Threads/spawning.py (Page 1 of 1)

```
#!/usr/bin/env python

#---------------------------------------------------------------------------
# spawning.py
# Author: Bob Dondero
#---------------------------------------------------------------------------

import threading

#---------------------------------------------------------------------------

class PrinterThread (threading.Thread):

    def __init__(self, color):
        threading.Thread.__init__(self)
        self._color = color

    def run(self):
        for i in range(5):
            print(self._color)
        print(self._color + ' thread terminated')

#---------------------------------------------------------------------------

def main():

    blueThread = PrinterThread('blue')
    redThread = PrinterThread('red')

    blueThread.start()
    redThread.start()

    print('main thread terminated')

#---------------------------------------------------------------------------

if __name__ == '__main__':
    main()
```


Threads/Spawning1.java (Page 1 of 1)

```
//---------------------------------------------------------------------------
// Spawning.java
// Author: Bob Dondero
//---------------------------------------------------------------------------

class PrinterThread extends Thread
{
   private String color;

   public PrinterThread(String color) { this.color = color; }

   public void run()
   {
      for (int i = 0; i < 5; i++)
         System.out.println(color);
      System.out.println(color + " thread terminated");
   }
}

//---------------------------------------------------------------------------

public class Spawning1
{
   public static void main(String[] args)
   {
      PrinterThread blueThread = new PrinterThread("blue");
      PrinterThread redThread = new PrinterThread("red");

      blueThread.start();
      redThread.start();

      System.out.println("main thread terminated");
   }
}
```

Threads/Spawning2.java (Page 1 of 1)

```
 1: //-------------------------------------------------------------------------
 2: // Spawning2.java
 3: // Author: Bob Dondero
 4: //-------------------------------------------------------------------------
 5:
 6: class PrinterRunnable implements Runnable
 7: {
 8:    private String color;
 9:
10:    public PrinterRunnable(String color) { this.color = color; }
11:
12:    public void run()
13:    {
14:       for (int i = 0; i < 5; i++)
15:          System.out.println(color);
16:       System.out.println(color + " thread terminated");
17:    }
18: }
19:
20: //-------------------------------------------------------------------------
21:
22: public class Spawning2
23: {
24:    public static void main(String[] args)
25:    {
26:       Runnable bluePrinterRunnable = new PrinterRunnable("blue");
27:       Runnable redPrinterRunnable = new PrinterRunnable("red");
28:
29:       Thread blueThread = new Thread(bluePrinterRunnable);
30:       Thread redThread = new Thread(redPrinterRunnable);
31:
32:       blueThread.start();
33:       redThread.start();
34:
35:       System.out.println("main thread terminated");
36:    }
37: }
38:
```

Threads/Spawning3.java (Page 1 of 1)

```
 1: //-------------------------------------------------------------------------
 2: // Spawning3.java
 3: // Author: Bob Dondero
 4: //-------------------------------------------------------------------------
 5:
 6: public class Spawning3
 7: {
 8:    public static void main(String[] args)
 9:    {
10:       Thread blueThread = new Thread(() -> {
11:          for (int i = 0; i < 5; i++)
12:             System.out.println("blue");
13:          System.out.println("blue thread terminated");
14:       });
15:
16:       Thread redThread = new Thread(() -> {
17:          for (int i = 0; i < 5; i++)
18:             System.out.println("red");
19:          System.out.println("red thread terminated");
20:       });
21:
22:       blueThread.start();
23:       redThread.start();
24:
25:       System.out.println("main thread terminated");
26:    }
27: }
28:
```

PennyJson/penny.py (Page 1 of 1)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# penny.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import os
import time
import json
import flask
import dotenv
import database

dotenv.load_dotenv()
_IO_DELAY = int(os.getenv('IO_DELAY', '0'))

#-----------------------------------------------------------------------

app = flask.Flask(__name__)

#-----------------------------------------------------------------------

@app.route('/', methods=['GET'])
@app.route('/index', methods=['GET'])
def index():

    return flask.send_file('index.html')

#-----------------------------------------------------------------------

@app.route('/searchresults', methods=['GET'])
def search_results():

    author = flask.request.args.get('author')
    if author is None:
        author = ''
    author = author.strip()

    # Simulate a slow database.
    time.sleep(_IO_DELAY)

    if author == '':
        books = []
    else:
        books = database.get_books(author) # Exception handling omitted

    json_doc = json.dumps(books)
    response = flask.make_response(json_doc)
    response.headers['Content-Type'] = 'application/json'
    return response
```

blank (Page 1 of 1)

```
This page is intentionally blank.
```

PennySwing1/runclient (Page 1 of 1)

```
#!/usr/bin/env bash

#-----------------------------------------------------------------------
# runclient
# Author: Bob Dondero
#-----------------------------------------------------------------------

if [ $# -ne 1 ]
then
   echo "usage: runserver serverURL"
   exit 1
fi

mvn clean
mvn compile
mvn exec:java -Dexec.mainClass="edu.princeton.penny.PennyDesktop" \
   -Dexec.args="$@"
```

PennySwing1/runclient.bat (Page 1 of 1)

