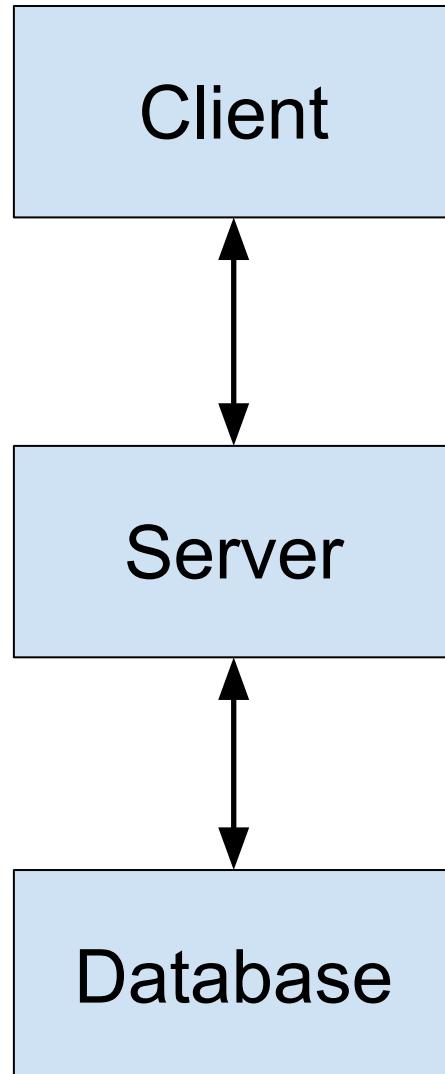


Client-Side Options (Part 2)

Copyright © 2024 by
Robert M. Dondero, Ph.D.
Princeton University

Objectives



Python

Browser/HTML/JavaScript (jQuery, React)

Desktop apps (PyQt5)

Android apps

iOS apps

blue => course default
red => other option

Python

Python/Flask/Jinja2

Java/Servlets

Java/Spring/Mustache

JavaScript/Express/Mustache

SQLite

PostgreSQL

Objectives

- We will cover:
 - Mobile programming
 - Android mobile programming
 - iOS mobile programming

Agenda

- **Aside: Function def expressions**
- Aside: Multithreaded programming
- Mobile programming
- PennyAndroid app: defining
- PennyAndroid app: running
- Pennylos app: defining
- Pennylos app: running

Function Def Exprs: JavaScript

With a function definition **statement**:

```
function compareLengths(word1, word2) {  
    return word1.length - word2.length;  
}  
...  
words.sort(compareLengths);  
...
```

With a function definition **expression**:

```
...  
words.sort(  
    function(word1, word2) {  
        return word1.length - word2.length;  
});  
...
```

Function Def Exprs: JavaScript

With a function definition **statement**:

```
function compareLengths(word1, word2) {  
    return word1.length - word2.length;  
}  
...  
words.sort(compareLengths);  
...
```

With a function definition **expression**:

```
...  
words.sort(  
    (word1, word2) => word1.length - word2.length);  
...
```

Arrow functions are a syntactic shortcut

Function Def Exprs: Python

Lambda Expressions



Alonzo
Church

Function Def Exprs: Python

```
lambda param1, param2 ...: expression
```

- The keyword `lambda`
- (optionally) Parameters separated by commas
- A colon
- A single expression that uses the parameters

Function Def Exprs: Python

Without a lambda expression:

```
def compare_lengths(word1, word2):  
    return len(word1) - len(word2)  
  
...  
words.sort(compare_lengths)  
...
```

With a lambda expression:

```
...  
words.sort(  
    lambda word1,word2: len(word1)-len(word2) )  
...
```

Function Def Exprs: Java

- *Java lambda expressions*
 - New to Java SE 8

Function Def Exprs: Java

Without a lambda expression:

```
class LengthComparator implements Comparator<String>
{
    public int compare(String word1, String word2)
    {
        return word1.length() - word2.length();
    }
}
...
String[] words;
...
Arrays.sort(words, new LengthComparator());
...
```

Function Def Exprs: Java

With a lambda expression:

```
...
String[] words;
...
Arrays.sort(words,
    (String word1, String word2) ->
        word1.length() - word2.length() );
...
```

Function Def Exprs: Java

With a lambda expression:

```
...
String[] words;
...
Arrays.sort(words,
    (word1, word2) -> word1.length() - word2.length()
);
...
```

Sometimes can omit parameter types

Function Def Exprs: Java

- ***Functional interface***
 - An interface that declares a single method
- It's OK to use a lambda expression in lieu of an **object of a class that implements a functional interface**

Function Def Exprs: Java

- Java lambda expression observations
 - Function def expressions in a language that doesn't have functions!
 - Handy!
 - Inelegant?

Agenda

- Aside: Function def expressions
- **Aside: Multithreaded programming**
- Mobile programming
- PennyAndroid app: defining
- PennyAndroid app: running
- Pennylos app: defining
- Pennylos app: running

Multithreaded Programming

- Recall spawning.py

```
$ python spawning.py
blue
blue
blue
blue
blue
blue
blue thread terminated
red
red
red
red
red
red
red thread terminated
main thread terminated
$
```

```
$ python spawning.py
blue
blue
blue
blue
blue
red
red
red
red
red
red
red thread terminated
blue
main thread terminated
blue thread terminated
$
```

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:

```

Multithreaded Programming

- See Spawning1.java

Multithreaded Programming

- **Problem:**
 - Java does not allow multiple inheritance
 - What if class PrinterThread already inherits from some class other than Thread?
- **Solution:** ...

Multithreaded Programming

- See Spawning2.java

```
$ java Spawning2
blue
blue
blue
blue
blue
blue
main thread terminated
red
red
red
red
red
blue thread terminated
red thread terminated
$
```

```
$ java Spawning2
blue
blue
main thread terminated
red
red
red
red
red
blue
blue
blue
blue thread terminated
red 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:

```

Multithreaded Programming

- **Note:**
 - `bluePrinterRunnable` is an object of a class that implements an interface that defines one method
 - `bluePrinterRunnable` can be replaced with a lambda expression
 - Same for `redPrinterRunnable`
- **And so...**

Multithreaded Programming

- See Spawning3.java

```
$ java Spawning3
main thread terminated
blue
blue
blue
blue
blue
blue
blue thread terminated
red
red
red
red
red
red
red thread terminated
$
```

