

## Threads/spawning.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # spawning.py
5: # Author: Bob Dondero
6: #-----
7:
8: import threading
9:
10: #-----
11:
12: class PrinterThread (threading.Thread):
13:
14:     def __init__(self, color):
15:         threading.Thread.__init__(self)
16:         self._color = color
17:
18:     def run(self):
19:         for i in range(5):
20:             print(self._color)
21:             print(self._color + ' thread terminated')
22:
23: #-----
24:
25: def main():
26:
27:     blueThread = PrinterThread('blue')
28:     redThread = PrinterThread('red')
29:
30:     blueThread.start()
31:     redThread.start()
32:
33:     print('main thread terminated')
34:
35: #-----
36:
37: if __name__ == '__main__':
38:     main()

```

## Threads/Spawning1.java (Page 1 of 1)

```

1: //-----
2: // Spawning.java
3: // Author: Bob Dondero
4: //-----
5:
6: class PrinterThread extends Thread
7: {
8:     private String color;
9:
10:    public PrinterThread(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 Spawning1
23: {
24:     public static void main(String[] args)
25:     {
26:         PrinterThread blueThread = new PrinterThread("blue");
27:         PrinterThread redThread = new PrinterThread("red");
28:
29:         blueThread.start();
30:         redThread.start();
31:
32:         System.out.println("main thread terminated");
33:     }
34: }
35:

```

## 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)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import time
10: import json
11: import flask
12: import dotenv
13: import database
14:
15: dotenv.load_dotenv()
16: _IO_DELAY = int(os.getenv('IO_DELAY', '0'))
17:
18: #-----
19:
20: app = flask.Flask(__name__)
21:
22: #-----
23:
24: @app.route('/', methods=['GET'])
25: @app.route('/index', methods=['GET'])
26: def index():
27:
28:     return flask.send_file('index.html')
29:
30: #-----
31:
32: @app.route('/searchresults', methods=['GET'])
33: def search_results():
34:
35:     author = flask.request.args.get('author')
36:     if author is None:
37:         author = ''
38:     author = author.strip()
39:
40:     # Simulate a slow database.
41:     time.sleep(_IO_DELAY)
42:
43:     if author == '':
44:         books = []
45:     else:
46:         books = database.get_books(author) # Exception handling omitted
47:
48:     json_doc = json.dumps(books)
49:     response = flask.make_response(json_doc)
50:     response.headers['Content-Type'] = 'application/json'
51:     return response

```

## PennyCommandLine/penny.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import json
10: import urllib.request
11: import urllib.parse
12:
13: #-----
14:
15: def fetch_books(server_url, author):
16:
17:     encoded_author = urllib.parse.quote_plus(author)
18:     request = server_url + '/searchresults?author=' + encoded_author
19:
20:     with urllib.request.urlopen(request) as flo:
21:         response = flo.read()
22:         json_doc = response.decode('utf-8')
23:         books = json.loads(json_doc)
24:
25:     return books
26:
27: #-----
28:
29: def write_books(books):
30:
31:     pattern = '%s: %s: %s'
32:     for book in books:
33:         row = pattern % (book['isbn'], book['author'], book['title'])
34:         print(row)
35:
36: #-----
37:
38: def main():
39:
40:     if len(sys.argv) != 3:
41:         print(f'usage: python {sys.argv[0]} serverurl author',
42:               file=sys.stderr)
43:         sys.exit(1)
44:
45:     server_url = sys.argv[1]
46:     author = sys.argv[2]
47:
48:     try:
49:         books = fetch_books(server_url, author)
50:         write_books(books)
51:
52:     except Exception as ex:
53:         print(f'{sys.argv[0]}: {str(ex)}', file=sys.stderr)
54:         sys.exit(1)
55:
56: if __name__ == '__main__':
57:     main()

```

**PennySwing1/runclient (Page 1 of 1)**

```
1: #!/usr/bin/env bash
2:
3: #-----
4: # runclient
5: # Author: Bob Dondero
6: #-----
7:
8: if [ $# -ne 1 ]
9: then
10:   echo "usage: runclient serverURL"
11:   exit 1
12: fi
13:
14: mvn exec:java -Dexec.mainClass="edu.princeton.penny.PennyDesktop" \
15:   -Dexec.args="$@"
```

**PennySwing1/runclient.bat (Page 1 of 1)**

