```java
// Fig. 16.16: RandomCharacters.java
// Class RandomCharacters demonstrates the Runnable interface
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class RandomCharacters extends JApplet implements ActionListener {
   private String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   private final static int SIZE = 3;
   private JLabel outputs[];
   private JCheckBox checkboxes[];
   private Thread threads[];
   private boolean suspended[];

   // set up GUI and arrays
   public void init()
   {
      outputs = new JLabel[ SIZE ];
      checkboxes = new JCheckBox[ SIZE ];
      threads = new Thread[ SIZE ];
      suspended = new boolean[ SIZE ];

      Container container = getContentPane();
      container.setLayout( new GridLayout( SIZE, 2, 5, 5 ) );

```

RandomCharacters.java

```
26         // create GUI components, register listeners and attach
27         // components to content pane
28         for ( int count = 0; count < SIZE; count++ ) {
29            outputs[ count ] = new JLabel();
30            outputs[ count ].setBackground( Color.GREEN );
31            outputs[ count ].setOpaque( true );
32            container.add( outputs[ count ] );
33
34            checkboxes[ count ] = new JCheckBox( "Suspended" );
35            checkboxes[ count ].addActionListener( this );
36            container.add( checkboxes[ count ] );
37         }
38
39      } // end method init
40
41      // create and start threads each time start is called (i.e., after
42      // init and when user revists Web page containing this applet)
43      public void start()
44      {
45         for ( int count = 0; count < threads.length; count++ ) {
46
47            // create Thread; initialize object that implements Runnable
48            threads[ count ] =
49               new Thread( new RunnableObject(), "Thread " + ( count + 1 ) );
50
51            threads[ count ].start(); // begin executing Thread
52         }
```

Outline

RandomCharacters.java

Line 43

Lines 48-49

Applet start method

Create three `Thread` objects and initialize each with a `RunnableObject`

Call thread start method

```
53     }
54
55     // determine thread location in threads array
56     private int getIndex( Thread current )
57     {
58        for ( int count = 0; count < threads.length; count++ )
59           if ( current == threads[ count ] )
60              return count;
61
62        return -1;
63     }
64
65     // called when user switches Web pages; stops all threads
66     public synchronized void stop()
67     {
68        // set references to null to terminate each thread's run method
69        for ( int count = 0; count < threads.length; count++ )
70           threads[ count ] = null;
71
72        notifyAll(); // notify all waiting threads, so they can terminate
73     }
74
75     // handle button events
76     public synchronized void actionPerformed( ActionEvent event )
77     {
```

Method `stop` stops all threads

Set thread references in array `threads` to

Invoke method `notifyAll` to ready waiting threads

Outline

RandomCharacters.java

Line 66

Line 70

Line 72

```
78         for ( int count = 0; count < checkboxes.length; count++ ) {
79
80            if ( event.getSource() == checkboxes[ count ] ) {
81               suspended[ count ] = !suspended[ count ];
82
83               // change label color on suspend/resume
84               outputs[ count ].setBackground(
85                  suspended[ count ] ? Color.RED : Color.GREEN );
86
87               // if thread resumed, make sure it starts executing
88               if ( !suspended[ count ] )
89                  notifyAll();
90
91               return;
92            }
93         }
94
95      } // end method actionPerformed
96
97      // private inner class that implements Runnable to control threads
98      private class RunnableObject implements Runnable {
99
100        // place random characters in GUI, variables currentThread and
101        // index are final so can be used in an anonymous inner class
102        public void run()
103        {
```

Outline

RandomCharacter

Line 89

Line 98

Line 102

Toggle `boolean` value in array `suspended`

Call `notifyAll` to start ready threads

Class `RunnableObject` implements `Runnable` interface

Declare method `run`

```
104        // get reference to executing thread
105        final Thread currentThread = Thread.currentThread();
106
107        // determine thread's position in array
108        final int index = getIndex( currentThread );
109
110        // loop condition determines when thread should stop; loop
111        // terminates when reference threads[ index ] becomes null
112        while ( threads[ index ] == currentThread ) {
113
114           // sleep from 0 to 1 second
115           try {
116              Thread.sleep( ( int ) ( Math.random() * 1000 ) );
117
118              // determine whether thread should suspend execution;
119              // synchronize on RandomCharacters applet object
120              synchronized( RandomCharacters.this ) {
121
122                 while ( suspended[ index ] &&
123                    threads[ index ] == currentThread ) {
124
125                    // temporarily suspend thread execution
126                    RandomCharacters.this.wait();
127                 }
128              } // end synchronized statement
```

Outline

RandomCharacters.java

Line 126

The `while` loop executes as long as the `index` of array `threads` equals `currentThread`

The `synchronized` block helps suspend currently executing thread

Invoke method `wait` on applet to place thread in waiting state

```java
129
130             } // end try
131
132             // if thread interrupted during wait/sleep, print stack trace
133             catch ( InterruptedException exception ) {
134                exception.printStackTrace();
135             }
136
137             // display character on corresponding JLabel
138             SwingUtilities.invokeLater(
139                new Runnable() {
140
141                   // pick random character and display it
142                   public void run()
143                   {
144                      char displayChar =
145                         alphabet.charAt( ( int ) ( Math.random() * 26 ) );
146
147                      outputs[ index ].setText(
148                         currentThread.getName()  + ": " + displayChar );
149                   }
150
151                } // end inner class
152
153             ); // end call to SwingUtilities.invokeLater
```

Anonymous inner class implements `Runnable` interface

Outline

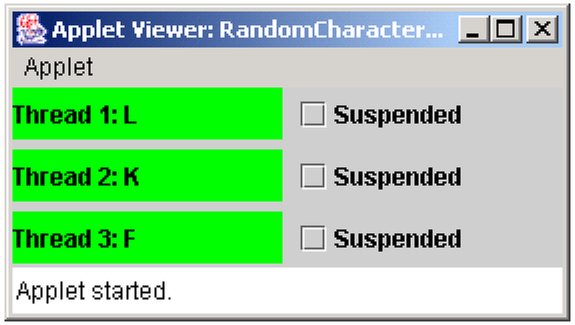
RandomCharacters.java

Line 139

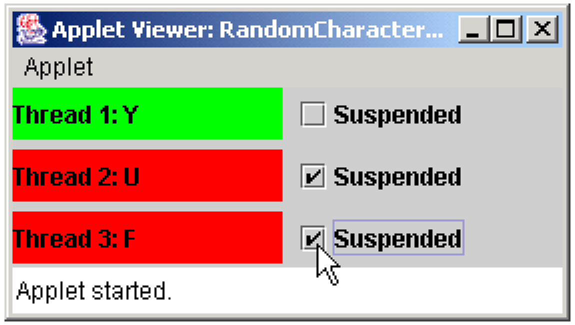
```java

            } // end while

            System.err.println( currentThread.getName() + " terminating" );

         } // end method run

      } // end private inner class RunnableObject

} // end class RandomCharacters
```

Outline

RandomCharacters.java



```
Thread 1: L    ☐ Suspended
Thread 2: K    ☐ Suspended
Thread 3: F    ☐ Suspended
Applet started.
```



```
Thread 1: Y    ☐ Suspended
Thread 2: U    ☑ Suspended
Thread 3: F    ☑ Suspended
Applet started.
```