

## Chap 9 - GUIs Part II

- Arranging components with JPanel.
- Resizing.
- More about Graphics.
- Graphics2D
- Menus

Note that the import statements have been excluded from many of the slides in this chapter to save space on the slide.

1

---

---

---

---


---

---

---

---

## BorderLayout Layout Manager



2

---

---

---

---

---

---

---

---

```
class BorderLayoutTest {
    public static void main(String[] args) {
        JFrame frame = new JFrame("BorderLayout");
        Container pane = frame.getContentPane();
        pane.add(
            new JButton("North Button"),BorderLayout.NORTH);
        pane.add(
            new JButton("South Button"),BorderLayout.SOUTH);
        pane.add(
            new JButton("East Button"), BorderLayout.EAST);
        pane.add(
            new JButton("West Button"), BorderLayout.WEST);
        pane.add(
            new JButton("Center Button"),BorderLayout.CENTER);
        frame.pack();
        frame.show();
    }
}
```

3

---

---

---

---

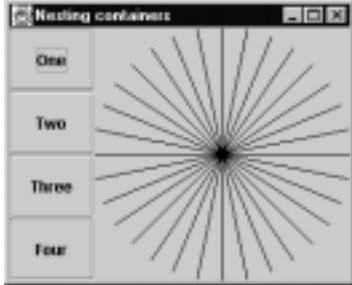
---

---

---

---

## Nesting Containers



4

---

---

---

---

---

---

---

---

```
class Nesting {  
    public static void main(String[] args) {  
        JFrame frame = new JFrame("Nesting containers");  
        Container pane = frame.getContentPane();  
        JPanel panel = new JPanel();  
        panel.setLayout(new GridLayout(4, 1));  
        panel.add(new JButton("One"));  
        panel.add(new JButton("Two"));  
        panel.add(new JButton("Three"));  
        panel.add(new JButton("Four"));  
        pane.add(panel, BorderLayout.WEST);  
        pane.add(new Star(), BorderLayout.CENTER);  
        frame.pack();  
        frame.show();  
    }  
}
```

5

---

---

---

---

---

---

---

---

## Resizing



The JButtons resized but the Star did not.

6

---

---

---

---

---

---

---

---

```

class Star2 extends JComponent {
    public void paint(Graphics g) {
        //same as in earlier Star class
    }
    public void setBounds(int x, int y,
                          int width, int height)
    {
        radius = Math.min(width, height) / 2;
        super.setBounds(x, y, width, height);
        repaint();
    }
    public Dimension getMinimumSize() {
        return new Dimension(2 * radius, 2 * radius);
    }
    public Dimension getPreferredSize() {
        return new Dimension(2 * radius, 2 * radius);
    }
    private int radius = 100;
}

```

7

---

---

---

---

---

---

---

---



8

---

---

---

---

---

---

---

---

### An Example: A Plot Component

- Create a component to plot any function of one variable.
- The component will be given a range of values or which to plot the function.
- The component will have a default size and will adjust the scaling of the plot to just fill the allotted screen space.
- When resized, the plot component will readjust its scaling so that the plot continues to just fill the allotted space.

9

---

---

---

---

---

---

---

---

## How do we specify the function?

- We can use an interface.

```
//Function.java
interface Function {
    public double valueAt(double x);
}
```

10

---

---

---

---

---

---

---

---

```
public class Plot extends JComponent {
    private Dimension dim;
    private double fmin, fmax;
    private double xScale, yScale;
    private double from, to, delta;
    private Function function;

    public Plot(Function f, double start, double end,
                double deltaX)
    {
        function = f;
        delta = deltaX;
        from = start;
        to = end;
        findRange(); // find max and min of f(x)
        setSize(200, 100); // default width, height
    }
}
```

11

---

---

---

---

---

---

---

---

```
public void paint(Graphics g)
{
    double x = from;
    double f_of_x = function.valueAt(x);
    while (x < to - delta) {
        double f_of_x_plus_delta;
        f_of_x_plus_delta = function.valueAt(x + delta);
        drawLine(g, x, f_of_x,
                x + delta, f_of_x_plus_delta);
        x = x + delta;
        f_of_x = f_of_x_plus_delta;
    }
    drawLine(g, x, f_of_x, to, function.valueAt(to));
}
```

