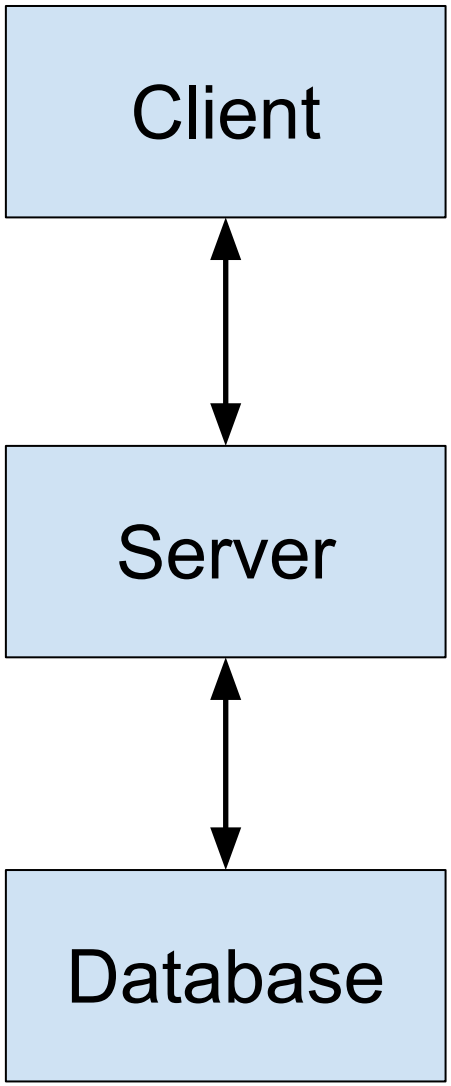
Client-Side Options (Part 2)

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 thread terminated
red
red
red
red
red
red thread terminated
main thread terminated
$
```

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

Threads/spawning.py (Page 1 of 1)

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

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

import threading

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

class PrinterThread (threading.Thread):

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

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

