```
1   package lifecycle;
2
3   import java.applet.Applet;
4   import java.awt.*;
5
6   /**
7    * A LyfeCycle Applets is an applet that demonstrates the major steps in
8    * the lyfe cycle of all applets by printing messages in the milestones of it's
9    * life cycle
10   */
11  public class LyfeCycleApplet extends Applet {
12
13          private int initialized, started, stopped, destroyed, painted;
14
15          public void init() {
16                  initialized++;
17                  repaint();
18          }
19
20          public void start() {
21                  started++;
22                  repaint();
23          }
24
25          public void stop() {
26                  stopped++;
27                  System.out.println("called stop!\n");
28
29                  repaint();
30          }
31
32          public void destroy() {
33                  destroyed++;
34                  repaint();
35                  System.out.println("called destroy!\n");
36          }
37
38          public void paint(Graphics painter) {
39                  painted++;
40                  painter.setColor(Color.blue);
41                  painter.drawString("Initialized: " + initialized, 15, 20);
42                  painter.setColor(Color.green);
43                  painter.drawString("Started: " + started, 15, 50);
44                  painter.setColor(Color.red);
45                  painter.drawString("Stopped: " + stopped, 15, 80);
46                  painter.setColor(Color.black);
47                  painter.drawString("Destroyed: " + destroyed, 15, 110);
48                  painter.drawString("Painted: " + painted, 15, 140);
49                  System.out.println(this);
50          }
51
52          public String toString() {
53                  StringBuffer result = new StringBuffer("LyfeCycleApplet");
54                  result.append("\nInitialized " + initialized);
55                  result.append("\nStarted " + started);
56                  result.append("\nStopped " + stopped);
57                  result.append("\nDestroyed " + destroyed);
58                  result.append("\nPainted " + painted);
59                  result.append("\n==========================\n\n\n");
60                  return result.toString();
61          }
62  }
63
```

```
1   package paintModel;
2
3   // <APPLET CODE="PaintModel1.class" WIDTH=200 HEIGHT=200></APPLET>
4
5   import java.applet.*;
6   import java.awt.event.*;
7   import java.awt.*;
8
9   public class PaintModel1 extends Applet {
10          // The paint model: the last click Point
11          private Point lastClick = null;
12
13          public void init() {
14                  addMouseListener(new MyModelRecorder());
15          }
16
17          public void paint(Graphics g) {
18                  if (lastClick != null) {
19                          g.drawString("Hello World!", lastClick.x, lastClick.y);
20                  }
21          }
22
23          private class MyModelRecorder extends MouseAdapter {
24                  public void mousePressed(MouseEvent e) {
25                          lastClick = e.getPoint();
26                          repaint();
27                  }
28          }
29  }
```

```
1   package paintModel;
2
3   // <APPLET CODE="PaintModel2.class" WIDTH=200 HEIGHT=200></APPLET>
4
5   import java.applet.*;
6   import java.awt.event.*;
7   import java.awt.*;
8
9   public class PaintModel2 extends Applet {
10          // The paint model: the last click Point
11          private Point lastClick = null;
12
13          public void init() {
14                  addMouseListener(new MyModelRecorder());
15          }
16
17          public void update(Graphics g) {
18                  paint(g);
19          }
20
21          public void paint(Graphics g) {
22                  if (lastClick != null) {
23                          g.drawString("Hello World!", lastClick.x, lastClick.y);
24                  }
25          }
26
27          private class MyModelRecorder extends MouseAdapter {
28                  public void mousePressed(MouseEvent e) {
29                          lastClick = e.getPoint();
30                          repaint();
31                  }
32          }
33  }
```

```
1   package paintModel;
2
3   // <APPLET CODE="PaintModel3.class" WIDTH=200 HEIGHT=200></APPLET>
4   import java.applet.*;
5   import java.awt.event.*;
6   import java.awt.*;
7   import java.util.List;
8   import java.util.ArrayList;
9
10  public class PaintModel3 extends Applet {
11          // The paint model: a list of click Points
12          private List mouseClicks = new ArrayList(5);
13
14          public void init() {
15                  addMouseListener(new MyModelRecorder());
16          }
17
18          public void update(Graphics g) {
19                  paint(g);
20          }
21
22          public void paint(Graphics g) {
23                  for (int x = 0; x < mouseClicks.size(); x++) {
24                          Point p = (Point) mouseClicks.get(x);
25                          g.drawString("Hello World!", p.x, p.y);
26                  }
27          }
28
29          private class MyModelRecorder extends MouseAdapter {
30                  public void mousePressed(MouseEvent e) {
31                          mouseClicks.add(e.getPoint());
32                          repaint();
33                  }
34          }
35  }
```

```
1   package freeHand;
2
3   import java.applet.Applet;
4   import java.awt.*;
5   import java.awt.event.*;
6   import java.util.Vector;
7
8   /**
9    * A Free Hand drawing application
10   */
11  public class FreeHandDrawing extends Applet implements MouseListener,
12                  MouseMotionListener, WindowListener {
13
14          protected Vector lines;
15