12

---

---

---

---

---

---

---

---

```

public void setBounds(int x, int y,
                    int width, int height)
{
    xScale = (width - 2) / (to - from);
    yScale = (height - 2) / (fmax - fmin);
    dim = new Dimension(width, height);
    super.setBounds(x, y, width, height);
    repaint();
}
public Dimension getMinimumSize() {
    return dim;
}

public Dimension getPreferredSize() {
    return getMinimumSize();
}

```

13

---

---

---

---

---

---

---

---

```

// scale and translate the points from the
// user's coordinate space to Java's
private void drawLine(Graphics g,
                    double x1, double y1,
                    double x2, double y2)
{
    g.drawLine(
        (int)Math.round((x1 - from) * xScale),
        (int)Math.round((fmax - y1) * yScale),
        (int)Math.round((x2 - from) * xScale),
        (int)Math.round((fmax - y2) * yScale));
}

```

14

---

---

---

---

---

---

---

---

```

// determine the minimum and maximum values of
// f(x) with x varying between from and to
private void findRange() {
    fmin = fmax = function.valueAt(from);
    for (double x = from + delta;
        x < to - delta;
        x = x + delta)
    {
        double f_of_x = function.valueAt(x);
        if (f_of_x < fmin)
            fmin = f_of_x;
        if (f_of_x > fmax)
            fmax = f_of_x;
    }
}
}

```

15

---

---

---

---

---

---

---

---

```
//SinFunction.java
class SinFunction implements Function {
    public double valueAt(double x) {
        return Math.sin(x);
    }
}
```

16

---

---

---

---

---

---

---

---

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

class PlotTest {
    public static void main(String[] args) {
        Function f = new SinFunction();
        JFrame frame = new JFrame("PlotTest");
        Container pane = frame.getContentPane();
        Plot plot = new Plot(f, 0, 2 * Math.PI, 0.1);
        pane.add(plot);
        frame.pack();
        frame.show();
    }
}
```

17

---

---

---

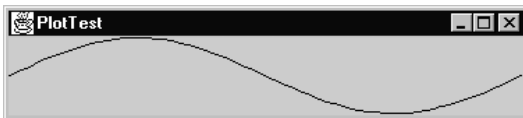
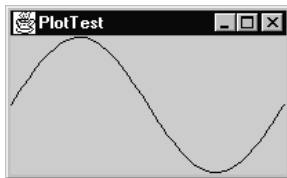
---

---

---

---

---



18

---

---

---

---

---

---

---

---

## java.awt.Graphics

- Handles basic drawing.
- Supports colors.
- Supports different text fonts.
- Draws lines, ovals, polygons, rectangles, and arcs, both filled and unfilled.

19

---

---

---

---

---

---

---

---

Minor variants of this class are used throughout section 9.4 to test components that draw different shapes.

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

class DrawingLines {
    public static void main(String[] args) {
        JFrame frame = new JFrame("DrawingLines");
        Container pane = frame.getContentPane();
        pane.add(new DrawTriangle());
        frame.pack();
        frame.show();
    }
}
```

20

---

---

---

---

---

---

---

---

```
// This class is miss named. It draws a triangle and a W.
class DrawTriangle extends JComponent {
```

```
    public void paint(Graphics g) {
        // draw a triangle
        g.drawLine(10, 10, 100, 60);
        g.drawLine(100, 60, 50, 80);
        g.drawLine(50, 80, 10, 10);
        // draw a W
        int[] x = {120, 140, 160, 180, 200};
        int[] y = {10, 80, 10, 80, 10};
        g.drawPolyline(x, y, 5);
    }
```

```
    public Dimension getMinimumSize() { return dim;}
    public Dimension getPreferredSize() {return dim;}
    private Dimension dim = new Dimension(220, 100);
}
```

