Home Work 1 - User Interface Event Notification and Handling

Introduction

The aim of this assignment is to provide experience in implementing Graphical User Interface components from scratch. In this assignment we will focus on the event notification and handling features of a button UI component. We will be implementing the code in Java. The sample project containing the template code has been shared in Collab under the resources folder (to be updated). It is recommended to use Java SDK 1.8 to compile and test the code, though it should be possible to use Java 9 as well.

Event notification and handling is commonly implemented using the Observer design pattern. The basic idea of the design pattern is to allow "observer" objects to register interest in specific "event types" on "observable" objects. Every time the corresponding event occurs on the "observable" object, a notification message is sent to the "observer" object. The Observer design pattern provides a flexible, reusable class design to achieve this. More details on the Observer design pattern can be found here http://www.oodesign.com/observer-pattern.html.

Class Design

The following UML class diagram describes the design for the Homework1 project. Detailed description of each method, the arguments passed and the expected return values/objects is provided in the interface files (filenames starting with "I") in the project.

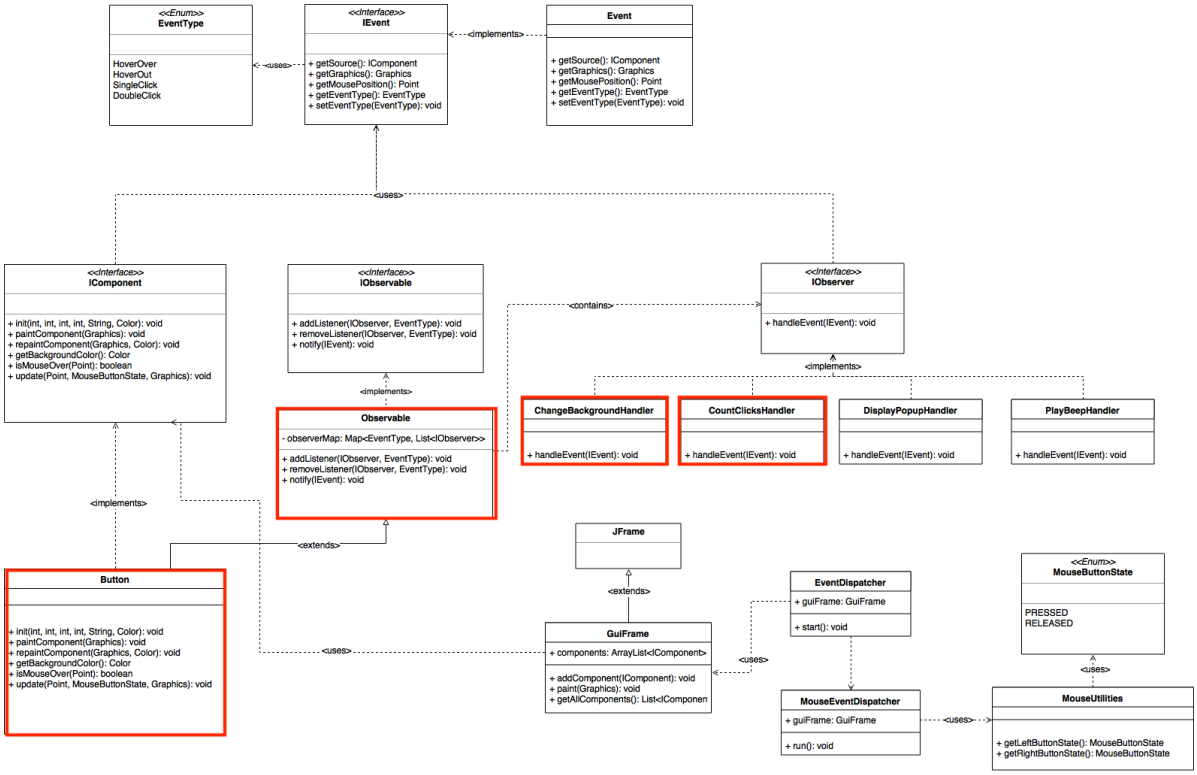


Figure 1 - UML Class Diagram

Third-party Libraries Used

The project requires using APIs in the lib/mouse-utils.jar third-party library.

This library internally has a dependency on the lib/jnativehook-2.1.0.jar library.

These libraries are used to get the current status of a mouse button.