```
$ java Spawning3
main thread terminated
blue
blue
blue
blue
red
red
red
red
red
red
red thread terminated
blue
blue
blue 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:

```

Agenda

- Aside: Function def expressions
- Aside: Multithreaded programming
- **Mobile programming**
- PennyAndroid app: defining
- PennyAndroid app: running
- Pennylos app: defining
- Pennylos app: running

Mobile Programming

- Suppose you want your app to run on a mobile device...
- So far:
 - Mobile web apps
- Now:
 - Native mobile apps

Mobile Programming

- Which option (mobile web app vs native mobile app) is right for you?
 - **Step 1:** Consider whether you have an option!

Mobile Programming

- See <https://whatwebcando.today/>
- Examples: If you need ____ do you have an option?
 - **Offline mode:** Yes
 - **Audio & video capture:** Probably
 - **Proximity sensors:** Probably not
 - **Contacts:** No

Mobile Programming

- Which option (mobile web app vs. native mobile app) is right for you?
 - **Step 1:** Consider whether you have an option!
 - **Step 2:** Consider the broader context...

Mobile Programming

Desirable Factor	Mobile Web App	Native Mobile App
Easy discoverability	✓	
Native look & feel		✓
Good performance (speed)		✓
Easy installation	✓	
Low development cost *	✓	
Low maintenance cost *	✓	
Few content restrictions, easy approval process, low/no fees	✓	

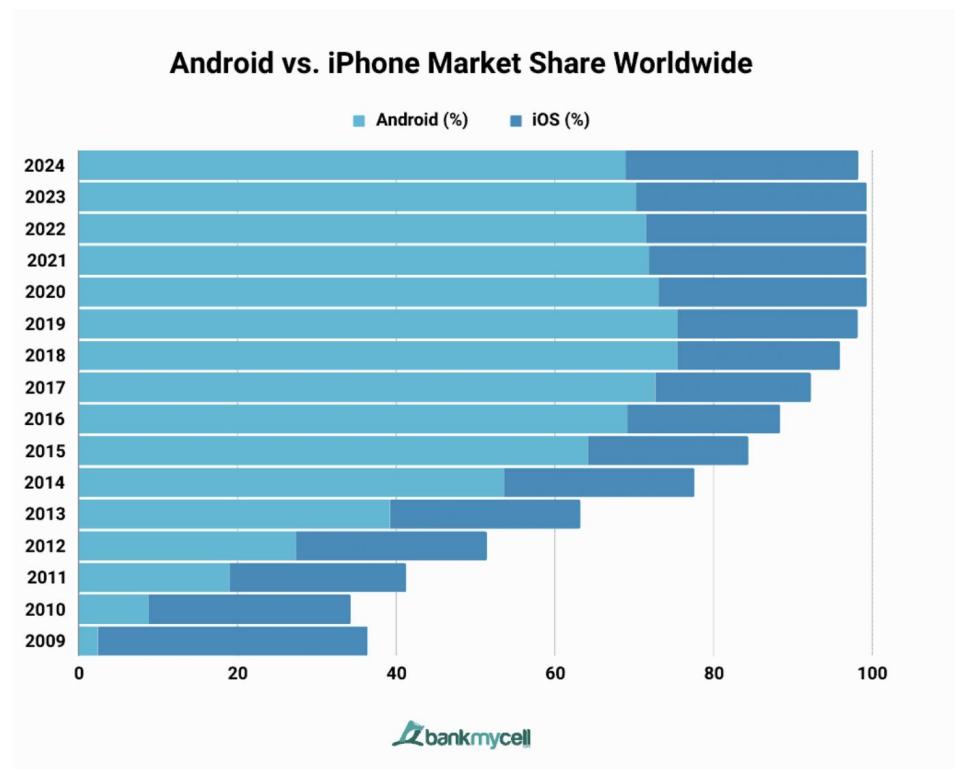
* May need multiple native mobile apps

<https://www.nngroup.com/articles/mobile-native-apps/>

Mobile Programming

Android vs. iOS?

Mobile operating systems' market share worldwide



In 2024:
Android: 69.88%
iOS: 29.39%

<https://www.bankmycell.com/blog/android-vs-apple-market-share/>

Agenda

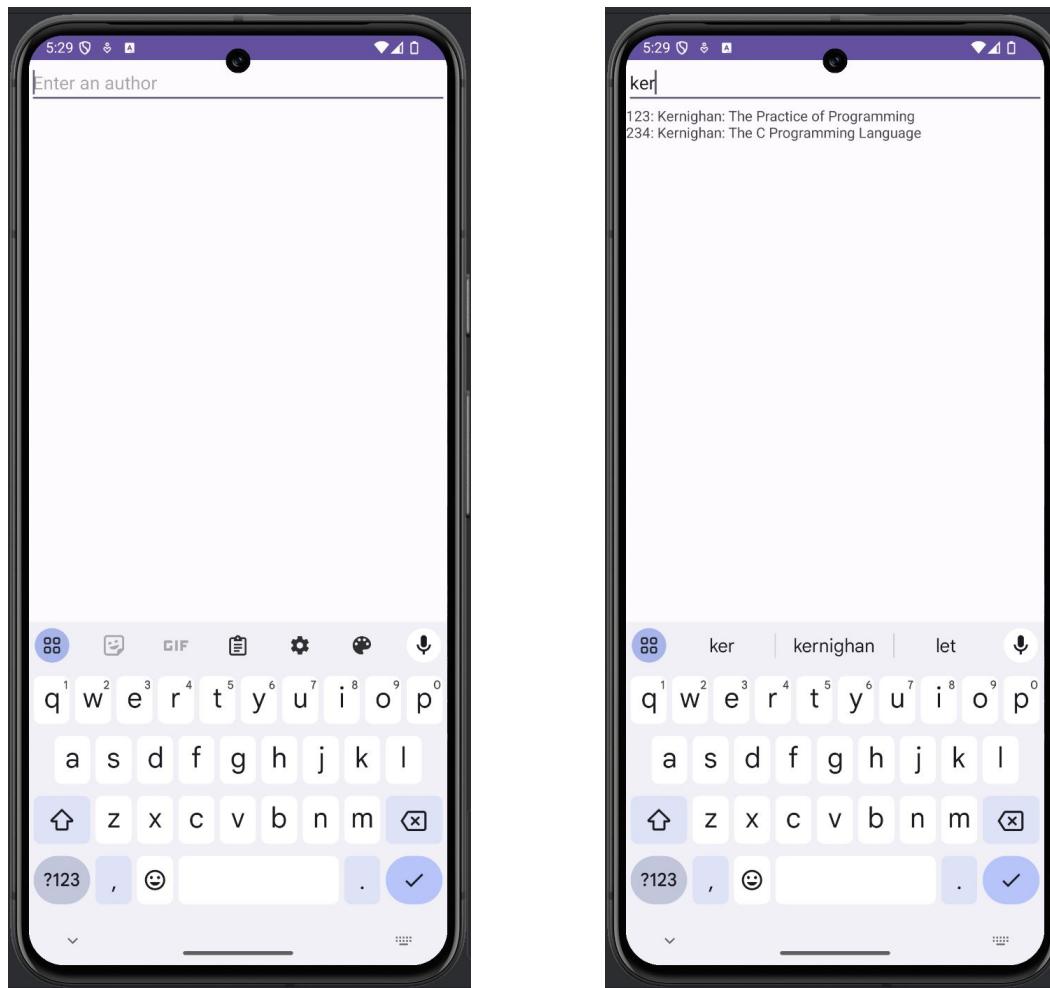
- Aside: Function def expressions
- Aside: Multithreaded programming
- Mobile programming
- **PennyAndroid app: defining**
- PennyAndroid app: running
- Pennylos app: defining
- Pennylos app: running

PennyAndroid App: Defining

- Preliminary
 - Deploy PennyJson server to
<https://pennyjson.onrender.com>
 - So PennyAndroid client can access it

PennyAndroid App: Defining

The
goal:



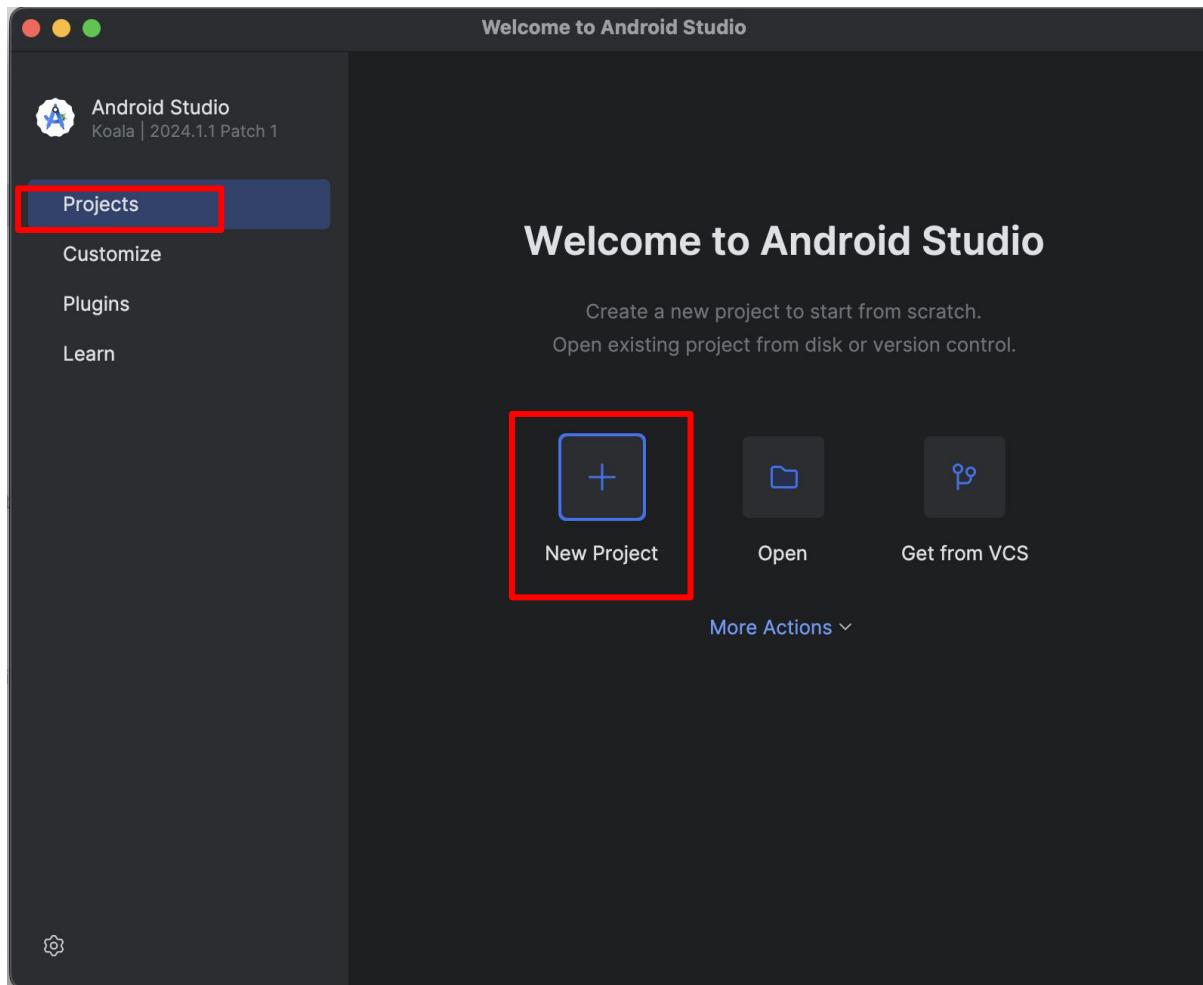
An Android client for the PennyJson server

PennyAndroid App: Defining

- Download and install *Android Studio*
 - Browse to
<https://developer.android.com/studio/install.html>
 - Complete the wizard; use defaults

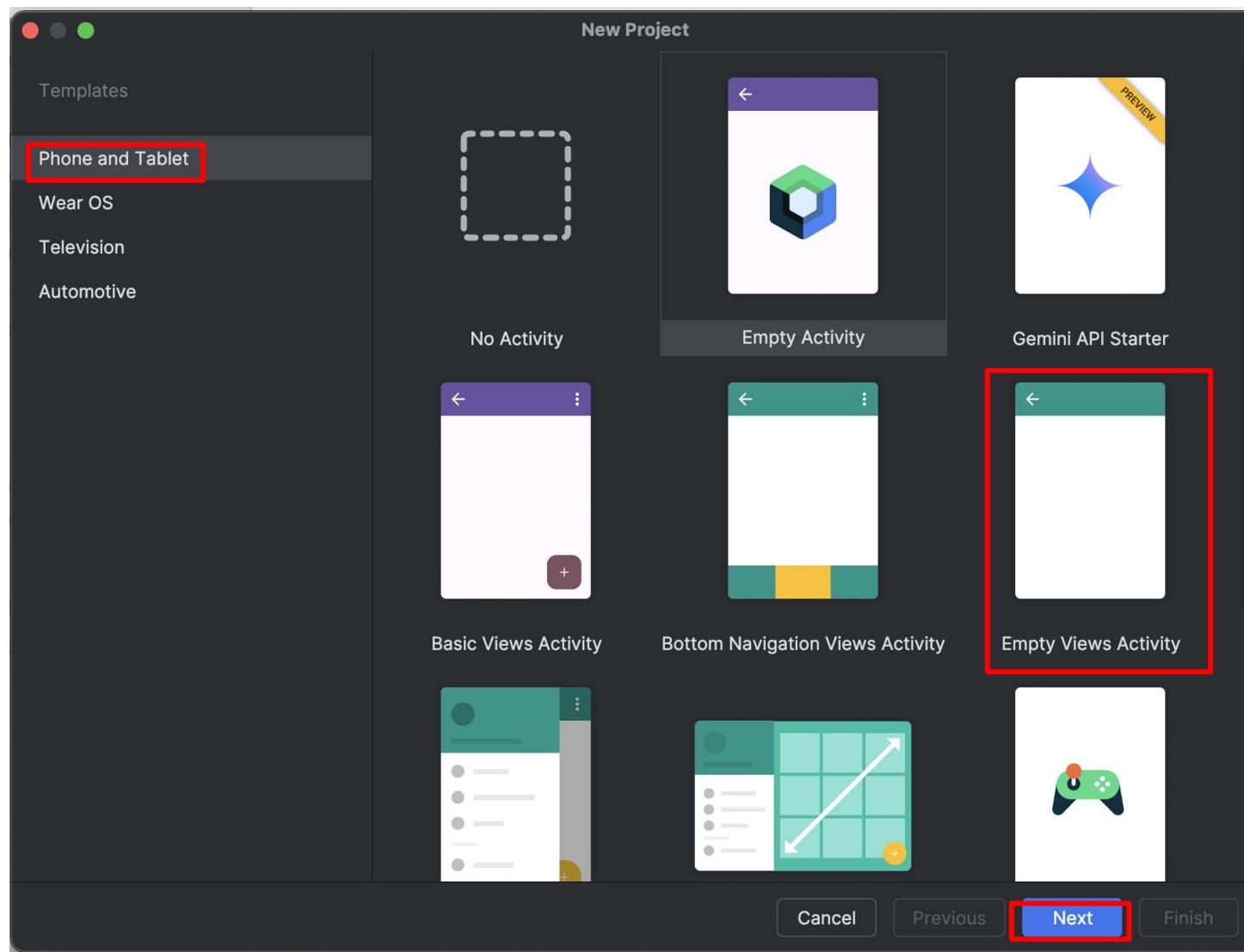
PennyAndroid App: Defining

Launch Android Studio; select *Projects*; click on *New Project*



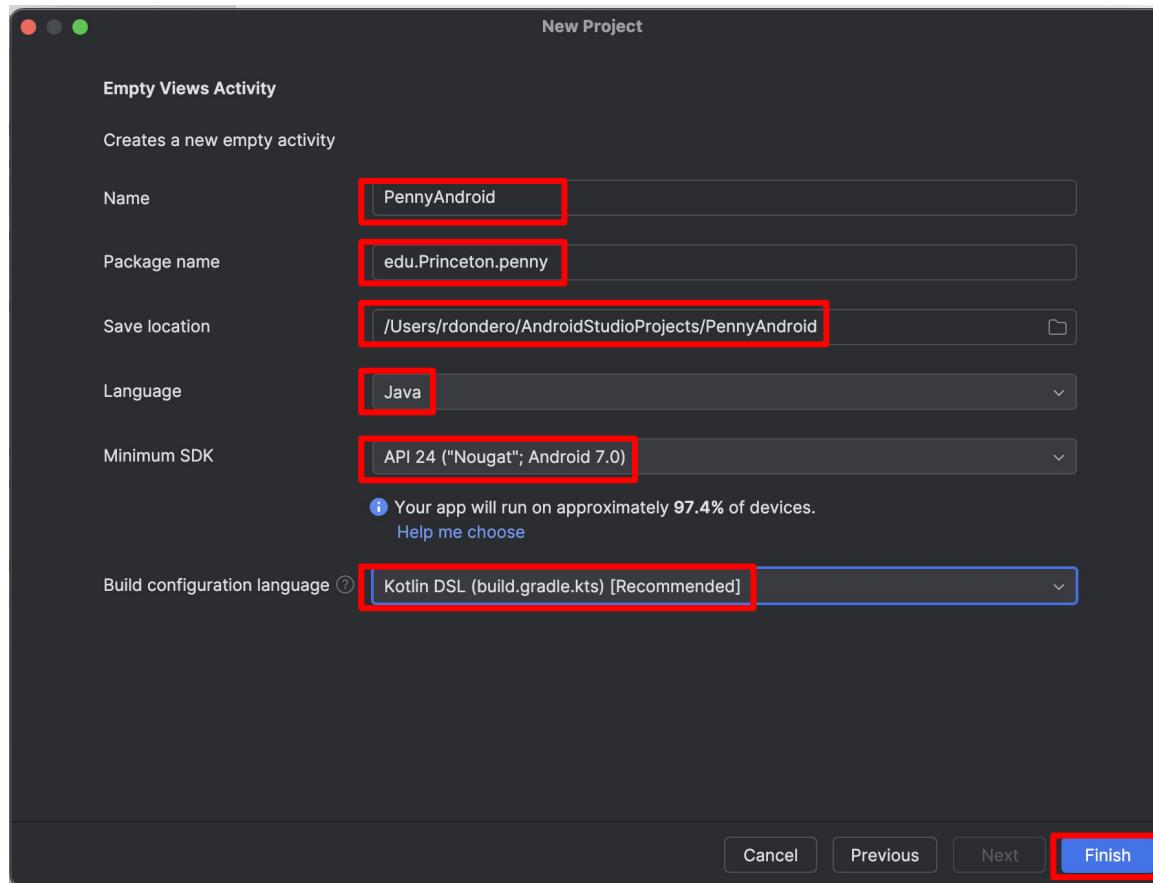
PennyAndroid App: Defining

Select Phone and Tablet; select *Empty Views Activity*; click on *Next*



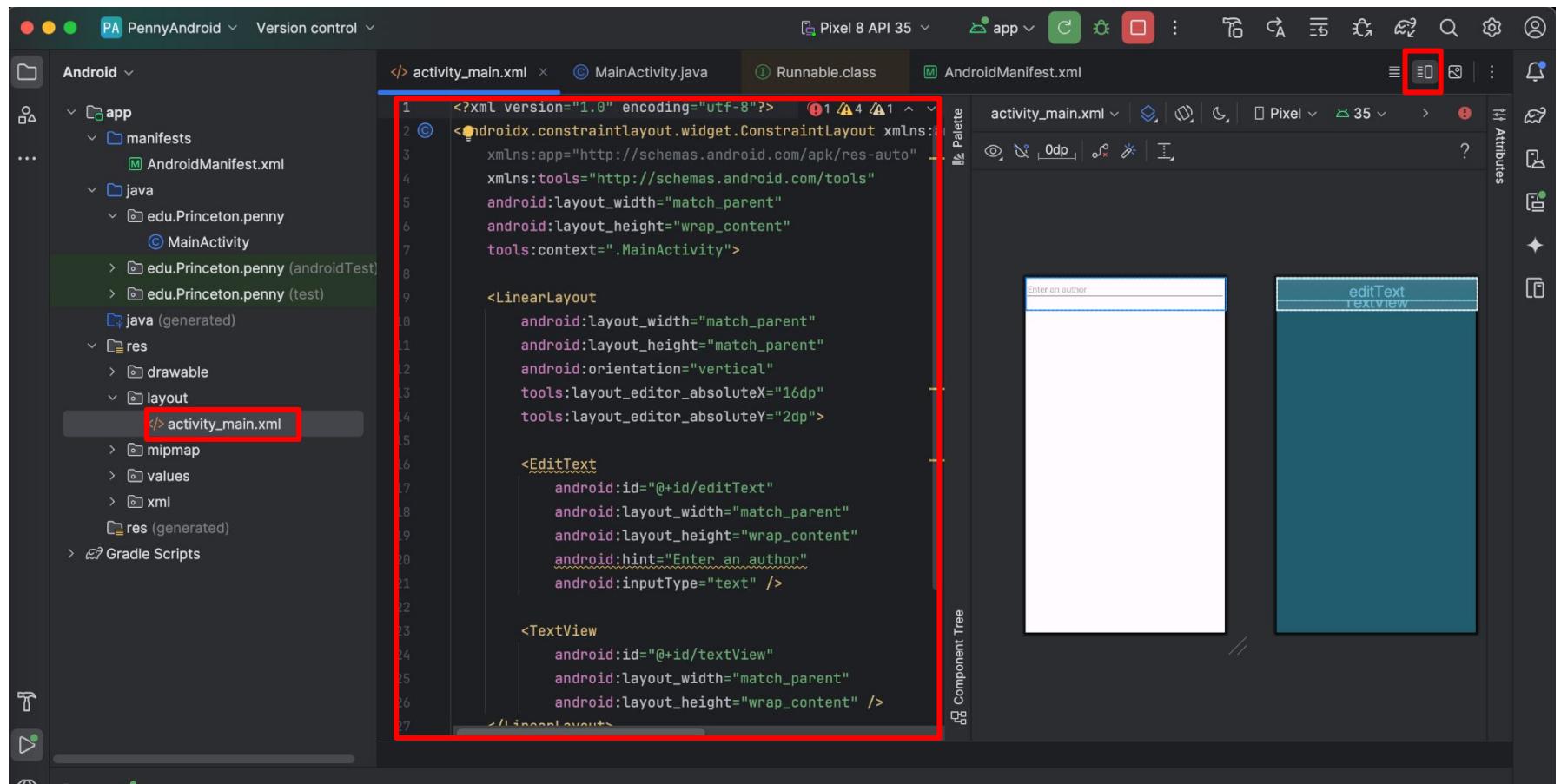
PennyAndroid App: Defining

For *Name* enter PennyAndroid; for *Package Name* enter edu.Princeton.penny; for *Save Location* choose whatever directory you want; for *Language* select Java; for *Minimum SDK* choose whatever you want; for Build configuration language choose Kotlin DSL; click on *Finish*



PennyAndroid App: Defining

In left panel double click on *app > res > layout > activity_main.xml*; at upper right, click on *Split icon*; copy **activity_main.xml** into editor



PennyAndroid/activity_main.xml (Page 1 of 1)

