HOW TO WRITE *MOUSELISTENER* AND *MOUSEMOTIONLISTENER* EVENTS

A mouse event is generated when the mouse cursor is moved into or out of the area occupied by a component, when one of the mouse buttons is pressed, released or clicked, or when the mouse is moved or dragged with a button pressed.

In Java, mouse event objects belong to a class called **MouseEvent**. Any objects that need to listen for key events must implement the **MouseListener** or **MouseMoveListener** interface. The object must be registered with a component by calling the component's **addMouseListener()** method. The registration is done with the following command:

```
component.addMouseListener(listener);
component.addMouseMotionListener(listener);
```

The **listener** is the object that listens for mouse events, and **component** is the object that will generate the mouse events (when it has the input focus).

The **MouseListener** interface defines the following methods to respond to events arising when the mouse cursor is moved into or out of the area occupied by a component, or one of the mouse buttons is pressed, released, or clicked:

METHODS	DESCRIPTION
mouseClicked(MouseEvent e)	Called when a mouse button is clicked on a component (i.e. when the button is pressed and released.
mousePressed(MouseEvent e)	Called when a mouse button is pressed on a component.
mouseReleased(MouseEvent e)	Called when a mouse button is released on a component.
mouseEntered(MouseEvent e)	Called when the mouse enters the area occupied by a component.
mouseExited(MouseEvent e)	Called when the mouse exits the area occupied by a component.

The **MouseMotionListener** interface defines the following methods that are called when the mouse is moved or dragged with a button pressed:

METHODS	DESCRIPTION
mouseMoved(MouseEvent e)	Called when the mouse is moved within a component.
mouseDragged(MouseEvent e)	Called when the mouse is moved within a component while a mouse button is held down.

The following are just some of the methods included in the **MouseEvent** class:

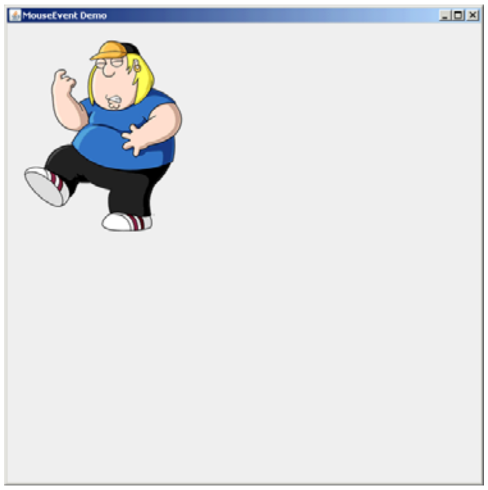
METHODS	DESCRIPTION
int getButton()	Indicates which mouse button has been clicked. Possible values include **MouseEvent.BUTTON1**, **MouseEvent.BUTTON2** and **MouseEvent.BUTTON3.**
int getClickCount()	Returns the number of mouse clicks associated with this event.
Point getPoint()	Returns a Point object containing the x and y coordinates.
int getX()	Returns the x-coordinate of the mouse point.
int get ()	Returns the y-coordinate of the mouse point.

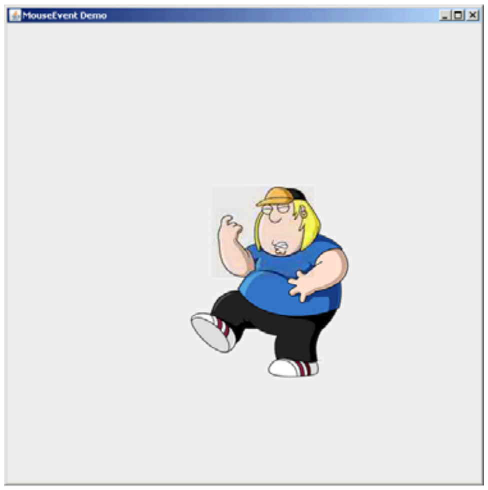
The following example will demonstrate how to create a program that allows the user to move an image by dragging on it and repositioning the image once the user releases the mouse button:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class MouseEventDemo extends JPanel implements
   MouseMotionListener {

   // Declare global variables
   private ImageIcon pic;
   private int x = 20, y = 20;

   public static void main(String[] args) {

      new MouseEventDemo ();
   }

   public MouseEventDemo() {

      // Initialize ImageIcon object
      pic = new ImageIcon("chrisGriffin.png");

      // Set properties of the JPanel object
      setLayout(null);
      addMouseMotionListener(this);

      // Declare and initialize JFrame object
      JFrame frame = new JFrame();
      frame.setContentPane(this);
      frame.setTitle("MouseEvent Demo");
      frame.setSize(600, 600);
```

```
         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         frame.setLocationRelativeTo(null);
         frame.setVisible(true);
      }

      public void mouseDragged(MouseEvent e) {

         // Get the x- and y-coordinates of the mouse point
         x = e.getX();
         y = e.getY();
         repaint ();
      }

      public void mouseMoved(MouseEvent e) {}

      public void paintComponent(Graphics g) {

         // Refresh the panel and draw the image at the x- and y-
            coordinate of the mouse point
         super.paintComponent(g);
         Graphics2D g2 = (Graphics2D) g;
         g2.drawImage(pic.getImage(), x, y, this);
      }
   }
```

When you run the above program, you will see an image located at 20, 20. When you drag the mouse over the image the image moves to wherever the pointer is at the time.





EXTENDING THE *MouseMotionAdapter* CLASS

You'll notice that even though I did not use the **mouseMoved()** method, I still need to declare the method in my program. To avoid having to do this extra step, you can create an inner class that extends the **MouseMotionAdapter** class.

The first thing we would have to change from our original version is the class declaration. In the class declaration we would need to remove **implements MouseMotionListener**.

```
public class MouseEventDemo extends JPanel ~~implements~~
     ~~MouseMotionListener~~
```

The next step would be removing the reference to the **JPanel** when adding the key listener and instead have it point to a new instance of our inner class, which we'll call **ImageListener**:

```
addMouseMotionListener(new ImageListener());
```

The final step requires us to declare our inner class and include the instructions for the **keyPressed()** method:

```
class ImageListener extends MouseMotionAdapter {

   public void mouseDragged(MouseEvent e) {
      ...
   }
}
```