Lecture 23

CompSci 101

Components

Section 16.5 – 16.7
Components

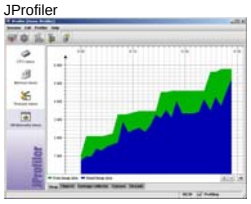
Java Desktop Applications

LimeWire

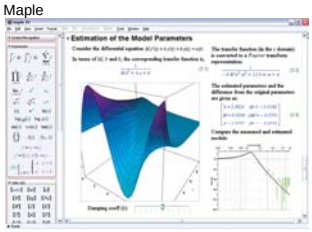
Azureus

Java Desktop Applications

JProfiler



Maple



jlGui

See hundreds more examples at the Swing Connection:

http://java.sun.com/products/jfc/tsc/sightings/

J2ME Applications

"More than 1.5 billion Java enabled phones in the world"
www.sun.com

Gmail mobile client

Mapping application on a BlackBerry

Another drawing example

Consider the following picture of a village:

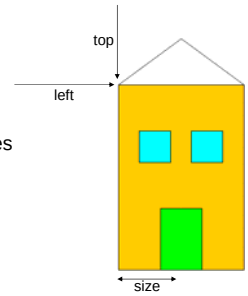
Each house is of the same design, but has a different size and position.

How would you write a program to do this?

House

A house can be defined by:
- its *size*
- its *position*

What *size* actually represents is arbitrary, but it can be used to calculate the correct relative sizes of all parts of the house.



width of house = size * 2

height of house = size * 3

size of window = size / 2

A House class

```
public class House {

	private int size;
	private int left;
	private int top;

	public House(int size, int left, int top) {
		this.size = size;
		this.left = left;
		this.top = top;
	}

	....
}
```

```
public void draw(Graphics g) {

    int houseWidth = size * 2;
    int houseHeight = houseWidth * 3/2;
    int houseRight = left + houseWidth;
    int houseBottom = top + houseHeight;

    int doorWidth = houseWidth/3;
    int doorHeight = houseHeight/3;
    int doorLeft = left + (houseWidth-doorWidth)/2;
    int doorTop = houseBottom - doorHeight;

    int windowSize = houseWidth/4;
    int windowLeft = left + (houseWidth-windowSize*2)/3;
    int windowRight = windowLeft + (windowSize*2)/3+windowSize;
    int windowTop = top + houseHeight/4;

    int roofPeakX = left + houseWidth/2;
    int roofPeakY = top - houseHeight/4;

    ....
```

```
    ....

    // Fill in the main structure of the house
    g.setColor( Color.orange );
    g.fillRect( left, top, houseWidth, houseHeight );

    // Paint the door
    g.setColor( Color.green );
    g.fillRect( doorLeft, doorTop, doorWidth, doorHeight );

    // Paint the windows
    g.setColor( Color.cyan );
    g.fillRect( windowLeft, windowTop, windowSize, windowSize );
    g.fillRect( windowRight, windowTop, windowSize, windowSize );

    // Draw all the borders
    g.setColor( Color.black );
    g.drawRect( left, top, houseWidth, houseHeight );
    g.drawRect( doorLeft, doorTop, doorWidth, doorHeight );
    g.drawRect( windowLeft, windowTop, windowSize, windowSize );
    g.drawRect( windowRight, windowTop, windowSize, windowSize );
    g.drawLine( left, top, roofPeakX, roofPeakY );
    g.drawLine( roofPeakX, roofPeakY, houseRight, top );
}
```

Drawing several houses

We can create and draw two houses:

```
public class MyJPanel extends JPanel {

	private House h1, h2;

	public MyJPanel() {
		h1 = new House(40, 50, 80);
		h2 = new House(75, 200, 120);
	}

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

		h1.draw(g);
		h2.draw(g);
	}
}
```

Drawing a village

To draw the village as before, we can declare an array of House objects:

```
public class MyJPanel extends JPanel {

	House[] houses;

	public MyJPanel() {

	}

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

	}
}
```

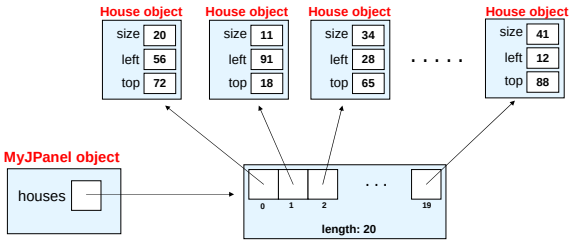
Drawing a village

We would construct the array and the House objects in the constructor method of the JPanel class:

```
public MyJPanel() {

	houses = new House[20];

	for (int i = 0; i < houses.length; i++) {
		int size = 5 + i*2;
		int left = (int)(Math.random()*300);
		int down = 10 + i*7;
		houses[i] = new House(size, left, down);
	}
}
```

Drawing a village

We can visualise the array of House objects as follows:



Drawing a village

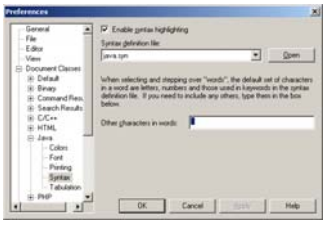
And then draw the objects in the paintComponent() method:

