Programmierpraktikum

Claudius Gros, SS2012

Institut für theoretische Physik
Goethe-University Frankfurt a.M.

Java Applets

Java applets

- embedding Java applications into webpages
- simulations, tutorials, web interfaces

applet example

- copy Java code to file *AppletDrawingLines.java*
- run the Java compiler `javac AppletDrawingLines`
- copy html code to file *anything.html*
- open the html file with a browser `firefox anything.html`, sometimes you need a reload

```
1. import javax.swing.*;    // for JApplet, ..
2. import java.awt.*;       // for Color
3. import java.util.*;      // for sleeping
4.
5. /** Java applet example based on Japplet.
```

```
 6.   * Has no main() function, init() executed before paint().
 7.   */
 8. public class AppletDrawingLines extends JApplet {
 9.
10.   int width, height;            // size of the html applet box
11.
12.   public void init() {          // executed when the applet starts
13.     width  = getSize().width;   // getSize() is a member function of JApplet
14.     height = getSize().height;  // returning the size of the html applet box
15.     setBackground(Color.black);
16.     }
17.
18. /** The .pain() function is executed  whenever the applet is
19.   * asked to redraw itself.
20.   */
21.   public void paint(Graphics g) {
22.     g.setColor(Color.green);
23.     for (int i=0; i<10; i++)
24.       {
25.       g.drawLine(width, height, i * width / 10, 0);
26.                                   // drawLine(x-start,y-start,x-end,y-end)
27.       try {
28.         Thread.sleep(700);        // sleep for a short time
```

```
29.          } catch (InterruptedException e) {
30.            System.out.println(e.toString());
31.            }
32.        } // end of loop over drawing lines
33.     }   // end of AppletDrawingLines.paint()
34. }       // end of AppletDrawingLines
```

- html code for embedding applet
- html tags are like brackets: ...

```
<html>
<head>
<meta charset="utf8" />
<title>Java applets</title>
</head>
<!--  this is a comment: a html page has a header and a body -->
<body>
<center>
  <h1>Embedded Java applet</hr>
  <br><br>

  <applet width=400 height=400 code="AppletDrawingLines.class"> </applet>
</center>
</body>
</html>
```

Mouse events

- here `Applet` instead of `JApplet`
  and `java.awt` instead of `javax.swing`
- the interfaces `MouseListener` and `MouseMotionListener` are implemented,
  all their member functions need to be explicitly declared, even when void

```
1. import java.applet.*;     // this example with Applet instead of JApplet
2. import java.awt.*;
3. import java.awt.event.*;
4.
5. public class AppletMouseFollow extends Applet
6.         implements MouseListener, MouseMotionListener {
7.
8.   int width, height;              // size of html applet box
9.   int mx, my;                     // the mouse coordinates
10.  boolean isButtonPressed = false;
11.
12.  public void init() {
```

```
13.     width = getSize().width;
14.     height = getSize().height;
15.     setBackground(Color.black);
16.
17.     mx = width/2;                // starting position of
18.     my = height/2;               // square
19.
20.     addMouseListener( this );    // adding event listeners
21.     addMouseMotionListener( this );
22.     }
23.
24. /** Called when the pointer enters the applet's box.
25.   * All functions of the implemented interfaces
26.   * MouseListener and MouseMotionListener need to be implemented.
27.   */
28.   public void mouseEntered( MouseEvent e ) {
29.      setBackground(Color.blue);
30.     }
31.   public void mouseExited( MouseEvent e ) {
32.     setBackground(Color.orange);
33.     }
34. /** Called after a press and release of a mouse button
35.   * with no motion in between.
```

```
36.    * If the user presses, drags, and then releases,
37.    * there will be no click event generated.
38.    */
39.   public void mouseClicked( MouseEvent e ) {
40.     setBackground(Color.cyan);
41.     }
42. /** Called when mouse down but not released.
43.    */
44.   public void mousePressed( MouseEvent e ) {
45.     isButtonPressed = true;
46.     setBackground(Color.gray);
47.     repaint();                    // re
48. // "Consume" the event so it won't be processed in the
49. // default manner by the source which generated it.
50.     e.consume();
51.     }
52. /** Called when released after drag.
53.    */
54.   public void mouseReleased( MouseEvent e ) {
55.     isButtonPressed = false;
56.     setBackground(Color.black);
57.     repaint();
58.     e.consume();
```

```
59.     }
60. /** Called during motion when no buttons are down.
61.   */
62.   public void mouseMoved( MouseEvent e ) {
63.     mx = e.getX();                // the mouse events provides the current
64.     my = e.getY();                // mouse position
65.     showStatus( "Mouse at (" + mx + "," + my + ")" );  // browser status line
66.     repaint();                    // calls the paint() method
67.     e.consume();
68.     }
69. /** Called during motion with buttons down.
70.   */
71.   public void mouseDragged( MouseEvent e ) {
72.     mx = e.getX();
73.     my = e.getY();
74.     showStatus( "Mouse at (" + mx + "," + my + ")" );
75.     repaint();
76.     e.consume();
77.     }
78.
79.   public void paint( Graphics g ) {
80.     if (isButtonPressed)
81.       g.setColor(Color.black);    // black rectangle if moused down
```

```
82.       else
83.       g.setColor(Color.gray);     // gray otherwise
84. //
85.       g.fillRect(mx-20, my-20, 40, 40); // filled and centered rectangle
86.     }
87. } // end of AppletMouseFollow
```