```
1: REM -----
2: REM runclient.bat
3: REM Author: Bob Dondero
4: REM -----
5:
6: mvn exec:java -Dexec.mainClass="edu.princeton.penny.PennyDesktop" \
7:   -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 PennySwing1/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java

```

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.URI;
24: import java.net.URL;
25: import java.net.HttpURLConnection;
26: import java.net.URLEncoder;
27: import java.nio.charset.StandardCharsets;
28: import java.util.ArrayList;
29: import org.json.JSONObject;
30: import org.json.JSONArray;
31:
32: //-----
33:
34: public class PennyDesktopRunnable implements Runnable {
35:     private String serverURL;
36:     private JTextField authorTextField = new JTextField("");
37:     private DefaultListModel<String> bookListModel =
38:         new DefaultListModel<String>();
39:
40:     public PennyDesktopRunnable(String serverURL) {
41:         this.serverURL = serverURL;
42:     }
43:
44:     //-----
45:
46:     private class AuthorActionListener implements ActionListener {
47:         public void actionPerformed(ActionEvent event) {
48:             String author = authorTextField.getText();
49:
50:             try {
51:                 String encodedAuthor =
52:                     URLEncoder.encode(author, StandardCharsets.UTF_8);
53:                 URL url = URI.create(serverURL +
54:                     "/searchresults?author=" + encodedAuthor).toURL();
55:                 HttpURLConnection con =
56:                     (HttpURLConnection) url.openConnection();
57:                 con.setRequestMethod("GET");
58:                 con.setDoInput(true);
59:                 con.connect();
60:
61:                 int responseCode = con.getResponseCode();
62:                 if (responseCode != HttpURLConnection.HTTP_OK)
63:                     throw new IOException(
64:                         "HTTP error code: " + responseCode);
65:                 BufferedReader in =
66:                     new BufferedReader(
67:                         new InputStreamReader(con.getInputStream()));
68:                 String jsonDoc = in.readLine();
69:                 in.close();
70:
71:                 ArrayList<String> bookList = new ArrayList<String>();
72:                 JSONArray jsonArray = new JSONArray(jsonDoc);
73:                 for (int i = 0; i < jsonArray.length(); i++) {
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:                 bookListModel.removeAllElements();
89:                 bookListModel.addElement(e.toString());
90:             }
91:         }
92:     }
93:
94:     //-----
95:
96:     public void run() {
97:         JLabel authorLabel = new JLabel("Author: ", JLabel.RIGHT);
98:         JButton submitButton = new JButton("Submit");
99:         JList<String> bookList = new JList<String>(bookListModel);
100:
101:         JPanel inputPanel = new JPanel();
102:         inputPanel.setLayout(new BorderLayout());
103:         inputPanel.add("West", authorLabel);
104:         inputPanel.add("Center", authorTextField);
105:         inputPanel.add("East", submitButton);
106:
107:         JFrame frame = new JFrame();
108:         frame.setTitle("Penny");
109:         frame.setLayout(new BorderLayout());
110:         frame.add("North", inputPanel);
111:         frame.add("Center", bookList);
112:         Toolkit kit = Toolkit.getDefaultToolkit();
113:         Dimension screenSize = kit.getScreenSize();
114:         int screenWidth = screenSize.width;
115:         int screenHeight = screenSize.height;
116:         frame.setSize(screenWidth/4, screenHeight/4);
117:
118:         AuthorActionListener authorActionListener =
119:             new AuthorActionListener();
120:         submitButton.addActionListener(authorActionListener);
121:
122:         frame.setLocationByPlatform(true);
123:         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
124:         frame.setVisible(true);
125:     }
126: }

