

Keyboard Input Events

Time required: 45 minutes

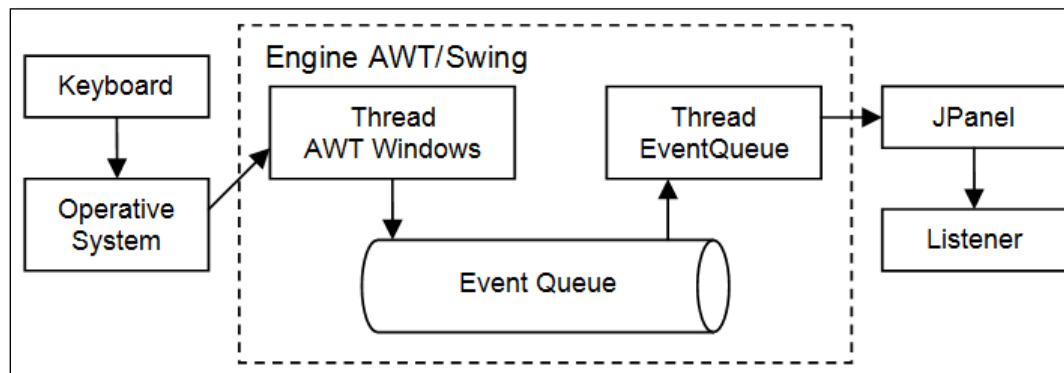
Contents

Keyboard Input Events	1
How do Events Work in AWT/Swing?.....	1
Keyboard Reading Example	2
Anonymous Class	4
Assignment Submission.....	5

In this tutorial we will see how the events work and how to obtain information about keyboard events. We will explain the concept and the use of the anonymous classes, which are the most common way of managing events in java. We will leave aside the game for a moment and explain the capture of events in a simple example.

How do Events Work in AWT/Swing?

The keyboard and mouse events are controlled by the operating system. The AWT engine and the AWT-Window thread communicate with the operating system and knows when an event occurs. When a new event comes along it is placed in the "Event queue" so that it is attended by the AWT-EventQueue thread in its turn.



When the AWT-EventQueue thread attends an event, it pays attention to what component it affects and it informs it. In our case the component is the JPanel, which informs all the listeners which are registered to receive notifications for this event.

In the case of the keyboard, the call to `addKeyListener(KeyListener listener)` is the one which carries out this register. If we want to register an object to listen to the events of the mouse we can use `addMouseListener(MouseListener listener)`.

Keyboard Reading Example

To read from the keyboard it is necessary to register an object which will be in charge of "listening if a key is pressed". This object is known as a "Listener" and it will have methods that will be called when someone presses a key.

In our example the Listener is registered in the JPanel (or KeyboardExample) using the `addKeyListener(KeyListener listener)` method.

```

1 import java.awt.event.KeyEvent;
2 import java.awt.event.KeyListener;
3 import javax.swing.JFrame;
4 import javax.swing.JPanel;
5
6 public class KeyboardExample extends JPanel {
7     private static final long serialVersionUID = 1L;
8
9     public KeyboardExample() {
10         // Create new KeyListener object
11         KeyListener listener = new KeyListener() {
12             @Override
13             public void keyTyped(KeyEvent e) {
14             }
15
16             // Listen to key pressed events and get key pressed
17             @Override
18             public void keyPressed(KeyEvent e) {
19                 System.out.println("keyPressed=" + KeyEvent.getKeyText(e.getKeyCode()));
20             }
21
22             // Listen to key released events and get key released
23             @Override
24             public void keyReleased(KeyEvent e) {
25                 System.out.println("keyReleased=" + KeyEvent.getKeyText(e.getKeyCode()));
26             }
27         };
28         // Register KeyListener to JPanel
29         addKeyListener(listener);
30         // Allow focus to JPanel
31         setFocusable(true);
32     }
33
34     public static void main(String[] args) {
35         JFrame frame = new JFrame("Keyboard Events");
36         KeyboardExample keyboardExample = new KeyboardExample();
37         frame.add(keyboardExample);
38         frame.setSize(200, 200);
39         frame.setVisible(true);
40         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
41     }
42 }

```

In the constructor of the KeyboardExample class, we create the listener and we register it. To ensure that the JPanel object receives the keyboard notifications it is necessary to include the instruction `setFocusable(true)`, which allows KeyboardExample to receive the focus.

```
public KeyboardExample() {  
    // Create new KeyListener object  
    KeyListener listener = new KeyListener() {  
        @Override  
        public void keyTyped(KeyEvent e) {  
        }  
    }  
}
```

The **KeyListener** class is used to create the **Listener** object.

KeyListener is an abstract class. We must override all the methods whether we use them or not. We are not going to use the **keyTyped** method, we implement it anyway as an empty method.

This Listener will write to the console the name of the method and the key which are affected by the event.

Once it is registered, when KeyboardExample (our JPanel) has the focus and someone presses a key, KeyboardExample will report it to the listener registered. The Listener of our example implements the KeyListener interface which has the keyTyped(), keyPressed() and keyReleased() methods. The keyPressed method will be called each time the key is pressed (and several times if the key is maintained pressed).

The keyTyped(), keyPressed() and keyReleased() methods receive a KeyEvent object as a parameter, which contains information on which key has been pressed or released. Using e.getKeyCode() we can obtain the key. If we pass a key code to KeyEvent.getKeyText(...), we can obtain the text which is associated to the key.

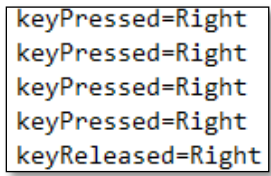
Anonymous Class

This example shows how to use an anonymous class. It can be thought of as a class inside of a class.

Even if it looks a bit strange, this is the best way of implementing the event Listeners and it is the way you will see it in most advanced java code.

Example run:

A little window will show up. When you press the cursor/arrow keys, the results will be shown in the console.



```
keyPressed=Right  
keyPressed=Right  
keyPressed=Right  
keyPressed=Right  
keyReleased=Right
```

Assignment Submission

Attach the .java file to the assignment in Blackboard.