21

---

---

---

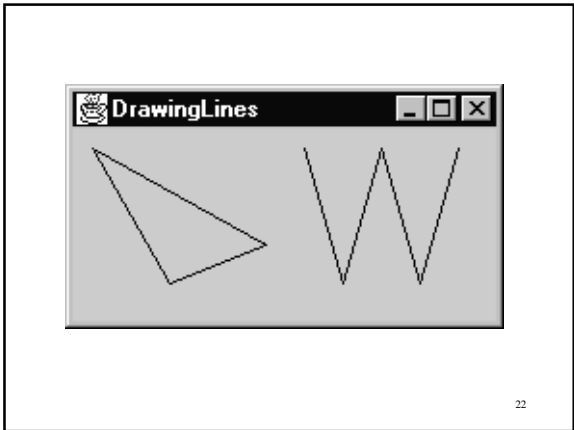
---

---

---

---

---



22

---

---

---

---

---

---

---

---

```

class DrawRectangles extends JComponent
{
    public void paint(Graphics g) {
        g.drawRect(0, 0, 100, 60);
        g.drawRoundRect(110, 0, 100, 60, 40, 20);
        g.setColor(Color.gray);
        g.drawOval(170, 40, 40, 20);
    }
    public Dimension getMinimumSize()
    {
        return dim;
    }
    public Dimension getPreferredSize()
    {
        return dim;
    }
    private Dimension dim = new Dimension(220,70);
}

```

23

---

---

---

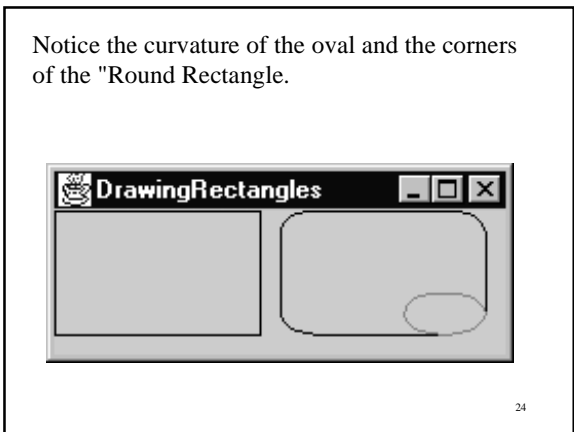
---

---

---

---

---



24

---

---

---

---

---

---

---

---



```

class DrawOvals extends JComponent
{
    public void paint(Graphics g) {
        g.drawRect(10, 10, 100, 60);
        g.drawOval(10, 10, 100, 60);
    }
    public Dimension getMinimumSize()
    { return dim;
    }
    public Dimension getPreferredSize()
    { return dim;
    }
    private Dimension dim = new Dimension(500,100);
}

```

25

---

---

---

---

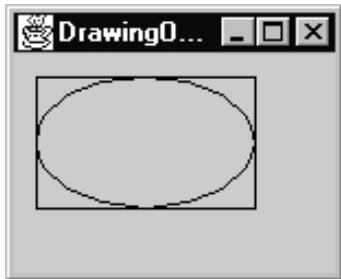
---

---

---

---

See how the oval fits inside of the rectangle, called the "bounding box".



26

---

---

---

---

---

---

---

---

```

class DrawArcs extends JComponent
{
    public void paint(Graphics g) {
        g.drawArc(0, 0, 100, 60, 0, 160);
        g.fillArc(110, 0, 60, 60, 45, -135);
    }
    public Dimension getMinimumSize()
    { return dim;
    }
    public Dimension getPreferredSize()
    { return dim;
    }
    private Dimension dim = new Dimension(500,70);
}

```

27

---

---

---

---

---

---

---

---

Arcs are just incomplete Ovals.



28

---

---

---

---

---

---

---

---

```
class DrawPolygons extends JComponent
{
    public void paint(Graphics g) {
        Polygon p = new Polygon();
        p.addPoint(10, 10);
        p.addPoint(60, 30);
        p.addPoint(40, 50);
        p.addPoint(25, 40);
        g.drawPolygon(p); // draw a Polygon object
        // draw a polygon using 2 arrays
        int[] x = {110, 160, 140, 125};
        int[] y = {10, 30, 50, 40};
        g.fillPolygon(x, y, 4);
        //move 200 pixels right and 20 down
        p.translate(200, 20);
        g.drawPolygon(p); // draw translated polygon
    }
}
```

