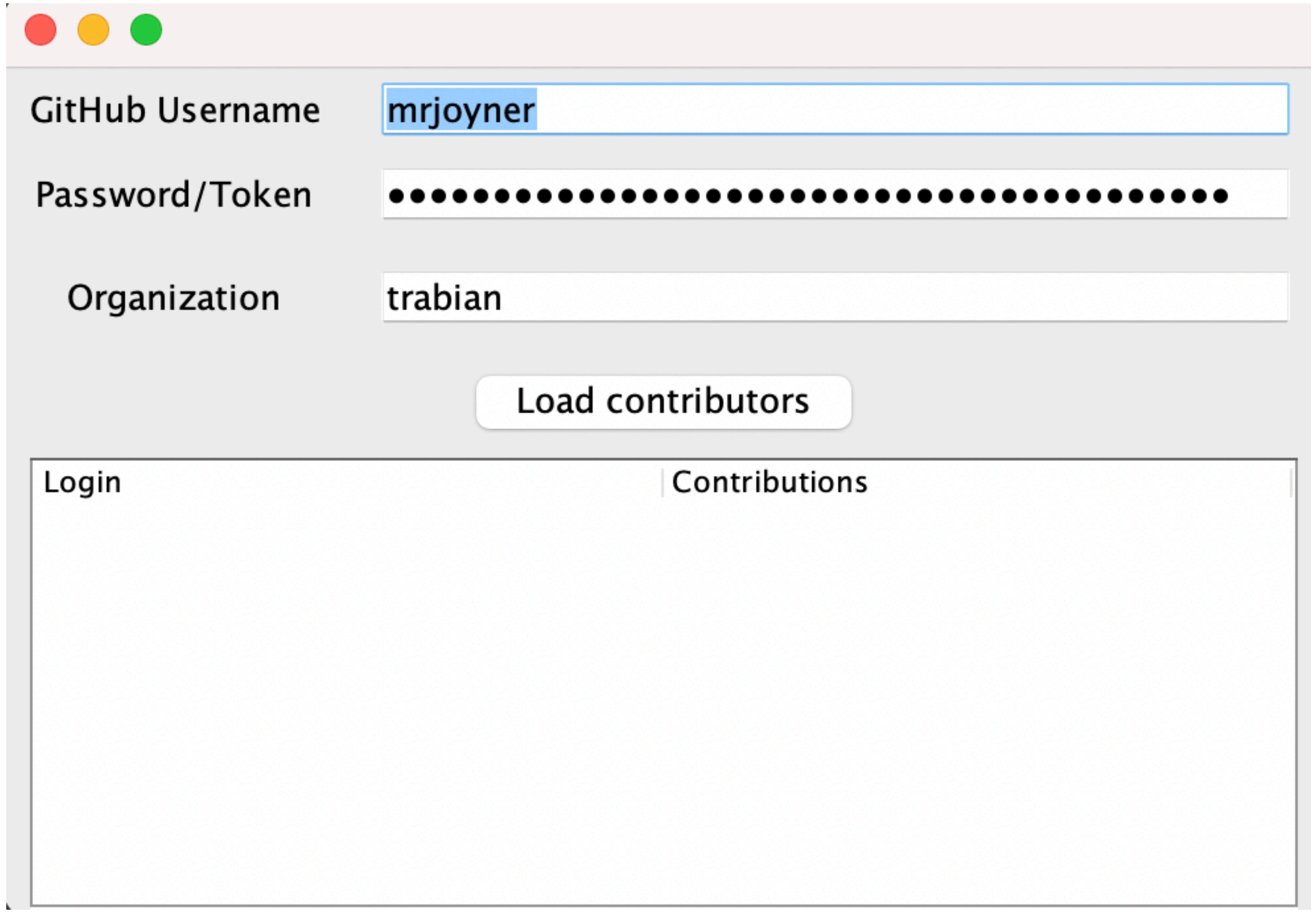
```
<button value="Click Me" onclick="alert('Thank you')" />
```

JavaScript

Java Swing

- Swing enables you to build a GUI in Java and respond to user events
- Containers (e.g. JFrame)
- Components
  - JButton
  - JLabel
  - JTextField
- Users interact with the GUI and trigger actions (events)
- ActionListeners are setup for a component to respond to the event