```

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

```

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: import java.io.BufferedReader;
20: import java.io.IOException;
21: import java.io.InputStreamReader;
22: import java.net.URI;
23: import java.net.URL;
24: import java.net.HttpURLConnection;
25: import java.net.URLEncoder;
26: import java.nio.charset.StandardCharsets;
27: import java.util.ArrayList;
28: import org.json.JSONObject;
29: import org.json.JSONArray;
30:
31: //-----
32:
33: public class PennyDesktopRunnable implements Runnable {
34:     private String serverURL;
35:     private JTextField authorTextField = new JTextField("");
36:     private DefaultListModel<String> bookListModel =
37:         new DefaultListModel<String>();
38:
39:     //-----
40:
41:     public PennyDesktopRunnable(String serverURL) {
42:         this.serverURL = serverURL;
43:     }
44:
45:     //-----
46:
47:     private class AuthorDocumentListener implements DocumentListener {
48:         private void update() {
49:             String author = authorTextField.getText();
50:
51:             try {
52:                 String encodedAuthor =
53:                     URLEncoder.encode(author, StandardCharsets.UTF_8);
54:                 URL url = URI.create(serverURL +
55:                     "/searchresults?author=" + encodedAuthor).toURL();
56:                 HttpURLConnection con =
57:                     (HttpURLConnection) url.openConnection();
58:
59:                 con.setRequestMethod("GET");
60:                 con.setDoInput(true);
61:                 con.connect();
62:                 int responseCode = con.getResponseCode();
63:                 if (responseCode != HttpURLConnection.HTTP_OK)
64:                     throw new IOException(
65:                         "HTTP error code: " + responseCode);
66:
67:                 BufferedReader in =
68:                     new BufferedReader(
69:                         new InputStreamReader(con.getInputStream()));
70:                 String jsonDoc = in.readLine();
71:                 in.close();
72:
73:                 ArrayList<String> bookList = new ArrayList<String>();
74:                 JSONArray jsonArray = new JSONArray(jsonDoc);
75:                 for (int i = 0; i < jsonArray.length(); i++) {
76:                     JSONObject jsonObject = jsonArray.getJSONObject(i);
77:                     bookList.add(
78:                         jsonObject.getString("isbn")
79:                         + ": " + jsonObject.getString("author")
80:                         + ": " + jsonObject.getString("title")
81:                     );
82:                 }
83:
84:                 bookListModel.removeAllElements();
85:                 for (String book: bookList)
86:                     bookListModel.addElement(book);
87:             }
88:             catch (Exception e) {
89:                 bookListModel.removeAllElements();
90:                 bookListModel.addElement(e.toString());
91:             }
92:         }
93:
94:         public void changedUpdate(DocumentEvent event) { update(); }
95:         public void removeUpdate(DocumentEvent event) { update(); }
96:         public void insertUpdate(DocumentEvent event) { update(); }
97:     }
98:
99:     //-----
100:
101:     public void run() {
102:         JLabel authorLabel = new JLabel("Author: ", JLabel.RIGHT);
103:         JList<String> bookList = new JList<String>(bookListModel);
104:
105:         JPanel inputPanel = new JPanel();
106:         inputPanel.setLayout(new BorderLayout());
107:         inputPanel.add("West", authorLabel);
108:         inputPanel.add("Center", authorTextField);
109:
110:         JFrame frame = new JFrame();
111:         frame.setTitle("Penny");
112:         frame.setLayout(new BorderLayout());
113:         frame.add("North", inputPanel);
114:         frame.add("Center", bookList);
115:         Toolkit kit = Toolkit.getDefaultToolkit();
116:         Dimension screenSize = kit.getScreenSize();
117:         int screenWidth = screenSize.width;
118:         int screenHeight = screenSize.height;
119:         frame.setSize(screenWidth/4, screenHeight/4);
120:
121:         AuthorDocumentListener authorDocumentListener =
122:             new AuthorDocumentListener();
123:         authorTextField.getDocument().
124:             addDocumentListener(authorDocumentListener);
125:
126:         frame.setLocationByPlatform(true);
127:         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
128:         frame.setVisible(true);
129:     }
130: }

```

PennySwing3/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java PennySwing3/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java

```

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:     private String serverURL;
24:     private JTextField authorTextField = new JTextField("");
25:     private DefaultListModel<String> bookListModel =
26:         new DefaultListModel<String>();
27:
28:     //-----
29:
30:     public PennyDesktopRunnable(String serverURL) {
31:         this.serverURL = serverURL;
32:     }
33:
34:     //-----
35:
36:     private class AuthorDocumentListener implements DocumentListener {
37:         private void update() {
38:             String author = authorTextField.getText();
39:
40:             WorkerThread workerThread =
41:                 new WorkerThread(serverURL, author, bookListModel);
42:             workerThread.start();
43:         }
44:
45:         public void changedUpdate(DocumentEvent event) { update(); }
46:
47:         public void removeUpdate(DocumentEvent event) { update(); }
48:
49:         public void insertUpdate(DocumentEvent event) { update(); }
50:     }
51:
52:     //-----
53:
54:     public void run() {
55:         JLabel authorLabel = new JLabel("Author: ", JLabel.RIGHT);
56:         JList<String> bookList = new JList<String>(bookListModel);
57:
58:         JPanel inputPanel = new JPanel();
59:         inputPanel.setLayout(new BorderLayout());
60:         inputPanel.add("West", authorLabel);
61:         inputPanel.add("Center", authorTextField);
62:
63:         JFrame frame = new JFrame();
64:         frame.setTitle("Penny");
65:         frame.setLayout(new BorderLayout());
66:         frame.add("North", inputPanel);
67:         frame.add("Center", bookList);
68:         Toolkit kit = Toolkit.getDefaultToolkit();
69:         Dimension screenSize = kit.getScreenSize();
70:         int screenWidth = screenSize.width;
71:         int screenHeight = screenSize.height;
72:         frame.setSize(screenWidth/4, screenHeight/4);
73:
74:         AuthorDocumentListener authorDocumentListener =
75:             new AuthorDocumentListener();
76:
77:         authorTextField.getDocument().
78:             addDocumentListener(authorDocumentListener);
79:
80:         frame.setLocationByPlatform(true);
81:         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
82:         frame.setVisible(true);
83:     }
84: }