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

def main():

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

    blueThread.start()
    redThread.start()

    print('main thread terminated')

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

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

Threads/Spawning1.java (Page 1 of 1)

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

class PrinterThread extends Thread
{
   private String color;

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

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

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

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

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

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

```

Multithreaded Programming

- See **Spawning1.java**

```
$ java Spawning1
main thread terminated
blue
blue
blue
blue
blue
red
red
red
red
red
red thread terminated
blue thread terminated
$
```

```
$ java Spawning1
blue
blue
blue
blue
blue
main thread terminated
red
red
red
red
red
red thread terminated
blue thread terminated
$
```

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
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 thread terminated
red
red
red
red
red
red thread terminated
$
```

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

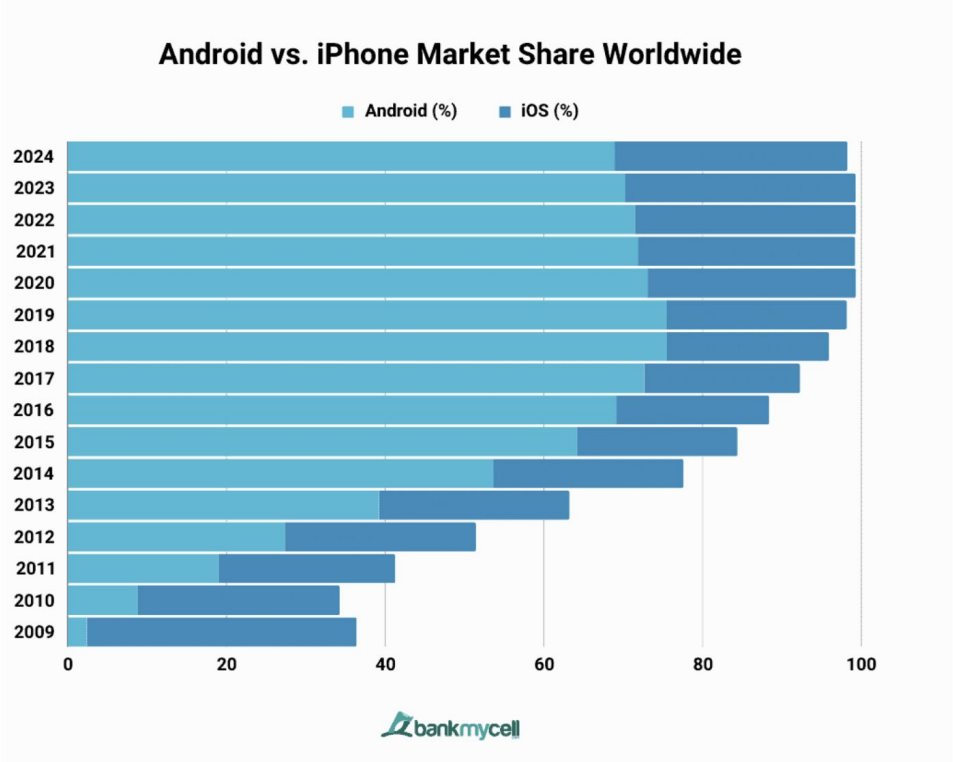
* May need multiple native mobile apps

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

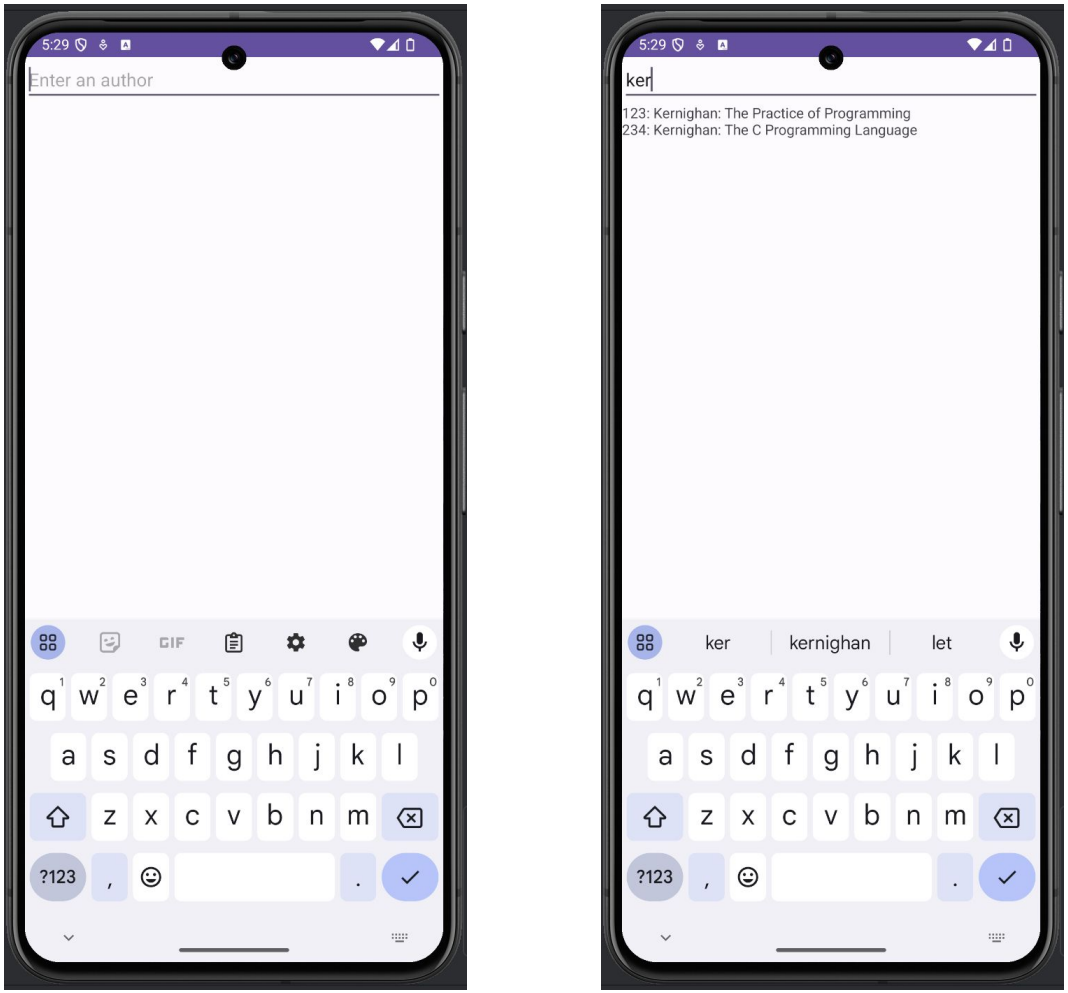
- Aside: Function def expressions
- Aside: Multithreaded programming
- Mobile programming
- **PennyAndroid app: defining**
- PennyAndroid app: running
- Pennylos app: defining
- Pennylos app: running

PennyAndroid App: Defining

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

PennyAndroid App: Defining

