

Lecture 18

Mouse Events

Mouse events are reported by two listening events `MouseListener` and `MouseMotionListener`. `MouseListener` has five methods:

```
public void mousePressed(MouseEvent e)
public void mouseClicked(MouseEvent e)
public void mouseReleased(MouseEvent e)
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
```

In addition, `MouseEvent` has the methods `getX()` and `getY()` which get the x and y coordinates of the mouse's location.

`MouseMotionListener` has two methods:

```
public void mouseDragged(MouseEvent e)
public void mouseMoved(MouseEvent e)
```

The application below implements both listeners. Consequently, all seven methods have to be defined in the body of the class.

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

public class MouseTracker extends JFrame
    implements MouseListener, MouseMotionListener
{
    private JLabel statusBar;

    public MouseTracker()
    {
        super("Mouse Events");

        statusBar = new JLabel();
        statusBar.setBackground(Color.white);

        //some problem with "setBackground(Color.white)"
        // on labels. See output below.
        statusBar.setForeground(Color.red);

        statusBar.setFont(new Font("Monospaced", Font.BOLD,20));
        getContentPane().add(statusBar, BorderLayout.SOUTH);
        addMouseListener(this);
        addMouseMotionListener(this);
        setSize(275,300);
        setLocation(200,200);
        show();
    }
} // end MouseTracker()
```

```

public void mouseClicked(MouseEvent e)
{
    statusBar.setText("Clicked at ["+
        e.getX()+ ", "+e.getY()+"]");
}

public void mousePressed(MouseEvent e)
{
    statusBar.setText("Pressed at ["+
        e.getX()+ ", "+e.getY()+"]");
}

public void mouseReleased(MouseEvent e)
{
    statusBar.setText("Released at ["+
        e.getX()+ ", "+e.getY()+"]");
}

public void mouseEntered(MouseEvent e)
{
    statusBar.setText("Mouse in window");
}

public void mouseExited(MouseEvent e)
{
    statusBar.setText("Mouse outside window");
}

public void mouseDragged(MouseEvent e)
{
    statusBar.setText("Dragged at ["+
        e.getX()+ ", "+e.getY()+"]");
}

public void mouseMoved(MouseEvent e)
{
    statusBar.setText("Moved at ["+
        e.getX()+ ", "+e.getY()+"]");
}

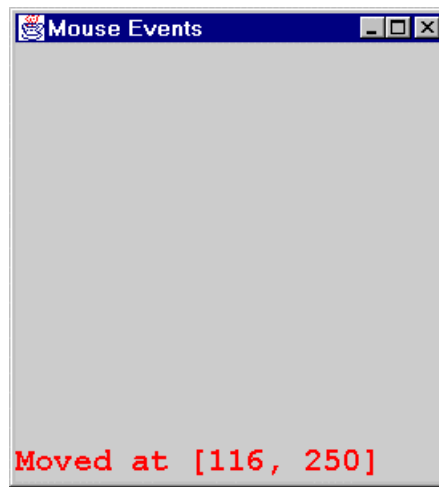
public static void main(String args[])
{
    MouseTracker app = new MouseTracker();

    app.addWindowListener
    (
        new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                System.exit(0);
            }
        }
    );
}

} // end    main()
} // end    class MouseTracker

```

The output is a window that reports on the various methods of the interfaces:



Adapter Classes

If you wish to use a mouse listener and only want to use one of the methods of an interface you still have to define all the methods listed in the interface. This problem can be finessed by using the appropriate adapter class.

This is done below (blue text) by constructing an anonymous listener. The method `mouseDragged(MouseEvent e)`, the only method from the two in the interface `MouseMotionListener`, is defined. The `MouseMotionAdapter()` provides an empty definition for the classes not employed.

This technique was used earlier `WindowListener`, which is an interface with seven methods. A `WindowAdapter()` was used to restrict the definition to the method `windowClosing()`

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

public class Painter extends JFrame
{
    private int xValue = -10, yValue = -10;

    public Painter()
    {
        getContentPane().add
        (
            new Label("Drag the mouse to draw"),
            BorderLayout.SOUTH
        );
    }
}
```

```

addMouseMotionListener
(
    new MouseMotionAdapter()
    {
        public void mouseDragged(MouseEvent e)
        {
            xValue = e.getX();
            yValue = e.getY();
            repaint();
        }
    }
);

setSize(300,300);
setLocation(300,300);
show();
} // end Painter()

public void paint(Graphics g)
{
    g.fillOval(xValue, yValue, 4, 4);
}

public static void main(String args[])
{
    Painter app = new Painter();

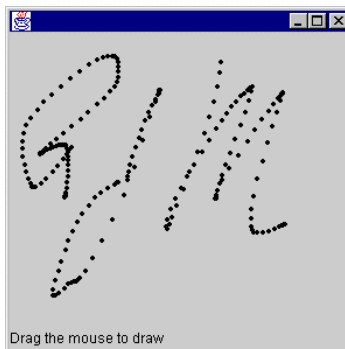
    app.addWindowListener
    (
        new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                System.exit(0);
            }
        }
    );

} // end main()

} // end class Painter

```

This application lets one make rough drawings:



Left mouse, Center mouse, Right mouse

The next application shows how to distinguish which mouse button was clicked. It also provides another example of an inner handler.

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

public class MouseDetails extends JFrame
{
    private int xPos, yPos;
    private String s = "";

    public MouseDetails()
    {
        addMouseListener(new MouseClickHandler() );

        setSize(500,100);
        setLocation(300,300);
        show();
    } // end MouseDetails()

    public void paint(Graphics g)
    {
        g.drawString(
            "Clicked @ [" + xPos+ ", " +yPos+"] ", xPos,yPos);
    }

    public static void main(String args[])
    {
        MouseDetails app = new MouseDetails();

        app.addWindowListener
        (
            new WindowAdapter()
            {
                public void windowClosing(WindowEvent e)
                {
                    System.exit(0);
                }
            }
        );
    } // end main()

    private class MouseClickHandler extends MouseAdapter
    {
        public void mouseClicked(MouseEvent e)
        {
            xPos = e.getX();
            yPos = e.getY();
            String s = "clicked " + e.getClickCount();
            s += " time(s)";
        }
    }
}
```

```
        if(e.isMetaDown())
            s += " with right mouse button";

        else if (e.isAltDown())
            s += " with center mouse button";

        else
            s += " with left mouse button";

        setTitle(s);

        repaint();

    } // end mouseClicked(MouseEvent e)
} // end class MouseClickhandler
} // end class MouseDetails
```