```
REM -------------------------------------------------------------------------
REM runclient.bat
REM Author: Bob Dondero
REM -------------------------------------------------------------------------

mvn clean
mvn compile
mvn exec:java -Dexec.mainClass="edu.princeton.penny.PennyDesktop" \
   -Dexec.args="%1"
```

PennySwing1/pom.xml (Page 1 of 1)

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>edu.princeton.penny</groupId>
    <artifactId>PennyDesktop</artifactId>
    <version>1.0</version>
    <name>PennyDesktop</name>

    <properties>
       <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
       <maven.compiler.source>21</maven.compiler.source>
       <maven.compiler.target>21</maven.compiler.target>
    </properties>

    <dependencies>
       <dependency>
           <groupId>org.json</groupId>
           <artifactId>json</artifactId>
           <version>20250107</version>
       </dependency>
    </dependencies>

</project>
```

PennySwing1/src/main/java/edu/princeton/penny/PennyDesktop.java (Page 1

```
//---------------------------------------------------------------------------
// PennyDesktop.java
// Author: Bob Dondero
//---------------------------------------------------------------------------

package edu.princeton.penny;

import javax.swing.SwingUtilities;

//---------------------------------------------------------------------------

public class PennyDesktop
{
   public static void main(String[] args)
   {
      String serverURL = args[0];
      SwingUtilities.invokeLater(new PennyDesktopRunnable(serverURL));
   }
}
```

```
 1: //-------------------------------------------------------------------------
 2: // PennyDesktopRunnable.java
 3: // Author: Bob Dondero
 4: //-------------------------------------------------------------------------
 5:
 6: package edu.princeton.penny;
 7:
 8: import javax.swing.JLabel;
 9: import javax.swing.JButton;
10: import javax.swing.JPanel;
11: import javax.swing.JFrame;
12: import javax.swing.JTextField;
13: import javax.swing.DefaultListModel;
14: import javax.swing.JList;
15: import java.awt.BorderLayout;
16: import java.awt.Toolkit;
17: import java.awt.Dimension;
18: import java.awt.event.ActionListener;
19: import java.awt.event.ActionEvent;
20: import java.io.BufferedReader;
21: import java.io.IOException;
22: import java.io.InputStreamReader;
23: import java.net.URL;
24: import java.net.HttpURLConnection;
25: import java.util.ArrayList;
26: import org.json.JSONObject;
27: import org.json.JSONArray;
28:
29: //-------------------------------------------------------------------------
30:
31: public class PennyDesktopRunnable implements Runnable
32: {
33:    private String serverURL;
34:    private JTextField authorTextField = new JTextField("");
35:    private DefaultListModel bookListModel = new DefaultListModel();
36:
37:    public PennyDesktopRunnable(String serverURL)
38:    {
39:       this.serverURL = serverURL;
40:    }
41:
42:    //-------------------------------------------------------------------------
43:
44:    private class AuthorActionListener implements ActionListener
45:    {
46:       public void actionPerformed(ActionEvent event)
47:       {
48:          String author = authorTextField.getText();
49:
50:          try
51:          {
52:             URL url = new URL(serverURL +
53:                "/searchresults?author=" + author);
54:             HttpURLConnection con =
55:                (HttpURLConnection) url.openConnection();
56:             con.setRequestMethod("GET");
57:             con.setDoInput(true);
58:             con.connect();
59:
60:             int responseCode = con.getResponseCode();
61:             if (responseCode != HttpURLConnection.HTTP_OK)
62:                throw new IOException(
63:                   "HTTP error code: " + responseCode);
64:             BufferedReader in =
65:                new BufferedReader(
```



```
66:                   new InputStreamReader(con.getInputStream()));
67:             String jsonDoc = in.readLine();
68:             in.close();
69:
70:             ArrayList<String> bookList = new ArrayList<String>();
71:             JSONArray jsonArray = new JSONArray(jsonDoc);
72:             for (int i = 0; i < jsonArray.length(); i++)
73:             {
74:                JSONObject jsonObject = jsonArray.getJSONObject(i);
75:                bookList.add(
76:                   jsonObject.getString("isbn")
77:                   + ": " + jsonObject.getString("author")
78:                   + ": " + jsonObject.getString("title")
79:                );
80:             }
81:
82:             bookListModel.removeAllElements();
83:             for (String book: bookList)
84:                bookListModel.addElement(book);
85:          }
86:
87:          catch (Exception e)
88:          {
89:             bookListModel.removeAllElements();
90:             bookListModel.addElement(e.toString());
91:          }
92:       }
93:    }
94:
95:    //-------------------------------------------------------------------------
96:
97:    public void run()
98:    {
99:       JLabel authorLabel = new JLabel("Author:  ", JLabel.RIGHT);
100:      JButton submitButton = new JButton("Submit");
101:      JList bookList = new JList(bookListModel);
102:
103:      JPanel inputPanel = new JPanel();
104:      inputPanel.setLayout(new BorderLayout());
105:      inputPanel.add("West", authorLabel);
106:      inputPanel.add("Center", authorTextField);
107:      inputPanel.add("East", submitButton);
108:
109:      JFrame frame = new JFrame();
110:      frame.setTitle("Penny");
111:      frame.setLayout(new BorderLayout());
112:      frame.add("North", inputPanel);
113:      frame.add("Center", bookList);
114:      Toolkit kit = Toolkit.getDefaultToolkit();
115:      Dimension screenSize = kit.getScreenSize();
116:      int screenWidth = screenSize.width;
117:      int screenHeight = screenSize.height;
118:      frame.setSize(screenWidth/4, screenHeight/4);
119:
120:      AuthorActionListener authorActionListener =
121:         new AuthorActionListener();
122:      submitButton.addActionListener(authorActionListener);
123:
124:      frame.setLocationByPlatform(true);
125:      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
126:      frame.setVisible(true);
127:   }
128: }
```