```

PennySwing3/src/main/java/edu/princeton/penny/WorkerThread.java (Page 9 of 18) PennySwing3/src/main/java/edu/princeton/penny/WorkerThread.java (Page 9 of 18)

```

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++) {
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 PennySwing4/src/main/java/edu/princeton/penny/PennyDesktopRunnable.java

```

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: public class PennyDesktopRunnable implements Runnable {
23:     private String serverURL;
24:     private JTextField authorTextField = new JTextField("");
25:     private DefaultListModel<String> bookListModel =
26:         new DefaultListModel<String>();
27:
28:     //-----
29:
30:     public PennyDesktopRunnable(String serverURL) {
31:         this.serverURL = serverURL;
32:     }
33:
34:     //-----
35:
36:     private class AuthorDocumentListener implements DocumentListener {
37:         private WorkerThread workerThread = null;
38:
39:         private void update() {
40:             String author = authorTextField.getText();
41:             if (workerThread != null)
42:                 workerThread.setShouldStop();
43:             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:         JLabel authorLabel = new JLabel("Author: ", JLabel.RIGHT);
59:         JList<String> bookList = new JList<String>(bookListModel);
60:
61:         JPanel inputPanel = new JPanel();
62:         inputPanel.setLayout(new BorderLayout());
63:         inputPanel.add("West", authorLabel);
64:         inputPanel.add("Center", authorTextField);
65:
66:         JFrame frame = new JFrame();
67:         frame.setTitle("Penny");
68:         frame.setLayout(new BorderLayout());
69:         frame.add("North", inputPanel);
70:         frame.add("Center", bookList);
71:         Toolkit kit = Toolkit.getDefaultToolkit();
72:         Dimension screenSize = kit.getScreenSize();
73:         int screenWidth = screenSize.width;
74:         int screenHeight = screenSize.height;
75:         frame.setSize(screenWidth/4, screenHeight/4);
76:
77:         AuthorDocumentListener authorDocumentListener =
78:             new AuthorDocumentListener();
79:
80:         authorTextField.getDocument().
81:             addDocumentListener(authorDocumentListener);
82:
83:         frame.setLocationByPlatform(true);
84:         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
85:         frame.setVisible(true);
86:     }
87: }

```

PennySwing4/src/main/java/edu/princeton/penny/WorkerThread.java (Page 11 of 18) PennySwing4/src/main/java/edu/princeton/penny/WorkerThread.java (Page 11 of 18)

```

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.InputStream;
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(
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:         } catch (Exception e) {
93:             SwingUtilities.invokeLater(
94:                 () -> {
95:                     if (! shouldStop)
96:                     {
97:                         bookListModel.removeAllElements();
98:                         bookListModel.addElement(e.toString());
99:                     }
100:                 }
101:             );
102:         }
103:     }
104: }
105: }

```

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