```

1: <?xml version="1.0" encoding="utf-8"?>
2: <androidx.constraintlayout.widget.ConstraintLayout
3:   xmlns:android="http://schemas.android.com/apk/res/android"
4:   xmlns:app="http://schemas.android.com/apk/res-auto"
5:   xmlns:tools="http://schemas.android.com/tools"
6:   android:layout_width="match_parent"
7:   android:layout_height="wrap_content"
8:   tools:context=".MainActivity">
9:
10:  <LinearLayout
11:    android:layout_width="match_parent"
12:    android:layout_height="match_parent"
13:    android:orientation="vertical"
14:    tools:layout_editor_absoluteX="16dp"
15:    tools:layout_editor_absoluteY="2dp">
16:
17:    <EditText
18:      android:id="@+id/editText"
19:      android:layout_width="match_parent"
20:      android:layout_height="wrap_content"
21:      android:hint="Enter an author"
22:      android:inputType="text" />
23:
24:    <TextView
25:      android:id="@+id/textView"
26:      android:layout_width="match_parent"
27:      android:layout_height="wrap_content" />
28:  </LinearLayout>
29:
30: </androidx.constraintlayout.widget.ConstraintLayout>

```

PennyAndroid/MainActivity.java (Page 1 of 3)

```

1: //-----
2: // MainActivity.java
3: // Author: Bob Dondero
4: //-----
5:
6: package edu.Princeton.penny;
7:
8: import android.widget.EditText;
9: import android.app.Activity;
10: import android.os.Bundle;
11: import android.text.method.ScrollingMovementMethod;
12: import android.widget.TextView;
13: import android.text.TextWatcher;
14: import android.text.Editable;
15:
16: import java.io.BufferedReader;
17: import java.io.IOException;
18: import java.io.InputStreamReader;
19: import java.net.URL;
20: import javax.net.ssl.HttpsURLConnection;
21: import java.util.Timer;
22: import java.util.TimerTask;
23:
24: import org.json.JSONObject;
25: import org.json.JSONArray;
26:
27: public class MainActivity extends Activity
28: {
29:     private EditText editText;
30:     private TextView textView;
31:     private AuthorSearchRunnable authorSearchRunnable = null;
32:     private Timer timer = null;
33:
34:     //-----
35:
36:     private class AuthorSearchRunnable implements Runnable
37:     {
38:         private String author;
39:         private boolean shouldStop = false;
40:
41:         AuthorSearchRunnable(String author)
42:         {
43:             this.author = author;
44:         }
45:
46:         public void setShouldStop()
47:         {
48:             shouldStop = true;
49:         }
50:
51:         public void run()
52:         {
53:             String books;
54:
55:             try
56:             {
57:                 URL url = new URL("https://pennyjson.onrender.com" +
58:                                     "/searchresults?author=" + author);
59:                 HttpsURLConnection con =
60:                     (HttpsURLConnection) url.openConnection();
61:
62:                 con.setRequestMethod("GET");
63:                 con.setDoInput(true);
64:                 con.connect();
65:                 int responseCode = con.getResponseCode();

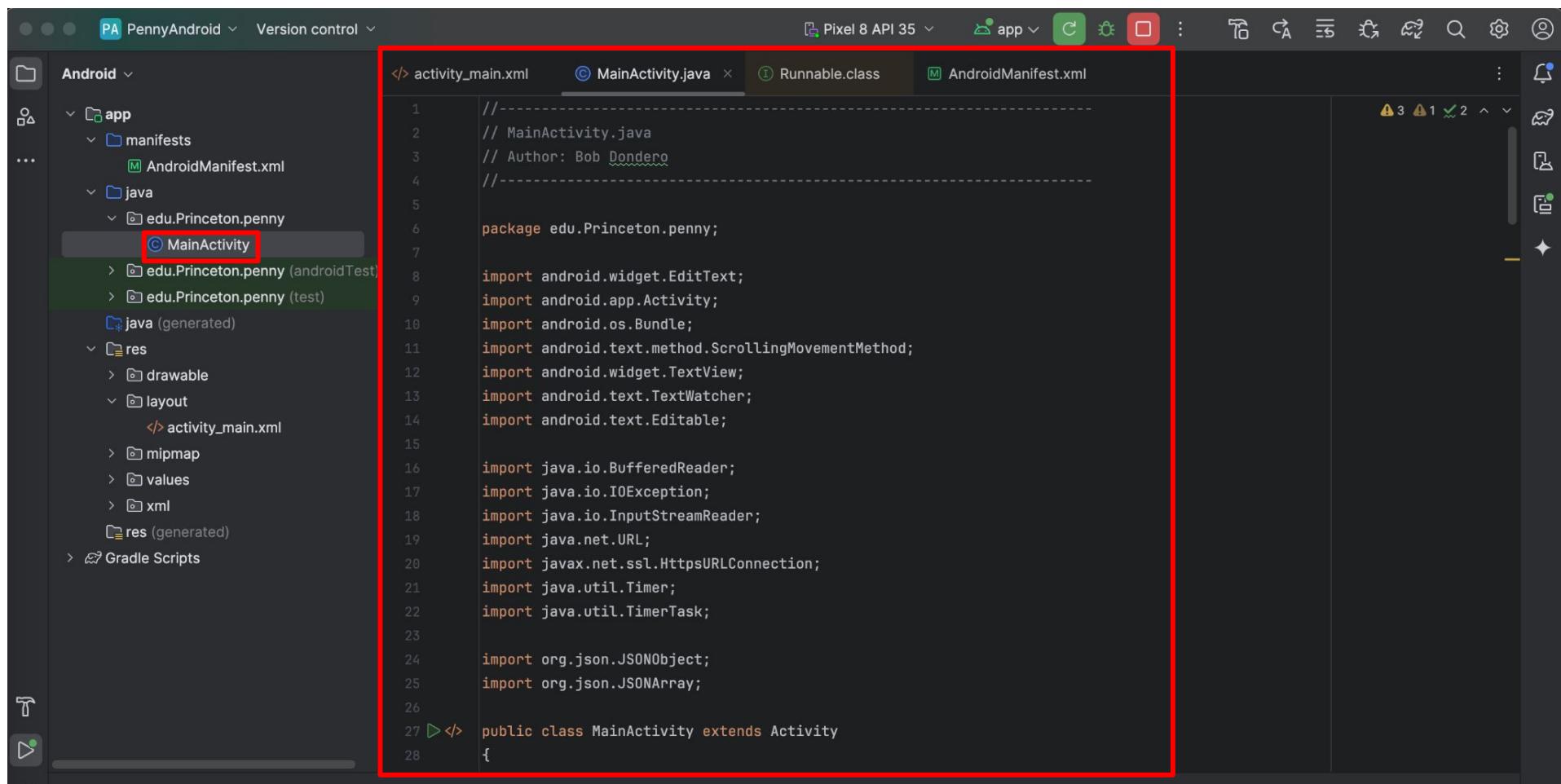
```

PennyAndroid App: Defining

- Suggestion:
 - Experiment with the graphical editor
 - Look at resulting XML code

PennyAndroid App: Defining

In left panel, double-click on *app > java > edu.Princeton.edu > MainActivity*; copy **MainActivity.java** into editor



The screenshot shows the Android Studio interface. On the left is the Project Navigational Bar, which includes sections for Android, app, manifests, Java (with subfolders for edu.Princeton.penny and generated), res (with drawable, layout, mipmap, values, and xml), and Gradle Scripts. A red box highlights the 'MainActivity' file under the Java section of the edu.Princeton.penny folder. The main workspace shows the code editor with the 'MainActivity.java' tab selected. The code itself is as follows:

```
// Main Activity
// Author: Bob Dondero
//
package edu.Princeton.penny;

import android.widget.EditText;
import android.app.Activity;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.widget.TextView;
import android.text.TextWatcher;
import android.text.Editable;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import javax.net.ssl.HttpsURLConnection;
import java.util.Timer;
import java.util.TimerTask;

import org.json.JSONObject;
import org.json.JSONArray;
public class MainActivity extends Activity
{
```

PennyAndroid App: Defining

- **Android design constraint 1**
 - Main/GUI thread is not allowed to do networking
- **Implications**
 - Main/GUI thread must spawn a child/worker thread
 - Child/worker thread must comm with PennyJson

PennyAndroid App: Defining

- **Android design constraint 2**
 - Main/GUI thread must remain responsive
 - Main/GUI thread laggy => typing/tapping fast generates “App is unresponsive” messages
- **Implications**
 - Main/GUI thread cannot wait for child/worker thread to finish
 - Main/GUI thread and child/worker thread must run concurrently

PennyAndroid App: Defining

- **Android design constraint 3**
 - Child/worker thread is not allowed to update GUI
- **Implications**
 - Child/worker thread must send changes to main/GUI thread, and ask main/GUI thread to update the GUI

PennyAndroid App: Defining

- **MainActivity.java**

- Informal overview:

- `onCreate()` instantiates `MyTextWatcher` object, and installs it as the listener for the `EditText` object
 - When user enters text into `EditText` object, Android calls `afterTextChanged()` method in `MyTextWatcher` object

PennyAndroid App: Defining

- **MainActivity.java**

- Informal overview:

- `afterTextChanged()` method handles debouncing via `Timer` and `MyTimerTask` objects
 - `MyTimerTask` object spawns & starts new `AuthorSearch` thread
 - `AuthorSearch` thread sends author to server, receives JSON response from server, calls `runOnUiThread()` to tell main/GUI thread to update GUI
 - main/GUI thread updates GUI

PennyAndroid/activity_main.xml (Page 1 of 1)

```

1: <?xml version="1.0" encoding="utf-8"?>
2: <androidx.constraintlayout.widget.ConstraintLayout
3:   xmlns:android="http://schemas.android.com/apk/res/android"
4:   xmlns:app="http://schemas.android.com/apk/res-auto"
5:   xmlns:tools="http://schemas.android.com/tools"
6:   android:layout_width="match_parent"
7:   android:layout_height="wrap_content"
8:   tools:context=".MainActivity">
9:
10:  <LinearLayout
11:    android:layout_width="match_parent"
12:    android:layout_height="match_parent"
13:    android:orientation="vertical"
14:    tools:layout_editor_absoluteX="16dp"
15:    tools:layout_editor_absoluteY="2dp">
16:
17:    <EditText
18:      android:id="@+id/editText"
19:      android:layout_width="match_parent"
20:      android:layout_height="wrap_content"
21:      android:hint="Enter an author"
22:      android:inputType="text" />
23:
24:    <TextView
25:      android:id="@+id/textView"
26:      android:layout_width="match_parent"
27:      android:layout_height="wrap_content" />
28:  </LinearLayout>
29:
30: </androidx.constraintlayout.widget.ConstraintLayout>

```

PennyAndroid/MainActivity.java (Page 1 of 3)

```

1: //-----
2: // MainActivity.java
3: // Author: Bob Dondero
4: //-----
5:
6: package edu.Princeton.penny;
7:
8: import android.widget.EditText;
9: import android.app.Activity;
10: import android.os.Bundle;
11: import android.text.method.ScrollingMovementMethod;
12: import android.widget.TextView;
13: import android.text.TextWatcher;
14: import android.text.Editable;
15:
16: import java.io.BufferedReader;
17: import java.io.IOException;
18: import java.io.InputStreamReader;
19: import java.net.URL;
20: import javax.net.ssl.HttpsURLConnection;
21: import java.util.Timer;
22: import java.util.TimerTask;
23:
24: import org.json.JSONObject;
25: import org.json.JSONArray;
26:
27: public class MainActivity extends Activity
28: {
29:     private EditText editText;
30:     private TextView textView;
31:     private AuthorSearchRunnable authorSearchRunnable = null;
32:     private Timer timer = null;
33:
34:     //-----
35:
36:     private class AuthorSearchRunnable implements Runnable
37:     {
38:         private String author;
39:         private boolean shouldStop = false;
40:
41:         AuthorSearchRunnable(String author)
42:         {
43:             this.author = author;
44:         }
45:
46:         public void setShouldStop()
47:         {
48:             shouldStop = true;
49:         }
50:
51:         public void run()
52:         {
53:             String books;
54:
55:             try
56:             {
57:                 URL url = new URL("https://pennyjson.onrender.com" +
58:                                     "/searchresults?author=" + author);
59:                 HttpsURLConnection con =
60:                     (HttpsURLConnection) url.openConnection();
61:
62:                 con.setRequestMethod("GET");
63:                 con.setDoInput(true);
64:                 con.connect();
65:                 int responseCode = con.getResponseCode();

```

PennyAndroid/MainActivity.java (Page 2 of 3)