PennySwing2/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java (Page 1 of 2)

```
//-------------------------------------------------------------------------
// PennyDesktopRunnable.java
// Author: Bob Dondero
//-------------------------------------------------------------------------

package edu.princeton.penny;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.DefaultListModel;
import javax.swing.JList;
import javax.swing.event.DocumentListener;
import javax.swing.event.DocumentEvent;
import java.awt.BorderLayout;
import java.awt.Toolkit;
import java.awt.Dimension;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.HttpURLConnection;
import java.util.ArrayList;
import org.json.JSONObject;
import org.json.JSONArray;

//-------------------------------------------------------------------------

public class PennyDesktopRunnable implements Runnable
{
   private String serverURL;
   private JTextField authorTextField = new JTextField("");
   private DefaultListModel bookListModel = new DefaultListModel();

   //----------------------------------------------------------------------
   public PennyDesktopRunnable(String serverURL)
   {
      this.serverURL = serverURL;
   }
   //----------------------------------------------------------------------

   private class AuthorDocumentListener implements DocumentListener
   {
      private void update()
      {
         String author = authorTextField.getText();

         try
         {
            URL url = new URL(serverURL +
                              "/searchresults?author=" + author);
            HttpURLConnection con =
               (HttpURLConnection) url.openConnection();

            con.setRequestMethod("GET");
            con.setDoInput(true);
            con.connect();
            int responseCode = con.getResponseCode();
            if (responseCode != HttpURLConnection.HTTP_OK)
               throw new IOException(
                  "HTTP error code: " + responseCode);

            BufferedReader in =
               new BufferedReader(
```



```
                  new InputStreamReader(con.getInputStream()));
            String jsonDoc = in.readLine();
            in.close();

            ArrayList<String> bookList = new ArrayList<String>();
            JSONArray jsonArray = new JSONArray(jsonDoc);
            for (int i = 0; i < jsonArray.length(); i++)
            {
               JSONObject jsonObject = jsonArray.getJSONObject(i);
               bookList.add(
                  jsonObject.getString("isbn")
                  + ": " + jsonObject.getString("author")
                  + ": " + jsonObject.getString("title")
               );
            }

            bookListModel.removeAllElements();
            for (String book: bookList)
               bookListModel.addElement(book);
         }
         catch (Exception e)
         {
            bookListModel.removeAllElements();
            bookListModel.addElement(e.toString());
         }
      }

      public void changedUpdate(DocumentEvent event) { update(); }
      public void removeUpdate(DocumentEvent event) { update(); }
      public void insertUpdate(DocumentEvent event) { update(); }
   }

   //----------------------------------------------------------------------

   public void run()
   {
      JLabel authorLabel = new JLabel("Author:  ", JLabel.RIGHT);
      JList bookList = new JList(bookListModel);

      JPanel inputPanel = new JPanel();
      inputPanel.setLayout(new BorderLayout());
      inputPanel.add("West", authorLabel);
      inputPanel.add("Center", authorTextField);

      JFrame frame = new JFrame();
      frame.setTitle("Penny");
      frame.setLayout(new BorderLayout());
      frame.add("North", inputPanel);
      frame.add("Center", bookList);
      Toolkit kit = Toolkit.getDefaultToolkit();
      Dimension screenSize = kit.getScreenSize();
      int screenWidth = screenSize.width;
      int screenHeight = screenSize.height;
      frame.setSize(screenWidth/4, screenHeight/4);

      AuthorDocumentListener authorDocumentListener =
         new AuthorDocumentListener();
      authorTextField.getDocument().
         addDocumentListener(authorDocumentListener);

      frame.setLocationByPlatform(true);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

PennySwing3/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java (Page 1 of 2)

```
 1: //---------------------------------------------------------------------
 2: // PennyDesktopRunnable.java
 3: // Author: Bob Dondero
 4: //---------------------------------------------------------------------
 5:
 6: package edu.princeton.penny;
 7:
 8: import javax.swing.JLabel;
 9: import javax.swing.JPanel;
10: import javax.swing.JFrame;
11: import javax.swing.JTextField;
12: import javax.swing.DefaultListModel;
13: import javax.swing.JList;
14: import javax.swing.event.DocumentListener;
15: import javax.swing.event.DocumentEvent;
16: import java.awt.BorderLayout;
17: import java.awt.Toolkit;
18: import java.awt.Dimension;
19:
20: //---------------------------------------------------------------------
21:
22: class PennyDesktopRunnable implements Runnable
23: {
24:    private String serverURL;
25:    private JTextField authorTextField = new JTextField("");
26:    private DefaultListModel bookListModel = new DefaultListModel();
27:
28:    //------------------------------------------------------------------
29:
30:    public PennyDesktopRunnable(String serverURL)
31:    {
32:       this.serverURL = serverURL;
33:    }
34:
35:    //------------------------------------------------------------------
36:
37:    private class AuthorDocumentListener implements DocumentListener
38:    {
39:       private void update()
40:       {
41:          String author = authorTextField.getText();
42:
43:          WorkerThread workerThread =
44:             new WorkerThread(serverURL, author, bookListModel);
45:          workerThread.start();
46:       }
47:
48:       public void changedUpdate(DocumentEvent event) { update(); }
49:
50:       public void removeUpdate(DocumentEvent event) { update(); }
51:
52:       public void insertUpdate(DocumentEvent event) { update(); }
53:    }
54:
55:    //------------------------------------------------------------------
56:
57:    public void run()
58:    {
59:       JLabel authorLabel = new JLabel("Author:  ", JLabel.RIGHT);
60:       JList bookList = new JList(bookListModel);
61:
62:       JPanel inputPanel = new JPanel();
63:       inputPanel.setLayout(new BorderLayout());
64:       inputPanel.add("West", authorLabel);
65:       inputPanel.add("Center", authorTextField);
```

PennySwing3/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java (Page 2 of 2)

