Swing and Java2D

Albert Weichselbraun

Graphical User Interfaces with Swing

- Swing is based on AWT
- lightweight components
- implements the MVC (Model-View-Controller) architecture
- For a visual guide to swing components see the guide[1] in the Java Swing tutorial.
- components, which are sublasses of `Container`, may contain other components (e.g. JPanel, JFrame, . . .)

[1] `http://download.oracle.com/docs/cd/E17409\_01/javase/tutorial/ui/features/components.html`

Approach

1. determine layout
   - position of the components
   - position of the JPanels / layout manager
2. add Swing components
3. add event handling

Determine Layout

```
1  package at.ac.wuwien.examples.swingdemo;

3  import javax.swing.*;

5  public class SimpleGUI extends JFrame {

7    // optional
8    public SimpleGUI(String title) {
9      super(title);
10   }

12   // main program
13   public static void main(String[] args) {
14     SimpleGUI sg = new SimpleGUI("Swing Example");
```

```
15        sg.pack();
16        sg.setVisible(true);
17    }
18 }
```

Add Swing Components

```
1  package at.ac.wuwien.examples.swingdemo;
2  import java.awt.BorderLayout;
3  import javax.swing.*;

5  public class SimpleGUI2 extends JFrame {

7    private JTextArea textoutput;
8    private JTextField textinput;
9    private JButton addButton, clearButton;

11   public SimpleGUI2(String title) {
12     super(title);
13     // create Text input/output fields
14     textoutput = new JTextArea("Output",3,10);
```

```
15      textinput  = new JTextField("Input");

17      // Create Buttons
18      JPanel controlPanel = new JPanel();
19      addButton   = new JButton("Add");
20      clearButton = new JButton("Clear");

22      // add buttons to the controlPanel
23      controlPanel.add(addButton);
24      controlPanel.add(clearButton);

26      // add everything to the layout
27      add(controlPanel, BorderLayout.NORTH);
28      add(textinput, BorderLayout.CENTER);
29      add(textoutput, BorderLayout.SOUTH);
30    }
```

```
32    // main program
33    public static void main(String[] args) {
34      SimpleGUI2 sg = new SimpleGUI2("Swing Example");
35      sg.pack();
36      sg.setVisible(true);
37    }
38  }
```

Add Event Handling

```
1  public class SimpleGUI3 extends JFrame
2     implements ActionListener {

4     ...

6     public SimpleGUI3(String title) {
7        super(title);

9        ...

11       // eventhandling
12       addButton.addActionListener( this );
13       clearButton.addActionListener( this );
14    }
```

```
16      public void actionPerformed(ActionEvent e) {
17          if (e.getSource() ==  clearButton) {
18              textoutput.setText("");
19          }
20          if (e.getSource() ==  addButton) {
21              textoutput.setText( textoutput.getText()
22                      + "\n" + textinput.getText() );
23          }
24      }
```

Extension - scrollable TextField

```
1           ...

3           // scrolling
4           JScrollPane sp = new JScrollPane(textoutput);
5           this.add(sp, BorderLayout.SOUTH);

7           ...
```

Exercise: Foreign Exchange Calculator

The foreign exchange calculator takes an amount and computes its value in the following currencies:

- USD (1 EUR = 1.32 USD)
- AUD (1 EUR = 1.689 AUD)
- JPY (1 EUR = 154.85 JPY)

Attention: `getText()` returns a `String` value. You need the static method `Double.parseDouble()` to convert its value to a `double` (see next slide).

String, Double, double

- `String`: sequence of characters $\rightarrow$ `"12.1"+ "13" = "12.113"`
- `double`: primitive $\rightarrow$ `12.1 + 13 = 25.1`
- `Double`: wrapper-object for the datatype `double`; contains methods for converting doubles to strings and vice versa

- Conversion: String `s` $\rightarrow$ double `d`:

```
1   double d = Double.parseDouble(s);
```

- Conversion: double `d` $\rightarrow$ String `s`:

```
1   String s = new Double(d).toString();
```

Java2D

- Customized Swing Components
- by overwriting of the method `paintComponent()` of the superclass `JComponent`
  - execute `paintCompontent()` of the superclass
  - add your extensions
  - `repaint()` makes your changes visible
- coordinate system
- use and benefits of using buffers:

```
BufferedImage buffer=new BufferedImage(x_size,
y_size, BufferedImage.TYPE_INT_RGB)
```

Customized Swing Components

- Approach
  1. inherit your class from `JComponent`
  2. create a `BufferedImage` of the size of your component
     $\rightarrow$ perform all operations on the buffer first
  3. overwrite `paintComponent()` so that it pasts the content of the `BufferedImage` to the customized component
  4. call `repaint` after all drawing operations.