```

66:         if (responseCode != HttpURLConnection.HTTP_OK)
67:             throw new IOException(
68:                 "HTTP error code: " + responseCode);
69:
70:         // Read the response, a JSON document containing books,
71:         // from the server.
72:         BufferedReader in =
73:             new BufferedReader(
74:                 new InputStreamReader(con.getInputStream()));
75:         String jsonDoc = in.readLine();
76:         in.close();
77:
78:         // Use the JSON response to build a String, and assign
79:         // it to books.
80:         JSONArray jsonArray = new JSONArray(jsonDoc);
81:         StringBuilder booksBuilder = new StringBuilder();
82:         for (int i = 0; i < jsonArray.length(); i++)
83:         {
84:             JSONObject jsonObject = jsonArray.getJSONObject(i);
85:             booksBuilder.append(jsonObject.getString("isbn"));
86:             booksBuilder.append(": ");
87:             booksBuilder.append(jsonObject.getString("author"));
88:             booksBuilder.append(": ");
89:             booksBuilder.append(jsonObject.getString("title"));
90:             booksBuilder.append("\n");
91:         }
92:         books = booksBuilder.toString();
93:
94:         if (! shouldStop)
95:             runOnUiThread(() -> { textView.setText(books); });
96:     }
97:
98:     catch (Exception e)
99:     {
100:         if (! shouldStop)
101:             runOnUiThread(() -> { textView.setText(e.toString()); });
102:     }
103: }
104:
105: //-----
106:
107: private class MyTimerTask extends TimerTask
108: {
109:     public void run()
110:     {
111:         String author = editText.getText().toString();
112:
113:         if (authorSearchRunnable != null)
114:             authorSearchRunnable.setShouldStop();
115:
116:         authorSearchRunnable = new AuthorSearchRunnable(author);
117:         Thread authorSearchThread = new Thread(authorSearchRunnable);
118:         authorSearchThread.start();
119:
120:     }
121: }
122:
123: //-----
124:
125: private class MyTextWatcher implements TextWatcher
126: {
127:     public void onTextChanged(CharSequence s, int start,
128:                             int before, int count)
129:     {
130:     }

```

PennyAndroid/MainActivity.java (Page 3 of 3)

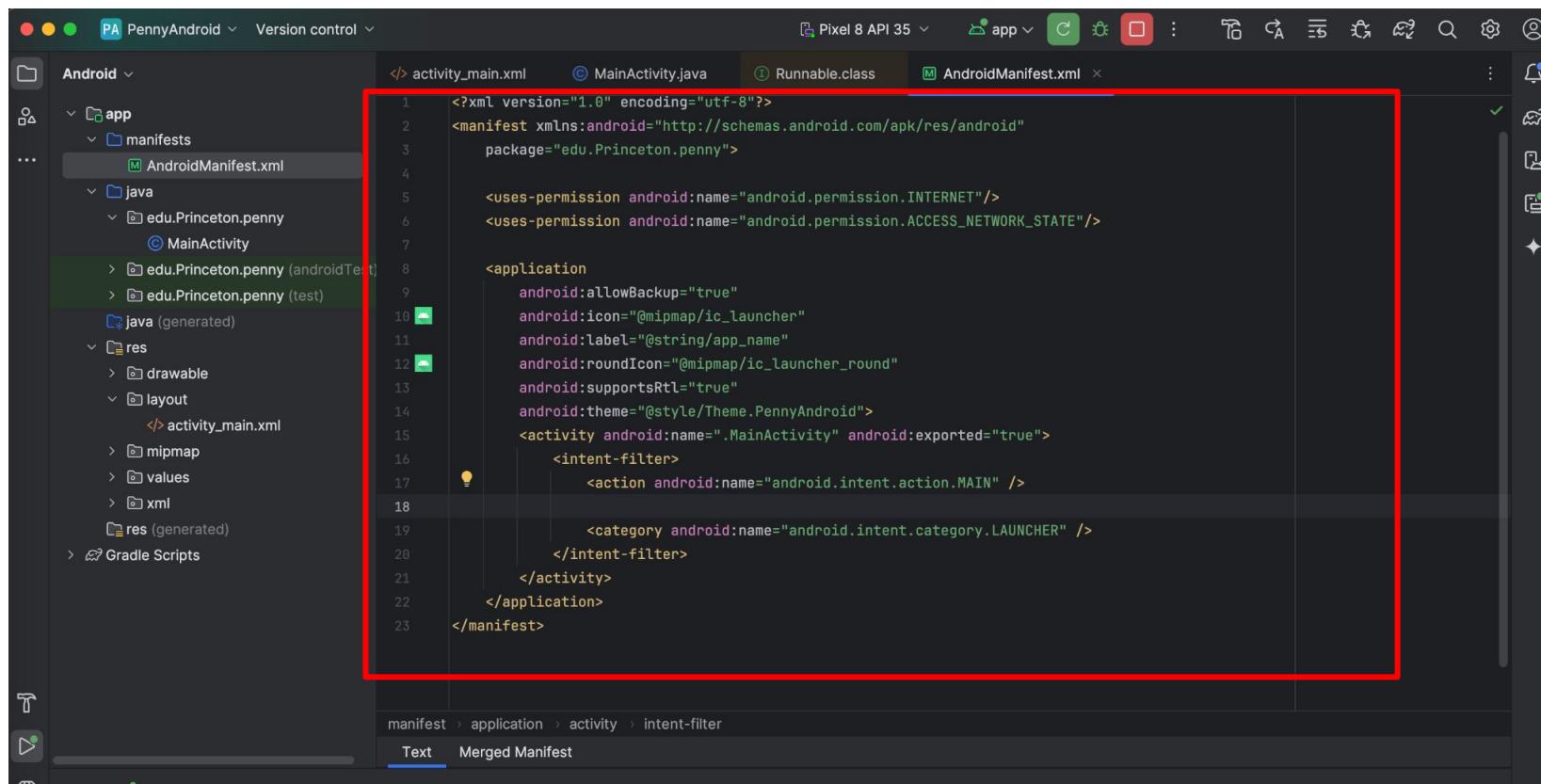
```

131:     public void beforeTextChanged(CharSequence s, int start,
132:                                   int count, int after)
133:     {
134:     }
135:
136:     public void afterTextChanged(Editable s)
137:     {
138:         if (timer != null)
139:             timer.cancel();
140:         timer = new Timer();
141:         timer.schedule(new MyTimerTask(), 500);
142:
143:     }
144: }
145:
146: //-----
147:
148: protected void onCreate(Bundle activityState)
149: {
150:     super.onCreate(activityState);
151:
152:     // Set the content view of the window to activity_main,
153:     // as defined in activity_main.xml.
154:     setContentView(R.layout.activity_main);
155:
156:     // Find the EditText object.
157:     editText = findViewById(R.id.editText);
158:
159:     // Instantiate a TextWatcher object, and install it as a
160:     // "text changed" listener on the EditText object. Subsequently,
161:     // immediately after the user changes the text within the
162:     // EditText object, the afterTextChanged() method will be called.
163:     editText.addTextChangedListener(new MyTextWatcher());
164:
165:     // Find the TextView object.
166:     textView = findViewById(R.id.textView);
167:
168:     // Allow the TextView object to scroll, just in case there are
169:     // many books.
170:     textView.setMovementMethod(new ScrollingMovementMethod());
171:
172: }

```

PennyAndroid App: Defining

In left panel double-click on *app > manifests > AndroidManifest.xml*; copy text from **AndroidManifest.xml** into editor



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "Android". The "app" module is expanded, showing "manifests" which contains "AndroidManifest.xml". Other sections like "java", "res", and "Gradle Scripts" are also visible.
- Editor:** The main editor area displays the content of "AndroidManifest.xml". The code is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.Princeton.penny">

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.PennyAndroid">
        <activity android:name=".MainActivity" android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

A large red box highlights the entire content of the "AndroidManifest.xml" file in the editor.

PennyAndroid/AndroidManifest.xml (Page 1 of 1)

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3:   package="edu.Princeton.penny">
4:
5:   <uses-permission android:name="android.permission.INTERNET"/>
6:   <uses-permission
7:     android:name="android.permission.ACCESS_NETWORK_STATE"/>
8:
9:   <application
10:    android:allowBackup="true"
11:    android:icon="@mipmap/ic_launcher"
12:    android:label="@string/app_name"
13:    android:roundIcon="@mipmap/ic_launcher_round"
14:    android:supportsRtl="true"
15:    android:theme="@style/Theme.PennyAndroid">
16:      <activity android:name=".MainActivity" android:exported="true">
17:        <intent-filter>
18:          <action android:name="android.intent.action.MAIN" />
19:          <category android:name="android.intent.category.LAUNCHER" />
20:        </intent-filter>
21:      </activity>
22:    </application>
23: </manifest>
```

blank (Page 1 of 1)

```
1: This page is intentionally blank.
```

PennyAndroid App: Defining

- **AndroidManifest.xml**
 - `<uses-permission>` elements give app permission to access Internet

Agenda

- Aside: Function def expressions
- Aside: Multithreaded programming
- Mobile programming
- PennyAndroid app: defining
- **PennyAndroid app: running**
- Pennylos app: defining
- Pennylos app: running

PennyAndroid App: Running

- To run an Android app...
- Option 1:
 - Use an Android **device**
 - Pro: Fast
- Option 2:
 - Create an Android **virtual device**
 - Run it (on any computer) using the **Android emulator**
 - Pro: Convenient

PennyAndroid App: Running

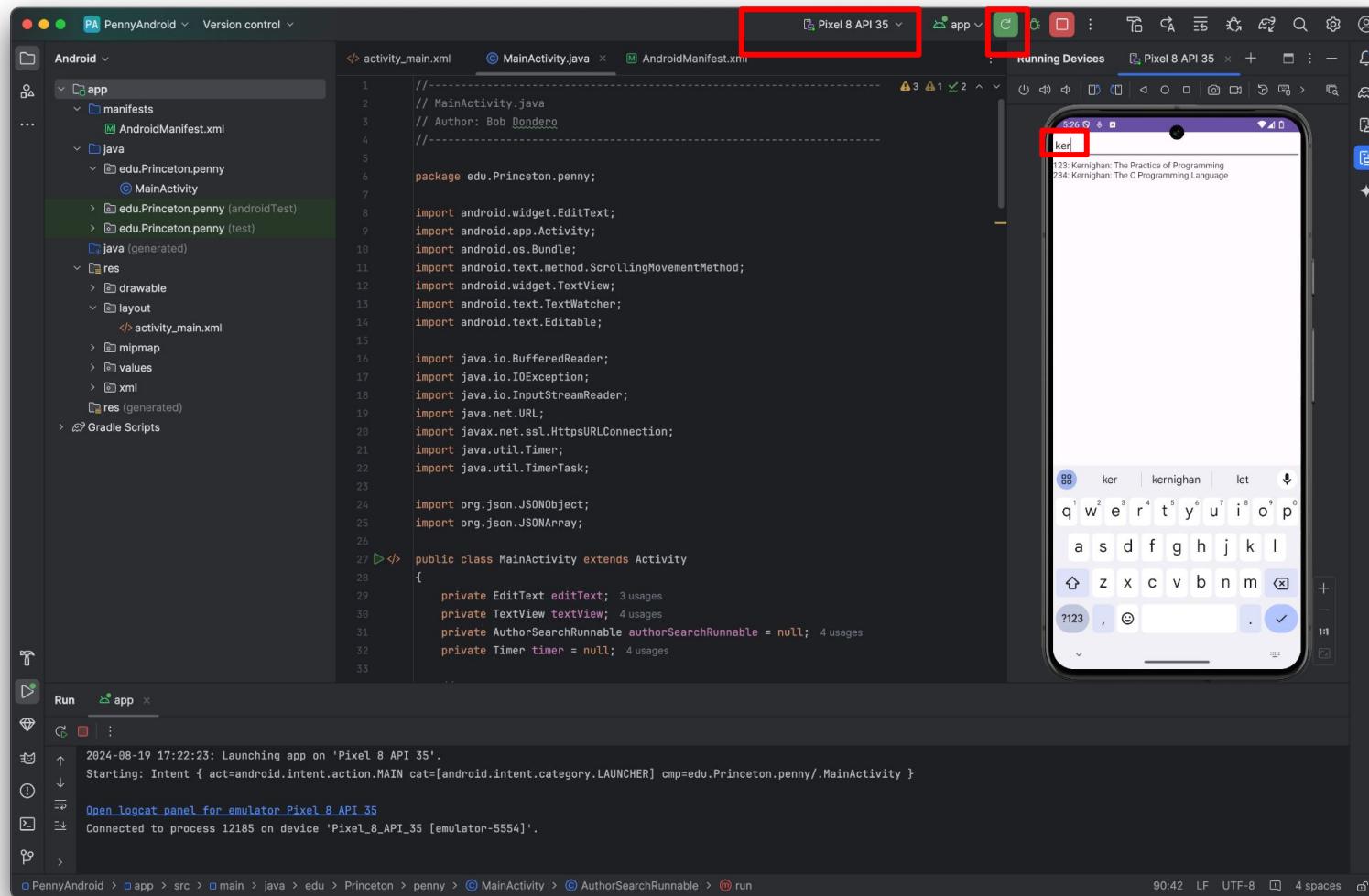
- To run on an Android emulator...

PennyAndroid App: Running

- Create an Android virtual device
 - In Android Studio
 - From the menu bar click *Tools*
 - Click *DeviceManager*
 - In the *Device Manager* panel,
 - Click the “+” button
 - Click *Create Virtual Device*
 - Click *Pixel 8*; click *Next*
 - Click *API 35*; click *Next*
 - Use the default *PIXEL 8 API 35* name
 - Click *Finish*

PennyAndroid App: Running

Select the Pixel 8 API 35 emulator; click *run* button; type an author!

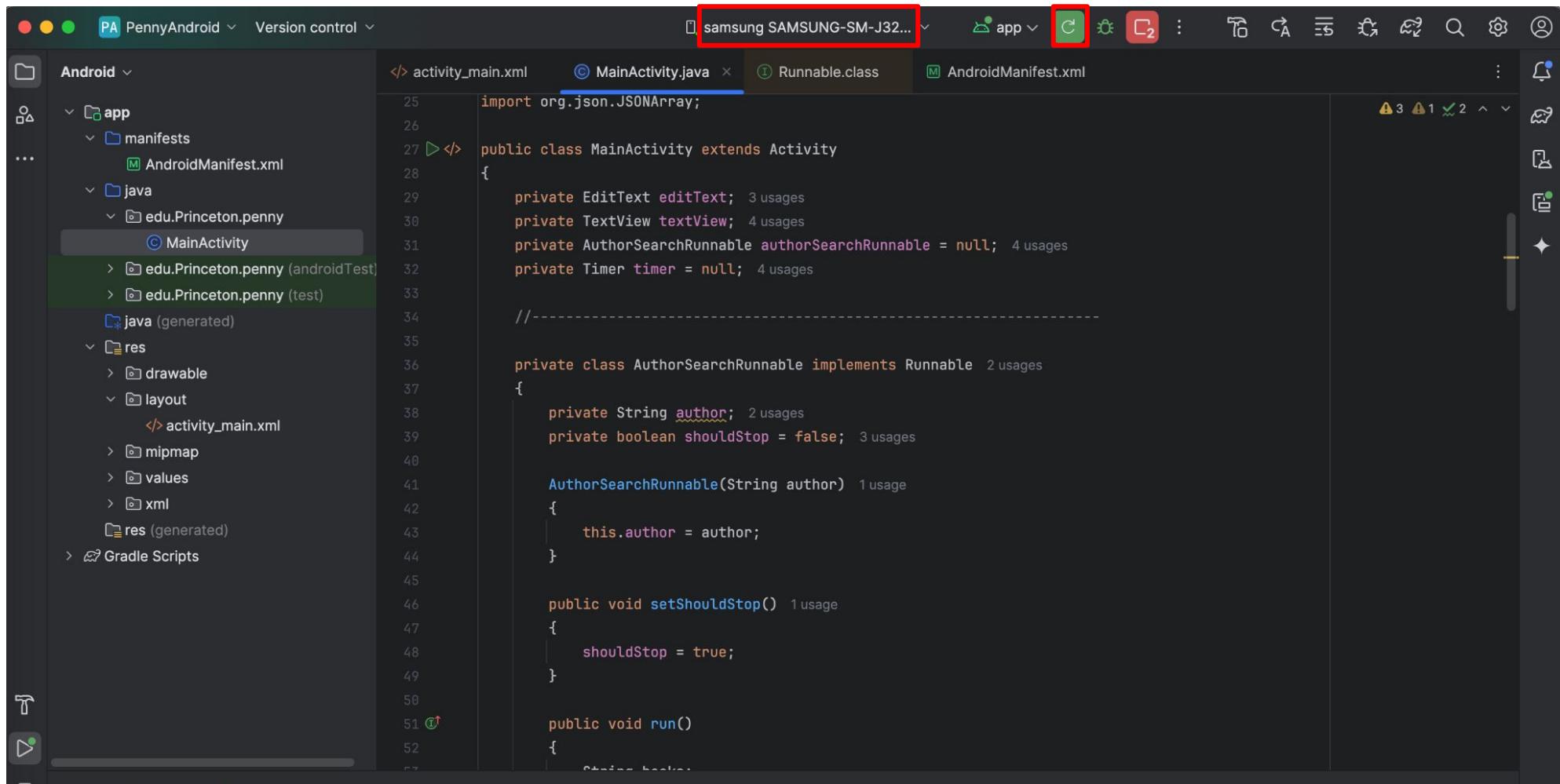


PennyAndroid App: Running

- To run on your Android phone...
 - Attach your Android phone to your computer's USB port
 - Respond to messages on your phone
 - Make sure your computer can access files stored on your phone

PennyAndroid App: Running

Select your phone; click on *run* button; type an author!