29

---

---

---

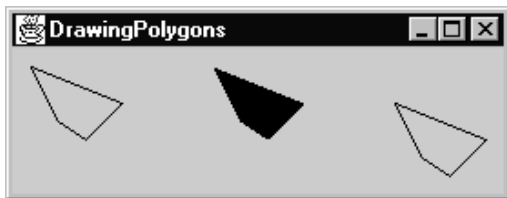
---

---

---

---

---



30

---

---

---

---

---

---

---

---

```

class DrawText extends JComponent
{
    public void paint(Graphics g) {
        g.drawString("Starts at (10, 20)", 10, 20);
        g.fillRect(10, 20, 4, 4);
        g.setFont(new Font("TimesRoman", Font.BOLD, 20));
        g.drawString("20pt bold starting at (100, 40)",
            100, 40);
        g.fillRect(100, 40, 4, 4);
    }
    public Dimension getMinimumSize()
    { return dim;
    }
    public Dimension getPreferredSize()
    { return dim;
    }
    private Dimension dim = new Dimension(220,100);
}

```

31

---

---

---

---

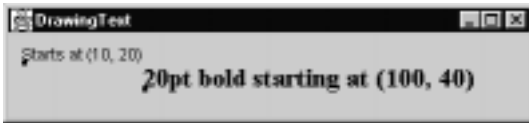
---

---

---

---

Notice where the small box is in relation to the text. That is the "location" of the text.



32

---

---

---

---

---

---

---

---

```

class DrawColors extends JComponent
{
    public void paint(Graphics g) {
        Color c = Color.blue;
        g.setColor(c.brighter());
        g.fillRect(10, 10, 200, 100);
        g.setColor(c);
        g.fillRect(20, 20, 180, 80);
        g.setColor(c.darker());
        g.fillRect(30, 30, 160, 60);
        g.setColor(c.darker().darker());
        g.fillRect(40, 40, 140, 40);
        g.setColor(Color.red);
        g.drawRect(10, 10, 200, 100);
    }
}

```

33

---

---

---

---

---

---

---

---

**Color.blue** appears to be the brightest blue already, so **Color.blue.brighter()** didn't do anything.



---

---

---

---

---

---

---

---

## Graphics2D

- "This Graphics2D class extends the Graphics class to provide more sophisticated control over geometry, coordinate transformations, color management, and text layout. This is the fundamental class for rendering 2-dimensional shapes, text and images on the Java(tm) platform." [From Standard javadoc documentation of Graphics2D.]

35

---

---

---

---

---

---

---

---

## Changing the drawing "stroke"

- We can control the following parameters of a "brush" stroke.
  - the width of the brush,
  - the dash pattern of the brush,
  - the shape of the end of a brush stroke,
  - what happens when two ends join, and
  - for dashed lines, where in the dash pattern does the stroke begin.

36

---

---

---

---

---

---

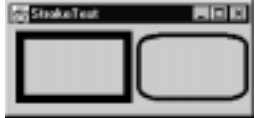
---

---

```

class StrokeSampler extends JComponent {
    static final BasicStroke wideStroke =
        new BasicStroke(8.0f);
    static final BasicStroke mediumStroke =
        new BasicStroke(4.0f);
    public void paint(Graphics g) {
        Graphics2D g2d = (Graphics2D)g;
        // draw some rectangles with wide lines
        g2d.setStroke(wideStroke);
        g2d.drawRect(10, 10, 100, 60);
        g2d.setStroke(mediumStroke);
        g2d.drawRoundRect(120, 10, 100, 60, 40, 20);
    }
}

```



37



```

// These definitions are shown out of order.
// They belong outside of the method.

static final BasicStroke wideRound =
    new BasicStroke(16.0f,
        BasicStroke.CAP_ROUND /* end style */,
        BasicStroke.JOIN_ROUND /* join style */);

static final BasicStroke wideBevel =
    new BasicStroke(16.0f,
        BasicStroke.CAP_BUTT /* end style */,
        BasicStroke.JOIN_BEVEL /* join style */);

