Event Listeners

I. Definitions
   a. **Events**: Any action that occurs while in your user interface
      i. Such as moving the mouse, pressing the mouse, releasing the mouse, selecting a button, or selecting a menu
      ii. There are many events happening constantly so a program must specify which events to focus on
   b. **Event Listener**: classes that contain the code that executes when the event occurs
   c. **Event Source**: the component in your user interface that generates an event

II. Adding event listeners to your program
   a. Must create a separate class for each event that you want to listen to
      i. Each of these classes must implement the ActionListener interface
      ii. **Warning:** Remember that polymorphism requires that the signature be the same for the method you are implementing from an interface
   b. The actionPerformed() method, which is the only method that needs to be implemented from the ActionListener interface, has a parameter of ActionEvent event.
      i. This event parameter has more information about the event, such as the time the even occurred
   c. In your main class, need to attach an event listener to the event source using addActionListener of the event source object
      i. This part is extremely important because without this step, your buttons or menus will **NOT** work.
   d. Example code:

```
JButton button = new JButton("Click me!");
ActionListener listener = new MySuperDuperListener();
button.addActionListener(listener); //this line adds the listener to the button
```

III. Timer events are also known as an event listener because there is an event generated each time the defined interval passes
   a. Sample code:
   MyListener listener = new MyListener();
   Timer t = new Timer(interval, listener);
   t.start();

IV. If you want to use the mouse as an event listener, there is a separate mouseListener interface that you must implement instead of the actionListener interface
   a. The mouseListener interface has 5 methods that must be implemented: mousePressed(), mouseReleased(), mouseClicked(), mouseEntered(), and mouseExited()
      i. All five of the methods have the parameter MouseEvent event
      ii. mouseClicked() is called when the mouse is pressed and then released in a short amount of time and the mouse does not move. If all of these conditions are met, there are three methods called: mouseClicked(), mousePressed(), and mouseReleased().
   b. Most common event implemented for the user interface is the mousePressed() method and therefore most users expect the code to execute when they press on the mouse
   c. Can have methods of the MouseListener interface do nothing by simply not inserting code into the methods

i. **Important:** The class must still implement all of the methods because MouseListener is an interface. The methods that don't do anything, known as do-nothing methods, just don't need any code within the method.

d. Instead of implementing the MouseListener interface and having multiple methods do nothing, you can extend the MouseAdapter class which implements the MouseListener interface and implements all five of the methods as do-nothing methods.
    i. Allows the use of polymorphism to change only the events that your program will use.

V. Event Listener classes should be added ***INSIDE*** the classes that will use it ('tis good practice)

a. Example (partly from book): trigger an event every certain time interval

```
class TimerStuff{
     class MyListener implements ActionListener
     {
          public void actionPerformed(ActionEvent event)
          {
               //a bunch of awesome code to execute every time interval
          }
     }

     MyListener listener = new MyListener();
     Timer t = new Timer(interval, listener);
     t.start();
}
```

b. This creates less clutter because the listener is in the same place as the event source that has the event listener attached to it

c. Also allows the programmer to use variables defined elsewhere in the code within the event listener class and vice versa
    i. **Important:** These variables must be declared as final

VI. To create a menu with an event listener:
    i. Create the menu (ex: JMenuItem fileExitItem = new JMenuItem("Exit");)
    ii. Add the file menu (ex: fileMenu.add(fileExitItem);)
    iii. Add the event listener (ex: fileExitItem.addActionListener(listener);)
    iv. No action is sent from opening the submenus from the main menu, only when selecting the sub menu

VII. When drawing shapes such as rectangles, circles, etc. when an event occurs, it is important to always call the repaint() method in order to update the screen to the shape object

a. Example code:

```
public void moveBy(int dx, int dy){

     box.translate(dx, dy);

     repaint();

}
```

b. **Important:** It is vital that you call the repaint method or the box will not be redrawn in the user interface

c. The implemented actionPerformed() method can now call the moveBy method every time the event source is called, such as a timer, and the box will repaint moved dx by dy