GitHub Contributors

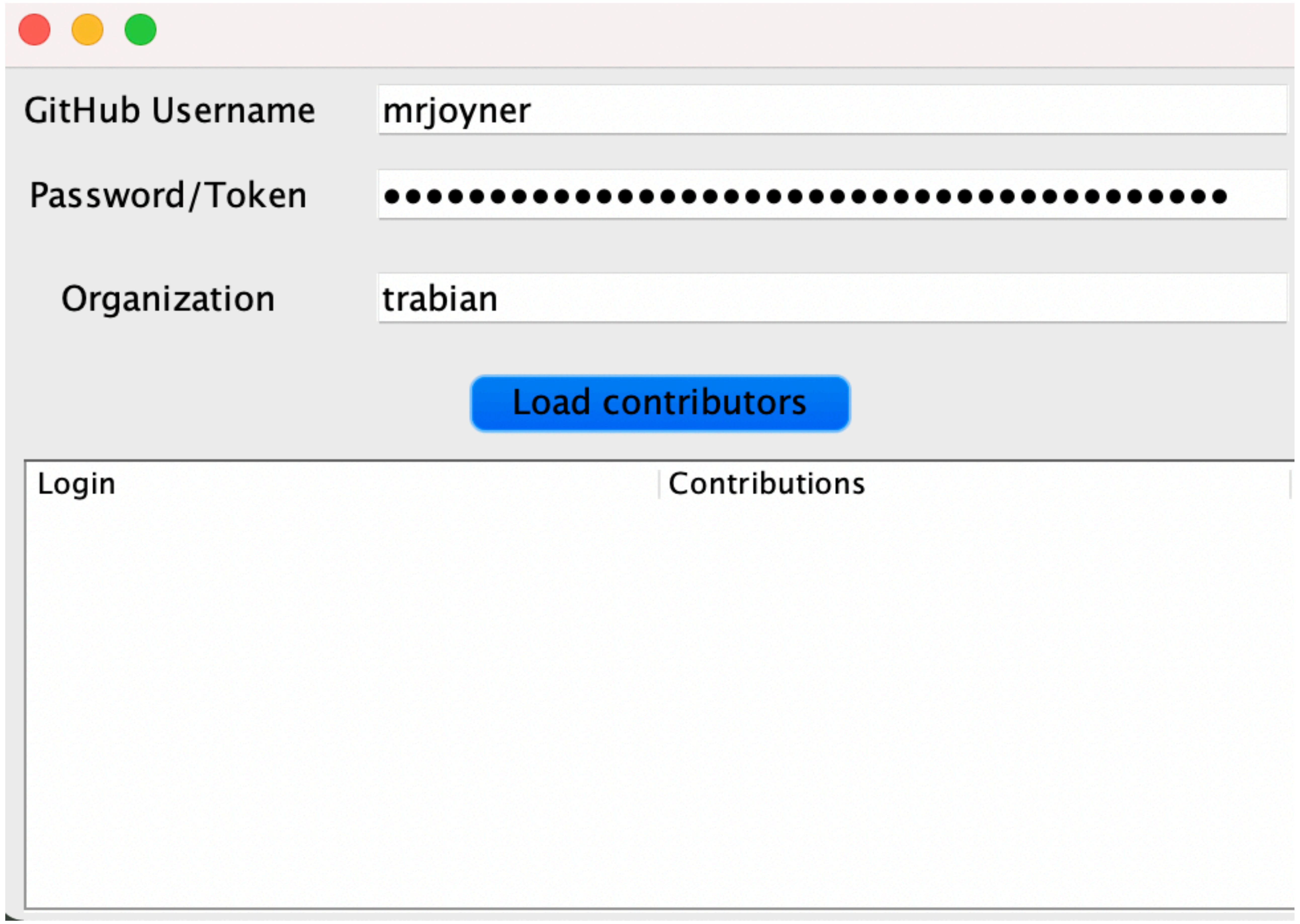


GitHub Username: mrjoyner

Password/Token: ●●

Organization: trabian

GitHub Contributors Event Handling with ActionListener


GitHub Username: mrjoyner

Password/Token: ••

Organization: trabian

Login	Contributions

ActionListeners

Adding ActionListener without a lambda

```
public class MultiListener ... implements ActionListener {
    ...
    //where initialization occurs:
        button1.addActionListener(this);
        button2.addActionListener(this);

        button2.addActionListener(new Eavesdropper(bottomTextArea));
    }

    public void actionPerformed(ActionEvent e) {
        topTextArea.append(e.getActionCommand() + newline);
    }
}

class Eavesdropper implements ActionListener {
    ...
    public void actionPerformed(ActionEvent e) {
        myTextArea.append(e.getActionCommand() + newline);
    }
}
```