The goal:



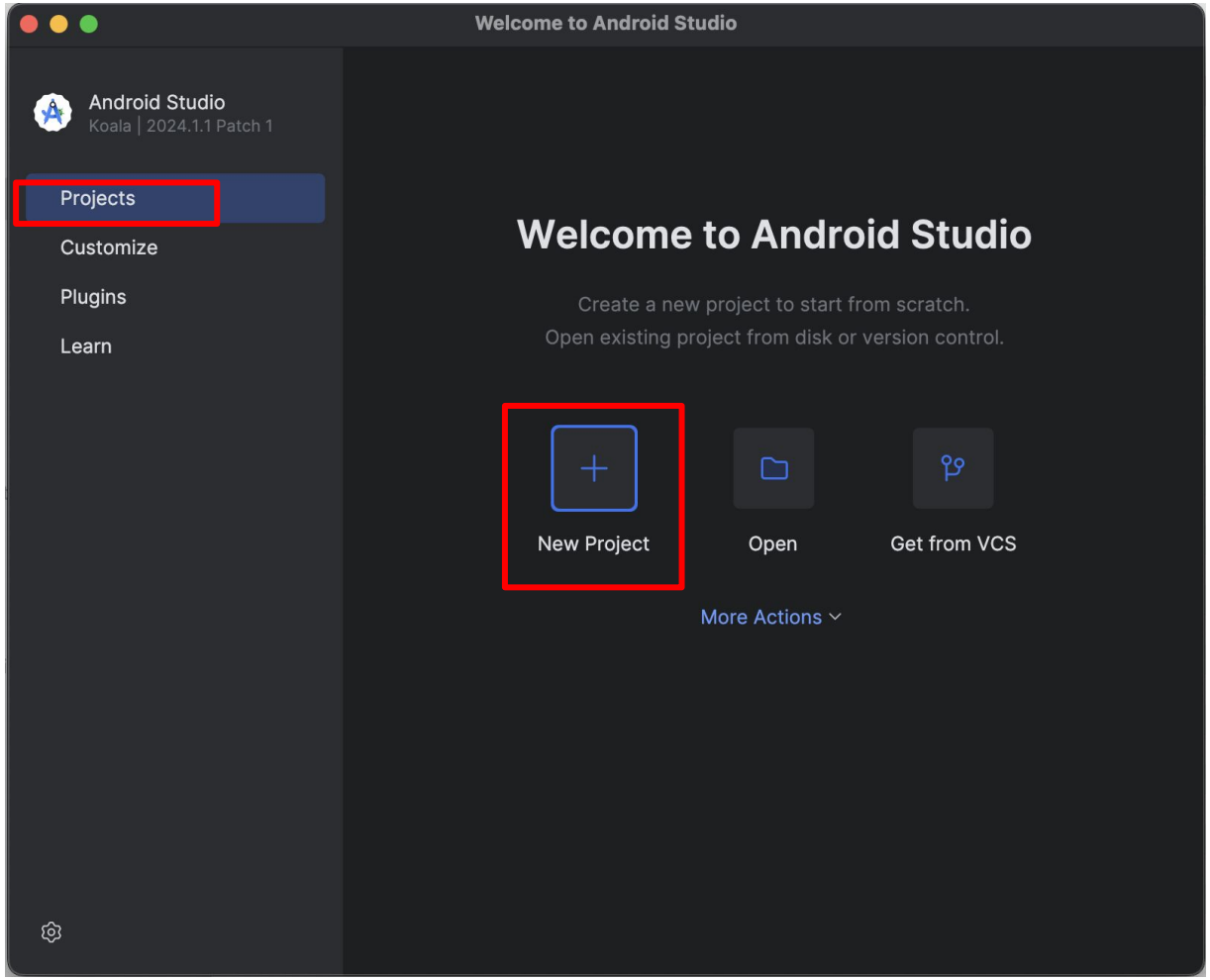
An Android client for the PennyJson server

PennyAndroid App: Defining

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

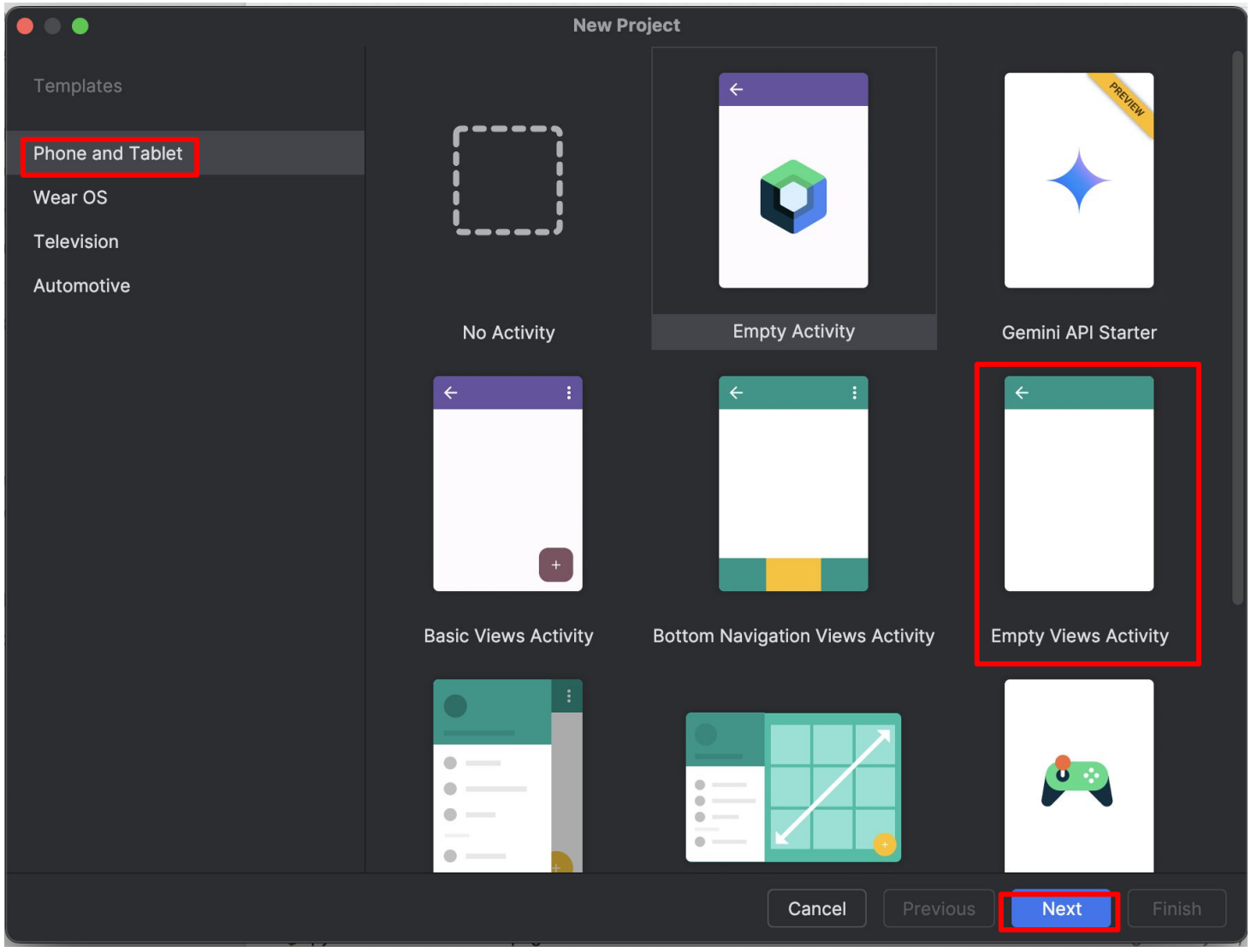
PennyAndroid App: Defining

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



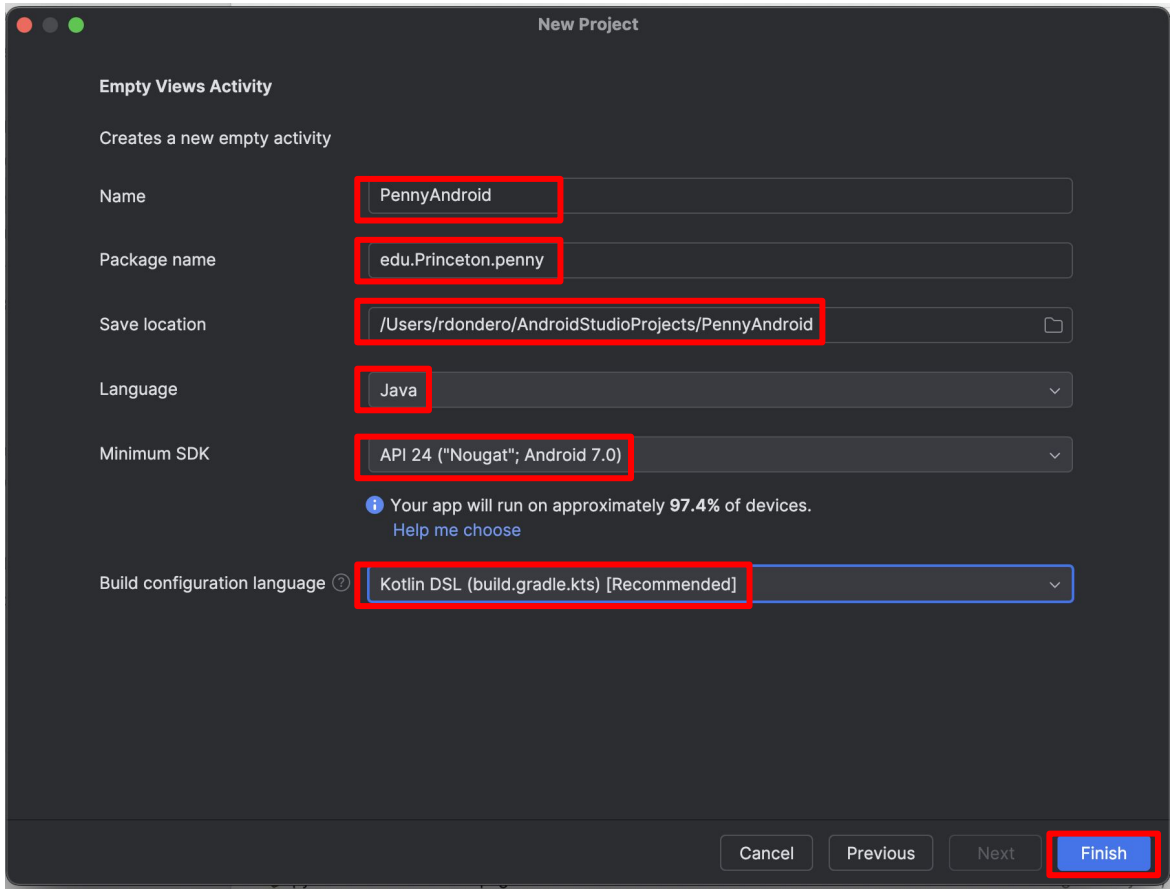
PennyAndroid App: Defining

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



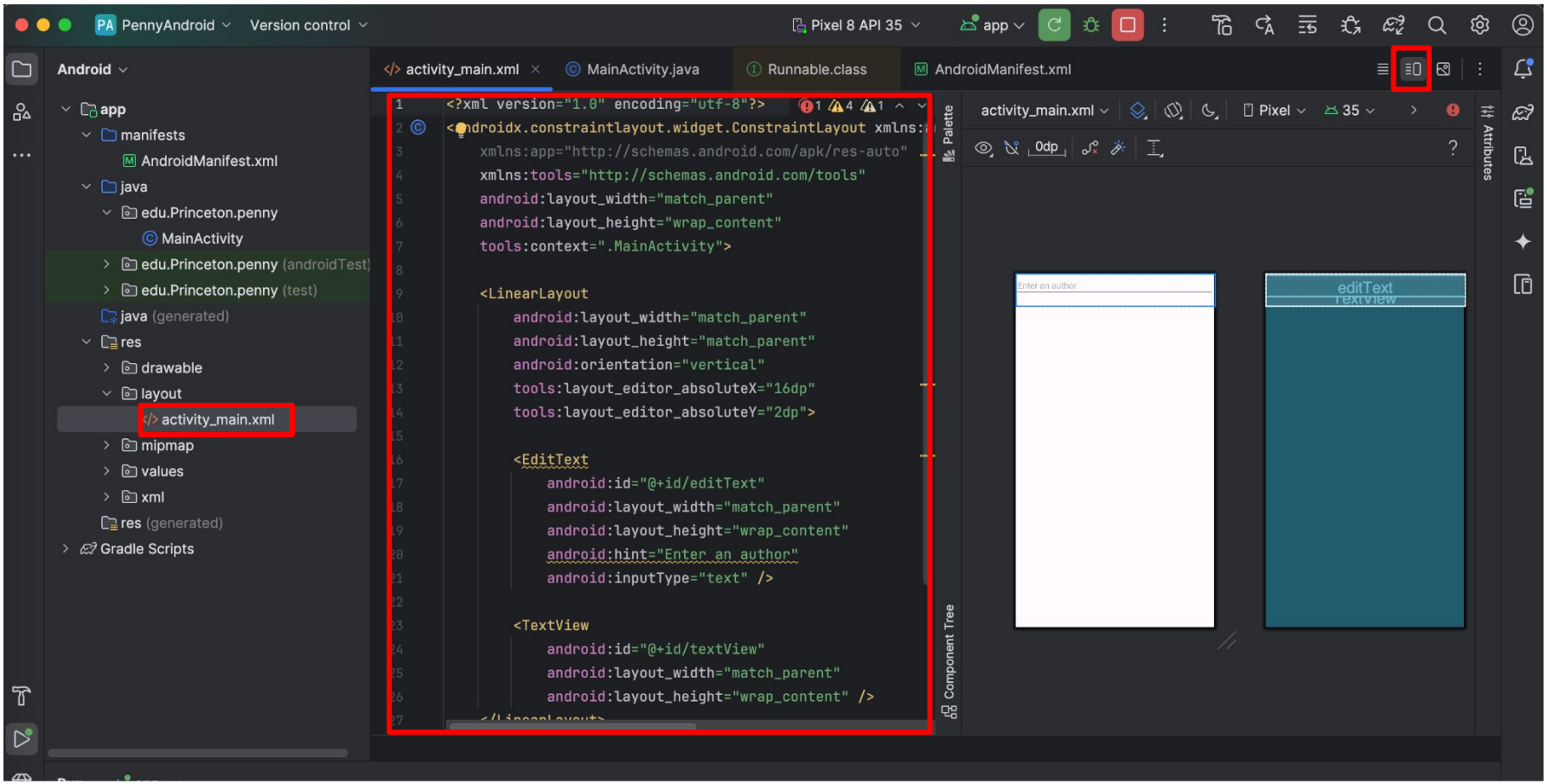
PennyAndroid App: Defining

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)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   tools:context=".MainActivity">

