

## Week 9 Displaying Graphics & Handling Events



1

## This Week

Lecture: Displaying graphics &  
Handling events

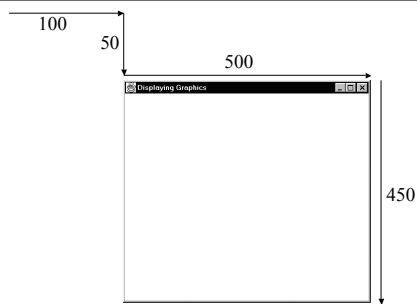
*Java Genesis:*

–Ch9: Graphics and event handling

Lab Assessment 7

Quick Quiz for Chapter 8

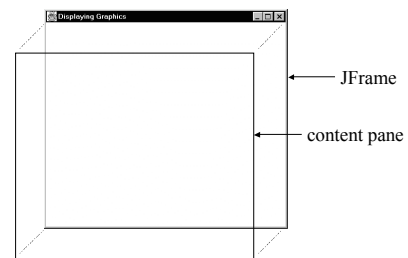
2



`setBounds(100, 50, 500, 450)`

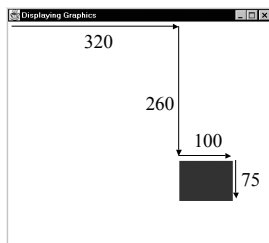
3

## Structure of JFrame objects (abbreviated)



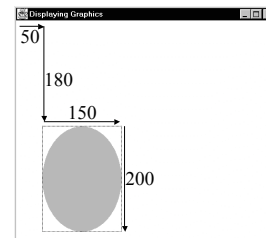
Graphical objects are painted on a **JPanel** object and this panel is then added to the frame's content pane.

4



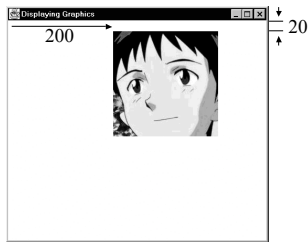
`fillRect(320, 260, 100, 75)`

5



`fillOval(50, 180, 150, 200)`

6



`drawImage(im, 200, 20, this)`

7

```
import genesis.*;
public class Main {

    public static void main (String [ ] args) {
        OurFrame frame = new OurFrame();
        frame.setVisible(true);
        while (true) {
            Delay.milliseconds(20);
            frame.moveGraphic();
        }
    }
}
```

8

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

public class OurFrame extends JFrame {

    // instance variable
    private OurPanel panel = new OurPanel();

    public OurFrame ( ) { // constructor
        setTitle("Displaying Graphics");
        setBounds(100, 50, 500, 450);
        Container c = getContentPane();
        c.add(panel);
    }
    public void moveGraphic ( ) {
        panel.moveGraphic();
    }
}
```

9

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

public class OurPanel extends JPanel {

    //instance variables
    private int xCorner = 200, yCorner = 200;
    private int step = 5;

    // constructor
    public OurPanel ( ) {
        setBackground(Color.white);
    }
}
```

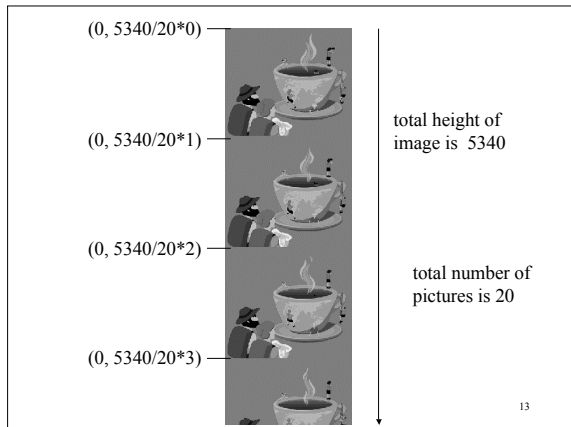
10

```
// paint graphical objects
public void paintComponent (Graphics g) {
    super.paintComponent(g);
    g.setColor(Color.red);
    g.fillRect(320, 260, 100, 75);
    g.setColor(Color.green);
    g.fillOval(50, 180, 150, 200);
    Toolkit tk = Toolkit.getDefaultToolkit();
    Image im = tk.getImage("Evangelion.gif");
    g.drawImage(im, 200, 20, this);
    g.setColor(Color.magenta);
    g.fillOval(xCorner, yCorner, 100, 100);
}
```

11

```
public void moveGraphic ( ) {
    if (xCorner > 390 || xCorner < 0) {
        step = -step;
    }
    xCorner = xCorner + step;
    repaint();
}
```

12



```
import genesis.*;
public class Main {

    public static void main (String [ ] args)
    {
        ImageFrame imageWin = new ImageFrame( );
        imageWin.setVisible(true);
        while (true) {
            Delay.milliseconds(200);
            imageWin.nextPicture( );
        }
    }
}
```