component has multiple listeners

called on each button click

event information

See: https://docs.oracle.com/javase/tutorial/uiswing/events/intro.html

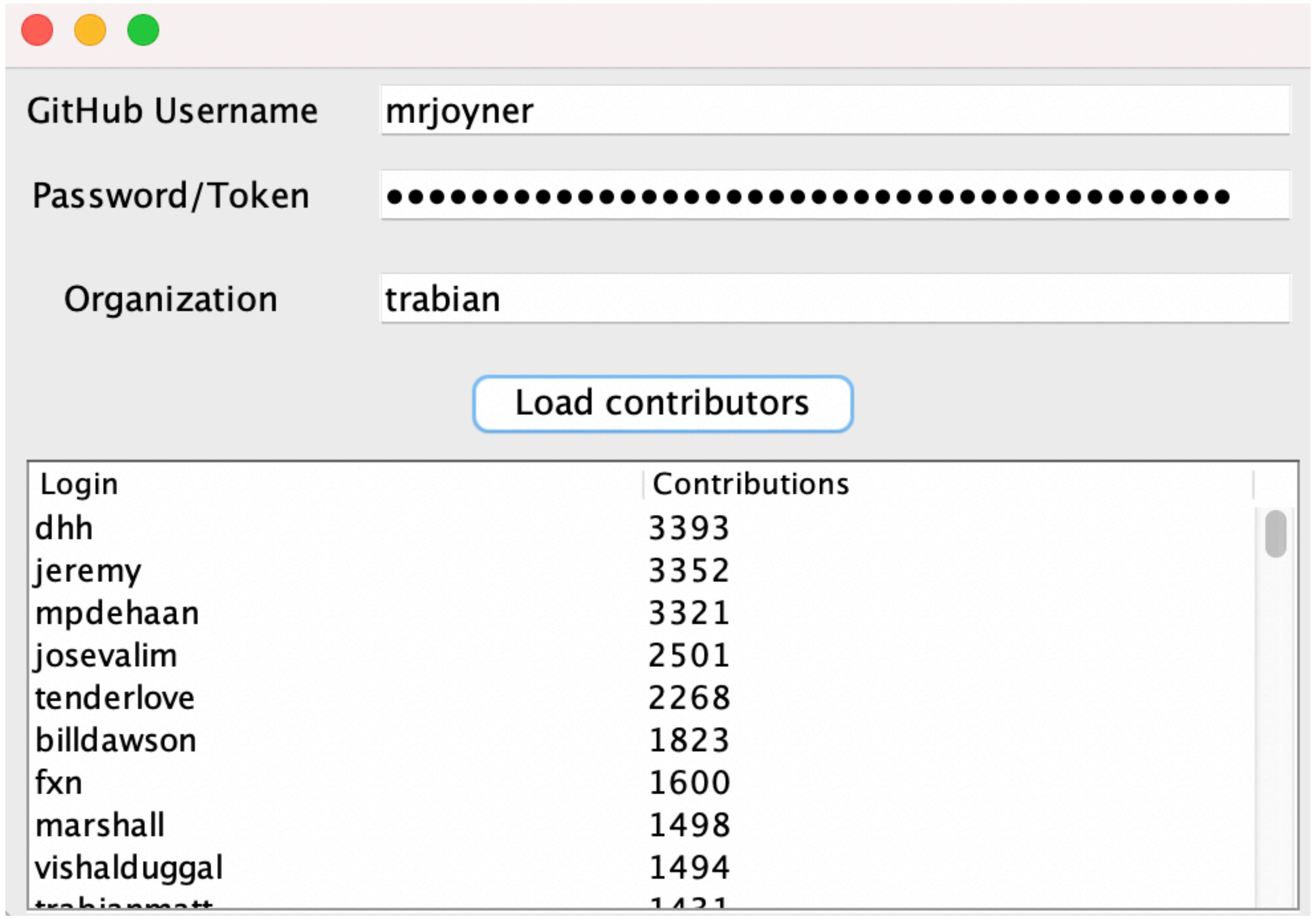
ActionListeners

Adding ActionListener with a lambda

```
/**
 * Adds action listener for load button.
 */
private void addLoadListener() {
    load.addActionListener(e -> {
        String userParam = username.getText();
        String passParam = String.valueOf(password.getPassword());
        String orgParam = org.getText();
        if (!userParam.isEmpty() && !passParam.isEmpty()) {
            saveParams(userParam, passParam, orgParam);
        }
        try {
            System.out.println("Loading Users ...");
            loadContributorsSeq(userParam, passParam, orgParam); //TODO change to use parallel implementation
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    });
}
```

lambda body instead of actionPerformed method

GitHub Contributors



GitHub Username	mrjoyner
Password/Token	●●
Organization	trabian

Login	Contributions
dhh	3393
jeremy	3352
mpdehaan	3321
josevalim	2501
tenderlove	2268
billdawson	1823
fxn	1600
marshall	1498
vishalduggal	1494
trabianmatt	1421

```java
private final String[] COLUMNS = {"Login", "Contributions"};
private final DefaultTableModel resultsModel = new DefaultTableModel(COLUMNS, 0);
public List<User> users = new ArrayList<>();
private final JTable results = new JTable(resultsModel);
private final JScrollPane resultsScroll = new JScrollPane(results);
```

```java
/**
 * Updates the contributors list displayed on the user-interface
 * @param users a list of Users
 */
public void updateContributors(List<User> users){
    Object[][] values = new Object[users.size()][2];
    for(int i = 0; i<users.size(); i++){
        values[i] = new Object[]{users.get(i).login, users.get(i).contributions};
    }
    this.users = users;
    resultsModel.setDataVector(values, COLUMNS);
}
```

Update GUI table with user contributions

Demo: GitHub Contributors