```
66:
67:       JFrame frame = new JFrame();
68:       frame.setTitle("Penny");
69:       frame.setLayout(new BorderLayout());
70:       frame.add("North", inputPanel);
71:       frame.add("Center", bookList);
72:       Toolkit kit = Toolkit.getDefaultToolkit();
73:       Dimension screenSize = kit.getScreenSize();
74:       int screenWidth = screenSize.width;
75:       int screenHeight = screenSize.height;
76:       frame.setSize(screenWidth/4, screenHeight/4);
77:
78:       AuthorDocumentListener authorDocumentListener =
79:          new AuthorDocumentListener();
80:
81:       authorTextField.getDocument().
82:          addDocumentListener(authorDocumentListener);
83:
84:       frame.setLocationByPlatform(true);
85:       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
86:       frame.setVisible(true);
87:    }
88: }
```

PennySwing3/src/main/java/edu/princeton/penny/WorkerThread.java (Page 1 of 2)

```
  1: //-------------------------------------------------------------------------
  2: // WorkerThread.java
  3: // Author: Bob Dondero
  4: //-------------------------------------------------------------------------
  5:
  6: package edu.princeton.penny;
  7:
  8: import javax.swing.DefaultListModel;
  9: import java.io.BufferedReader;
 10: import java.io.IOException;
 11: import java.io.InputStreamReader;
 12: import java.net.URL;
 13: import java.net.HttpURLConnection;
 14: import java.util.ArrayList;
 15: import org.json.JSONObject;
 16: import org.json.JSONArray;
 17:
 18: //-------------------------------------------------------------------------
 19:
 20: public class WorkerThread extends Thread
 21: {
 22:    private String serverURL;
 23:    private String author;
 24:    private DefaultListModel bookListModel;
 25:
 26:    //----------------------------------------------------------------------
 27:
 28:    public WorkerThread(String serverURL, String author,
 29:       DefaultListModel bookListModel)
 30:    {
 31:       this.serverURL = serverURL;
 32:       this.author = author;
 33:       this.bookListModel = bookListModel;
 34:    }
 35:
 36:    //----------------------------------------------------------------------
 37:
 38:    public void run()
 39:    {
 40:       try
 41:       {
 42:          URL url = new URL(serverURL +
 43:                            "/searchresults?author=" + author);
 44:          HttpURLConnection con =
 45:             (HttpURLConnection) url.openConnection();
 46:
 47:          con.setRequestMethod("GET");
 48:          con.setDoInput(true);
 49:          con.connect();
 50:          int responseCode = con.getResponseCode();
 51:          if (responseCode != HttpURLConnection.HTTP_OK)
 52:             throw new IOException(
 53:                "HTTP error code: " + responseCode);
 54:
 55:          BufferedReader in =
 56:             new BufferedReader(
 57:                new InputStreamReader(con.getInputStream()));
 58:          String jsonDoc = in.readLine();
 59:          in.close();
 60:
 61:          ArrayList<String> bookList = new ArrayList<String>();
 62:          JSONArray jsonArray = new JSONArray(jsonDoc);
 63:          for (int i = 0; i < jsonArray.length(); i++)
 64:          {
 65:             JSONObject jsonObject = jsonArray.getJSONObject(i);
```

PennySwing3/src/main/java/edu/princeton/penny/WorkerThread.java (Page 2

```
 66:             bookList.add(
 67:                jsonObject.getString("isbn")
 68:                + ": " + jsonObject.getString("author")
 69:                + ": " + jsonObject.getString("title")
 70:             );
 71:          }
 72:
 73:          bookListModel.removeAllElements();
 74:          for (String book: bookList)
 75:             bookListModel.addElement(book);
 76:       }
 77:       catch (Exception e)
 78:       {
 79:          bookListModel.removeAllElements();
 80:          bookListModel.addElement(e.toString());
 81:       }
 82:    }
 83: }
```

PennySwing4/src/main/java/edu/princeton/penny/WorkerThread.java (Page 1 of 2)