Drawing with Java2D

- Approach
  1. create a Graphics2D Object and
  2. a shape (Ellipse2D.Double, Rectangle2D.Double, ...) object
  3. paste the shape on your image using `draw(element)` or `fill(element)`.

Paste Pictures

- Approach
  1. import `java.awt.image.*`
  2. always access your image file using `getClass().getResource(fname)` (not limited to the file system)
  3. initialize the `Image`-object with `ImageIO.read( url )`
  4. draw the image using `drawImage()`

```
1  package at.ac.wu.examples.java2d;

3  import java.awt.*;
4  import java.awt.event.*;
5  import javax.swing.*;

7  public class Java2DDemo extends JFrame
8    implements ActionListener{

10   private static enum Demos { SHAPE_DEMO, IMAGE_DEMO };
11   private Demos displayDemo = Demos.SHAPE_DEMO;
12   private MyComponent mc;

14   private JButton bShapes;
15   private JButton bImage;
16   private JButton bNorth, bSouth, bWest, bEast;
```

```
18    public Java2DDemo(String title) {
19      super(title);

21      // show control panel
22      JPanel demoControlPanel = new JPanel();
23      bShapes = new JButton("Shapes");
24      bShapes.addActionListener(this);
25      bImage = new JButton("Image");
26      bImage.addActionListener(this);
27      demoControlPanel.add(bShapes);
28      demoControlPanel.add(bImage);

30      this.add(demoControlPanel, BorderLayout.SOUTH);
```

```
33      // create cursor control buttons
34      bNorth = new JButton("N");
35      bNorth.addActionListener(this);
36      bSouth= new JButton("S");
37      bSouth.addActionListener(this);
38      bWest = new JButton("<-");
39      bWest.addActionListener(this);
40      bEast = new JButton("->");
41      bEast.addActionListener(this);

43      JPanel cursorControlPanel = new JPanel();
44      cursorControlPanel.add(bWest);
45      cursorControlPanel.add(bNorth);
46      cursorControlPanel.add(bSouth);
47      cursorControlPanel.add(bEast);
48      this.add(cursorControlPanel, BorderLayout.CENTER);
```

```
50        // display custom component
51        mc = new MyComponent(320, 200);
52        this.add(mc, BorderLayout.NORTH);
53        mc.drawShapes();
54      }

56      // input event handling (buttons) and
57      // implementation of the ActionListener interface
58      public void actionPerformed(ActionEvent e) {
59        if (e.getSource()==bShapes) {
60          displayDemo = Demos.SHAPE_DEMO;
61        }
62        else if (e.getSource()==bImage) {
63          displayDemo = Demos.IMAGE_DEMO;
64        }
```

```
65      else if (e.getSource()==bWest) { mc.moveWest(); }
66      else if (e.getSource()==bEast ) { mc.moveEast(); }
67      else if (e.getSource()==bNorth) { mc.moveNorth(); }
68      else if (e.getSource()==bSouth) { mc.moveSouth(); }

70      // display the selected demo
71      System.out.println(displayDemo);
72      switch (displayDemo) {
73        case SHAPE_DEMO: mc.drawShapes(); break;
74        case IMAGE_DEMO: mc.drawImage(); break;
75      }
76    }

78    // main method (static)
79    // create an instance of the demo and execute it
80    public static void main(String[] args) {
```

```
81        Java2DDemo demo = new Java2DDemo("Java2D Demo");
82        demo.pack();
83        demo.setVisible( true );
84    }
85 }
```

```
1  package at.ac.wu.examples.java2d;

3  import javax.swing.*;
4  import java.awt.*;
5  import java.awt.geom.*;
6  import java.awt.image.*; // BufferdImage
7  import java.net.URL;

9  import javax.imageio.*;
10 import java.io.*;

12 public class MyComponent extends JComponent {

14   private BufferedImage buffer;
15   private Image img;
```