Bouncing with and without threads

- try to to change speed while bouncing
- **double buffering** for professional applications, avoiding flickering: paint first everything offscreen and then upload full graphics (not done here)

```
 1. import java.awt.*;
 2. import java.awt.event.*;
 3. import java.applet.Applet;
 4.
 5. /** A class can extend only a single parent class
 6.   * but it can implement any number of interfaces.
 7.   */
 8. public class AppletBounce extends Applet
 9.         implements ActionListener, AdjustmentListener, Runnable
10. {
11.   private int width, height;       // width, height of applet box
12.   private int tick = 100;          // length of pausini, speed control
13.   private int initialY = 70;       // bouncing between these
```

```
14.   private int lowestY;             // two values
15.
16.   private Button goThreaded    = new Button("go threaded");
17.   private Button goNotThreaded = new Button("go not threaded");
18.   private Scrollbar speed      =
19.       new Scrollbar(Scrollbar.HORIZONTAL,tick,30,20,400);
20.
21. /** Called when applet is loaded.
22.   */
23.   public void init() {
24.     setBackground(new Color(230,230,230));
25.     width    = getSize().width;
26.     height   = getSize().height;
27.     lowestY = (int)(height*0.8);   // use relative units
28.
29. // --- select layout manager
30.     setLayout(new BorderLayout()); // a layout manger
31.     Panel buttons = new Panel();   // a seperate panel for the buttons
32.
33. // --- button for starting animation with current thread
34.     goNotThreaded.setBackground(Color.lightGray);
35.     goNotThreaded.addActionListener(this);
36.     buttons.add(goNotThreaded);
```

```
37.
38. // --- button for starting animation in an additional thread
39.     goThreaded.setBackground(Color.lightGray);
40.     goThreaded.addActionListener(this);
41.     buttons.add(goThreaded);
42.     add("North", buttons);          // add both bottoms to the top
43.
44. // --- the Scrollbar 'speed' is inside the Panel 'speedControl'
45.     Panel speedControl = new Panel();
46.     speedControl.setBackground(Color.lightGray);
47.     speed.setPreferredSize(new Dimension((int)(width*0.6),20));
48.     speedControl.add(new Label("faster",Label.RIGHT));
49.     speedControl.add(speed);
50.     speed.addAdjustmentListener(this);
51.     speedControl.add(new Label("slower"));
52.     add("South", speedControl);    // add the speed-control bar to the bottom
53.
54.     }  // end of AppletBounce.init()
55.
56. /** Catch scrollbar event.
57.   */
58.   public void adjustmentValueChanged(AdjustmentEvent e) {
59.     tick = speed.getValue();
```

```
60.     }
61.
62. /** Catch button events.
63.   */
64.   public void actionPerformed(ActionEvent e) {
65.     if (e.getSource() == goThreaded)      // the new Thread calls run()
66.       new Thread(this).start();           // running the object 'this'
67.     if (e.getSource() == goNotThreaded)   // calling directly
68.       doAnimation();                      // doAnimation()
69.     }
70.
71. /** AppletBounce is implementing runnable and run() is
72.   * called when a new thread is started.
73.   */
74.   public void run() {
75.     doAnimation();
76.     }
77.
78. /** Let a ball at random position x bounce downwards
79.   * and upwards, with increasing speed (step).
80.   */
81.   public void doAnimation() {
82. // --- setup of bounds
```

```
83.     int minX = (int)(width*0.1);                    // allways use
84.     int maxX = (int)(width*0.9);                    // relative units
85.     int x = (int)(minX+Math.random()*(maxX-minX)); // fixed random x for bouncing
86.     int y = initialY;                               // variable y of bouncing
87.     int step = 1;                                   // (in)decrement for y
88.
89. // --- setup of graphics
90.     Graphics g = getGraphics();
91.     Color background = getBackground();             // needed for repainting
92.     g.setColor(Color.black);
93.     g.drawLine(minX, lowestY+20,                    // bottom line
94.                maxX, lowestY+20);
95.
96. // --- bouncing downwards
97.     while (y <= lowestY) {
98.       g.fillOval(x, y, 20, 20);             // paint ball
99.       pause(tick);                          // pause animation method
100.      g.setColor(background);
101.      g.fillOval(x, y, 20, 20);             // turn ball to background
102.      g.setColor(Color.black);
103.      y = y+step;
104.      step++;
105.      }
```

```
106.
107. // --- hitting bottom
108.     g.setColor(new Color(204,51,51));
109.     g.fillOval(x, lowestY, 20, 15);          // extra image for lowest point
110.     g.drawString("boing!",x-10,lowestY+35); // its 'boing'
111.     pause(tick*3);                           // pause for reading
112.     g.setColor(background);
113.     g.drawString("boing!",x-10,lowestY+35);
114.     g.fillOval(x, lowestY, 20, 15);          // erase the extra squeeze ball
115.
116. // --- bouncing upwards
117.     g.setColor(Color.black);
118.     while (step>0) {
119.       step--;
120.       y = y-step;
121.       g.fillOval(x, y, 20, 20);
122.       pause(tick);
123.       g.setColor(background);
124.       g.fillOval(x, y, 20, 20);
125.       g.setColor(Color.black);
126.     }
127.   }    // end of AppletBounce.doAnimation()
128.
```

```
129. /** Just a wrapper for the exception capture.
130.   */
131.   private void pause(int pauseLength) {
132.     try { Thread.sleep(pauseLength); } catch (InterruptedException e) {}
133.     }
134. }       // end of AppletBounce
```

What to do with applets?

physics / science simulations - visualization

- 1D quantum wave packet scattering
- one / two player collision game
- Conway's Game of Life
- ...

interaction

- web-based GUI (graphical user interface)
- web-based user-user interactions (maps, chat, mail, ...)
- browser games