```
1: //-------------------------------------------------------------------------
2: // WorkerThread.java
3: // Author: Bob Dondero
4: //-------------------------------------------------------------------------
5:
6: package edu.princeton.penny;
7:
8: import javax.swing.DefaultListModel;
9: import javax.swing.SwingUtilities;
10: import java.io.BufferedReader;
11: import java.io.IOException;
12: import java.io.InputStreamReader;
13: import java.net.URL;
14: import java.net.HttpURLConnection;
15: import java.util.ArrayList;
16: import org.json.JSONObject;
17: import org.json.JSONArray;
18:
19: //-------------------------------------------------------------------------
20:
21: class WorkerThread extends Thread
22: {
23:    private String serverURL;
24:    private String author;
25:    private DefaultListModel bookListModel;
26:
27:    //----------------------------------------------------------------------
28:
29:    public WorkerThread(String serverURL, String author,
30:       DefaultListModel bookListModel)
31:    {
32:       this.serverURL = serverURL;
33:       this.author = author;
34:       this.bookListModel = bookListModel;
35:    }
36:
37:    //----------------------------------------------------------------------
38:
39:    public void run()
40:    {
41:       try
42:       {
43:          URL url = new URL(serverURL +
44:                            "/searchresults?author=" + author);
45:          HttpURLConnection con =
46:             (HttpURLConnection) url.openConnection();
47:
48:          con.setRequestMethod("GET");
49:          con.setDoInput(true);
50:          con.connect();
51:          int responseCode = con.getResponseCode();
52:          if (responseCode != HttpURLConnection.HTTP_OK)
53:             throw new IOException(
54:                "HTTP error code: " + responseCode);
55:
56:          BufferedReader in =
57:             new BufferedReader(
58:                new InputStreamReader(con.getInputStream()));
59:          String jsonDoc = in.readLine();
60:          in.close();
61:
62:          ArrayList<String> bookList = new ArrayList<String>();
63:          JSONArray jsonArray = new JSONArray(jsonDoc);
64:          for (int i = 0; i < jsonArray.length(); i++)
65:          {
```

PennySwing4/src/main/java/edu/princeton/penny/WorkerThread.java (Page 2)

```
66:             JSONObject jsonObject = jsonArray.getJSONObject(i);
67:             bookList.add(
68:                jsonObject.getString("isbn")
69:                + ": " + jsonObject.getString("author")
70:                + ": " + jsonObject.getString("title")
71:             );
72:          }
73:
74:          SwingUtilities.invokeLater(
75:             () -> {
76:                bookListModel.removeAllElements();
77:                for (String book: bookList)
78:                   bookListModel.addElement(book);
79:             }
80:          );
81:       }
82:       catch (Exception e)
83:       {
84:          SwingUtilities.invokeLater(
85:             () -> {
86:                bookListModel.removeAllElements();
87:                bookListModel.addElement(e.toString());
88:             }
89:          );
90:       }
91:    }
92: }
```

PennySwing5/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java (Page 1 of 2)

```
//-----------------------------------------------------------------------
// PennyDesktopRunnable.java
// Author: Bob Dondero
//-----------------------------------------------------------------------

package edu.princeton.penny;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.DefaultListModel;
import javax.swing.JList;
import javax.swing.event.DocumentListener;
import javax.swing.event.DocumentEvent;
import java.awt.BorderLayout;
import java.awt.Toolkit;
import java.awt.Dimension;

//-----------------------------------------------------------------------

public class PennyDesktopRunnable implements Runnable
{
   private String serverURL;
   private JTextField authorTextField = new JTextField("");
   private DefaultListModel bookListModel = new DefaultListModel();

   //--------------------------------------------------------------------

   public PennyDesktopRunnable(String serverURL)
   {
      this.serverURL = serverURL;
   }

   //--------------------------------------------------------------------

   private class AuthorDocumentListener implements DocumentListener
   {
      private WorkerThread workerThread = null;

      private void update()
      {
         String author = authorTextField.getText();
         if (workerThread != null)
            workerThread.setShouldStop();
         workerThread =
            new WorkerThread(serverURL, author, bookListModel);
         workerThread.start();
      }

      public void changedUpdate(DocumentEvent event) { update(); }

      public void removeUpdate(DocumentEvent event) { update(); }

      public void insertUpdate(DocumentEvent event) { update(); }
   }

   //--------------------------------------------------------------------

   public void run()
   {
      JLabel authorLabel = new JLabel("Author:  ", JLabel.RIGHT);
      JList bookList = new JList(bookListModel);

      JPanel inputPanel = new JPanel();
      inputPanel.setLayout(new BorderLayout());
      inputPanel.add("West", authorLabel);
      inputPanel.add("Center", authorTextField);

      JFrame frame = new JFrame();
      frame.setTitle("Penny");
      frame.setLayout(new BorderLayout());
      frame.add("North", inputPanel);
      frame.add("Center", bookList);
      Toolkit kit = Toolkit.getDefaultToolkit();
      Dimension screenSize = kit.getScreenSize();
      int screenWidth = screenSize.width;
      int screenHeight = screenSize.height;
      frame.setSize(screenWidth/4, screenHeight/4);

      AuthorDocumentListener authorDocumentListener =
         new AuthorDocumentListener();

      authorTextField.getDocument().
         addDocumentListener(authorDocumentListener);

      frame.setLocationByPlatform(true);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

PennySwing5/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java (Page 1 of 2)