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

       for (int i = 0; i < houses.length; i++) {
              houses[i].draw(g);
       }
}
```

Components

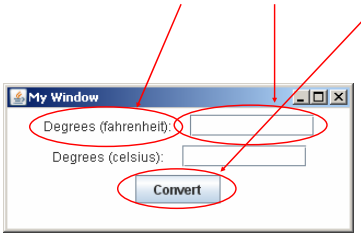
- Most GUIs contain components such as buttons and textfields with which users interact

For example, consider the Preferences dialog box in TextPad:

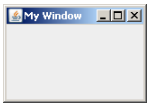


Components

- This window contains a label, a textfield and a button:

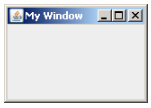


JButtons



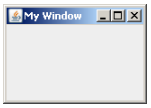

```
public class MyJPanel extends JPanel {


       public MyJPanel() {



       }

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

       }
}
```

JButtons




```
public class MyJPanel extends JPanel {

       private JButton clickMe;

       public MyJPanel() {



       }

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

       }
}
```

JButtons

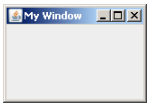



```
public class MyJPanel extends JPanel {

	private JButton clickMe;

	public MyJPanel() {

		clickMe = new JButton("Click me!");


	}

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

	}
}
```

JButtons

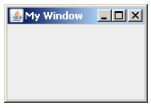

```
public class MyJPanel extends JPanel {

	private JButton clickMe;

	public MyJPanel() {

		clickMe = new JButton("Click me!");

		add(clickMe);
	}

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

	}
}
```

JTextFields



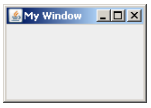

```
public class MyJPanel extends JPanel {


	public MyJPanel() {



	}

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

	}
}
```

JTextFields



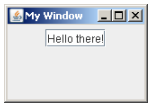

```
public class MyJPanel extends JPanel {

	private JTextField words;

	public MyJPanel() {



	}

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

	}
}
```

JTextFields



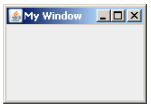

```
public class MyJPanel extends JPanel {

	private JTextField words;

	public MyJPanel() {

		words = new JTextField("Hello there!");


	}

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

	}
}
```

JTextFields



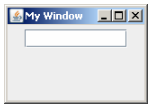

```
public class MyJPanel extends JPanel {

	private JTextField words;

	public MyJPanel() {

		words = new JTextField("Hello there!");

		add(words);
	}

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

	}
}
```

JTextFields



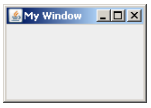

```
public class MyJPanel extends JPanel {

	private JTextField words;

	public MyJPanel() {

		words = new JTextField(10);


	}

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

	}
}
```

JTextFields



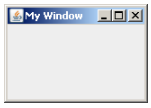

```
public class MyJPanel extends JPanel {

	private JTextField words;

	public MyJPanel() {

		words = new JTextField(10);

		add(words);
	}

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

	}
}
```

JLabels




```
public class MyJPanel extends JPanel {

	private JLabel promptUser;

	public MyJPanel() {



	}

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

	}
}
```

JLabels




```
public class MyJPanel extends JPanel {

	private JLabel promptUser;

	public MyJPanel() {

		promptUser = new JLabel("Enter value:");


	}

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

	}
}
```

JLabels


```
public class MyJPanel extends JPanel {

	private JLabel promptUser;

	public MyJPanel() {

		promptUser = new JLabel("Enter value:");

		add(promptUser);
	}

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

	}
}
```

Multiple components

```
private JButton aButton, anotherButton;
private JTextField aTextField;

public MyJPanel() {

	aButton = new JButton("Click here");
	aTextField = new JTextField("Great - some text!");
	anotherButton = new JButton("No, click here!");

	add(aButton);
	add(aTextField);
	add(anotherButton);
}
```

Layout

- The default layout of components in the window is called a *FlowLayout*
- Components are positioned from left to right, top to bottom, in the order they were added to the window
- If the window is resized, the components will be automatically repositioned

Null Layout

- We can choose not to use an automatic layout, by making the method call:

```
setLayout(null);
```

- In this case, we have to explicitly set the positions of each component on the screen by calling the setBounds() method on each component:

```
setBounds(int x, int y, int width, int height);
```

Null Layout

```
public class MyJPanel extends JPanel {

    private JButton button;

    public MyJPanel() {

        setLayout(null);

        button = new JButton("Click me!");
        button.setBounds(200, 10, 100, 50);
        add(button);
    }

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

    }
}
```

Background colour

- By default, the background colour of our window is light gray
- We can change this background colour by calling the following method in the constructor of the JPanel class:

```
setBackground(Color color)
```

Background colour

```
public MyJPanel() {

    setBackground(Color.white);

    ...
}
```

```
public MyJPanel() {

    setBackground(Color.yellow);

    ...
}
```

getText() and setText()

- Given a JTextField object called **`input`**, we can call the following two methods on this object:

```
String userInput = input.getText();

input.setText("The answer is 42");
```

- We need to learn about event handling to see how this can be useful