The MouseListener interface is used to track the preceding events of the mouse on the area occupied by the graphical component.

Methods of MouseListener

- public void mousePressed (MouseEvent e)

  This method is called when a mouse button is pressed within the component to which we are listening.

- public void mouseReleased (MouseEvent e)
  This is called when a mouse button is released within the component to which we are listening.
- public void mouseClicked (MouseEvent e)
  This is called when the mouse is clicked (a press followed quickly by a release) within the component to which we are listening.
- public void mouseEntered (MouseEvent e)
  This is called when the mouse cursor enters the bounds of the component to which we are listening.
- public void mouseExited (MouseEvent e)
  This is called when the mouse cursor exits the bounds of the component to which we are listening.

Methods of MouseMotionListener

- public void mouseMoved (MouseEvent e)
  This is called when a mouse is moved, without a button being pressed, over the component to which we are listening.
- public void mouseDragged (MouseEvent e)
  This is called when a mouse is moved, with a button held down, over the component to which we are listening. If a mouse is dragged over more than one component, the events produced by dragging the mouse are delivered to the component where the dragging originated.

  The MouseEvent object e that appears as a parameter of each of these methods is supplied automatically by Java. It can be used to get a great deal of useful information about the event.

- public int getX ()
  This method returns the horizontal coordinate of the mouse when the event occurred.
- public int getY ()
  This method returns the vertical coordinate of the mouse when the event occurred.

import java.awt.*;

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

public class MouseCircles implements MouseListener
{
  int x = -1;
  int y;
  static final int RADIUS = 10;
  Drawing draw = new Drawing();

  public MouseCircles()
  {
    JFrame frame = new JFrame("Mouse Droppings");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    draw.addMouseListener(this);
    frame.add(draw);
    frame.setSize(300,100);
    frame.setVisible(true);
  }

  public void mouseClicked (MouseEvent e)
  {
    x = e.getX();
    y = e.getY();
    draw.repaint();
  }

  public void mousePressed (MouseEvent e)
  {
  }

  public void mouseReleased (MouseEvent e)
  {
  }

  public void mouseEntered (MouseEvent e)
  {
  }

  public void mouseExited (MouseEvent e)
  {
  }

  class Drawing extends JComponent
  {
    public void paint(Graphics g)
```

```
    {
      if (x != -1)
        g.fillOval(x - RADIUS, y - RADIUS, 2 * RADIUS, 2 * RADIUS);
    }
  }

  public static void main(String[] args)
  {
    new MouseCircles();
  }
}
```

Using Adapter class

Exercise 13.5

1. Explain the difference between the terms.
    a. mousePressed and mouseClicked
    b. MouseListener and MouseMotionListener
    c. MouseListener and MouseAdapter
2. Although the designers of Java created adapter classes for MouseListener and MouseMotionListener, they did not do so for ActionListener. Do you think that this was an oversight, sheer laziness, or something else? Justify your answer.
3. Write a program that first allows the user to press the mouse at one point on the screen, move the mouse (keeping the button pressed) to another point, and then release the mouse. Once the user has done this, the program should draw the line segment that connects the two points.
4. Write a program that allows the user to click on two points and then draws the circle that has its centre at the first point and passes through the second point.

Exercise 13.4

1. Write a program that uses a flow layout to display a window containing two buttons, one labelled "On" and the other labelled "Off". If a user presses "On", the background colour should be set to white but if a user presses "Off", the background should be set to black.
2. Modify the program of the previous question so that the labels on the buttons are "Brighter" and "Dimmer". If a user presses "Brighter", the background colour should be set closer to white (if it is not already white) while pressing "Dimmer" moves the

background closer to black (if it is not already black). Have each press of a button change the intensity by one-sixteenth of the difference between pure white and pure black.

3. Modify the code in example 3 to add two buttons labelled "Larger" and "Smaller". Clicking "Larger" will add 5 pixels to the size of the shape that is showing (it will affect any shape shown after as well). Clicking "Smaller" will subtract 5 pixels from the size.

Exercise 13.3

1. Suppose that a grid layout is to be used to display fourteen buttons. How many rows and columns would be displayed in each case if the call to the grid layout constructor had the form shown?
    a. new GridLayout(3,5)
    b. new GridLayout(5,4)
    c. new GridLayout(4,0)
    d. new GridLayout(0,6)
2. Write a program that will produce the following image in a window that is 140 pixels wide and 70 pixels high.

3. Write a program that will produce the following display in a square window whose sides are 200 pixels in length.

4. Write a program that will display two solid red circles in a window. The diameter of each circle should be one quarter of the smaller of the width and height of the region. One circle should be centred in the upper left quadrant while the other should be centred in the lower right quadrant. Resizing the window should cause the circles to resize appropriately.

Exercise 13.2

1. If a region is 300 pixels wide and 200 pixels high, determine the coordinates of each point.

    (a) the upper left corner

    (b) the midpoint of the bottom

    (c) the centre of the left half

    (d) the centre of the lower right quadrant

2. Write a paint method that could be used to draw a square each of whose sides are 100 pixels long. The square should be positioned so that it would be at the centre of a window that is 400 pixels wide and 200 pixels high. The top and bottom lines defining the square should be green while the sides should be red.
3. Write a paint method that could be used to draw a standard, octagonal stop sign that is 200 pixels wide.
4. Write a paint method that could be used to draw the following pattern of squares. The centre of the smallest square has coordinates(200,100) and the length of each of its sides is 20 pixels.

5. Write a complete program containing a paint method that draws a bull's eye with a red centre circle, an orange ring around that, a yellow ring around that, and a green ring around that. The radius of the centre circle and the width of each of the surrounding rings should be 10 pixels. The bull's eye should be located in the centre of a square region that is 100 pixels wide and 100 pixels high.
6. Write a program that will produce a window that can be used as a simple calculator. The window should have two input fields and one output field plus buttons for each of the operations +, -, *, and /. The illustration shows the window that would be produced by calculating 7 × 5.