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
```

PennyCommandLine/penny.py (Page 1 of 1)

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

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

import sys
import json
import urllib.request
import urllib.parse

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

def fetch_books(server_url, author):

    encoded_author = urllib.parse.quote_plus(author)
    request = server_url + '/searchresults?author=' + encoded_author

    with urllib.request.urlopen(request) as flo:
        response = flo.read()
    json_doc = response.decode('utf-8')
    books = json.loads(json_doc)

    return books

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

def write_books(books):

    pattern = '%s: %s: %s'
    for book in books:
        row = pattern % (book['isbn'], book['author'], book['title'])
        print(row)

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

def main():

    if len(sys.argv) != 3:
        print(f'usage: python {sys.argv[0]} serverurl author',
              file=sys.stderr)
        sys.exit(1)

    server_url = sys.argv[1]
    author = sys.argv[2]

    try:
        books = fetch_books(server_url, author)
        write_books(books)

    except Exception as ex:
        print(f'{sys.argv[0]}: {str(ex)}', file=sys.stderr)
        sys.exit(1)

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

PennySwing1/runclient (Page 1 of 1)

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

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

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

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

PennySwing1/pom.xml (Page 1 of 1)

```
 1: <?xml version="1.0" encoding="UTF-8"?>
 2:
 3: <project xmlns="http://maven.apache.org/POM/4.0.0"
 4:    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 5:    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
 6:    http://maven.apache.org/xsd/maven-4.0.0.xsd">
 7:
 8:    <modelVersion>4.0.0</modelVersion>
 9:
10:    <groupId>edu.princeton.penny</groupId>
11:    <artifactId>PennyDesktop</artifactId>
12:    <version>1.0</version>
13:    <name>PennyDesktop</name>
14:
15:    <properties>
16:       <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
17:       <maven.compiler.source>21</maven.compiler.source>
18:       <maven.compiler.target>21</maven.compiler.target>
19:    </properties>
20:
21:    <dependencies>
22:       <dependency>
23:           <groupId>org.json</groupId>
24:           <artifactId>json</artifactId>
25:           <version>20250107</version>
26:       </dependency>
27:    </dependencies>
28:
29: </project>
```

PennySwing1/src/main/java/edu/princeton/penny/PennyDesktop.java (P

```
 1: //---------------------------------------------------------------------
 2: // PennyDesktop.java
 3: // Author: Bob Dondero
 4: //---------------------------------------------------------------------
 5:
 6: package edu.princeton.penny;
 7:
 8: import javax.swing.SwingUtilities;
 9:
10: //---------------------------------------------------------------------
11:
12: public class PennyDesktop {
13:    public static void main(String[] args) {
14:       if (args.length != 1) {
15:          System.err.println("usage: java PennyDesktop serverURL");
16:          System.exit(1);
17:       }
18:
19:       String serverURL = args[0];
20:       SwingUtilities.invokeLater(new PennyDesktopRunnable(serverURL));
21:    }
22: }
```

PennySwing1/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java

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

package edu.princeton.penny;

import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.DefaultListModel;
import javax.swing.JList;
import java.awt.BorderLayout;
import java.awt.Toolkit;
import java.awt.Dimension;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import java.net.URL;
import java.net.HttpURLConnection;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
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<String> bookListModel =
      new DefaultListModel<String>();

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

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

   private class AuthorActionListener implements ActionListener {
      public void actionPerformed(ActionEvent event) {
         String author = authorTextField.getText();

         try {
            String encodedAuthor =
               URLEncoder.encode(author, StandardCharsets.UTF_8);
            URL url = URI.create(serverURL +
               "/searchresults?author=" + encodedAuthor).toURL();
            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 run() {
      JLabel authorLabel = new JLabel("Author:  ", JLabel.RIGHT);
      JButton submitButton = new JButton("Submit");
      JList<String> bookList = new JList<String>(bookListModel);

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

      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);

      AuthorActionListener authorActionListener =
         new AuthorActionListener();
      submitButton.addActionListener(authorActionListener);

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

PennySwing1/src/main/java/edu/princeton/penny/PennyDesktopRunnable

PennySwing2/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java

```
//--------------------------------------------------------------------------
// 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.URI;
import java.net.URL;
import java.net.HttpURLConnection;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
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<String> bookListModel =
      new DefaultListModel<String>();

