**Event Handling in Java**

***Event:***

* Change in the state of an object is known as event
* Event describes the change in state of source.
* Events are generated as result of user interaction with the graphical user interface components.
* For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that causes an event to happen.

***Important Events***

* Window Event
* Mouse Event
* Acton Event
* Key Event

***Handling Event in java***

* Event generated as event object.
* Event object is passed to event listener.
* Event listener handles the event
* Events are represented as classes and Event Listeners are represented as interfaces.

***Steps to create a java program that handles the event***

* Create Class that implements Listener Interface
* Implement Listener methods
* Add listener in class constructor

**Window Event**

1. Lifecycle stages of a window
   * Window Opened
   * Window Closing
   * Window Closed
   * Window Activated
   * Window Deactivated
   * Window Deiconified
   * Window Iconified

**Eg Program**

import java.awt.Canvas;

import java.awt.Frame;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.event.WindowEvent;

import java.awt.event.WindowListener;

import java.awt.geom.Rectangle2D;

class MyCanvas extends Canvas{

public void paint(Graphics g) {

Graphics2D g2d = (Graphics2D)g;

Rectangle2D.Float r2d = new Rectangle2D.Float(100, 100, 100.67F, 100.657F);

g2d.draw(r2d);

}

}

class MyFrame extends Frame implements WindowListener{

public MyFrame() {

add(new MyCanvas());

setSize(500, 500);

// Adding Window Listener

addWindowListener(this);

setTitle("Frame Demo");

setVisible(true);

}

//Window Listener Implementation

@Override

public void windowActivated(WindowEvent e) {

System.out.println("windowActivated");

}

@Override

public void windowClosed(WindowEvent e) {

System.out.println("windowClosed");

}

@Override

public void windowClosing(WindowEvent e) {

System.out.println("windowClosing");

//Method the close the awt window

dispose();

}

@Override

public void windowDeactivated(WindowEvent e) {

System.out.println("windowDeactivated");

}

@Override

public void windowDeiconified(WindowEvent e) {

System.out.println("windowDeiconified");

}

@Override

public void windowIconified(WindowEvent e) {

System.out.println("windowIconified");

}

@Override

public void windowOpened(WindowEvent e) {

System.out.println("windowOpened");

}

}

public class Main {

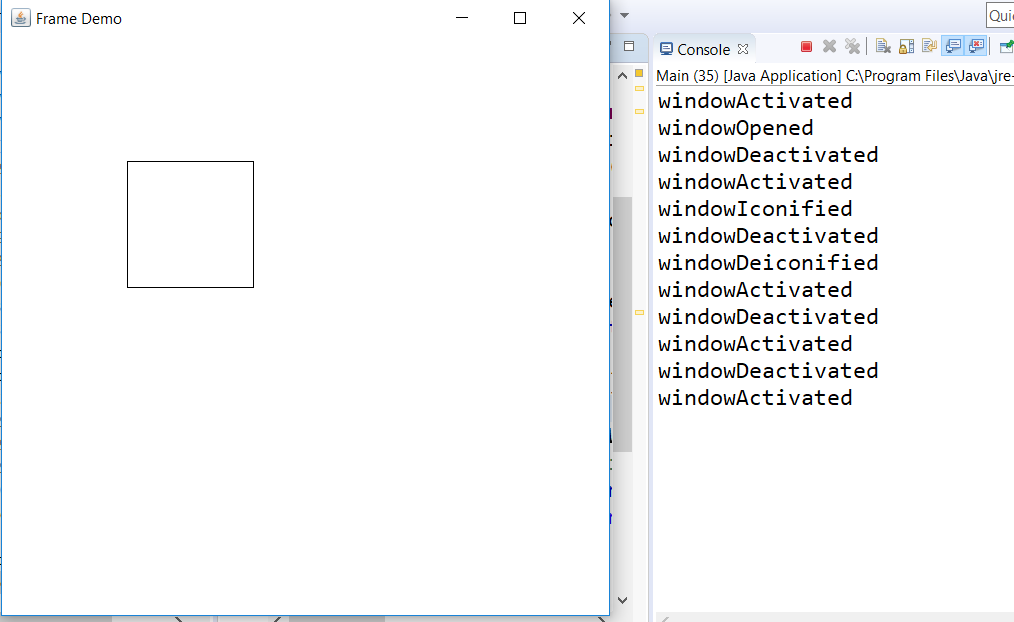
public static void main(String[] args) {

new MyFrame();

}

}

**Output**

****

**Adapter Class**

* Adapter classes are used handle the events like listener interface.
* It provides empty implementations for same interface that representing.
* When developer needs implementation required only for certain methods of some event interface they use equaling adapter classes