14

```
import javax.swing.*;
import java.awt.*;
public class ImageFrame extends JFrame {

    // instance variables
    private ImagePanel panel = new ImagePanel( );

    // constructor
    public ImageFrame ( ) {
        setTitle("Image Animation");
        setBounds(100, 50, 310, 295);
        Container c = getContentPane( );
        c.add(panel);
    }

    public void nextPicture ( ) {
        panel.nextPicture( );
    }
}
```

15

```
import javax.swing.*;
import java.awt.*;
public class ImagePanel extends JPanel {

    // instance variables
    private Image im;
    private int imageCount = 20;
    private int imageHeight = 5340;
    private int current = 0;

    // constructor
    public ImagePanel ( ) {
        Toolkit tk = Toolkit.getDefaultToolkit( );
        im = tk.getImage("Javabig.gif");
    }
}
```

16

```
public void paintComponent (Graphics g) {
    super.paintComponent(g);
    int offset =
        -(imageHeight/imageCount)*current;
    g.drawImage(im, 0, offset, null);
}

public void nextPicture ( ) {
    current = (current+1)%imageCount;
    repaint();
}
}
```

17

## Event Handling

To handle (i.e. detect and respond) to events such as

- pressing the mouse button
- moving the mouse
- pressing a key on the keyboard
- etc.....

we need to add **event listeners** to our window.

18

## Event Listeners

When an event listener is added to a window it will

- listen out for specific events
- respond in its own specific way when an event it is listening for occurs

19

To specify the way an event listener responds when an event it is listening for occurs, we use Java's **anonymous inner class** construct.

20

```
public class NumberToy {  
    // instance variables  
    public int number;  
  
    // constructor  
    public NumberToy (int num) {  
        number = num%10;  
        if (number == 0) number = 10;  
    }  
  
    public void pressButton ( ) {  
        number = (number + 1)%10;  
        if (number == 0) number = 10;  
    }  
}
```

21

```
----(code not shown)-----  
  
public class NumberToyGUI extends JFrame{  
    // instance variables  
    public NumberToy toy = new NumberToy(1);  
  
    ----(code not shown)-----
```

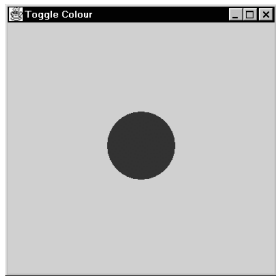
22

```
public class AnotherNumberToy extends NumberToy {  
    // constructor  
    public AnotherNumberToy (int num) {  
        super(num);  
    }  
  
    public void pressButton ( ) {  
        number = (number - 1)%10;  
        if (number == 0) number = 10;  
    }  
}
```

23

```
----(code not shown)-----  
  
public class NumberToyGUI extends JFrame{  
    // instance variables  
    public NumberToy toy = new NumberToy(1){  
        public void pressButton ( ) {  
            number = (number - 1)%10;  
            if (number == 0) number = 10;  
        }  
    };  
  
    ----(code not shown)-----
```

24



25

```
public class Main {
    public static void main (String [ ] args) {
        TogColourFrame togWin =
            new TogColourFrame( );
        togWin.setVisible(true);
    }
}
```

26

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

public class TogColourFrame extends JFrame {
    // constructor
    public TogColourFrame ( ) {
        setTitle("Toggle Colour");
        setBounds(400, 100, 400, 400);
        TogColourPanel panel =
            new TogColourPanel( );
        Container c = getContentPane( );
        c.add(panel);
    }
}
```

27

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

public class TogColourPanel extends JPanel {
    // instance variables
    private Color colour = Color.red;
    private int xPosn = 196, yPosn = 180;

    public void paintComponent (Graphics g) {
        super.paintComponent(g);
        g.setColor(colour);
        g.fillOval(xPosn-50, yPosn-50, 100, 100);
    }
}
```

28

### The class MouseAdapter

```
public abstract class MouseAdapter {

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

29

### The class MouseMotionAdapter

```
public abstract class MouseMotionAdapter {

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

30

```

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

public class TogColourPanel extends JPanel {

    // instance variable
    private Color colour = Color.red;
    private int xPosn = 196, yPosn = 180;

```

31

```

// constructor
public TogColourPanel ( ) {
    addMouseListener(new MouseAdapter ( ) {
        public void mousePressed (MouseEvent e) {
            if (colour == Color.red)
                colour = Color.blue;
            else colour = Color.red;
            repaint();
        }
        public void mouseEntered(MouseEvent e) {
            colour = Color.yellow;
            repaint();
        }
        public void mouseExited(MouseEvent e) {
            colour = Color.magenta;
            repaint();
        }
    });
}

```

32

```

        addMouseMotionListener
            (new MouseMotionAdapter ( ) {
                public void mouseMoved(MouseEvent e) {
                    xPosn = e.getX();
                    yPosn = e.getY();
                    repaint();
                }
            });
    }

    public void paintComponent (Graphics g) {
        super.paintComponent(g);
        g.setColor(colour);
        g.fillOval(xPosn-50, yPosn-50, 100, 100);
    }
}

```

33