Appendix A Startup Files

Applets and Applications

A program can be started up by a web browser (in which case the program is called an applet), or it can be started up independently (in which case it is called an application). Once a program is started up, the differences between applets and applications are minor. All of the programs in the text can be run as either applets or applications. The programs in chapter 7, however, create files and require security clearance to run as applets; they are most easily run as applications.

Applets, Applications & Security

Programs that read from or write to files require security clearance to run as applets. Attempts to read or write files from applets generally result in security violation exceptions. It is much easier to run such programs as applications.

Startup Files

This appendix contains:

A list of (and in most cases, a listing of the contents of) startup files suitable for use with the programs in this book. Appropriate startup files are listed in 2 categories.

1. Applet and application startup files for the 'static' programs of chapters 1 & 2.
2. Applet and application startup files for the mouse & keyboard using programs of chapters 3 and up.

```
import java.io.*;
```

Starting 'Static' Programs — Chapters 1-2

Starting as an Applet

You must have an 'HTML' document (web page) to start a program as an applet. The following is the contents of a minimal HTML document to start an applet named *StaticProgram*.

```
<applet  code = StaticProgram  width = 400  height = 300> </applet>
```

The line above starts a program with a display window 400 pixels wide and 300 pixels high. Some Java development systems may store your program in such a way that additional information is required. Check the documentation with your Java development system (or ask your instructor).

Starting as an Application

You must have a startup program that puts a window on the computer screen and runs your program in that window. The class *PanelApplication* describes such a program. This class is listed on the next page.

Application Starter Program — PanelApplication

PanelApplication is a minimal application that displays a *Panel*. Since an *Applet* is a *Panel, PanelApplication* can be used to display the *Applets* that appear in chapters 1 & 2.

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

public class PanelApplication implements WindowListener
   {
   public static final int PANEL_WIDTH = 400;     // <-- put initial window size here
   public static final int PANEL_HEIGHT = 300;

   private static PanelApplication panelApplication;
   private Panel panel;
   private Frame frame;

   public static void main(String args[])
      {
      panelApplication = new PanelApplication();
      }

   private PanelApplication()
      {
      panel = new MyPanel();   // <-- your program name here
      panel.setLayout(null);
      panel.setSize(PANEL_WIDTH, PANEL_HEIGHT);

      frame = new Frame();
      frame.addWindowListener(this);
      frame.setTitle("My Program");
      frame.setLayout(new FlowLayout());
      frame.add(panel);
      frame.pack();
      frame.setVisible(true);
      if (panel instanceof KeyListener)
         frame.addKeyListener((KeyListener)panel);
      }

   public void windowActivated      (WindowEvent e) {}
   public void windowClosed         (WindowEvent e) {}
   public void windowClosing        (WindowEvent e)
      {
      frame.dispose();
      System.exit(0);
      }
   public void windowDeactivated    (WindowEvent e) {}
   public void windowDeiconified    (WindowEvent e) {}
   public void windowIconified      (WindowEvent e) {}
   public void windowOpened         (WindowEvent e) {}
   }
```

Starting 'Dynamic' Programs — Chapters 3 and Up

Starting up a program that responds to mouse and keyboard

The programs in chapters 3 through 8 and 14 of this book are built on the class: *EventPanel*. *EventPanel* is built upon the standard Java class *Panel*. *EventPanel* contains 13 fairly dense lines of code concerning the handling of mouse and keyboard events. These lines are required by the Java 1.1 event model.

EventPanel is used as a means of reducing clutter in the programs by hiding code that is common to all programs. The full text of *EventPanel* appears on the next page.

Starting as an Applet

The items required are:

An 'HTML' file	starts up *PanelApplet.class*
PanelApplet	starts up your program, must contain the name of your program
EventPanel	use unchanged, may be supplied as *EventPanel.class*
Your Program	extends *EventPanel*

The 'HTML' file is the same as on page A–1 except that the code option must be changed to *PanelApplet*.

The panel name in *PanelApplet.java* must be the name of your program. The full text of *PanelApplet* appears on the next page.

Starting as an Application

The items required are:

PanelApplication	starts up your program, must contain the name of your program
EventPanel	use unchanged, may be supplied as *EventPanel.class*
Your Program	extends *EventPanel*

The panel name in *PanelApplication.java* must be that of your program. The full text of *PanelApplication* appears on page A–2.

Applet Starter Program — PanelApplet

PanelApplet is a minimal applet that displays a *Panel*.

```
import java.awt.*;
import java.applet.Applet;

public class PanelApplet extends Applet
   {
   private Panel panel;

   public void init()
      {
      setLayout(null);
      panel = new MyPanel();     // <-- your program name must go here
      add(panel);
      panel.setSize(getSize().width, getSize().height);
      panel.setLocation(0,0);
      panel.setVisible(true);
      panel.requestFocus();
      if (panel instanceof KeyListener)
         addKeyListener(panel);
      }
   }
```

Mouse and Keyboard Event Director — EventPanel

EventPanel directs mouse and keyboard events to itself and hence to any object that extends it.

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

public class EventPanel extends Panel
                        implements MouseListener, MouseMotionListener, KeyListener
   {

   public EventPanel()
      {
      addMouseListener(this);
      addMouseMotionListener(this);
      addKeyListener(this);
      }

   // Mouse Events
   public void mouseClicked         (MouseEvent e) {}
   public void mouseEntered         (MouseEvent e) {}
   public void mouseExited          (MouseEvent e) {}
   public void mousePressed         (MouseEvent e) {}
   public void mouseReleased        (MouseEvent e) {}

   // Mouse Motion Events
   public void mouseDragged         (MouseEvent e) {}
   public void mouseMoved           (MouseEvent e) {}

   // Key Events
   public void keyPressed           (KeyEvent e) {}
   public void keyReleased          (KeyEvent e) {}
   public void keyTyped             (KeyEvent e) {}
   }
```