***Eg: Program that uses WindowAdapter Class***

* WindowListner interface has an alternative of WindowAdapter

import java.awt.Canvas;

import java.awt.Frame;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.awt.event.WindowListener;

import java.awt.geom.Rectangle2D;

class MyCanvas extends Canvas{

public void paint(Graphics g) {

Graphics2D g2d = (Graphics2D)g;

Rectangle2D.Float r2d = new Rectangle2D.Float(100, 100, 100.67F, 100.657F);

g2d.draw(r2d);

}

}

class MyWindowAdapter extends WindowAdapter{

public void windowClosing(WindowEvent e) {

System.out.println("windowClosing");

//Method the close the awt window

e.getWindow().dispose();

}

}

class MyFrame extends Frame{

MyWindowAdapter myadapter;

public MyFrame() {

add(new MyCanvas());

setSize(500, 500);

myadapter = new MyWindowAdapter();

// Adding Window Listener

addWindowListener(myadapter);

setTitle("Frame Demo");

setVisible(true);

}

}

public class Main {

public static void main(String[] args) {

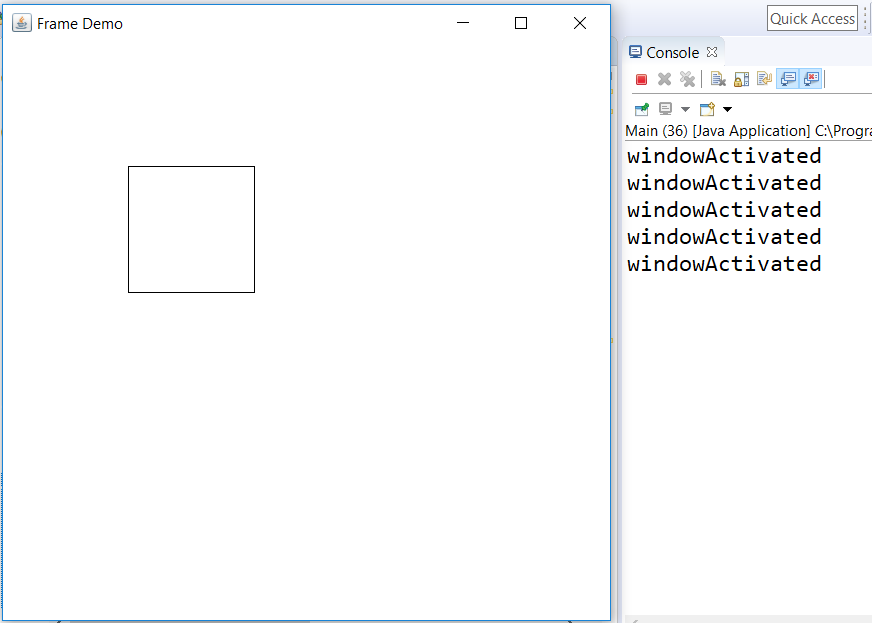
new MyFrame();

}

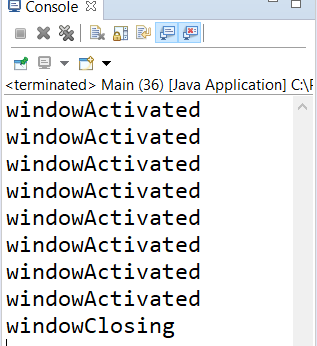
}

**Ouput:**

***Window Activated***



***Window Closing***



**Mouse Event**

Mouse contains following mouse life cycle states

* Mouse Entered
* Mouse Exited
* Mouse Pressed
* Mouse Released
* Mouse Clicked

Program that responds for mouse events

import java.awt.Canvas;

import java.awt.Color;

import java.awt.Frame;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.event.WindowEvent;

import java.awt.event.WindowListener;

import java.awt.geom.Rectangle2D;

class MyCanvas extends Canvas{

public void paint(Graphics g) {

Graphics2D g2d = (Graphics2D)g;

Rectangle2D.Float r2d = new Rectangle2D.Float(100, 100, 100.67F, 100.657F);

g2d.draw(r2d);

}

}

class MyFrame extends Frame implements MouseListener{

public MyFrame() {

//add(new MyCanvas());

setSize(500, 500);

// Adding Mouse Listener

addMouseListener(this);

setTitle("Mouse Listener Demo");

setVisible(true);

}

@Override

public void mouseClicked(MouseEvent e) {

System.out.println("mouseClicked");

}

@Override

public void mouseEntered(MouseEvent e) {

System.out.println("mouseEntered");

}

@Override

public void mouseExited(MouseEvent e) {

System.out.println("mouseExited");

}

@Override

public void mousePressed(MouseEvent e) {

System.out.println("mousePressed");

}

@Override

public void mouseReleased(MouseEvent e) {

System.out.println("mouseReleased");

}

}

public class Main {

public static void main(String[] args) {

new MyFrame();

}

}