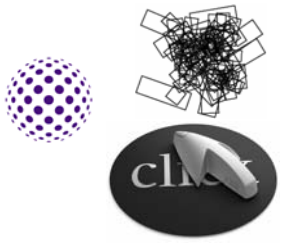
Lecture 25



Points

Rectangles

Mouse Events

Section 16.8 Points and Rectangles	*Section 17.3 – 17.5* Repainting, Mouse Events

Introduction

Strings

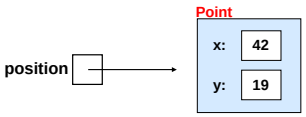
JButtons, JTextFields

Points, Rectangles

The Point class

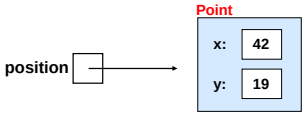
- For storing an **x** and a **y** position in a single object

```
int xPos = 42;
int yPos = 19;

Point position;

position = new Point(xPos, yPos);
```



Using Point objects

- The instance variables, **x** and **y**, are public and can be referred to directly:

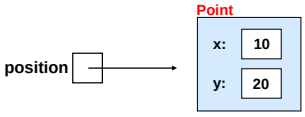



```
System.out.println( position.x );
System.out.println( position.y );
```

```
42
19
```

Using Point objects

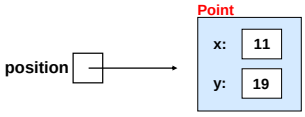
- The instance variables, **x** and **y**, are public and can be referred to directly:




```
position.x = 10;
position.y = 20;
```

Using Point objects

We can also call methods on Point objects:

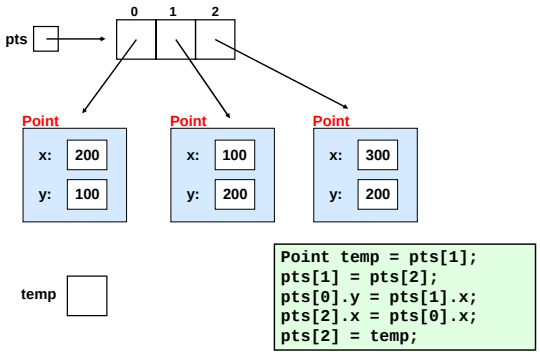



```
position.move(10, 20);
position.translate(1, -1);
```

Point object exercise

```
Point[] pts = new Point[3];

pts[0] = new Point(200, 100);
pts[1] = new Point(100, 200);
pts[2] = new Point(300, 200);

Point temp = pts[1];
pts[1] = pts[2];
pts[0].y = pts[1].x;
pts[2].x = pts[0].x;
pts[2] = temp;

System.out.println(pts[0].x + " , " + pts[0].y);
System.out.println(pts[1].x + " , " + pts[1].y);
System.out.println(pts[2].x + " , " + pts[2].y);
```

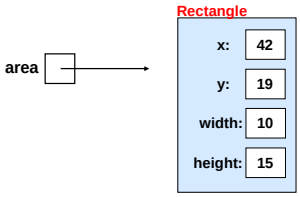
Point object exercise




```
Point temp = pts[1];
pts[1] = pts[2];
pts[0].y = pts[1].x;
pts[2].x = pts[0].x;
pts[2] = temp;
```

The Rectangle class

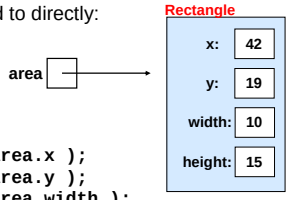
- For storing **x**, **y**, **width** and **height** values in a single object




```
int xPos = 42;
int yPos = 19;
int width = 10;
int height = 15;

Rectangle area;
area = new Rectangle(xPos, yPos, width, height);
```

Using Rectangle objects

- The instance variables, **x**, **y**, **width** and **height** are public and can be referred to directly:

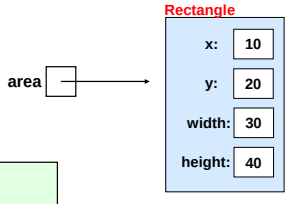



```
System.out.println( area.x );
System.out.println( area.y );
System.out.println( area.width );
System.out.println( area.height );
```

```
42
19
10
15
```