The screenshot shows the Android Studio interface with the following details:

- Title Bar:** Shows the project name "PennyAndroid" and a "Version control" dropdown.
- Device Selection:** A dropdown menu for selecting a device, currently set to "samsung SAMSUNG-SM-J32...".
- Run Button:** A green triangle icon representing the "Run" button, located in the top right of the toolbar, which is also highlighted with a red box.
- Project Structure:** On the left, the project tree shows the "Android" view with the "app" module expanded. It contains "manifests", "java" (with "edu.Princeton.penny" selected), "res", and "Gradle Scripts".
- Main Activity Code:** The main content area displays the code for "MainActivity.java". The code defines a class that extends Activity and implements Runnable. It includes fields for EditText, TextView, AuthorSearchRunnable, and Timer, along with methods for setting an author and running the search.

PennyAndroid App: Running

- To learn more about Android programming:
 - <https://developer.android.com/guide>

Agenda

- Aside: Function def expressions
- Aside: Multithreaded programming
- Mobile programming
- PennyAndroid app: defining
- PennyAndroid app: running
- **Pennylos app: defining**
- Pennylos app: running

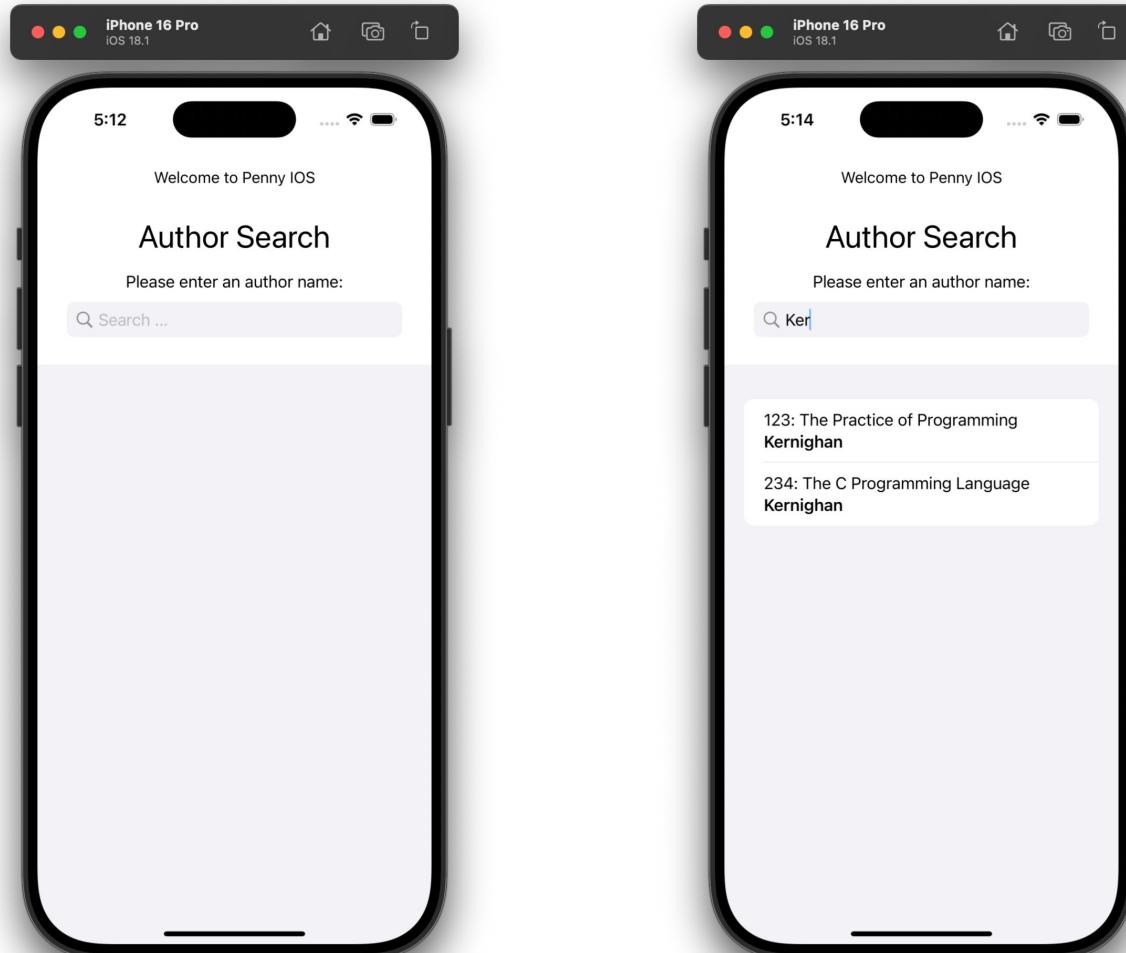
Thanks to
Katie DiPaola ('26)...

PennyAndroid App: Defining

- Preliminary
 - Deploy PennyJson server to
<https://pennyjson.onrender.com>
 - So Pennylos client can access it

Pennylos App: Defining

The
goal:



