ESS 201: Programming II
Term 1, 2019-20
Java Lab Assignment 4
Version 3

GUI with Mouse events
(Please see highlighted parts below. These are changes from the first and second versions)

Most Graphical User Interface (GUI) frameworks are designed as a class hierarchy of graphical elements such as panels, buttons, text boxes, labels, whose appearance and behaviour can be customized by the developer to create the effects needed for an application. We will try to model the functionality of a simple GUI.

Assume we want to create a UI consisting of a text box, a button and a text label. They each have the following properties:

- TextBox:
    - Gets focus when the mouse enters the text box region. After that, all keyboard events (characters) are sent to this text box, and the text box displays the text that was typed in. Stops getting text input when the mouse moves outside the box. For simplicity, we will not worry about editing commands like delete/backspace etc. Clicking within the TextBox has no effect.
- Label:
    - DIsplays a static string. No response to mouse events
- Button:
    - If the mouse is clicked inside the box, it executes a method associated with this mouse event.
- Widget:
    - This is the base class of the above elements. All widgets are rectangular, have a size (width and height). A Widget can be queried for its size.
- Panel:
    - This is a special widget. The program creates a “root” widget and attaches widgets (labels, buttons, text boxes) as needed. Widgets are added to the panel by specifying the position of the lower-left corner of the widget in the pixel coordinates of the Panel. Position (0,0) corresponds to the lower left corner of the Panel. We assume that panels cannot contain other panels, and that widgets do not overlap.

A derived class of widget, that wants to react to mouse or keyboard events, should implement either or both of the following interfaces:

1. **interface MouseWatcher**, which contains the following methods:
    a. public void onEnter() - called when the mouse enters the widget
    b. public void onExit() - called when the mouse exits the widget

c. public void moveTo(int x, int y) - when the mouse moves to x,y within the widget. **Note: this is a new method added in v2**
d. public void onClick(int x, int y) - called when the mouse is clicked within the widget. The values x, y are in the pixel coordinates of the widget (and should be the same values as the last moveTo() call)
e. public Widget getWidget() - returns the Widget reference if the derived class is also a subclass of Widget, and null otherwise **(Note. added in v3. Please see note at end of description)**

2. **interface KeyboardWatcher,** which contains:
   a. public void onKbdEvent(character x) - called when the character x is typed, and the mouse is within the widget
   b. public Widget getWidget() - returns the Widget reference if the derived class is also a subclass of Widget, and null otherwise. **(Note. added in v3)**

All keyboard and mouse events are processed first by the root panel, and it, in turn, sends them to the appropriate widgets. Panel **implements** both the MouseWatcher and KeyboardWatcher interfaces. What should the other sub-types of Widget implement?

The Panel class contains two methods that are used to register widgets for either mouse events or keyboard events or both:

- public void addMouseWatcher(MouseWatcher m) { … }
- public void addKeyboardWatcher(KeyboardWatcher w) { … }

Note that an instance of (a derived class of) widget can be registered for both types of events if it is appropriately designed.

The Panel class provides a method by which widgets are added to it.

- public void addWidget(Widget w, int x, int y)
  adds Widget w with position (x,y) in the pixel coordinates of the Panel. Note that the addWidget only informs the Panel about the existence and position of a Widget. The Widget still needs to be registered for mouse or keyboard events as needed.

When the Panel receives a mouse event (through its implementation of the MouseWatcher methods), it checks among all the widgets that have registered for mouse events to see which widget this might correspond to (that is, which widget covers the position of the mouse event), converts the position to the pixel values relative to the position of the widget, and invokes the appropriate MouseWatcher method of that widget. Note that, for a mouse movement, we are only interested in the end points of the move, and therefore the widgets that the mouse might have exited and/or entered. Other widgets in between are not notified.

When the Panel receives a keyboard event, it invokes the KeyboardWatcher event of all widgets that have registered with it for keyboard events.

For our example, we have the following:

1. A root Panel, of width 600 and height 800. This contains the following at the specified locations:
    a. A Label of width 200 and height 100 positioned at (50, 500)
    b. A TextBox of width 400 and height 200, positioned at (100, 300)
    c. A Button of width 200 and height 100, positioned at (250, 100)
2. When the textbox receives keyboard inputs (which will happen when the mouse is within the text box area), it prints out the complete string typed in so far - for each character input
3. When the button receives a mouse event, it print out the message: “Selected point: x y”, where x and y are the coordinates as sent in the onClick event

The main program does the following:

1. Instantiates the root panel
2. Creates each of the widgets defined above and adds them to the root panel
3. Reads the standard input, and for each line of input, invokes the appropriate MouseWatcher or KeyboardWatcher event of the root panel. Format of the input is defined below:
    a. The first word is one of MoveTo, MouseClick, KeyPressed or End
    b. if MoveTo, the next two integers are the position (in pixel coordinates of the root panel)
    c. if MouseClick, there is no other data. The clicked point is the same as the end point of the last MoveTo
    d. if KeyPressed, the next character is the character that was typed
    e. if End, that signals the end of input

    Assume the mouse starts at 0,0

Note (changes in v3):

The changes in v2 do not completely solve the problem, since we have to do something similar for KeyboardWatcher.

To keep things simple, we drop the idea of WidgetWithMouse and instead we add a method

```
public Widget getWidget()
```

to each of the interfaces MouseWatcher and KeyboardWatcher.
Implementationally, any class that extends Widget and implements one of these interfaces should return a reference to itself in this method. i.e.

```
return this;
```

The Panel class can invoke this method to get the reference to the Widget object corresponding to the Watchers. If this method returns null, then we can assume this is not a Widget.

Sample Input

```
MoveTo 400 400
MouseClick
KeyPressed H
KeyPressed i
KeyPressed T
KeyPressed h
KeyPressed e
KeyPressed r
MoveTo 50 20
KeyPressed e
MouseClick
MoveTo 350 150
MouseClick
KeyPressed N
MoveTo 150 550
MouseClick
KeyPressed M
MoveTo 250 350
KeyPressed e
End
```

Expected output

```
H
Hi
HiT
HiTh
HiThe
HiTher
Selected point: 100 50
HiThere
```