The class MouseUtilities provides the following methods:

- MouseButtonState getLeftButtonState()

  This method returns:

  - MouseButtonState.PRESSED if the left button is currently in pressed state
  - MouseButtonState.RELEASED if the left button is currently in released state

- MouseButtonState getRightButtonState()

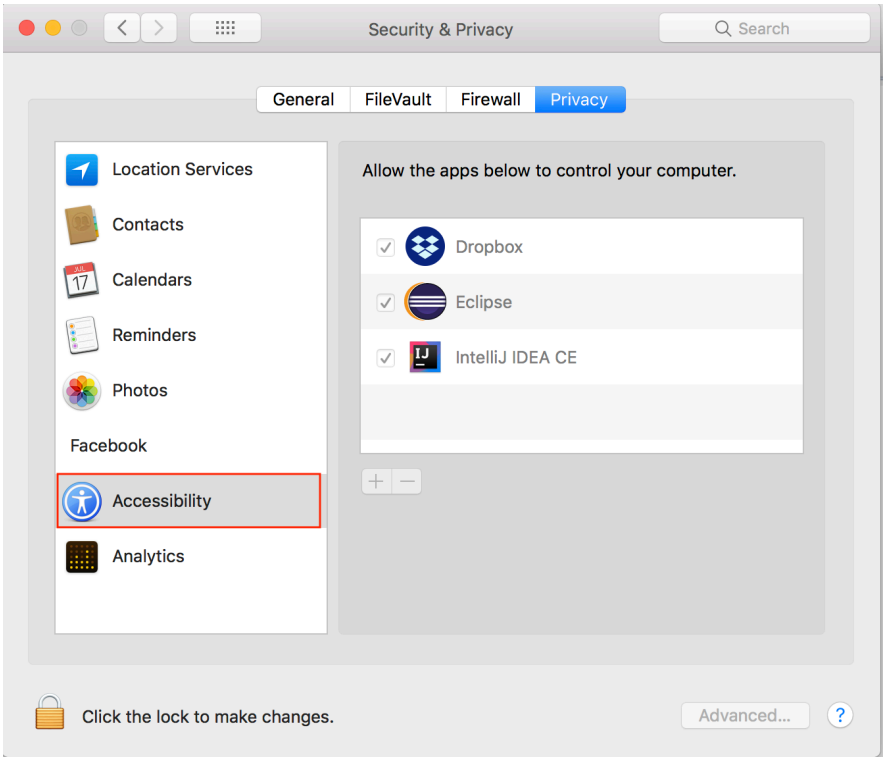
  This return behavior for this method is the same as the getLeftButtonState() method except that it checks the current state of the right button of the mouse.

The third-party libraries used in the project are supported on all OS platforms (Mac OSX, Windows and Linux), however, they have been tested on Windows and Mac OSX only. So it is recommended to use either one of these OS platforms for the assignment.

Mac OSX extra permissions

To use jnativehook on Mac OSX, you will need to add the process that is running the application to access the system resources. Please follow the below steps to do so:

1. Open System Preferences
2. Open Security & Privacy settings
3. Click on the “Accessibility” tab in the left panel.



4. Click on the lock icon (bottom left corner) to make changes.
5. Click on the “+” button to add the process that is running the application. For example, if you are running the application from Eclipse IDE, you should add “Eclipse” to the list of processes (apps in Mac OSX terminology) to control your computer, i.e., access system hardware such as the mouse buttons.

Template Project

A Java Eclipse project has been provided which contains the template Java class files with all the interface methods defined for each of the classes listed in the UML class diagram.

The classes that require implementation are highlighted with a red border in Figure 1. Certain methods / parts of certain methods in these classes have been left blank where the implementation needs to added. **These code blocks can be identified with the comment "TODO". Please add your implementation in these code blocks as per the description of the method and expected return behavior**.

The main method has been implemented in ButtonTester.java class. The code to register observers for "Button 1" has been added. **In the same manner, the code to register observers for "Button 2" needs to be added as per the requirements in the following section**. The main method constructs all the required UI components (Observables), Event Handlers (Observers) and registers them to their corresponding UI components.

Homework Deliverables

The following use cases need to be implemented for this assignment.

Use Case 1 - Hover over Button 1

Hovering the mouse over Button 1 should change the background of the button to RED color. On moving the mouse out of Button 1's area, the background color of the button should be restored to its original color (BLUE).

Use Case 2 - Hover over Button 2

Hovering the mouse over Button 2 should change the background of the button to GREEN color. On moving the mouse out of Button 2's area, the background color of the button should be restored to its original color (GRAY).

Use Case 3 - Single Click on Button 1

A single left click of the mouse over Button 1 should play a beep sound.

Use Case 4 - Single Click on Button 2

A single left click of the mouse over Button 2 should play a beep sound and display a popup message.

Use Case 5 – Double Click on Button 2

A double left click of the mouse over Button 2 should print the number of times the button was double clicked upon to the console output.