PennySwing5/src/main/java/edu/princeton/penny/WorkerThread.java (Page 1 of 2)

```
//-----------------------------------------------------------------------
// WorkerThread.java
// Author: Bob Dondero
//-----------------------------------------------------------------------

package edu.princeton.penny;

import javax.swing.DefaultListModel;
import javax.swing.SwingUtilities;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.HttpURLConnection;
import java.util.ArrayList;
import org.json.JSONObject;
import org.json.JSONArray;

//-----------------------------------------------------------------------

public class WorkerThread extends Thread
{
   private String serverURL;
   private String author;
   private DefaultListModel bookListModel;
   private Boolean shouldStop = false;

   //--------------------------------------------------------------------

   public WorkerThread(String serverURL, String author,
      DefaultListModel bookListModel)
   {
      this.serverURL = serverURL;
      this.author = author;
      this.bookListModel = bookListModel;
   }

   //--------------------------------------------------------------------

   public void setShouldStop()
   {
      shouldStop = true;
   }

   //--------------------------------------------------------------------

   public void run()
   {
      try
      {
         URL url = new URL(serverURL +
                           "/searchresults?author=" + author);
         HttpURLConnection con =
            (HttpURLConnection) url.openConnection();

         con.setRequestMethod("GET");
         con.setDoInput(true);
         con.connect();
         int responseCode = con.getResponseCode();
         if (responseCode != HttpURLConnection.HTTP_OK)
            throw new IOException(
               "HTTP error code: " + responseCode);

         BufferedReader in =
            new BufferedReader(
```

PennySwing5/src/main/java/edu/princeton/penny/WorkerThread.java (Page 2)

```
                new InputStreamReader(con.getInputStream()));
         String jsonDoc = in.readLine();
         in.close();

         ArrayList<String> bookList = new ArrayList<String>();
         JSONArray jsonArray = new JSONArray(jsonDoc);
         for (int i = 0; i < jsonArray.length(); i++)
         {
            JSONObject jsonObject = jsonArray.getJSONObject(i);
            bookList.add(
               jsonObject.getString("isbn")
               + ": " + jsonObject.getString("author")
               + ": " + jsonObject.getString("title")
            );
         }

         SwingUtilities.invokeLater(
            () -> {
               if (! shouldStop)
               {
                  bookListModel.removeAllElements();
                  for (String book: bookList)
                     bookListModel.addElement(book);
               }
            }
         );
      }
      catch (Exception e)
      {
         SwingUtilities.invokeLater(
            () -> {
               if (! shouldStop)
               {
                  bookListModel.removeAllElements();
                  bookListModel.addElement(e.toString());
               }
            }
         );
      }
   }
}
```

PennySwing6/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java (Page 1 of 2)

```
//-------------------------------------------------------------------------
// PennyDesktopRunnable.java
// Author: Bob Dondero
//-------------------------------------------------------------------------

package edu.princeton.penny;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.DefaultListModel;
import javax.swing.JList;
import javax.swing.event.DocumentListener;
import javax.swing.event.DocumentEvent;
import java.awt.BorderLayout;
import java.awt.Toolkit;
import java.awt.Dimension;
import java.util.Timer;
import java.util.TimerTask;

//-------------------------------------------------------------------------

public class PennyDesktopRunnable implements Runnable
{
   private String serverURL;
   private JTextField authorTextField = new JTextField("");
   private DefaultListModel bookListModel = new DefaultListModel();

   //----------------------------------------------------------------------

   public PennyDesktopRunnable(String serverURL)
   {
      this.serverURL = serverURL;
   }

   //----------------------------------------------------------------------

   private class AuthorDocumentListener implements DocumentListener
   {
      private WorkerThread workerThread = null;
      private Timer timer = null;

      private class MyTimerTask extends TimerTask
      {
         public void run()
         {
            String author = authorTextField.getText();
            if (workerThread != null)
               workerThread.setShouldStop();
            workerThread =
               new WorkerThread(serverURL, author, bookListModel);
            workerThread.start();
         }
      }

      private void update()
      {
         if (timer != null)
            timer.cancel();
         timer = new Timer();
         timer.schedule(new MyTimerTask(), 500);
      }

      public void changedUpdate(DocumentEvent event) { update(); }
```



```

      public void removeUpdate(DocumentEvent event) { update(); }

      public void insertUpdate(DocumentEvent event) { update(); }
   }

   //----------------------------------------------------------------------

   public void run()
   {
      JLabel authorLabel = new JLabel("Author:  ", JLabel.RIGHT);
      JList bookList = new JList(bookListModel);

      JPanel inputPanel = new JPanel();
      inputPanel.setLayout(new BorderLayout());
      inputPanel.add("West", authorLabel);
      inputPanel.add("Center", authorTextField);

      JFrame frame = new JFrame();
      frame.setTitle("Penny");
      frame.setLayout(new BorderLayout());
      frame.add("North", inputPanel);
      frame.add("Center", bookList);
      Toolkit kit = Toolkit.getDefaultToolkit();
      Dimension screenSize = kit.getScreenSize();
      int screenWidth = screenSize.width;
      int screenHeight = screenSize.height;
      frame.setSize(screenWidth/4, screenHeight/4);

      AuthorDocumentListener authorDocumentListener =
         new AuthorDocumentListener();

      authorTextField.getDocument().
         addDocumentListener(authorDocumentListener);

      frame.setLocationByPlatform(true);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

Threads/race.py (Page 1 of 2)

```
#!/usr/bin/env python

#---------------------------------------------------------------------------
# race.py
# Author: Bob Dondero
#---------------------------------------------------------------------------

import threading

#---------------------------------------------------------------------------

class BankAcct:

    def __init__(self):
        self._balance = 0

    def get_balance(self):
        return self._balance

    def deposit(self, amount):
        temp = self._balance
        temp += amount
        print(temp)
        self._balance = temp

    def withdraw(self, amount):
        temp = self._balance
        temp -= amount
        print(temp)
        self._balance = temp