An iOS client for the PennyJson server

Pennylos App: Defining

- Download and install **XCode**

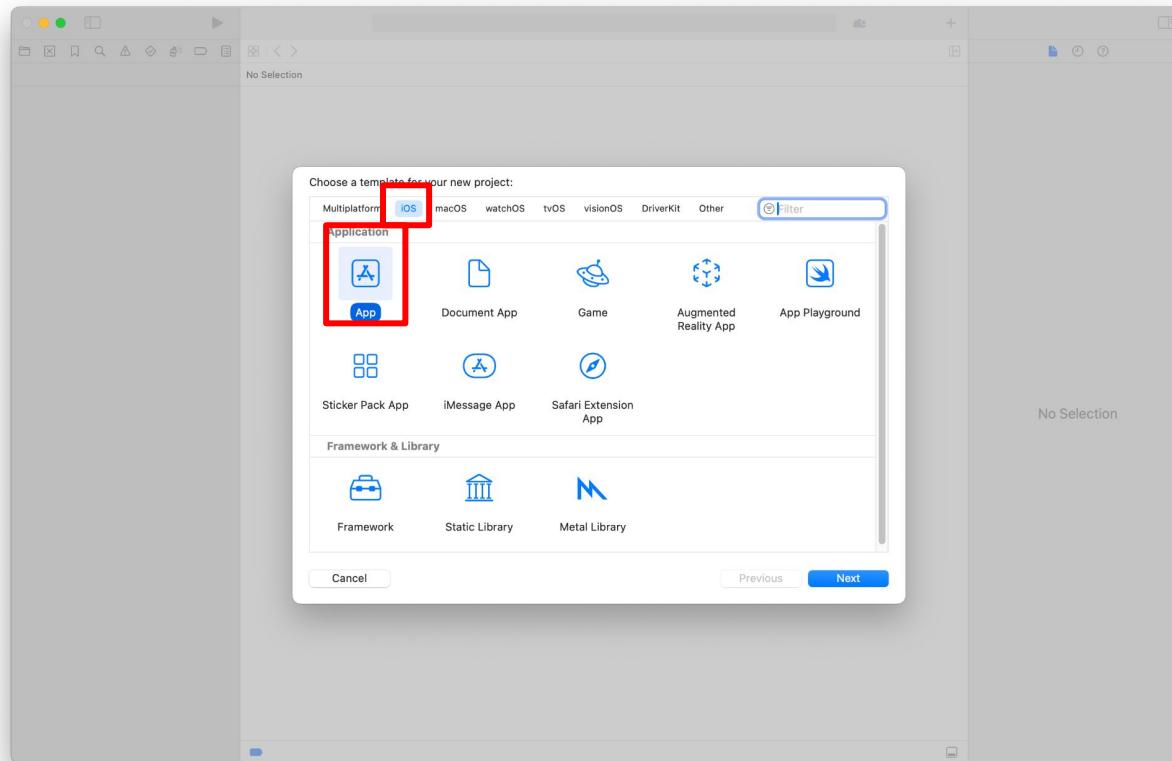
Pennylos App: Defining

Launch XCode; select *Create New Project...*



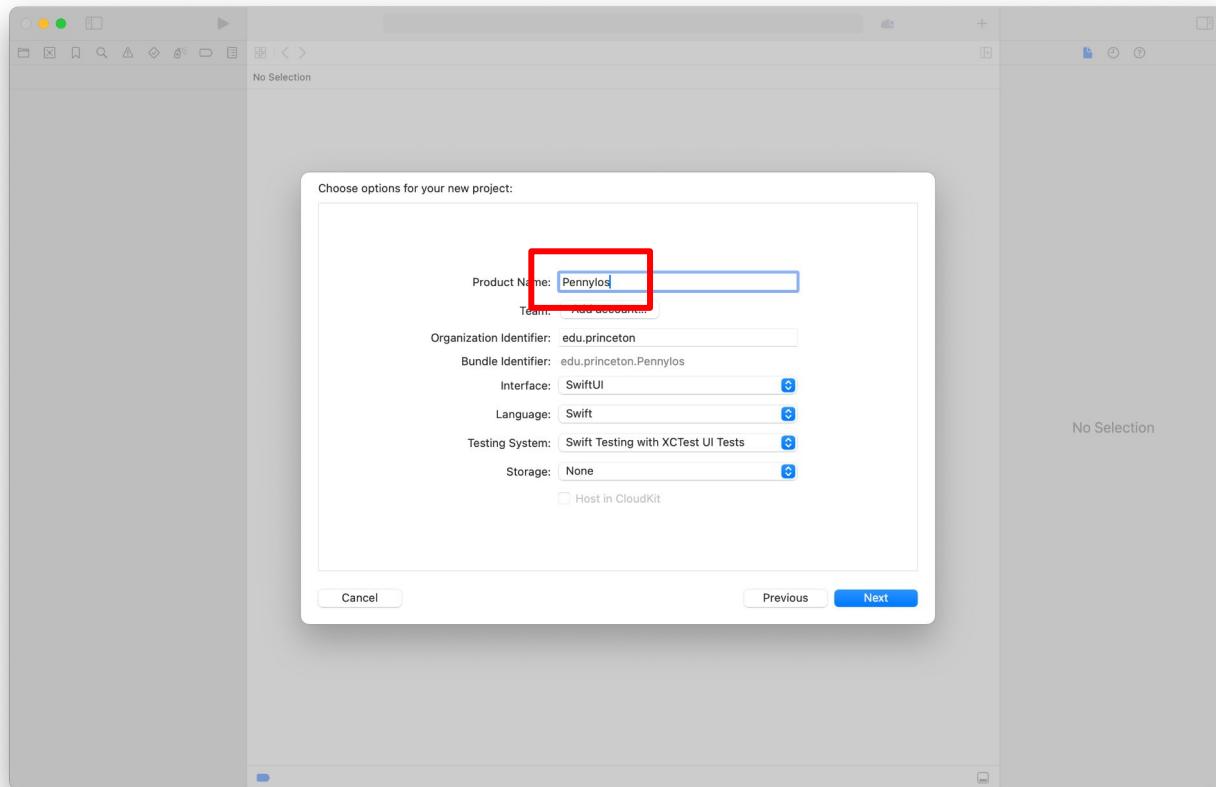
Pennylos App: Defining

Select *iOS, App*; click on *Next*



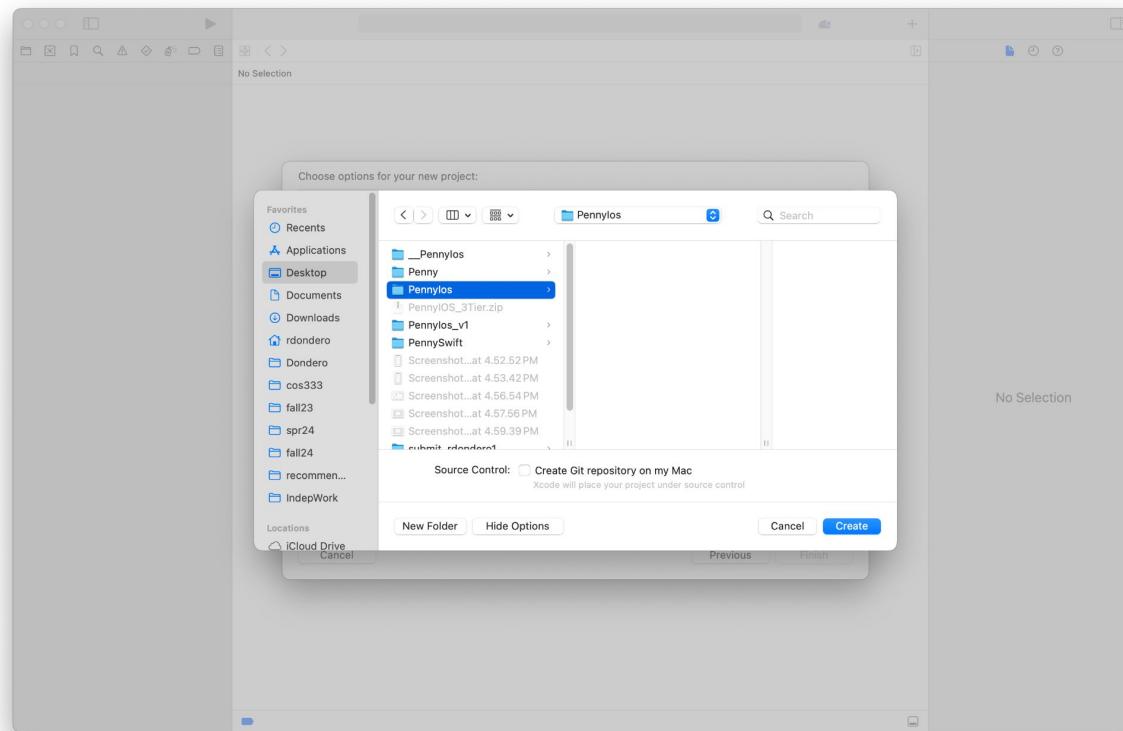
Pennylos App: Defining

For Product Name enter PennyIos; click on Next



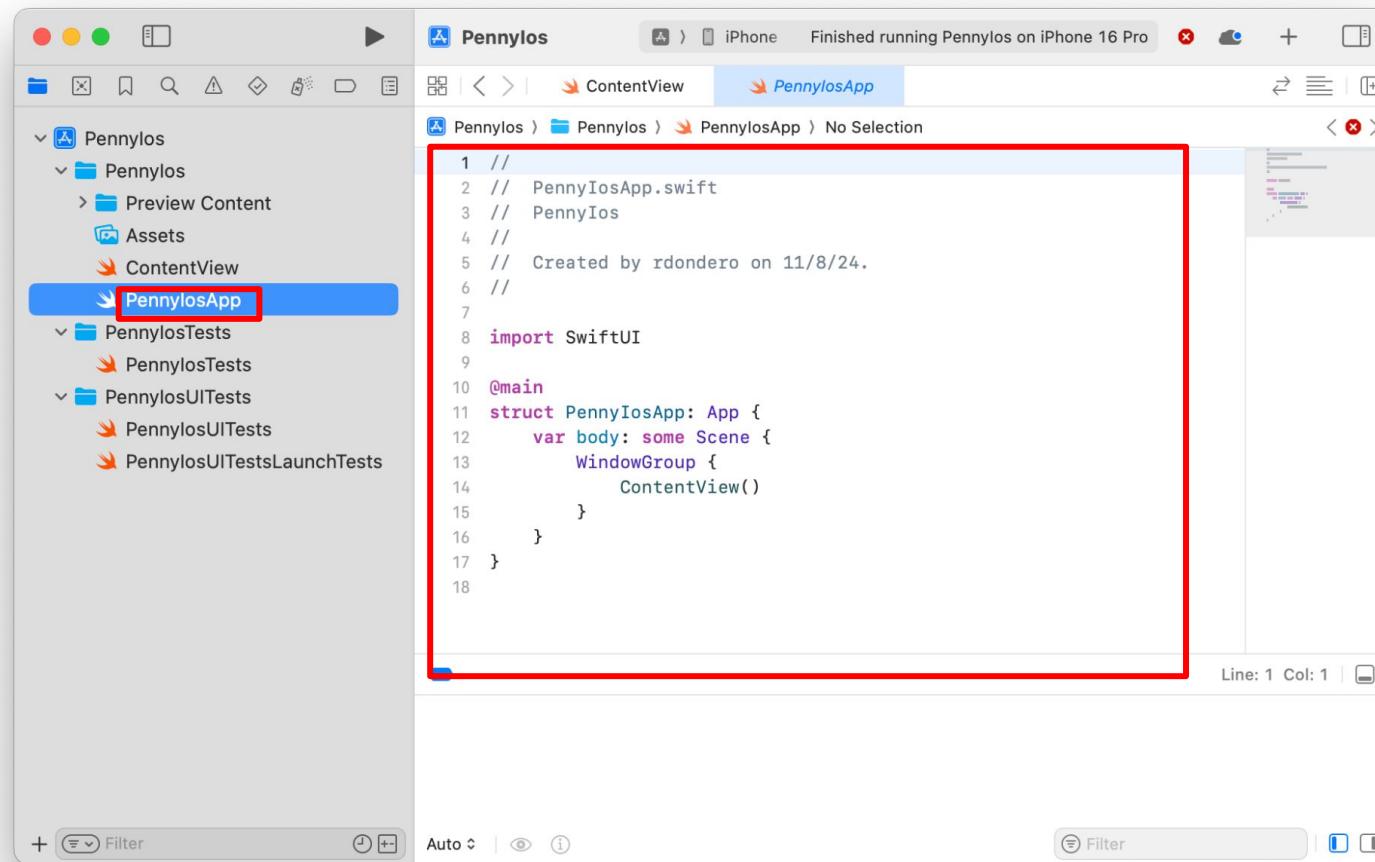
Pennylos App: Defining

Name a directory in which the app should be stored; click on *Create*



Pennylos App: Defining

Click on *PennylosApp*; examine the generated code



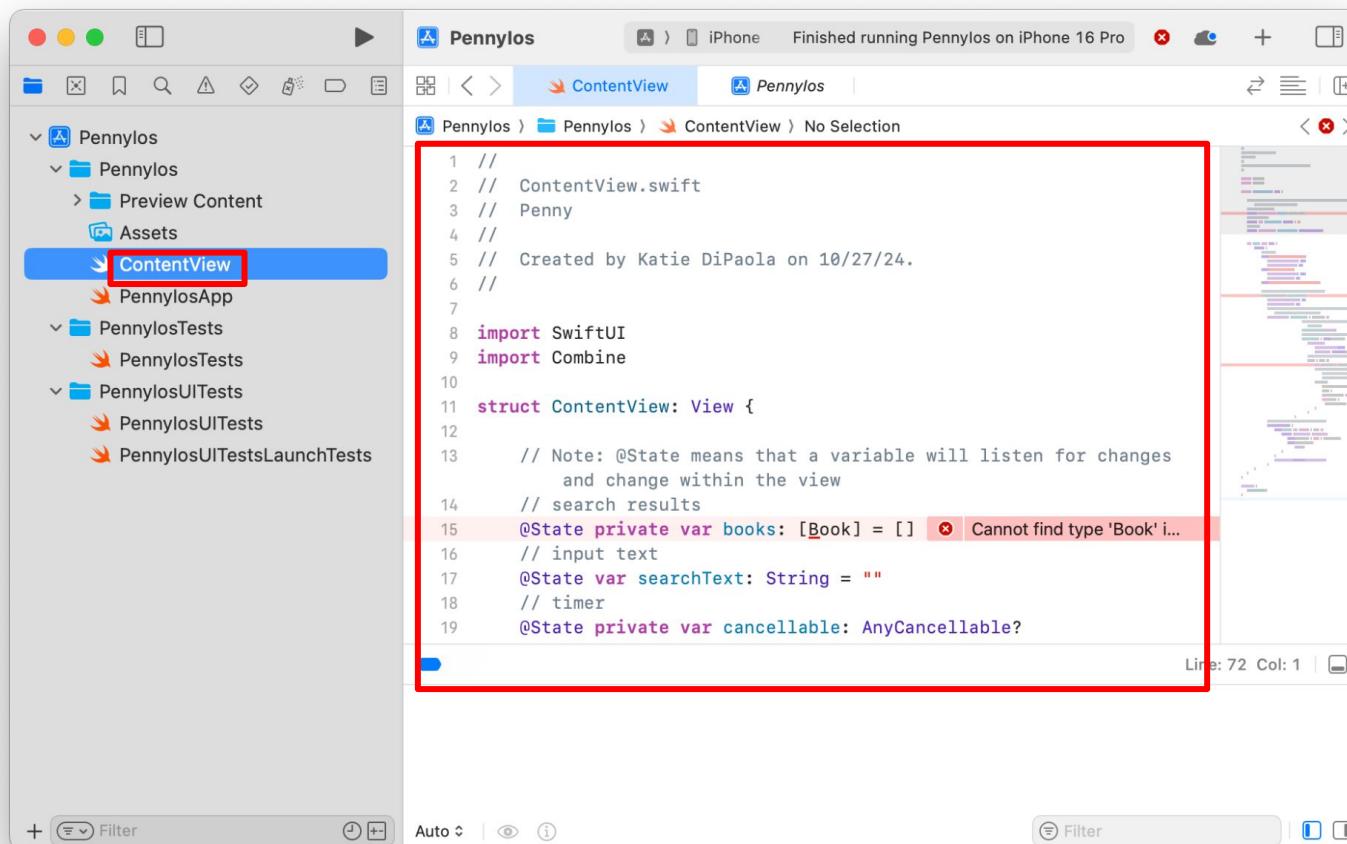
The screenshot shows the Xcode interface with the project 'Pennylos' open. The left sidebar displays the project structure, including 'Pennylos', 'Preview Content', 'Assets', 'ContentView', and 'PennylosApp'. The 'PennylosApp' item is selected and highlighted with a blue background. The main editor area shows the generated Swift code for 'PennylosApp.swift':

```
1 //  
2 // PennyIosApp.swift  
3 // PennyIos  
4 //  
5 // Created by rdondoro on 11/8/24.  
6 //  
7  
8 import SwiftUI  
9  
10 @main  
11 struct PennyIosApp: App {  
12     var body: some Scene {  
13         WindowGroup {  
14             ContentView()  
15         }  
16     }  
17 }  
18
```

A red rectangular box highlights the entire code block in the editor. The status bar at the bottom right of the editor shows 'Line: 1 Col: 1'.

Pennylos App: Defining

Click on *ContentView*; copy/paste the given code

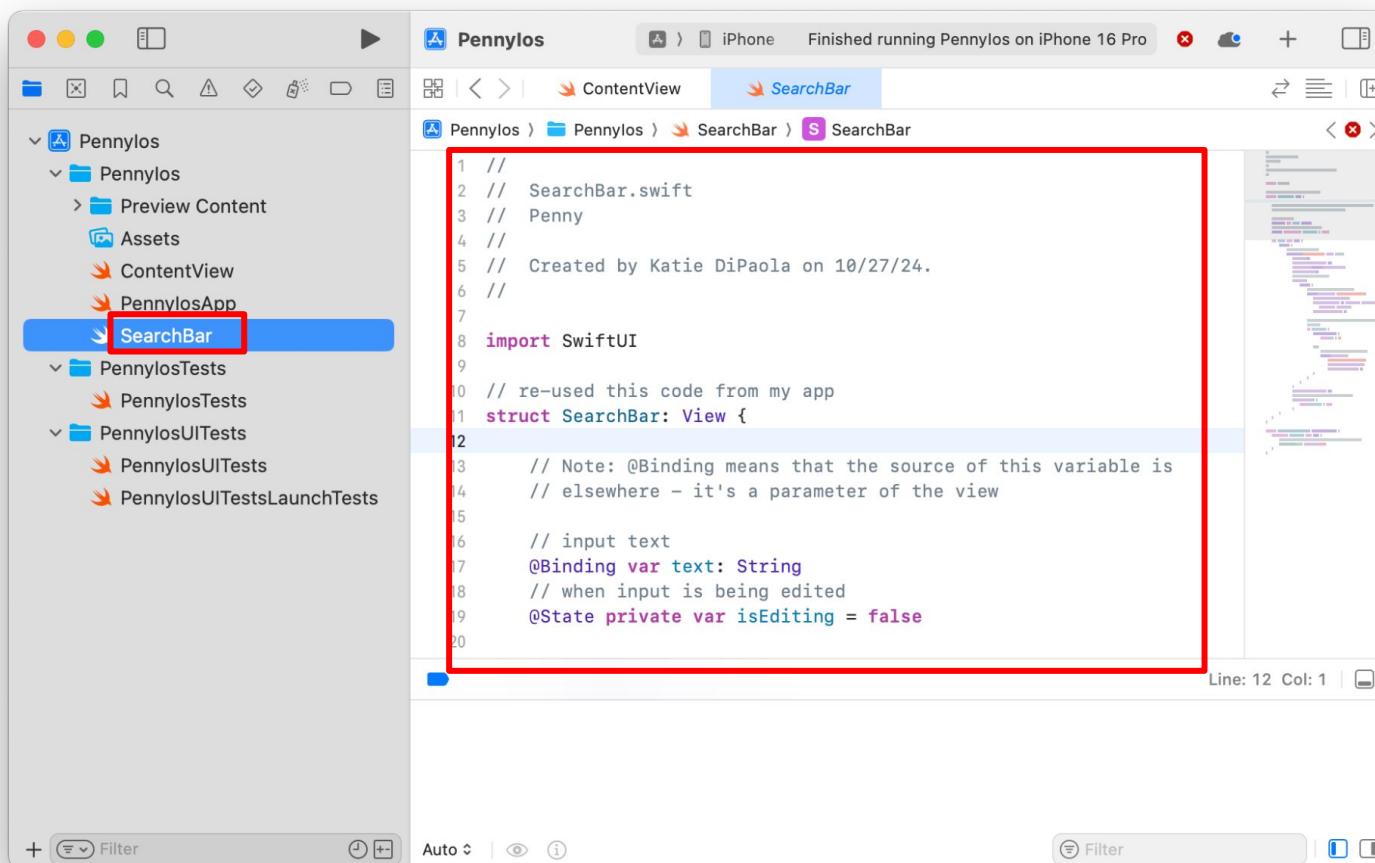


The screenshot shows the Xcode interface with the project 'Pennylos' open. The left sidebar displays the project structure, including 'ContentView' which is selected and highlighted with a blue background. The main editor area shows the 'ContentView.swift' file content. A red rectangular box highlights the entire code block in the editor. The code defines a struct ContentView that contains state variables for books, searchText, and cancellable, along with imports for SwiftUI and Combine.

```
1 //  
2 // ContentView.swift  
3 // Penny  
4 //  
5 // Created by Katie DiPaola on 10/27/24.  
6 //  
7  
8 import SwiftUI  
9 import Combine  
10  
11 struct ContentView: View {  
12  
13     // Note: @State means that a variable will listen for changes  
     // and change within the view  
14     // search results  
15     @State private var books: [Book] = [] ✖ Cannot find type 'Book' i...  
16     // input text  
17     @State var searchText: String = ""  
18     // timer  
19     @State private var cancellable: AnyCancellable?
```

Pennylos App: Defining

Create new file *SearchBar*; copy/paste the given code



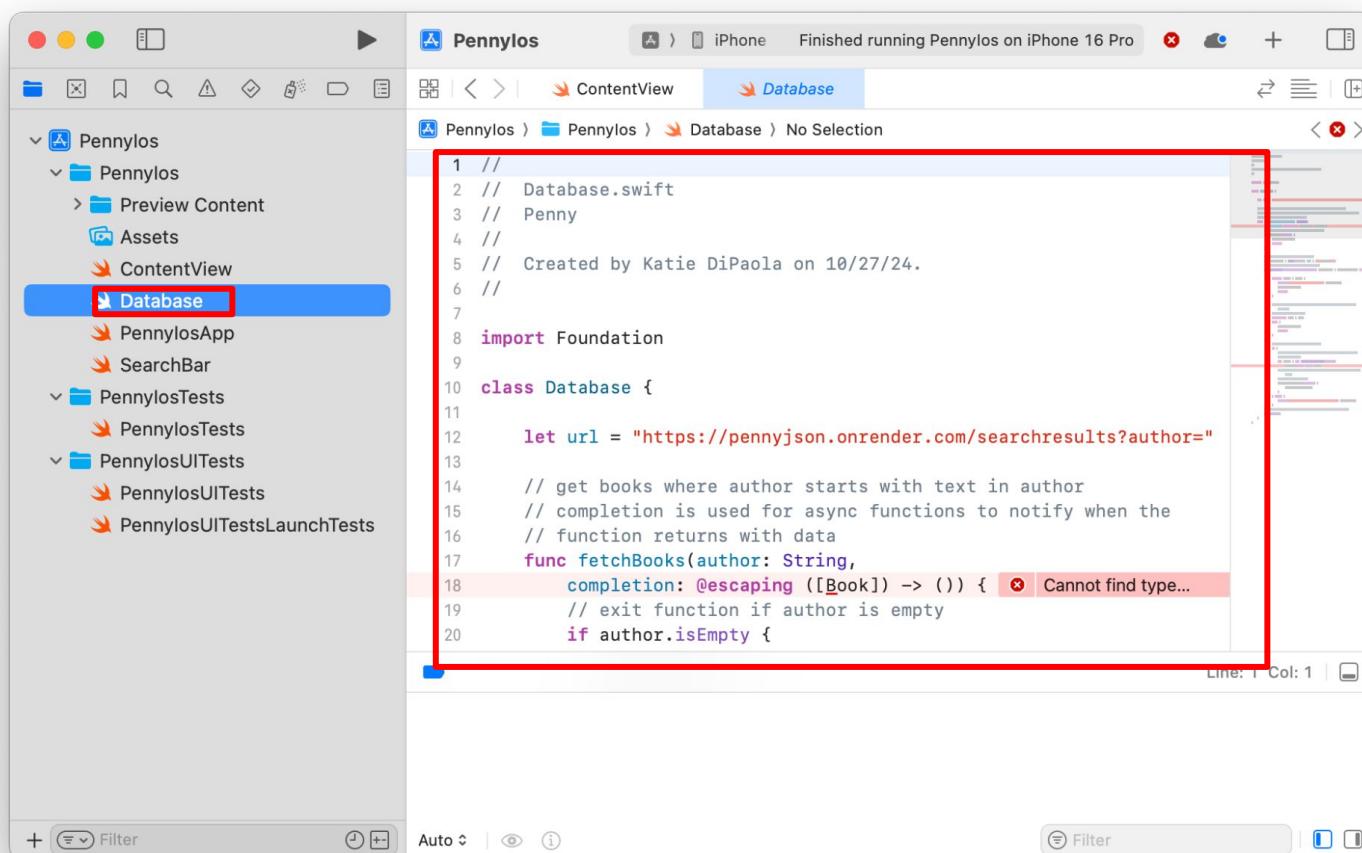
The screenshot shows the Xcode interface with the project 'Pennylos' open. The left sidebar displays the project structure, including 'Pennylos', 'Preview Content', 'Assets', 'ContentView', 'PennylosApp', and 'SearchBar'. The 'SearchBar' folder is selected and highlighted with a blue bar at the top of the sidebar. The main editor area shows the 'SearchBar.swift' file, which contains the following code:

```
1 //  
2 //  SearchBar.swift  
3 //  Penny  
4 //  
5 //  Created by Katie DiPaola on 10/27/24.  
6 //  
7  
8 import SwiftUI  
9  
10 // re-used this code from my app  
11 struct SearchBar: View {  
12  
13     // Note: @Binding means that the source of this variable is  
14     // elsewhere - it's a parameter of the view  
15  
16     // input text  
17     @Binding var text: String  
18     // when input is being edited  
19     @State private var isEditing = false  
20 }
```

A large red rectangle highlights the entire code block in the editor. The status bar at the bottom right of the editor indicates 'Line: 12 Col: 1'.