Using Rectangle objects

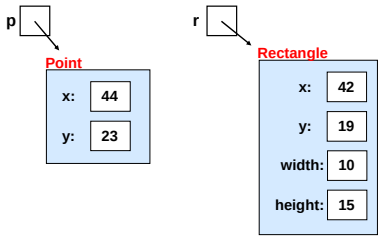
- The instance variables, **x**, **y**, **width** and **height** are public and can be referred to directly:




```
area.x = 10;
area.y = 20;
area.width = 30;
area.height = 40;
```

Using Rectangle objects

We can also call methods on Rectangle objects:

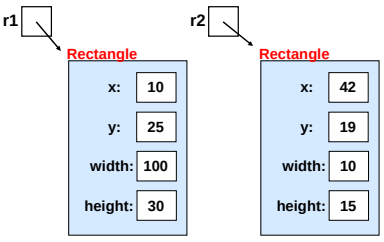



```
r.contains(p);
```

true

Using Rectangle objects

We can also call methods on Rectangle objects:




```
r1.intersects(r2);
```

true

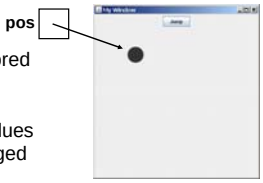
Repainting the screen

To redraw the screen when a certain event occurs we call:

```
repaint();
```

Consider this program:

- the position of the circle is stored in a **Point**, called **pos**.
- the circle moves when the values of **pos.x** and **pos.y** are changed and then **repaint()** is called



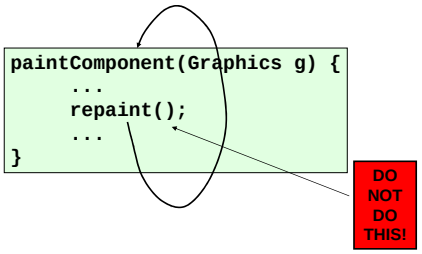
Method responsibilities

Each method in this program has well-defined responsibilities:

- The **constructor** method initialises the state of the program
  i.e. gives initial values to all of the instance variables
- The **actionPerformed()** method updates the state
  i.e. modifies the current values of the instance variables
- The **paintComponent()** method displays the current state
  i.e. uses the current values of the instance variables to draw the circle on the window

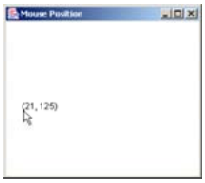
Warning

You should be careful never to call **`repaint()`** inside the **`paintComponent()`** method



Mouse Events

MouseEvents are generated when the mouse button is pressed or the mouse is moved.



mouseX 21

mouseY 125

```
public class MyJPanel extends JPanel {

	private int mouseX;
	private int mouseY;

	public MyJPanel() {
		mouseX = 100;
		mouseY = 100;

	}




	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		g.drawString("(" + mouseX + ", " + mouseY + ")",
                                                    mouseX, mouseY);
	}



}
```

```
public class MyJPanel extends JPanel implements MouseListener {
        private int mouseX;
        private int mouseY;
        public MyJPanel() {
                mouseX = 100;
                mouseY = 100;
        }
        public void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.drawString("(" + mouseX + ", " + mouseY + ")",
                                                        mouseX, mouseY);
        }
}
```