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

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

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

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

         try {
            String encodedAuthor =
               URLEncoder.encode(author, StandardCharsets.UTF_8);
            URL url = URI.create(serverURL +
               "/searchresults?author=" + encodedAuthor).toURL();
            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);
```

PennySwing2/src/main/java/edu/princeton/penny/PennyDesktopRunnable

```

            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<String> bookList = new JList<String>(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

```
//-----------------------------------------------------------------------
// 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;

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

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

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

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

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

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

         WorkerThread 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<String> bookList = new JList<String>(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

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 javax.swing.SwingUtilities;
10: import java.io.BufferedReader;
11: import java.io.IOException;
12: import java.io.InputStreamReader;
13: import java.net.URI;
14: import java.net.URL;
15: import java.net.HttpURLConnection;
16: import java.net.URLEncoder;
17: import java.nio.charset.StandardCharsets;
18: import java.util.ArrayList;
19: import org.json.JSONObject;
20: import org.json.JSONArray;
21:
22: //---------------------------------------------------------------------
23:
24: class WorkerThread extends Thread {
25:    private String serverURL;
26:    private String author;
27:    private DefaultListModel<String> bookListModel;
28:
29:    //-----------------------------------------------------------------
30:
31:    public WorkerThread(String serverURL, String author,
32:       DefaultListModel<String> bookListModel) {
33:       this.serverURL = serverURL;
34:       this.author = author;
35:       this.bookListModel = bookListModel;
36:    }
37:
38:    //-----------------------------------------------------------------
39:
40:    public void run() {
41:       try {
42:          String encodedAuthor =
43:             URLEncoder.encode(author, StandardCharsets.UTF_8);
44:          URL url = URI.create(serverURL +
45:             "/searchresults?author=" + encodedAuthor).toURL();
46:          HttpURLConnection con =
47:             (HttpURLConnection) url.openConnection();
48:
49:          con.setRequestMethod("GET");
50:          con.setDoInput(true);
51:          con.connect();
52:          int responseCode = con.getResponseCode();
53:          if (responseCode != HttpURLConnection.HTTP_OK)
54:             throw new IOException(
55:                "HTTP error code: " + responseCode);
56:
57:          BufferedReader in =
58:             new BufferedReader(
59:                new InputStreamReader(con.getInputStream()));
60:          String jsonDoc = in.readLine();
61:          in.close();
62:
63:          ArrayList<String> bookList = new ArrayList<String>();
64:          JSONArray jsonArray = new JSONArray(jsonDoc);
65:          for (int i = 0; i < jsonArray.length(); i++) {
```

PennySwing3/src/main/java/edu/princeton/penny/WorkerThread.java (P

```
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:
83:       catch (Exception e) {
84:          SwingUtilities.invokeLater(
85:             () -> {
86:                bookListModel.removeAllElements();
87:                bookListModel.addElement(e.toString());
88:             }
89:          );
90:       }
91:    }
92: }
```

PennySwing4/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java

```
//-------------------------------------------------------------------------
// 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<String> bookListModel =
       new DefaultListModel<String>();

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

   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<String> bookList = new JList<String>(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);
   }
}
```

PennySwing4/src/main/java/edu/princeton/penny/PennyDesktopRunnable

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.URI;
14: import java.net.URL;
15: import java.net.HttpURLConnection;
16: import java.net.URLEncoder;
17: import java.nio.charset.StandardCharsets;
18: import java.util.ArrayList;
19: import org.json.JSONObject;
20: import org.json.JSONArray;
21:
22: //---------------------------------------------------------------------
23:
24: public class WorkerThread extends Thread {
25:    private String serverURL;
26:    private String author;
27:    private DefaultListModel<String> bookListModel;
28:    private Boolean shouldStop = false;
29:
30:    //------------------------------------------------------------------
31:
32:    public WorkerThread(String serverURL, String author,
33:       DefaultListModel<String> bookListModel) {
34:       this.serverURL = serverURL;
35:       this.author = author;
36:       this.bookListModel = bookListModel;
37:    }
38:
39:    //------------------------------------------------------------------
40:
41:    public void setShouldStop() {
42:       shouldStop = true;
43:    }
44:
45:    //------------------------------------------------------------------
46:
47:    public void run() {
48:       try {
49:          String encodedAuthor =
50:             URLEncoder.encode(author, StandardCharsets.UTF_8);
51:          URL url = URI.create(serverURL +
52:             "/searchresults?author=" + encodedAuthor).toURL();
53:          HttpURLConnection con =
54:             (HttpURLConnection) url.openConnection();
55:
56:          con.setRequestMethod("GET");
57:          con.setDoInput(true);
58:          con.connect();
59:          int responseCode = con.getResponseCode();
60:          if (responseCode != HttpURLConnection.HTTP_OK)
61:             throw new IOException(
62:                "HTTP error code: " + responseCode);
63:
64:          BufferedReader in =
65:             new BufferedReader(
```

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