Pennylos App: Defining

Create new file *Database*; copy/paste the given code



The screenshot shows the Xcode interface with the following details:

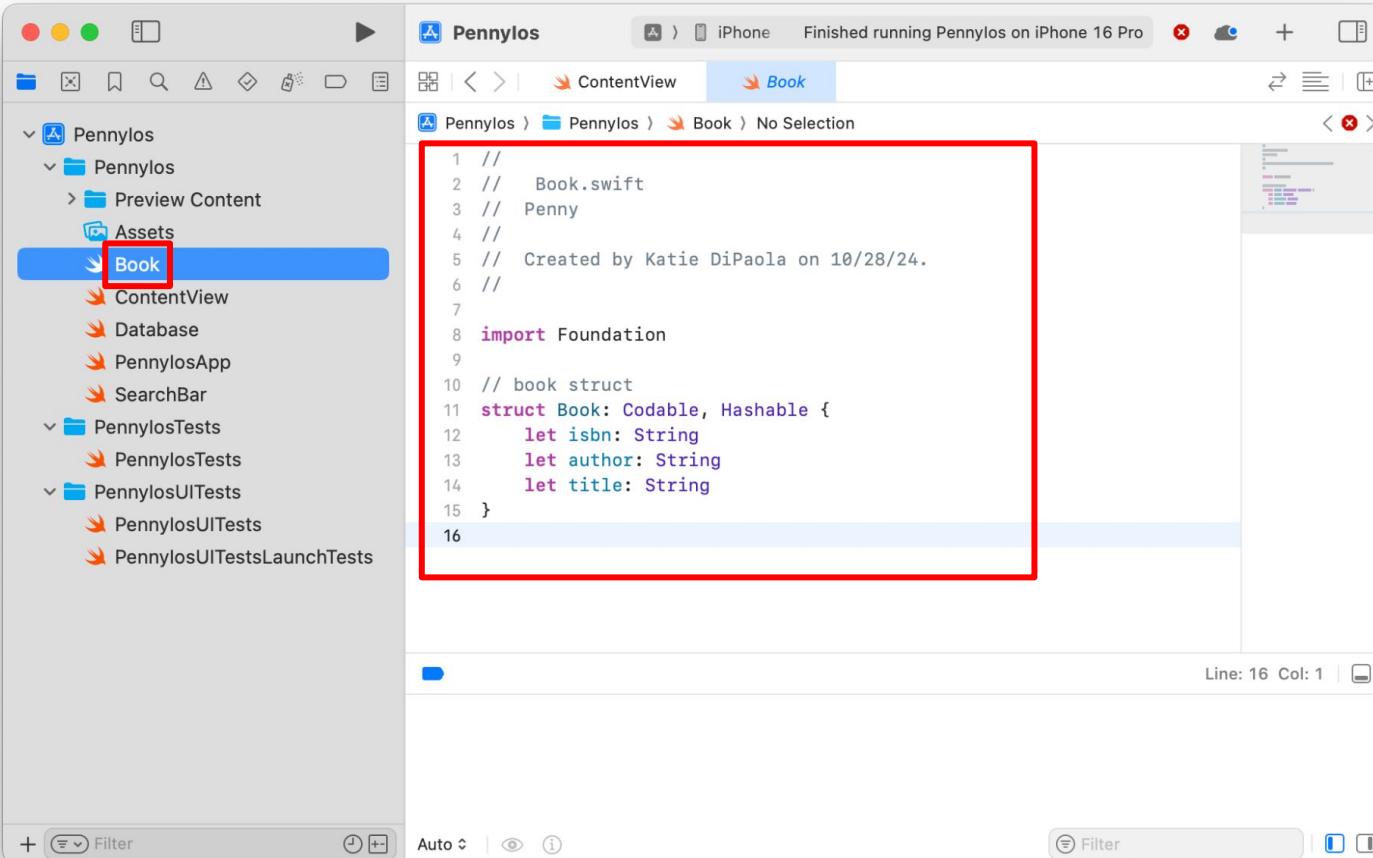
- Project Navigator:** Shows the project structure under "Pennylos". The "Database" file is selected and highlighted with a blue bar.
- Editor:** Displays the contents of the "Database.swift" file. The code is as follows:

```
1 //  
2 //  Database.swift  
3 //  Penny  
4 //  
5 //  Created by Katie DiPaola on 10/27/24.  
6 //  
7  
8 import Foundation  
9  
10 class Database {  
11  
12     let url = "https://pennyjson.onrender.com/searchresults?author="  
13  
14     // get books where author starts with text in author  
15     // completion is used for async functions to notify when the  
16     // function returns with data  
17     func fetchBooks(author: String,  
18                     completion: @escaping ([Book] -> ()) {  
19         // exit function if author is empty  
20         if author.isEmpty {  
21             completion([])  
22         } else {  
23             // Fetch books from API  
24             // ...  
25         }  
26     }  
27 }
```

A red rectangle highlights the entire code block in the editor.

Pennylos App: Defining

Create new file *Book*; copy/paste the given code



The screenshot shows the Xcode interface with the project 'Pennylos' open. In the Project Navigator on the left, there is a folder named 'Assets' containing a file named 'Book'. A red box highlights this 'Book' file. The main editor area shows the code for 'Book.swift'. The code defines a struct 'Book' with properties for isbn, author, and title.

```
1 //  
2 //  Book.swift  
3 //  Penny  
4 //  
5 //  Created by Katie DiPaola on 10/28/24.  
6 //  
7  
8 import Foundation  
9  
10 // book struct  
11 struct Book: Codable, Hashable {  
12     let isbn: String  
13     let author: String  
14     let title: String  
15 }  
16
```

PennyIos/PennyIosApp.swift (Page 1 of 1)

```
1: //  
2: //  PennyIosApp.swift  
3: //  PennyIos  
4: //  
5: //  Created by rdondoro on 11/8/24.  
6: //  
7:  
8: import SwiftUI  
9:  
10: @main  
11: struct PennyIosApp: App {  
12:     var body: some Scene {  
13:         WindowGroup {  
14:             ContentView()  
15:         }  
16:     }  
17: }
```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyIos/ContentView.swift (Page 1 of 2)

```

1: // ContentView.swift
2: // Penny
3: // Created by Katie DiPaola on 10/27/24.
4: //
5: // import SwiftUI
6: // import Combine
7:
8: import SwiftUI
9: import Combine
10:
11: struct ContentView: View {
12:
13:     // Note: @State means that a variable will listen for changes and
14:     // change within the view search results
15:     @State private var books: [Book] = []
16:     // input text
17:     @State var searchText: String = ""
18:     // timer
19:     @State private var cancellable: AnyCancellable?
20:
21:
22:     var body: some View {
23:         VStack {
24:             // header
25:             Text("Welcome to Penny IOS")
26:                 .font(.system(size: 16))
27:                 .padding(.vertical, 20)
28:             Text("Author Search")
29:                 .font(.system(size: 32))
30:                 .padding(.bottom, 10)
31:             Text("Please enter an author name:")
32:
33:             // searchBar view from SearchBar.swift
34:             searchBar(text: $searchText)
35:                 .padding(.horizontal, 20)
36:                 .padding(.bottom, 20)
37:                 // when text is entered (aka input changes), delay and
38:                 // then fetch from database
39:                 .onChange(of: searchText) { newValue in
40:                     // Cancel the previous timer if it exists
41:                     cancellable?.cancel()
42:                     // Start a new timer for delay
43:                     cancellable = Just(newValue)
44:                         .delay(for: .milliseconds(500),
45:                               scheduler: RunLoop.main)
46:                         // fetch books from database
47:                         .sink { text in
48:                             let database = Database()
49:                             // handle completion (aka await return from
50:                             // fetchBooks)
51:                             database.fetchBooks(author: text) {
52:                                 fetched_books in
53:                                     self.books = fetched_books
54:                             }
55:                         }
56:                     }
57:                     // display result of current search
58:                     NavigationView {
59:                         List(books, id: \.self) { book in
60:                             VStack (alignment: .leading){
61:                                 Text(book.isbn + ": " + book.title)
62:                                 Text(book.author)
63:                                     .bold()
64:                             }
65:                         }
66:                         .background(Color(.systemGray6))
67:                     }
68:                 }
69:             }
70:         }
71:
72: #Preview {
73:     ContentView()
74: }
```

PennyIos/ContentView.swift (Page 2 of 2)

```

66:                         .background(Color(.systemGray6))
67:                     }
68:                 }
69:             }
70:         }
71:
72: #Preview {
73:     ContentView()
74: }
```

PennyIos/SearchBar.swift (Page 1 of 2)

```

1: //  

2: //  SearchBar.swift  

3: //  Penny  

4: //  

5: //  Created by Katie DiPaola on 10/27/24.  

6: //  

7:  

8: import SwiftUI  

9:  

10: // re-used this code from my app  

11: struct SearchBar: View {  

12:  

13:     // Note: @Binding means that the source of this variable is  

14:     // elsewhere - it's a parameter of the view  

15:  

16:     // input text  

17:     @Binding var text: String  

18:     // when input is being edited  

19:     @State private var isEditing = false  

20:  

21:     var body: some View {  

22:         HStack {  

23:             // input text field  

24:             TextField("Search ...", text: $text)  

25:                 .padding(7)  

26:                 .padding(.horizontal, 25)  

27:                 .background(Color(.systemGray6))  

28:                 .cornerRadius(8)  

29:                 // make it look pretty  

30:                 .overlay(  

31:                     HStack {  

32:                         // uses a build-in IOS image  

33:                         Image(systemName: "magnifyingglass")  

34:                             .foregroundColor(.gray)  

35:                             .frame(minWidth: 0, maxWidth: .infinity,  

36:                                 alignment: .leading)  

37:                             .padding(.leading, 8)  

38:  

39:                         // provide functionality to delete full user  

40:                         // input  

41:                         if isEditing {  

42:                             Button(action: {  

43:                                 self.text = ""  

44:  

45:                             }) {  

46:                                 // uses a build-in IOS image  

47:                                 Image(systemName:  

48:                                     "multiply.circle.fill")  

49:                                     .foregroundColor(.gray)  

50:                                     .padding(.trailing, 8)  

51:                             }  

52:                         }  

53:                     }  

54:                 )  

55:                 .padding(.horizontal, 10)  

56:                 // track when user is editing input  

57:                 .onTapGesture {  

58:                     self.isEditing = true  

59:                 }  

60:             }  

61:         }  

62:     }  

63:  

64: struct SearchBar_Previews: PreviewProvider {  

65:     static var previews: some View {
```

PennyIos/SearchBar.swift (Page 2 of 2)

```

66:         // view paramter (@Binding variable from above!)  

67:         searchBar(text: .constant(""))  

68:     }  

69: }
```

PennyIos/Database.swift (Page 1 of 1)

```

1: //
2: //  Database.swift
3: //  Penny
4: //
5: //  Created by Katie DiPaola on 10/27/24.
6: //
7: 
8: import Foundation
9: 
10: class Database {
11: 
12:     let url = "https://pennyjson.onrender.com/searchresults?author="
13: 
14:     // get books where author starts with text in author
15:     // completion is used for async functions to notify when the
16:     // function returns with data
17:     func fetchBooks(author: String,
18:                     completion: @escaping ([Book]) -> ()) {
19:         // exit function if author is empty
20:         if author.isEmpty {
21:             completion([])
22:             return
23:         }
24: 
25:         // add author to database url
26:         let queryUrl = URL(string: url + "\\(author)")!
27:         // make asynchronous http request
28:         URLSession.shared.dataTask(with: queryUrl) { data, _, error in
29: 
30:             if let error = error {
31:                 print("Error fetching books: \(error)")
32:                 completion([])
33:                 return
34:             }
35: 
36:             // guard lets us check if an optional value exists without
37:             // throwing an error
38:             guard let data = data
39:             else {
40:                 completion([])
41:                 return
42:             }
43: 
44:             // otherwise handle json data
45:             do {
46:                 // decode into list of Book objects (defined in
47:                 // Book.swift)
48:                 let books = try JSONDecoder().decode(
49:                     [Book].self, from: data)
50:                 // completion on the main thread (UI updates must occur
51:                 // on main thread)
52:                 DispatchQueue.main.async {
53:                     completion(books)
54:                 }
55:             } catch {
56:                 print("Error decoding JSON response: \(error)")
57:             }
58:             // start task (send http request we defined above)
59:             .resume()
60:         }
61:     }
}