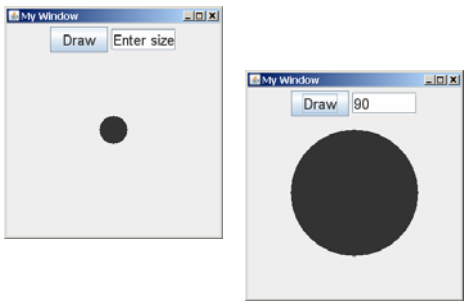
```
public class MyJPanel extends JPanel implements MouseListener {
        private int mouseX;
        private int mouseY;
        public MyJPanel() {
                mouseX = 100;
                mouseY = 100;
        }
        public void mousePressed(MouseEvent e) {
        }
        public void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.drawString("(" + mouseX + ", " + mouseY + ")",
                                                        mouseX, mouseY);
        }
        public void mouseReleased(MouseEvent e) {}
        public void mouseClicked(MouseEvent e) {}
        public void mouseEntered(MouseEvent e) {}
        public void mouseExited(MouseEvent e) {}
}
```

```
public class MyJPanel extends JPanel implements MouseListener {
        private int mouseX;
        private int mouseY;
        public MyJPanel() {
                mouseX = 100;
                mouseY = 100;
                addMouseListener(this);
        }
        public void mousePressed(MouseEvent e) {
        }
        public void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.drawString("(" + mouseX + ", " + mouseY + ")",
                                                        mouseX, mouseY);
        }
        public void mouseReleased(MouseEvent e) {}
        public void mouseClicked(MouseEvent e) {}
        public void mouseEntered(MouseEvent e) {}
        public void mouseExited(MouseEvent e) {}
}
```

```
public class MyJPanel extends JPanel implements MouseListener {
        private int mouseX;
        private int mouseY;
        public MyJPanel() {
                mouseX = 100;
                mouseY = 100;
                addMouseListener(this);
        }
        public void mousePressed(MouseEvent e) {
                mouseX = e.getX();
                mouseY = e.getY();
                repaint();
        }
        public void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.drawString("(" + mouseX + ", " + mouseY + ")",
                                                        mouseX, mouseY);
        }
        public void mouseReleased(MouseEvent e) {}
        public void mouseClicked(MouseEvent e) {}
        public void mouseEntered(MouseEvent e) {}
        public void mouseExited(MouseEvent e) {}
}
```

```
public class MyJPanel extends JPanel implements MouseListener {
        private int mouseX;
        private int mouseY;
        public MyJPanel() {
                mouseX = 100;
                mouseY = 100;
                addMouseListener(this);
        }
        public void mousePressed(MouseEvent e) {
                mouseX = e.getX();
                mouseY = e.getY();
                repaint();
        }
        public void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.drawString("(" + mouseX + ", " + mouseY + ")",
                                                        mouseX, mouseY);
        }
        public void mouseReleased(MouseEvent e) {}
        public void mouseClicked(MouseEvent e) {}
        public void mouseEntered(MouseEvent e) {}
        public void mouseExited(MouseEvent e) {}
}
```

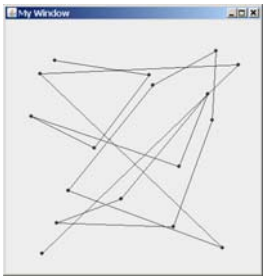
Exercise



```
public class MyJPanel extends JPanel implements ActionListener {

        private JButton drawButton;
        private JTextField sizeInput;
        private int size;

        public MyJPanel() {

                size = 20;

                drawButton = new JButton("Draw");
                drawButton.addActionListener(this);
                add(drawButton);

                sizeInput = new JTextField("Enter size");
                add(sizeInput);
        }

        public void actionPerformed(ActionEvent e) {
                size = Integer.parseInt(sizeInput.getText());
                repaint();
        }

        public void paintComponent(Graphics g) {
                super.paintComponent(g);

                g.fillOval(150-size, 150-size, 2*size, 2*size);
        }
}
```

Example: connect the dots

What variables will we need to store the data for this program?



points

an array of Point objects to store each point

count

to keep a count of how many Point objects are currently being stored in the array

```
public class MyJPanel extends JPanel implements MouseListener {

      private Point[] points;
      private int count;

      public MyJPanel() {

            addKeyListener(this);
      }

      public void mousePressed(MouseEvent e) {

      }

      public void paintComponent(Graphics g){
            super.paintComponent(g);

      }

      public void keyReleased(KeyEvent e) {}
      public void keyTyped(KeyEvent e) {}
}
```

Initialise the instance variables

Update the array by adding a new point

Draw the points

```
public MyJPanel() {

      points = new Point[1000];
      count = 0;

      addMouseListener(this);
}
```

```
public void mousePressed(MouseEvent e) {

      points[count] = new Point(e.getX(), e.getY());
      count++;

      repaint();
}
```

```
  public void paintComponent(Graphics g){
      super.paintComponent(g);

      for (int i = 0 ; i < count; i++) {
            g.fillOval(points[i].x-3, points[i].y-3, 6, 6);
      }

      for (int i = 1 ; i < count; i++) {
            g.drawLine(points[i].x, points[i].y,
                        points[i-1].x, points[i-1].y);
      }
}
```