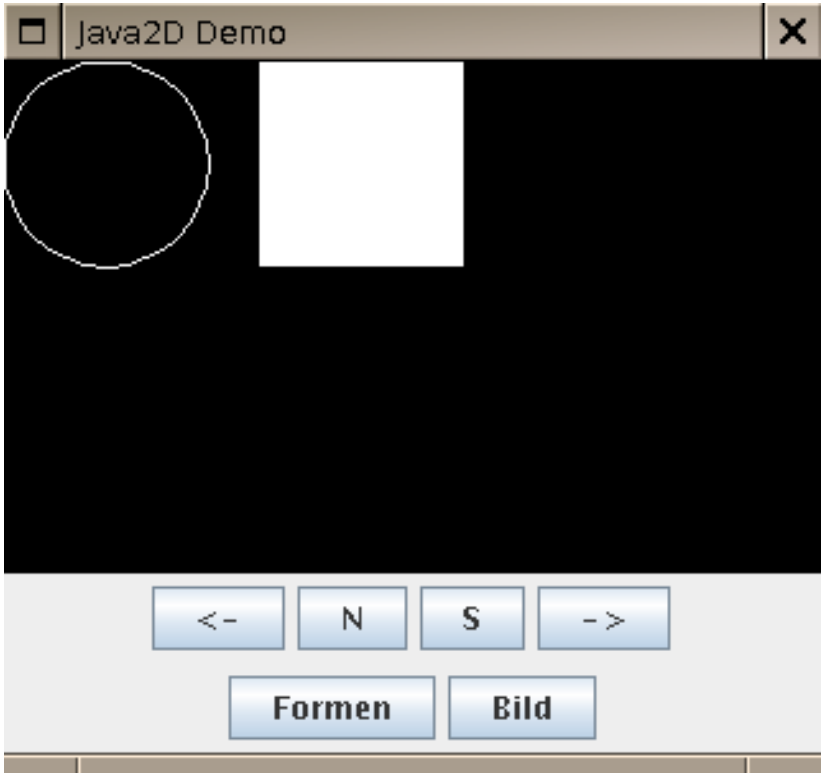
```
17    private int x_size, y_size;
18    private int x_pos, y_pos;

20    public MyComponent(int x_size, int y_size) {
21      // set MyComponent's size
22      this.x_size = x_size;
23      this.y_size = y_size;

25      // object start position
26      this.x_pos = 0;
27      this.y_pos = 0;
28      this.setPreferredSize(
29          new Dimension(x_size, y_size));
30      this.setMaximumSize(
31          new Dimension(x_size, y_size));
32      this.setMinimumSize(
```

```
33            new Dimension(x_size, y_size));
34        this.setDoubleBuffered(true);

36        // load image
37        try {
38          // retrieve the image from the web
39          // URL bildUrl = new URL("http://www.bugzilla.org

41          // or from the filesystem/jar file
42          URL bildUrl = getClass().
43            getResource("/at/ac/wu/examples/java2d/buggie.p
44          img = ImageIO.read( bildUrl );
45        } catch (IOException e) {
46          System.err.println(e);
47        }
```

```
50      // create buffer image
51      this.buffer =
52        new BufferedImage(x_size, y_size,
53            BufferedImage.TYPE_INT_RGB);
54      this.setBackground(Color.GRAY);
55    }


58    protected void paintComponent(Graphics g) {
59      super.paintComponent(g);
60      Graphics2D g2d = (Graphics2D)g.create();
61      g2d.drawImage(this.buffer, 0, 0, this);
62      g2d.dispose(); // dispose handle (not required)
63      }
```

```
65    // control object movement
66    public void moveNorth() { y_pos -=5; }
67    public void moveSouth() { y_pos +=5; }
68    public void moveEast()  { x_pos +=5; }
69    public void moveWest() { x_pos -=5; }

71    public void drawShapes() {
72      Graphics2D g2d = this.buffer.createGraphics();
73      g2d.clearRect( 0, 0, x_size, y_size);

75      Ellipse2D.Double ellipse =
76        new Ellipse2D.Double(x_pos, y_pos, 80, 80);
77      g2d.draw( ellipse );

79      Rectangle2D.Double box =
80        new Rectangle2D.Double(x_pos+100, y_pos, 100, 80)
```

```
81        g2d.fill( box );
82        repaint();
83    }

85    public void drawImage() {
86        Graphics2D g2d = this.buffer.createGraphics();
87        g2d.clearRect( 0, 0, x_size, y_size);

89        g2d.drawImage( img, x_pos, y_pos, null);
90        repaint();
91    }

93 }
```


Java2D Demo
<-
N
S
->
Formen
Bild


Java2D Demo
<-
N
S
->
Formen
Bild

Exercise: Dice

Write an application which paints a dice and a “dice”-button. Every time a user presses the dice-button the dice is expected to show a random number between one and six.

Note: You can compute a random number by importing the class `java.util.Random` creating a `Random()` object and calling its `nextInt` method.

```
1 import java.util.Random;

3 class RandomNumber{
4     private Random rnd = new Random();

6     public int getRandomNumber() {
7         return rnd.nextInt(6)+1;
8     }
9 }
```