#---------------------------------------------------------------------------

class DepositThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(10):
            self._bank_acct.deposit(1)

#---------------------------------------------------------------------------

class WithdrawThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(5):
            self._bank_acct.withdraw(2)

#---------------------------------------------------------------------------

def main():

    bank_acct = BankAcct()

    deposit_thread = DepositThread(bank_acct)
    withdraw_thread = WithdrawThread(bank_acct)

    deposit_thread.start()
```

Threads/race.py (Page 2 of 2)

```
    withdraw_thread.start()

    deposit_thread.join()
    withdraw_thread.join()

    print('Final balance:', bank_acct.get_balance())

#---------------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

Threads/Race.java (Page 1 of 2)

```
 1: //-------------------------------------------------------------------------
 2: // Race.java
 3: // Author: Bob Dondero
 4: //-------------------------------------------------------------------------
 5:
 6: class Bank
 7: {
 8:    private int balance = 0;
 9:
10:    public int getBalance()
11:    {
12:       return balance;
13:    }
14:
15:    public void deposit(int amount)
16:    {
17:       int temp = balance;
18:       temp += amount;
19:       System.out.println(temp);
20:       balance = temp;
21:    }
22:
23:    public void withdraw(int amount)
24:    {
25:       int temp = balance;
26:       temp -= amount;
27:       System.out.println(balance);
28:       balance = temp;
29:    }
30: }
31:
32: //-------------------------------------------------------------------------
33:
34: class DepositThread extends Thread
35: {
36:    private Bank bank;
37:
38:    public DepositThread(Bank bank)
39:    {
40:       this.bank = bank;
41:    }
42:
43:    public void run()
44:    {
45:       for (int i = 0; i < 10; i++)
46:          bank.deposit(1);
47:    }
48: }
49:
50: //-------------------------------------------------------------------------
51:
52: class WithdrawThread extends Thread
53: {
54:    private Bank bank;
55:
56:    public WithdrawThread(Bank bank)
57:    {
58:       this.bank = bank;
59:    }
60:
61:    public void run()
62:    {
63:       for (int i = 0; i < 5; i++)
64:          bank.withdraw(2);
65:    }
```

Threads/Race.java (Page 2 of 2)

```
66: }
67:
68: //-------------------------------------------------------------------------
69:
70: public class Race
71: {
72:    public static void main(String[] args)
73:    {
74:       Bank bank = new Bank();
75:
76:       Thread depositThread = new DepositThread(bank);
77:       Thread withdrawThread = new WithdrawThread(bank);
78:
79:       depositThread.start();
80:       withdrawThread.start();
81:
82:       try
83:       {
84:          depositThread.join();
85:          withdrawThread.join();
86:       }
87:       catch (InterruptedException e)
88:       {
89:          Thread.currentThread().interrupt();
90:       }
91:
92:       System.out.println("Final balance: " + bank.getBalance());
93:    }
94: }
```

Threads/locking.py (Page 1 of 2)

```
#!/usr/bin/env python

#---------------------------------------------------------------------
# locking.py
# Author: Bob Dondero
#---------------------------------------------------------------------

import threading

#---------------------------------------------------------------------

class BankAcct:

    def __init__(self):
        self._balance = 0
        self._lock = threading.RLock()

    def get_balance(self):
        self._lock.acquire()
        try:
            return self._balance
        finally:
            self._lock.release()

    def deposit(self, amount):
        self._lock.acquire()
        temp = self._balance
        temp += amount
        print(temp)
        self._balance = temp
        self._lock.release()

    def withdraw(self, amount):
        self._lock.acquire()
        temp = self._balance
        temp -= amount
        print(temp)
        self._balance = temp
        self._lock.release()

#---------------------------------------------------------------------

class DepositThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(10):
            self._bank_acct.deposit(1)

#---------------------------------------------------------------------

class WithdrawThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(5):
            self._bank_acct.withdraw(2)

#---------------------------------------------------------------------
```

Threads/locking.py (Page 2 of 2)

```

def main():

    bank_acct = BankAcct()

    deposit_thread = DepositThread(bank_acct)
    withdraw_thread = WithdrawThread(bank_acct)

    deposit_thread.start()
    withdraw_thread.start()

    deposit_thread.join()
    withdraw_thread.join()

    print('Final balance:', bank_acct.get_balance())

#---------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

Threads/Locking.java (Page 1 of 2)

```
 1: //-------------------------------------------------------------------------
 2: // Locking.java
 3: // Author: Bob Dondero
 4: //-------------------------------------------------------------------------
 5:
 6: class Bank
 7: {
 8:    private int balance = 0;
 9:
10:    public synchronized int getBalance()
11:    {
12:       return balance;
13:    }
14:
15:    public synchronized void deposit(int amount)
16:    {
17:       int temp = balance;
18:       temp += amount;
19:       System.out.println(temp);
20:       balance = temp;
21:    }
22:
23:    public synchronized void withdraw(int amount)
24:    {
25:       int temp = balance;
26:       temp -= amount;
27:       System.out.println(temp);
28:       balance = temp;
29:    }
30: }
31:
32: //-------------------------------------------------------------------------
33:
34: class DepositThread extends Thread
35: {
36:    private Bank bank;
37:
38:    public DepositThread(Bank bank)
39:    {
40:       this.bank = bank;
41:    }
42:
43:    public void run()
44:    {
45:       for (int i = 0; i < 10; i++)
46:          bank.deposit(1);
47:    }
48: }
49:
50: //-------------------------------------------------------------------------
51:
52: class WithdrawThread extends Thread
53: {
54:    private Bank bank;
55:
56:    public WithdrawThread(Bank bank)
57:    {
58:       this.bank = bank;
59:    }
60:
61:    public void run()
62:    {
63:       for (int i = 0; i < 5; i++)
64:          bank.withdraw(2);
65:    }
```

Threads/Locking.java (Page 2 of 2)