```

PennyIos/Book.swift (Page 1 of 1)

```

1: //
2: //  Book.swift
3: //  Penny
4: //
5: //  Created by Katie DiPaola on 10/28/24.
6: //
7: 
8: import Foundation
9: 
10: // book struct
11: struct Book: Codable, Hashable {
12:     let isbn: String
13:     let author: String
14:     let title: String
15: }

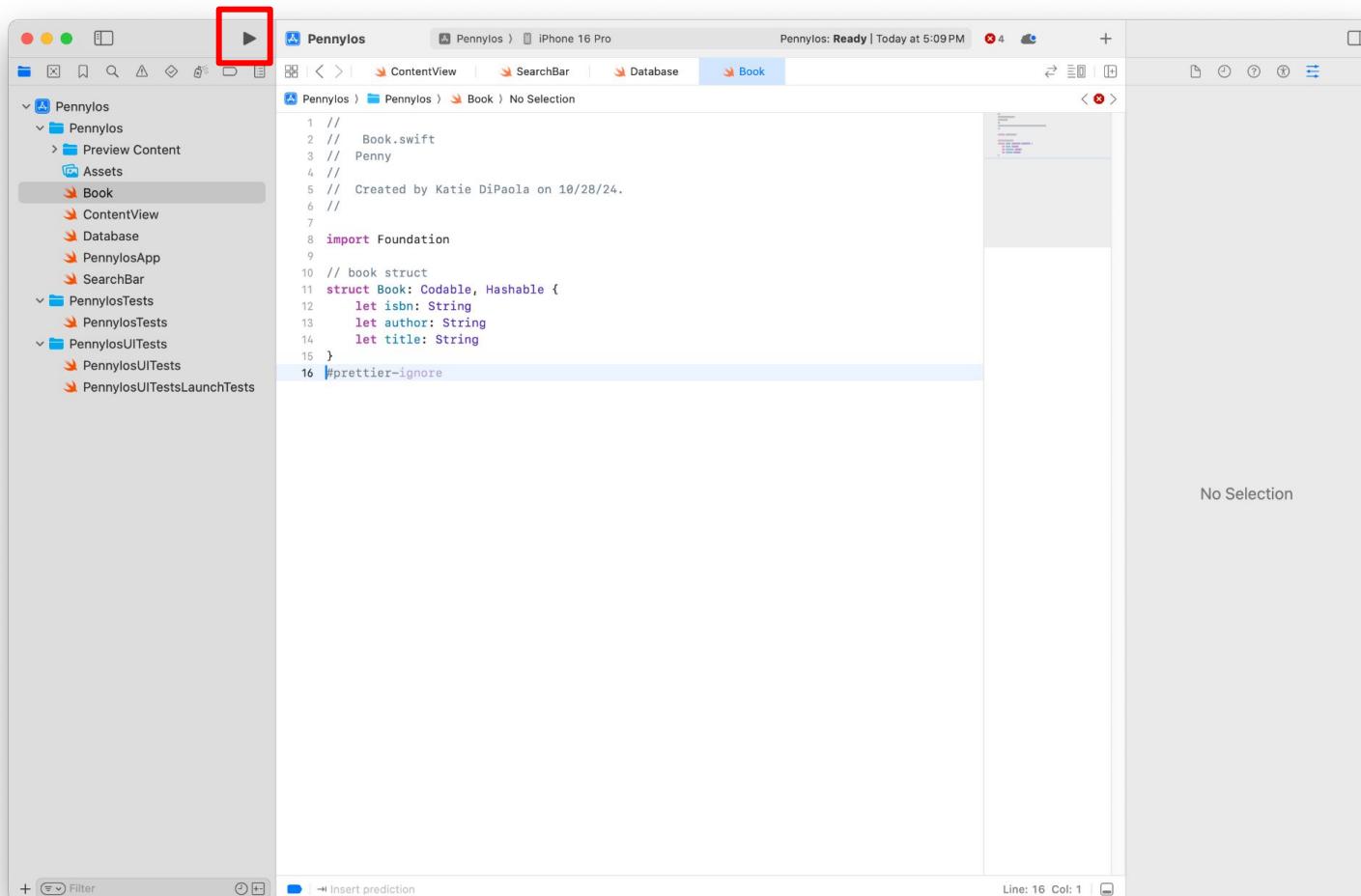
```

Agenda

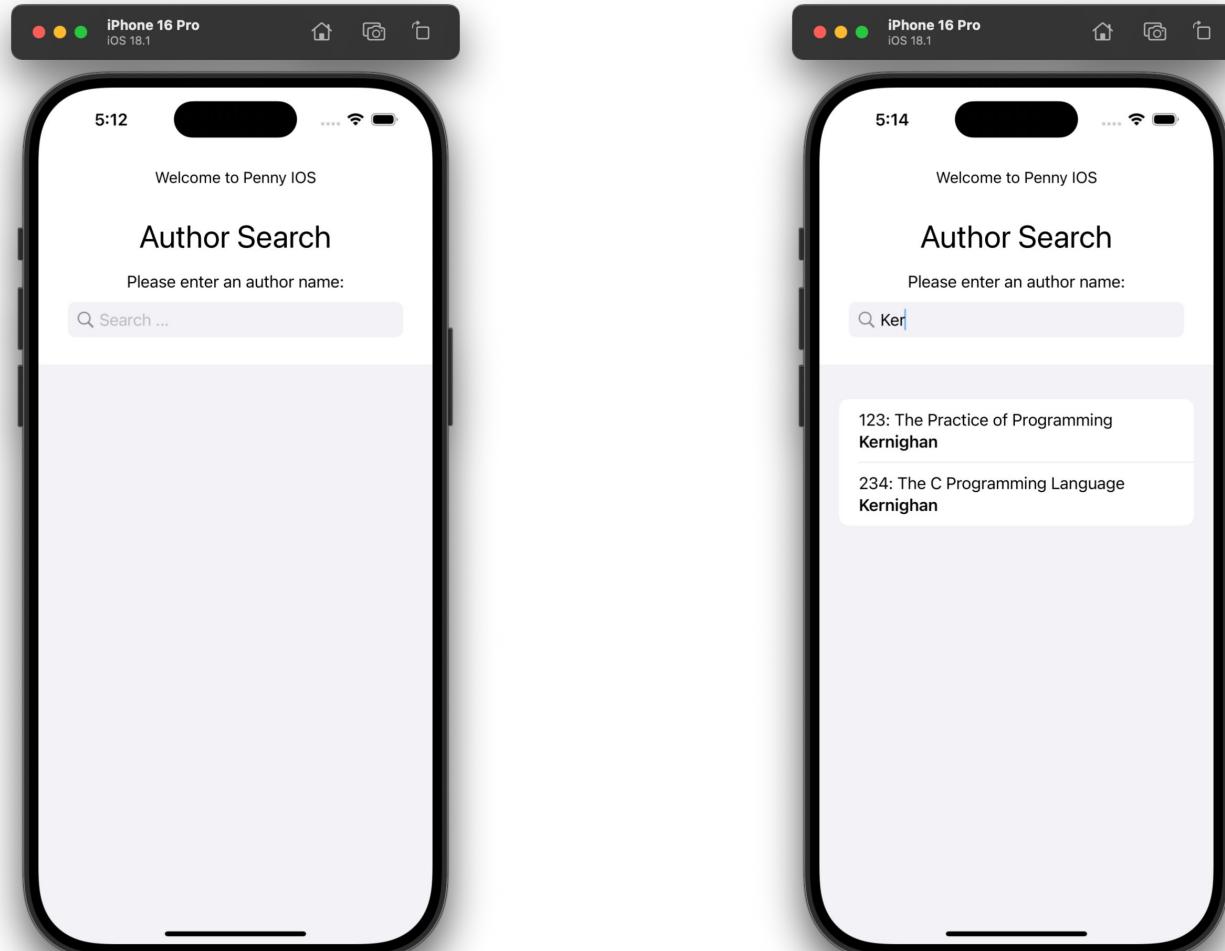
- Aside: Function def expressions
- Aside: Multithreaded programming
- Mobile programming
- PennyAndroid app: defining
- PennyAndroid app: running
- Pennylos app: defining
- **Pennylos app: running**

Pennylos App: Running

Click on the *Build/Run* button



Pennylos App: Running



Additional Resources

- Swift Basics Guide
 - <https://guides.codepath.com/ios/Swift-Basics>
- Uploading to the iOS App Store
 - <https://christinesun.notion.site/christinesun/How-to-get-your-app-onto-the-iOS-app-store-018bcd0de6434878acb81c527f2efcb7>

Our thanks go to **Katie DiPaola** ('26) for contributing the Pennylos application, and to **Christine Sun** ('24) for contributing a prior (now outdated) version

Summary

- We have covered:
 - Mobile programming
 - Android mobile programming
 - iOS mobile programming
- See also:
 - **Appendix 1:** Java Multithreaded Programming
 - **Appendix 2:** Cross-Platform Frameworks

Java Multithreaded Programming

- Recall race.py

```
$ python race.py  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
8  
6  
4  
2  
0  
Final balance: 0  
$
```

```
$ python race.py  
1  
2  
3  
4  
-1  
5  
6  
-3  
-5  
-7  
-9  
7  
8  
9  
10  
Final balance: 10  
$
```

```
$ python race.py  
1  
2  
3  
4  
5  
6  
7  
8  
9  
6  
4  
10  
2  
0  
-2  
Final balance: -2  
$
```

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: }

```

Java Multithreaded Programming

- See Race.java

```
$ java Race
1
2
3
4
5
6
7
8
0
-2
-4
-6
-8
9
10
Final balance: 10
$
```

```
$ java Race
1
2
3
4
5
6
7
8
0
-2
-4
9
10
-6
-8
Final balance: -10
$
```

Java Multithreaded Programming

- Recall lockinuser.py

```
$ python lockinuser.py  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
8  
6  
4  
2  
0  
Final balance: 0  
$
```

```
$ python lockinuser.py  
1  
2  
3  
4  
2  
0  
-2  
-4  
-6  
-5  
-4  
-3  
-2  
-1  
0  
Final balance: 0  
$
```

Threads/lockinuser.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # lockinuser.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, bank_acct_lock):
37:         threading.Thread.__init__(self)
38:         self._bank_acct = bank_acct
39:         self._bank_acct_lock = bank_acct_lock
40:
41:     def run(self):
42:         for _ in range(10):
43:             self._bank_acct_lock.acquire()
44:             self._bank_acct.deposit(1)
45:             self._bank_acct_lock.release()
46:
47: #-----
48:
49: class WithdrawThread (threading.Thread):
50:
51:     def __init__(self, bank_acct, bank_acct_lock):
52:         threading.Thread.__init__(self)
53:         self._bank_acct = bank_acct
54:         self._bank_acct_lock = bank_acct_lock
55:
56:     def run(self):
57:         for _ in range(5):
58:             self._bank_acct_lock.acquire()
59:             self._bank_acct.withdraw(2)
60:             self._bank_acct_lock.release()
61:
62: #-----
63:
64: def main():
65:
```

Threads/lockinuser.py (Page 2 of 2)

```

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

Threads/LockInUser.java (Page 1 of 2)

```

1: //-----
2: // LockInUser.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(temp);
28:        balance = temp;
29:    }
30: }
31:
32: //-----
33: class DepositThread extends Thread
34: {
35:     private Bank bank;
36:
37:     public DepositThread(Bank bank)
38:     {
39:         this.bank = bank;
40:     }
41:
42:
43:     public void run()
44:     {
45:         for (int i = 0; i < 10; i++)
46:             synchronized (bank)
47:             {
48:                 bank.deposit(1);
49:             }
50:     }
51: }
52:
53: //-----
54:
55: class WithdrawThread extends Thread
56: {
57:     private Bank bank;
58:
59:     public WithdrawThread(Bank bank)
60:     {
61:         this.bank = bank;
62:     }
63:
64:     public void run()
65:     {

```

Threads/LockInUser.java (Page 2 of 2)

```

66:         for (int i = 0; i < 5; i++)
67:             synchronized (bank)
68:             {
69:                 bank.withdraw(2);
70:             }
71:         }
72:     }
73:
74: //-----
75:
76: public class LockInUser
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: }

```

Java Multithreaded Programming

- See LockInUser.java

```
$ java LockInUser  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
8  
6  
4  
2  
0  
Final balance: 0  
$
```

```
$ java LockInUser  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
8  
6  
4  
2  
0  
Final balance: 0  
$
```

Java Multithreaded Programming

- Recall lockinresource.py

```
$ python lockinresource.py  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
8  
6  
4  
2  
0  
Final balance: 0  
$
```

```
$ python lockinresource.py  
1  
2  
3  
1  
-1  
-3  
-5  
-7  
-6  
-5  
-4  
-3  
-2  
-1  
0  
Final balance: 0  
$
```

Threads/lockinresource.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # lockinresource.py
5: # Author: Bob Dondero
6: #-----
7:
8: import threading
9:
10:#-----
11:
12class 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:
43class 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:
55class 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/lockinresource.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/LockInResource.java (Page 1 of 2)

```

1: //-----
2: // LockInResource.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/LockInResource.java (Page 2 of 2)

```

66: }
67:
68: //-----
69:
70: public class LockInResource
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: }

```

Java Multithreaded Programming

- See LockInResource.java

```
$ java LockInResource  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
8  
6  
4  
2  
0  
Final balance: 0  
$
```

```
$ java LockInResource  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
8  
6  
4  
2  
0  
Final balance: 0  
$
```

Java Multithreaded Programming

- Recall conditions.py

```
$ python conditions.py  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
8  
6  
4  
2  
0  
Final balance: 0  
$
```

```
$ python conditions.py  
1  
2  
3  
4  
5  
3  
1  
2  
3  
4  
5  
6  
4  
2  
0  
Final balance: 0  
$
```

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:     public void run()
67:     {
68:         for (int i = 0; i < 5; i++)
69:             bank.withdraw(2);
70:     }
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: }

```

Java Multithreaded Programming

- See Conditions.java

```
$ java Conditions  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
8  
6  
4  
2  
0  
Final balance: 10  
$
```

```
$ java Conditions  
1  
2  
3  
4  
2  
0  
1  
2  
3  
4  
5  
6  
4  
2  
0  
Final balance: 0  
$
```

Java Multithreaded Programming

- Multithreaded programming across languages

	Language Support	Library Support
Python	no	yes
C	no	yes
Java	yes	yes
JavaScript	no	no

Appendix 2: Cross-Platform Frameworks

Cross-Platform Frameworks

- **Observations:**
 - A native **Android** app works only on **Android** devices
 - A native **iOS** app works only on **iOS** devices
- **Problem:**
 - Develop for one kind of device => limit users
 - Develop both kinds of devices => develop & *Maintain* two code bases
- **Solution:**
 - Cross-platform frameworks

Cross-Platform Frameworks

Framework	Development Language	Developer	Kinds of Apps
Flutter	Dart	Google	Android, iOS, web, desktop
React Native	JavaScript	Facebook, now Meta Platforms	Android, iOS
Kotlin Multiplatform	Kotlin	Jet Brains	Android, iOS, web, desktop, server-side
Ionic	JavaScript	Drifty Co.	Android, iOS, Windows, web, desktop
.NET Multi-Platform App UI	C#	Microsoft	Android, iOS, macOS, Windows
NativeScript	JavaScript	Telerik Progress Software	Android, iOS

Each runs on an emulator, which runs on the Android or iOS device
React Native is not native!