```



38



```

// draw some lines with round ends/joints
int[] xcoords = {30, 60, 90};
int[] ycoords = {90, 140, 90};
g2d.setStroke(wideRound);
g2d.drawPolyline(xcoords, ycoords, 3);
// draw some lines with bevel joints
for (int i = 0; i < xcoords.length; i++)
    xcoords[i] = xcoords[i] + 110;
g2d.setStroke(wideBevel);
g2d.drawPolyline(xcoords, ycoords, 3);

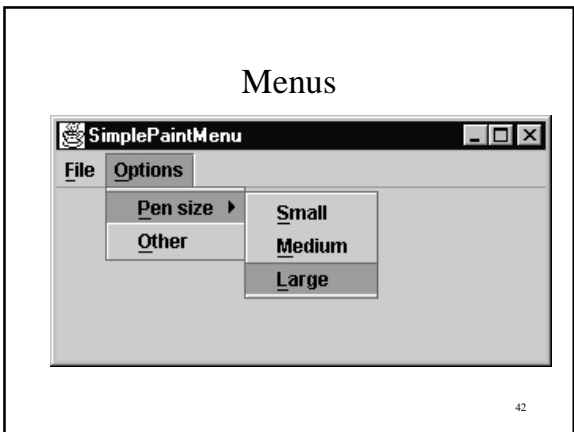
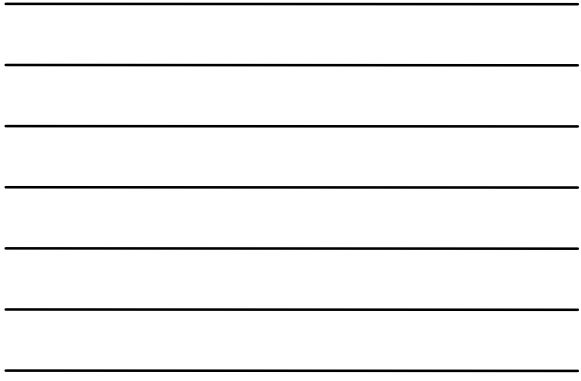
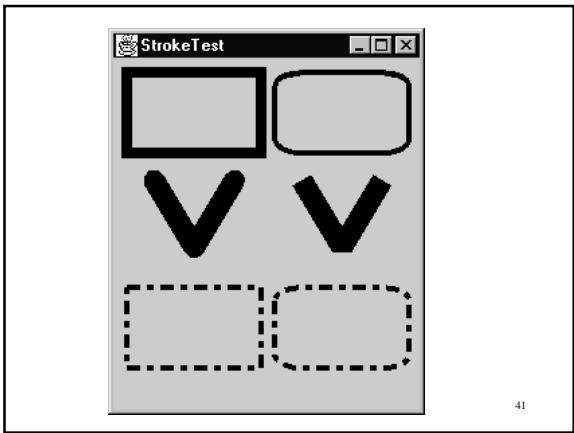
```

39



```
// use a dot-dash stroke
g2d.setStroke(dotDashStroke);
g2d.drawRect(10, 170, 100, 60);
g2d.drawRoundRect(120, 170, 100, 60, 40, 20);
}

static float[] dotDash = {10.0f, 5.0f, 5.0f, 5.0f};
static final BasicStroke dotDashStroke =
    new BasicStroke(4.0f /*width*/,
        BasicStroke.CAP_BUTT /*end style*/,
        BasicStroke.JOIN_MITER /*join style*/,
        1.0f /*miter trim limit */,
        dotDash /* pattern array */,
        0.0f /* offset to start of pattern */);
```



## SimplePaintMenu

- **SimplePaintMenu** - main().
- **PaintListener3** - extends **PaintListener2**, adding `setPenSize()`.
- **Painter** - an interface that isolates the mouse motion listener, `PaintListener3`, from the action event listener, `PenAdjuster`.
- **PenAdjuster** - handles menu selections.
- **DrawingCanvas2** - where the drawing occurs.