16          protected int x;
17
18          protected int y;
19
20          /**
21           * Constructs a new FreeHandDrawing
22           */
23          public FreeHandDrawing() {
24                  addMouseMotionListener(this);
25                  addMouseListener(this);
26                  lines = new Vector();
27          }
28
29          /**
30           * process the event of pressing the mouse - start drawing
31           */
32          public void mousePressed(MouseEvent e) {
33                  x = e.getX();
34                  y = e.getY();
35          }
36
37          /**
38           * process the event of dragging the mouse - draw
39           */
40          public void mouseDragged(MouseEvent e) {
41                  Rectangle rect = new Rectangle(x, y, e.getX(), e.getY());
42                  lines.addElement(rect);
43                  x = e.getX();
44                  y = e.getY();
45                  repaint();
46          }
47
48          // since this class is also the listener we could't use an adapter
49          // and must supply a blank implementation to these methods
50          // this is obviously not a good solution - try to think how to improve it
51          public void mouseClicked(MouseEvent e) {
52          }
53
54          public void mouseEntered(MouseEvent e) {
55          }
56
57          public void mouseExited(MouseEvent e) {
58          }
59
60          public void mouseMoved(MouseEvent e) {
61          }
62
63          public void mouseReleased(MouseEvent e) {
64          }
65
66          public void windowOpened(WindowEvent e) {
67          }
68
69          public void windowClosed(WindowEvent e) {
70          }
71
72          public void windowActivated(WindowEvent e) {
73          }
74
75          public void windowDeactivated(WindowEvent e) {
```

```
76              }
77
78              public void windowIconified(WindowEvent e) {
79              }
80
81              public void windowDeiconified(WindowEvent e) {
82              }
83
84              public void windowClosing(WindowEvent e) {
85                      System.exit(1);
86              }
87
88              /**
89               * Called to paint this panel
90               */
91              public void paint(Graphics painter) {
92                      Dimension size = this.getSize();
93
94                      // paint the back ground
95                      for (int y = 0; y < size.height; y += 2) {
96                              int blue = 255 - (y % 256);
97                              Color color = new Color(128, 128, blue);
98                              painter.setColor(color);
99                              painter.fillRect(0, y, size.width, 2);
100                     }
101
102                     // set the painter to black
103                     painter.setColor(Color.black);
104
105                     // paint the forground drawing
106                     int numberOfPoints = lines.size();
107                     for (int i = 0; i < numberOfPoints; i++) {
108                             Rectangle r = (Rectangle) lines.elementAt(i);
109                             painter.drawLine(r.x, r.y, r.width, r.height);
110                     }
111             }
112
113             public static void main(String[] args) {
114                     FreeHandDrawing drawing = new FreeHandDrawing();
115                     Frame frame = new Frame("Free Hand Drawing Application");
116                     frame.addWindowListener(drawing);
117                     frame.add(drawing);
118                     frame.setSize(300, 300);
119                     frame.setVisible(true);
120             }
121     }
```

```
1   package freeHand;
2
3   import java.applet.Applet;
4   import java.awt.*;
5   import java.awt.event.*;
6   import java.util.Vector;
7
8   /**
9    * A Free Hand drawing application. An improved version
10   * uses listeners and reduces the filckers in the display
11   */
12  public class FreeHandDrawingA extends Applet {
13
14          protected Vector lines;
15
16          protected int x;
17
18          protected int y;
19
20          /**
21           * Constructs a new FreeHandDrawing
22           */
23          public FreeHandDrawingA() {
24
25                  lines = new Vector();
26
27                  // this code creates an anonymos class that listens to mouse events
28                  addMouseListener(new MouseAdapter() {
29
30                          // overriding the method mousePressed to begin drawing
31                          public void mousePressed(MouseEvent e) {
32                                  x = e.getX();
33                                  y = e.getY();
34                          }
35                  });
36
37                  // this code creates an anonymos class that listens to mouse motion e
    vents
38                  addMouseMotionListener(new MouseMotionAdapter() {
39
40                          // overriding the method mouseDragged to draw
41                          public void mouseDragged(MouseEvent e) {
42                                  Rectangle rect = new Rectangle(x, y, e.getX(), e.getY
    ());
43                                  lines.addElement(rect);
44                                  x = e.getX();
45                                  y = e.getY();
46                                  repaint();
47                          }
48                  });
49          }
50
51          /**
52           * Called to paint this pane;
53           */
54          public void paint(Graphics painter) {
55
56                  // paint the back ground
57                  Dimension size = this.getSize();
58                  for (int y = 0; y < size.height; y++) {