```
66: }
67:
68: //-------------------------------------------------------------------------
69:
70: public class Locking
71: {
72:    public static void main(String[] args)
73:    {
74:       Bank bank = new Bank();
75:
76:       Thread depositThread = new DepositThread(bank);
77:       Thread withdrawThread = new WithdrawThread(bank);
78:
79:       depositThread.start();
80:       withdrawThread.start();
81:
82:       try
83:       {
84:          depositThread.join();
85:          withdrawThread.join();
86:       }
87:       catch (InterruptedException e)
88:       {
89:          Thread.currentThread().interrupt();
90:       }
91:
92:       System.out.println("Final balance: " + bank.getBalance());
93:    }
94: }
```

Threads/conditions.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# conditions.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import threading

#-----------------------------------------------------------------------

class BankAcct:

    def __init__(self):
        self._balance = 0
        self._lock = threading.RLock()
        self._condition = threading.Condition(self._lock)

    def get_balance(self):
        self._lock.acquire()
        try:
            return self._balance
        finally:
            self._lock.release()

    def deposit(self, amount):
        self._condition.acquire()
        self._balance += amount
        print(self._balance)
        self._condition.notify_all()
        self._condition.release()

    def withdraw(self, amount):
        self._condition.acquire()
        while self._balance < amount:
            self._condition.wait()
        self._balance -= amount
        print(self._balance)
        self._condition.release()

#-----------------------------------------------------------------------

class DepositThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(10):
            self._bank_acct.deposit(1)

#-----------------------------------------------------------------------

class WithdrawThread (threading.Thread):

    def __init__(self, bank_acct):
        threading.Thread.__init__(self)
        self._bank_acct = bank_acct

    def run(self):
        for _ in range(5):
            self._bank_acct.withdraw(2)

#-----------------------------------------------------------------------
```

Threads/conditions.py (Page 2 of 2)

```

def main():
    bank_acct = BankAcct()

    deposit_thread = DepositThread(bank_acct)
    withdraw_thread = WithdrawThread(bank_acct)

    deposit_thread.start()
    withdraw_thread.start()

    deposit_thread.join()
    withdraw_thread.join()

    print('Final balance:', bank_acct.get_balance())

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

Threads/Conditions.java (Page 1 of 2)

```
  1: //-------------------------------------------------------------------------
  2: // Conditions.java
  3: // Author: Bob Dondero
  4: //-------------------------------------------------------------------------
  5:
  6: class Bank
  7: {
  8:    private int balance = 0;
  9:
 10:    public synchronized int getBalance()
 11:    {
 12:       return balance;
 13:    }
 14:
 15:    public synchronized void deposit(int amount)
 16:    {
 17:       balance += amount;
 18:       System.out.println(balance);
 19:       notifyAll();
 20:    }
 21:
 22:    public synchronized void withdraw(int amount)
 23:    {
 24:       try
 25:       {
 26:          while (balance < amount)
 27:             wait();
 28:          balance -= amount;
 29:          System.out.println(balance);
 30:       }
 31:       catch (InterruptedException e)
 32:       {
 33:          Thread.currentThread().interrupt();
 34:       }
 35:    }
 36: }
 37:
 38: //-------------------------------------------------------------------------
 39:
 40: class DepositThread extends Thread
 41: {
 42:    private Bank bank;
 43:
 44:    public DepositThread(Bank bank)
 45:    {
 46:       this.bank = bank;
 47:    }
 48:
 49:    public void run()
 50:    {
 51:       for (int i = 0; i < 10; i++)
 52:          bank.deposit(1);
 53:    }
 54: }
 55:
 56: //-------------------------------------------------------------------------
 57:
 58: class WithdrawThread extends Thread
 59: {
 60:    private Bank bank;
 61:
 62:    public WithdrawThread(Bank bank)
 63:    {
 64:       this.bank = bank;
 65:    }
```

Threads/Conditions.java (Page 2 of 2)

```
 66:
 67:    public void run()
 68:    {
 69:       for (int i = 0; i < 5; i++)
 70:          bank.withdraw(2);
 71:    }
 72: }
 73:
 74: //-------------------------------------------------------------------------
 75:
 76: public class Conditions
 77: {
 78:    public static void main(String[] args)
 79:    {
 80:       Bank bank = new Bank();
 81:
 82:       Thread depositThread = new DepositThread(bank);
 83:       Thread withdrawThread = new WithdrawThread(bank);
 84:
 85:       depositThread.start();
 86:       withdrawThread.start();
 87:
 88:       try
 89:       {
 90:          depositThread.join();
 91:          withdrawThread.join();
 92:       }
 93:       catch (InterruptedException e)
 94:       {
 95:          Thread.currentThread().interrupt();
 96:       }
 97:
 98:       System.out.println("Final balance: " + bank.getBalance());
 99:    }
100: }
```