CST8221 – JAP
Lab 3 – Event Handling – Mouse Events and Cursors

Objectives

The purpose of the lab is to give you a hand-on experience of how to track the mouse actions and how to change the mouse cursor.

The Nature of Things

Event handling is one of the fundamental tasks in programming with GUI. To implement useful user interfaces, you must master the way in which Java handles events generated by different input devices such as mouse and keyboard.

Any operating environment that supports GUIs monitors events such as mouse movements and clicks or keystrokes. The operating environment reports these events to the running applications. Each application then decides what, if anything, to do in response to these events. In Java GUI API (both Swing and JavaFX), you completely control how events are transmitted from ***even sources*** (such as buttons) to ***event listeners***. You can designate ***any*** object to be an event listener. A listener object is an instance of a class that implements a special interface called a *listener interface*. The information about the event is encapsulated in an event object (such as *MouseEvent*) which is sent to an appropriate listener method. Which listener method is called on the listener object depends on the nature of the event generated by the event source. For example, whenever a user clicks on a button, the *JButton* (or *Button*) object creates an *ActionEvent* and a *MouseEvent* objects (both in Swing and JavaFX) and calls the appropriate listener method (*actionPerformed()* (or *handle(ActionEvent)*- see Example 4 in Lab 1*)* in the case of *ActionEvent,* or one of the mouse listener methods). In order to transmit the event form the event source to the event listener object, the event listener object must be registered with the event source. Event sources have specialized registering methods with convenient names. For example, *JButton* has a method called *addActionListener()* which is used to register action listeners. It has similar methods for registering different mouse listeners (*addMouseListener()* and so on).

When a mouse cursor hovers over a GUI it is customary to change the cursor shape to indicate to the user what action they can perform. Usually the operating environment provides a set of standard shapes, but the user can create shapes of their own.

Useful links: http://download.oracle.com/javase/tutorial/uiswing/events/intro.html
http://www.java2s.com/Code/Java/JavaFX/Listentoallmouseevents.htm

Tasks

Download the ***CST8221_Lab3_code.zip*** file from Blackboard. Extract the contents. Two sets of code files are provided for you – one for Swing GUI and one for JavaFX GUI. Swing GUI: ***MouseTest.jar, MouseTest.java***, ***CursorTest.jar***, ***CursorTest.java,*** and ***happy.gif***. JavaFX GUI: ***CursorDemoFX.java , MouseDemoFX.java*** and ***happy.gif***. The Swing classes ***MouseTest.java*** and ***CursorTest.java*** are incomplete. You are to complete them during this lab period.

Exercise 1 – Handling the Mouse Events

You have to modify ***MouseTest.java*** so it handles all possible mouse events defined by the three mouse listeners the class implements. To do so, follow the steps:

0. Run the Mouse Test reference implementation. To do so, open a command window and type at the command prompt `java -jar MouseTest.jar` and press **Enter**. Play with the application to see what you are expected to do. Your application should mimic the reference implementation.

1. Implement all listener methods with NOP operation (empty braces { }). You will find the names of the methods in the documentation of the corresponding interfaces which are part of *java.awt.event package*.
2. Register the three listeners with the ***button*** object.
3. Compile and run the application. When you click on the button the following message must be displayed in the Console window: `Method actionPerformed called`
4. Working method by method using System.out.println() print an appropriate message in each of the methods. For example, `Method mouseClicked called` and so on. Every time you implement a method, run the application and try to find what mouse action will invoke the method.
5. Once you finish all the methods, modify the *mouseClicked()* so that it reports the number of button clicks and the mouse button (left, middle, right) that has been clicked. Look at the *MouseEvent* class to find an appropriate method. Next, add more code that will report which mouse button is clicked. Use the *getButton()* method and compare the return value with one of the three constants (BUTTON1, BUTTON2, BUTTON3) defined in the MouseEvent class. Print the type of the button.
6. Modify the *mouseMoved()* so that it reports the coordinates of the mouse cursor when it enters the button. You will find two suitable methods inside the *MouseEvent* class. Record on paper the coordinates of the four corners of the button and the center of the image. You have to show the numbers to me during the demonstration.
7. Finally, modify the *mouseWheelMoved()* so that it displays the notches and the direction the wheel moved. Use the *getWheelRotation()* of the MouseWheelEvent class to get the notches.
8. Demonstrate your work if you want to earn some marks.

Exercise 2 – Changing the Mouse Cursor Shape

You have to modify ***CursorTest.java*** so it handles displays all standard cursor shape provider by the *Cursor* class in java.awt package. To do so, follow the steps:

0. Run the Cursor Test reference implementation. To do so, open a command window and type at the command prompt `java -jar CursorTest.jar` and press **Enter**. Play with the application to see what you are expected to do. Your application should mimic the reference implementation.

1. Initialize the cursor array with the following ***Cursor*** class constants:
   DEFAULT_CURSOR
   CROSSHAIR_CURSOR
   TEXT_CURSOR
   WAIT_CURSOR
   N_RESIZE_CURSOR
   S_RESIZE_CURSOR
   W_RESIZE_CURSOR
   E_RESIZE_CURSOR
   SW_RESIZE_CURSOR
   SE_RESIZE_CURSOR
   NW_RESIZE_CURSOR
   NE_RESIZE_CURSOR
   HAND_CURSOR
   MOVE_CURSOR

2. Add some code to the *actionPerformed()* so that it changes the cursor shape every the time the button is clicked. The logic of the code is very similar to the implementation of the same method in *SimpleSwingGUIe3.java* provided for you in Lab 1. Here you have to use the cursors array instead of the "look and feel" array.

3. Demonstrate your work if you want to earn some marks

Compile and run the JavaFX examples. Examine the code. See the differences between Swing and JavaFX in the implementation of the same tasks.

Before the lab
Enjoy Java.

During the lab
Ask questions and modify the programs.

Before leaving the lab
Demonstrate your work.
Sign the attendance sheet.

After the lab
Remember what you have learned. You will need it later.

Submission
No submission is required for this lab but you have to demonstrate your work before the end of the lab period if you want to earn some marks.

Marks: 2% (1%+1%) of your course mark
The lab exercises will be marked according to the following marking method:

```
public int markLab3(boolean demonstration,
                    boolean workingProgram){
    int mark = 0;
    while(!endOfLab3Period)
     if(demonstration & workingProgram) ++mark;++mark;
    return mark;
}
```

"*The best laid schemes o' mice an' men / Gang aft agley(Often go wrong)* " Robert Burns