```

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: import java.util.Timer;
20: import java.util.TimerTask;
21:
22: //-----
23:
24: public class PennyDesktopRunnable implements Runnable {
25:     private String serverURL;
26:     private JTextField authorTextField = new JTextField("");
27:     private DefaultListModel<String> bookListModel =
28:         new DefaultListModel<String>();
29:
30:     //-----
31:
32:     public PennyDesktopRunnable(String serverURL) {
33:         this.serverURL = serverURL;
34:     }
35:
36:     //-----
37:
38:     private class AuthorDocumentListener implements DocumentListener {
39:         private WorkerThread workerThread = null;
40:         private Timer timer = null;
41:
42:         private class MyTimerTask extends TimerTask {
43:             public void run() {
44:                 String author = authorTextField.getText();
45:                 if (workerThread != null)
46:                     workerThread.setShouldStop();
47:                 workerThread =
48:                     new WorkerThread(serverURL, author, bookListModel);
49:                 workerThread.start();
50:             }
51:         }
52:
53:         private void update() {
54:             if (timer != null)
55:                 timer.cancel();
56:             timer = new Timer();
57:             timer.schedule(new MyTimerTask(), 500);
58:         }
59:
60:         public void changedUpdate(DocumentEvent event) { update(); }
61:
62:         public void removeUpdate(DocumentEvent event) { update(); }
63:
64:         public void insertUpdate(DocumentEvent event) { update(); }
65:     }
66:
67:     //-----
68:
69:     public void run() {
70:         JLabel authorLabel = new JLabel("Author: ", JLabel.RIGHT);
71:         JList<String> bookList = new JList<String>(bookListModel);
72:
73:         JPanel inputPanel = new JPanel();
74:         inputPanel.setLayout(new BorderLayout());
75:         inputPanel.add("West", authorLabel);
76:         inputPanel.add("Center", authorTextField);
77:
78:         JFrame frame = new JFrame();
79:         frame.setTitle("Penny");
80:         frame.setLayout(new BorderLayout());
81:         frame.add("North", inputPanel);
82:         frame.add("Center", bookList);
83:         Toolkit kit = Toolkit.getDefaultToolkit();
84:         Dimension screenSize = kit.getScreenSize();
85:         int screenWidth = screenSize.width;
86:         int screenHeight = screenSize.height;
87:         frame.setSize(screenWidth/4, screenHeight/4);
88:
89:         AuthorDocumentListener authorDocumentListener =
90:             new AuthorDocumentListener();
91:
92:         authorTextField.getDocument().
93:             addDocumentListener(authorDocumentListener);
94:
95:         frame.setLocationByPlatform(true);
96:         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
97:         frame.setVisible(true);
98:     }
99: }

```

## Threads/race.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # race.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:
17:     def get_balance(self):
18:         return self._balance
19:
20:     def deposit(self, amount):
21:         temp = self._balance
22:         temp += amount
23:         print(temp)
24:         self._balance = temp
25:
26:     def withdraw(self, amount):
27:         temp = self._balance
28:         temp -= amount
29:         print(temp)
30:         self._balance = temp
31:
32: #-----
33:
34: class DepositThread (threading.Thread):
35:
36:     def __init__(self, bank_acct):
37:         threading.Thread.__init__(self)
38:         self._bank_acct = bank_acct
39:
40:     def run(self):
41:         for _ in range(10):
42:             self._bank_acct.deposit(1)
43:
44: #-----
45:
46: class WithdrawThread (threading.Thread):
47:
48:     def __init__(self, bank_acct):
49:         threading.Thread.__init__(self)
50:         self._bank_acct = bank_acct
51:
52:     def run(self):
53:         for _ in range(5):
54:             self._bank_acct.withdraw(2)
55:
56: #-----
57:
58: def main():
59:
60:     bank_acct = BankAcct()
61:
62:     deposit_thread = DepositThread(bank_acct)
63:     withdraw_thread = WithdrawThread(bank_acct)
64:
65:     deposit_thread.start()

```

## Threads/race.py (Page 2 of 2)

```

66:     withdraw_thread.start()
67:
68:     deposit_thread.join()
69:     withdraw_thread.join()
70:
71:     print('Final balance:', bank_acct.get_balance())
72:
73: #-----
74:
75: if __name__ == '__main__':
76:     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)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # locking.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:
18:     def get_balance(self):
19:         self._lock.acquire()
20:         try:
21:             return self._balance
22:         finally:
23:             self._lock.release()
24:
25:     def deposit(self, amount):
26:         self._lock.acquire()
27:         temp = self._balance
28:         temp += amount
29:         print(temp)
30:         self._balance = temp
31:         self._lock.release()
32:
33:     def withdraw(self, amount):
34:         self._lock.acquire()
35:         temp = self._balance
36:         temp -= amount
37:         print(temp)
38:         self._balance = temp
39:         self._lock.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/locking.py (Page 2 of 2)

```

66:
67: def main():
68:
69:     bank_acct = BankAcct()
70:
71:     deposit_thread = DepositThread(bank_acct)
72:     withdraw_thread = WithdrawThread(bank_acct)
73:
74:     deposit_thread.start()
75:     withdraw_thread.start()
76:
77:     deposit_thread.join()
78:     withdraw_thread.join()
79:
80:     print('Final balance:', bank_acct.get_balance())
81:
82: #-----
83:
84: if __name__ == '__main__':
85:     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: }

```