```
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:             JSONObject jsonObject = jsonArray.getJSONObject(i);
74:             bookList.add(
75:                jsonObject.getString("isbn")
76:                + ": " + jsonObject.getString("author")
77:                + ": " + jsonObject.getString("title")
78:             );
79:          }
80:
81:          SwingUtilities.invokeLater(
82:             () -> {
83:                if (! shouldStop)
84:                {
85:                   bookListModel.removeAllElements();
86:                   for (String book: bookList)
87:                      bookListModel.addElement(book);
88:                }
89:             }
90:          );
91:       }
92:
93:       catch (Exception e) {
94:          SwingUtilities.invokeLater(
95:             () -> {
96:                if (! shouldStop)
97:                {
98:                   bookListModel.removeAllElements();
99:                   bookListModel.addElement(e.toString());
100:               }
101:            }
102:         );
103:      }
104:   }
105: }
```

PennySwing5/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java

```
//-------------------------------------------------------------------------
// 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<String> bookListModel =
      new DefaultListModel<String>();

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

   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<String> bookList = new JList<String>(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

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)

```
 1: #!/usr/bin/env python
 2:
 3: #-----------------------------------------------------------------------
 4: # conditions.py
 5: # Author: Bob Dondero
 6: #-----------------------------------------------------------------------
 7:
 8: import threading
 9:
10: #-----------------------------------------------------------------------
11:
12: class BankAcct:
13:
14:     def __init__(self):
15:         self._balance = 0
16:         self._lock = threading.RLock()
17:         self._condition = threading.Condition(self._lock)
18:
19:     def get_balance(self):
20:         self._lock.acquire()
21:         try:
22:             return self._balance
23:         finally:
24:             self._lock.release()
25:
26:     def deposit(self, amount):
27:         self._condition.acquire()
28:         self._balance += amount
29:         print(self._balance)
30:         self._condition.notify_all()
31:         self._condition.release()
32:
33:     def withdraw(self, amount):
34:         self._condition.acquire()
35:         while self._balance < amount:
36:             self._condition.wait()
37:         self._balance -= amount
38:         print(self._balance)
39:         self._condition.release()
40:
41: #-----------------------------------------------------------------------
42:
43: class DepositThread (threading.Thread):
44:
45:     def __init__(self, bank_acct):
46:         threading.Thread.__init__(self)
47:         self._bank_acct = bank_acct
48:
49:     def run(self):
50:         for _ in range(10):
51:             self._bank_acct.deposit(1)
52:
53: #-----------------------------------------------------------------------
54:
55: class WithdrawThread (threading.Thread):
56:
57:     def __init__(self, bank_acct):
58:         threading.Thread.__init__(self)
59:         self._bank_acct = bank_acct
60:
61:     def run(self):
62:         for _ in range(5):
63:             self._bank_acct.withdraw(2)
64:
65: #-----------------------------------------------------------------------
```

Threads/conditions.py (Page 2 of 2)

```
66:
67: def main():
68:     bank_acct = BankAcct()
69:
70:     deposit_thread = DepositThread(bank_acct)
71:     withdraw_thread = WithdrawThread(bank_acct)
72:
73:     deposit_thread.start()
74:     withdraw_thread.start()
75:
76:     deposit_thread.join()
77:     withdraw_thread.join()
78:
79:     print('Final balance:', bank_acct.get_balance())
80:
81: #-----------------------------------------------------------------------
82:
83: if __name__ == '__main__':
84:     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: }
```