43

---

---

---

---

---

---

---

---

```
// SimplePaintMenu.java - add a menu to SimplePaint2
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class SimplePaintMenu {
    public static void main(String[] args) {
        JFrame frame = new JFrame("SimplePaintMenu");
        Container pane = frame.getContentPane();
        // create the canvas and mouse listener
        // just as in SimplePaint2
        DrawingCanvas2 canvas = new DrawingCanvas2();
        PaintListener3 painter = new PaintListener3();
        canvas.addMouseListener(painter);
        pane.add(canvas);
    }
}
```

44

---

---

---

---

---

---

---

---

```
// create the menu bar and top level menus
JMenuBar menuBar = new JMenuBar();
JMenu fileMenu = new JMenu("File");
fileMenu.setMnemonic('F');
JMenu optionsMenu = new JMenu("Options");
optionsMenu.setMnemonic('O');
menuBar.add(fileMenu);
menuBar.add(optionsMenu);
frame.setJMenuBar(menuBar);

// add items to the fileMenu
JMenuItem exit = new JMenuItem("Exit", 'x');
fileMenu.add(exit);
exit.addActionListener(new GoodBye());
```

45

---

---

---

---

---

---

---

---

```

// create and add items to the optionsMenu
// the first item is a submenu for pen size
JMenu penAdjusterMenu = new JMenu("Pen size");
penAdjusterMenu.setMnemonic('P');
// create and add items to the pen size sub menu
JMenuItem smallPen = new JMenuItem("Small", 'S');
JMenuItem mediumPen = new JMenuItem("Medium", 'M');
JMenuItem largePen = new JMenuItem("Large", 'L');
penAdjusterMenu.add(smallPen);
penAdjusterMenu.add(mediumPen);
penAdjusterMenu.add(largePen);

```

46

---

---

---

---

---

---

---

---

```

// add a listener to the pen selection items
PenAdjuster penAdjuster = new PenAdjuster(painter);
smallPen.addActionListener(penAdjuster);
mediumPen.addActionListener(penAdjuster);
largePen.addActionListener(penAdjuster);
optionsMenu.add(penAdjusterMenu);
// for demo purposes add second (unused) item
optionsMenu.add(new JMenuItem("Other", 'O'));

frame.pack();
frame.show();
}
}

```

47

---

---

---

---

---

---

---

---

```

public class PaintListener3 extends PaintListener2
    implements Painter
{
// specify one of three pen sizes, Small, Medium or Large
// radius and diameter are inherited
public void setPenSize(String size) {
    if (size.equals("Small")) {
        radius = 0;
        diameter = 1;
    }
    else if (size.equals("Medium")) {
        radius = 3;
        diameter = radius * 2;
    }
    else if (size.equals("Large")) {
        radius = 6;
        diameter = radius * 2;
    }
}
}
}

```

48

---

---

---

---

---

---

---

---



```

// Painter.java -
//   in practice this interface would contain
//   additional methods such as for changing pen
//   shape, color, pattern etc.
interface Painter {
    public void setPenSize(String size);
}

```

---

---

---

---

---

---

---

---

```

// PenAdjuster.java - pass pen adjustment requests
//   to the painter
import java.awt.event.*;

class PenAdjuster implements ActionListener {
    private Painter painter;
    PenAdjuster(Painter thePainter) {
        painter = thePainter;
    }
    public void actionPerformed(ActionEvent e) {
        painter.setPenSize(e.getActionCommand());
    }
}

```

---

---

---

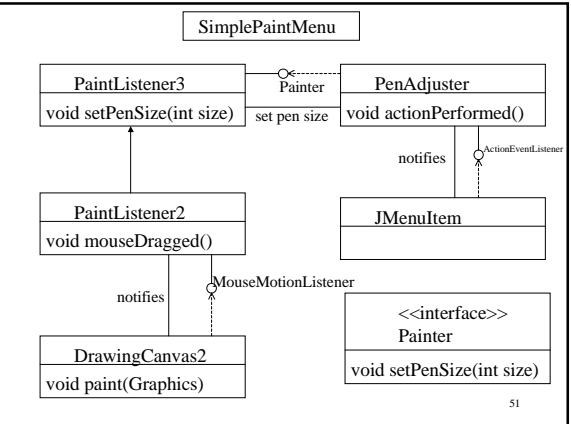
---

---

---

---

---




---

---

---

---

---

---

---

---

## Template for a Game Board



52

---

---

---

---

---

---

---

---

```
class GameTest {
    public static void main(String[] args)
    {
        byte[][] state = new byte[20][20];

        JFrame frame = new JFrame("GameTest");
        Container pane = frame.getContentPane();
        GameBoard board = new GameBoard(400, 400, state);
        GameEventHandler actor = new GameEventHandler();
        board.addMouseListener(actor);
        pane.add(board, BorderLayout.CENTER);
        JButton quit = new JButton("Quit");
        quit.addActionListener(actor);
        pane.add(quit, BorderLayout.NORTH);
        frame.pack();
        frame.show();
    }
}
```