59                          int blue = 255 - (y % 256);
60                          Color color = new Color(200, 100, blue);
61                          painter.setColor(color);
62                          painter.fillRect(0, y, size.width, 1);
63                  }
64
65                  // set the painter to black
66                  painter.setColor(Color.black);
67
68                  // draws the forground of the component
69                  int numberOfPoints = lines.size();
70                  for (int i = 0; i < numberOfPoints; i++) {
71                          Rectangle r = (Rectangle) lines.elementAt(i);
72                          painter.drawLine(r.x, r.y, r.width, r.height);
73                  }
```

```
74              }
75
76              // first improvement
77              // we override the method update so that it only calls paint without
78              // clearing the drawing rectangle, which is the main cause of flickering
79              public void update(Graphics painter) {
80                      paint(painter);
81              }
82
83              public static void main(String[] args) {
84                      FreeHandDrawingA drawing = new FreeHandDrawingA();
85                      Frame frame = new Frame("Free Hand Drawing Application");
86                      frame.addWindowListener(new WindowAdapter() {
87                              public void windowClosing(WindowEvent event) {
88                                      System.exit(0);
89                              }
90                      });
91                      frame.add(drawing);
92                      frame.setSize(300, 300);
93                      frame.setVisible(true);
94              }
95      }
```

```
1   package freeHand;
2
3   import java.applet.Applet;
4   import java.awt.*;
5   import java.awt.event.*;
6   import java.util.Vector;
7
8   /**
9    * A Free Hand drawing application. An improved version
10   * uses listeners and reduces the filckers in the display even more
11   */
12  public class FreeHandDrawingB extends Applet {
13
14          protected Vector lines;
15
16          protected int x;
17
18          protected int y;
19
20          /**
21           * Constructs a new FreeHandDrawing
22           */
23          public FreeHandDrawingB() {
24
25                  lines = new Vector();
26
27                  // this code creates an anonymos class that listens to mouse events
28                  addMouseListener(new MouseAdapter() {
29
30                          // overriding the method mousePressed to begin drawing
31                          public void mousePressed(MouseEvent e) {
32                                  x = e.getX();
33                                  y = e.getY();
34                          }
35                  });
36
37                  // this code creates an anonymos class that listens to mouse motion e
    vents
38                  addMouseMotionListener(new MouseMotionAdapter() {
39
40                          // overriding the method mouseDragged to draw
41                          public void mouseDragged(MouseEvent e) {
42                                  Rectangle rect = new Rectangle(x, y, e.getX(), e.getY
    ());
43                                  lines.addElement(rect);
44                                  x = e.getX();
45                                  y = e.getY();
46                                  repaint();
47                          }
48                  });
49          }
50
51          /**
52           * Called to paint this pane;
53           */
54          public void paint(Graphics painter) {
55                  paintBackground(painter);
56                  paintForeground(painter);
57          }
58
59          // first improvement
60          // we override the method update so that it only calls paint without
61          // clearing the drawing rectangle, which is the main cause of flickering
62          public void update(Graphics painter) {
63                  paintForeground(painter);
64          }
65
66          // paints only the back ground - will be called in the awt event of exposure
67          private void paintBackground(Graphics painter) {
68                  Dimension size = this.getSize();
69                  for (int y = 0; y < size.height; y += 2) {
70                          int blue = 255 - (y % 256);
71                          Color color = new Color(128, 128, blue);
72                          painter.setColor(color);
73                          painter.fillRect(0, y, size.width, 2);
```

```
74                  }
75          }
76
77          // paints only the forground - will be called in the program triggered event
78          // of repaint by the update method
79          private void paintForeground(Graphics painter) {
80                  painter.setColor(Color.black);
81                  int numberOfPoints = lines.size();
82                  for (int i = 0; i < numberOfPoints; i++) {
83                          Rectangle r = (Rectangle) lines.elementAt(i);
84                          painter.drawLine(r.x, r.y, r.width, r.height);
85                  }
86          }
87
88          public static void main(String[] args) {
89                  FreeHandDrawingB drawing = new FreeHandDrawingB();
90                  Frame frame = new Frame("Free Hand Drawing Application");
91                  frame.addWindowListener(new WindowAdapter() {
92                          public void windowClosing(WindowEvent event) {
93                                  System.exit(0);
94                          }
95                  });
96                  frame.add(drawing);
97                  frame.setSize(300, 300);
98                  frame.setVisible(true);
99          }
100 }
101
```