53

---

---

---

---

---

---

---

---

```
class GameEventHandler
    implements MouseListener, ActionListener
{
    public void actionPerformed(ActionEvent e) {
        System.out.println("Goodbye!");
        System.exit(0);
    }
    public void mouseClicked(MouseEvent e)
    {
        GameBoard board = (GameBoard)e.getSource();
        int row = e.getY() / board.getCellHeight();
        int col = e.getX() / board.getCellWidth();
        if (board.getCell(row, col) == board.STATE0)
            board.setCell( row, col, board.STATE1);
        else
            board.setCell( row, col, board.STATE0);
        board.repaint();
    }
}
```

54

---

---

---

---

---

---

---

---

```
public void mouseEntered(MouseEvent e) {};  
public void mouseExited(MouseEvent e) {};  
public void mousePressed(MouseEvent e) {};  
public void mouseReleased(MouseEvent e) {};  
}  
  
55
```

---

---

---

---

---

---

---

---

```
//GameBoard.java  
import java.awt.*;  
import javax.swing.*;  
  
class GameBoard extends JComponent {  
    // the example supports only two valid states  
    public static final byte INVALID = -1;  
    public static final byte STATE0 = 0;  
    public static final byte STATE1 = 1;  
  
    public GameBoard(int width, int height,  
                     byte[][] board) {  
        this.board = board;  
        setSize(new Dimension(width, height));  
    }  
  
56
```

---

---

---

---

---

---

---

---

```
// draw the cells of the board  
public void paint(Graphics g) {  
    for (int row = 0; row < board.length; row++)  
        for (int col=0; col < board[row].length; col++){  
            select_color(g, board[row][col]);  
            g.fillRect(col * cellWidth,  
                       row * cellHeight,  
                       cellWidth, cellHeight);  
        }  
}  
public Dimension getMinimumSize() {  
    return dim;  
}  
public Dimension getPreferredSize() {  
    return getMinimumSize();  
}  
}  
  
57
```

---

---

---

---

---

---

---

---

```

public void setBounds(int x, int y,
                    int width, int height)
{
    dim = new Dimension(width, height);
    //adjust cell size to fill the new board size
    cellWidth =
        Math.round((float)width / board.length);
    cellHeight = Math.round(
        (float)height / board[0].length);
    super.setBounds(x, y, width, height);
    repaint();
}

```

58

---

---

---

---

---

---

---

---

```

public void setCell(int row, int col, byte value) {
    board[row][col] = value;
}
public byte getCell(int row, int col) {
    return board[row][col];
}
public int getCellWidth() { return cellWidth; }
public int getCellHeight() { return cellHeight; }
private void select_color(Graphics g, byte cell) {
    if (cell == STATE0)
        g.setColor(Color.white);
    else
        g.setColor(Color.black);
}
private byte[][] board;
private int cellWidth, cellHeight;
private Dimension dim;
}

```

59

---

---

---

---

---

---

---

---

## Using an Adapter

```

class GameEventHandler2 extends MouseAdapter
    implements ActionListener
{
    public void mouseClicked(MouseEvent e) {
        GameBoard board = (GameBoard)e.getSource();
        int row = e.getY() / board.getCellHeight();
        int col = e.getX() / board.getCellWidth();
        if (board.getCell(row, col) == board.STATE0)
            board.setCell(row, col, board.STATE1);
        else
            board.setCell(row, col, board.STATE0);
        board.repaint();
    }
}

```

60

---

---

---

---

---

---

---

---

```
public void actionPerformed(ActionEvent e) {  
    System.out.println("Goodbye!");  
    System.exit(0);  
}  
}
```

61

---

---

---

---

---

---

---

---