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical"
      tools:layout_editor_absoluteX="16dp"
      tools:layout_editor_absoluteY="2dp">

      <EditText
         android:id="@+id/editText"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:hint="Enter an author"
         android:inputType="text" />

      <TextView
         android:id="@+id/textView"
         android:layout_width="match_parent"
         android:layout_height="wrap_content" />
   </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```


PennyAndroid/MainActivity.java (Page 1 of 3)

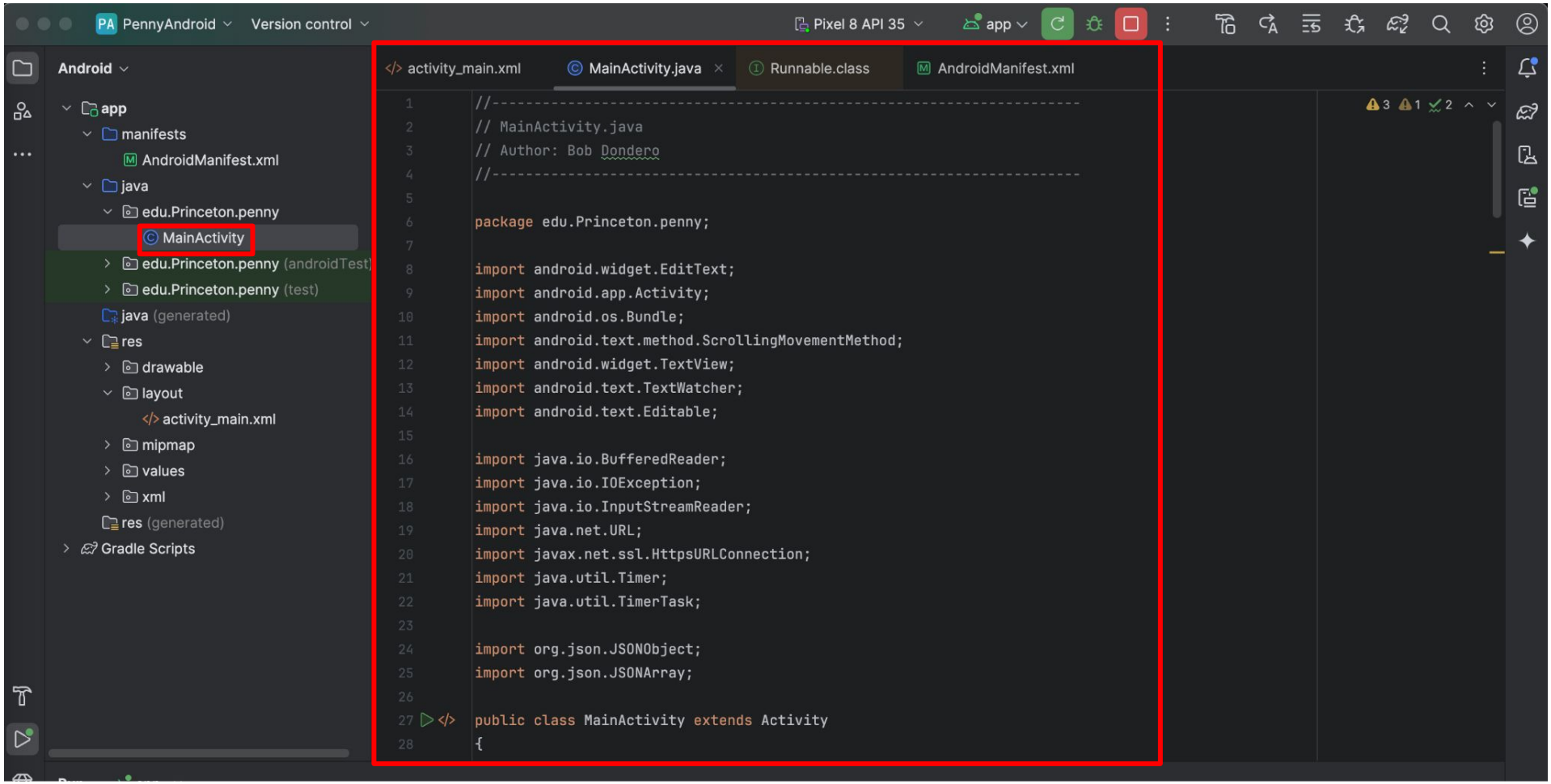
```
//---------------------------------------------------------------------------
// MainActivity.java
// Author: Bob Dondero
//---------------------------------------------------------------------------

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
{
   private EditText editText;
   private TextView textView;
   private AuthorSearchRunnable authorSearchRunnable = null;
   private Timer timer = null;

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

   private class AuthorSearchRunnable implements Runnable
   {
      private String author;
      private boolean shouldStop = false;

      AuthorSearchRunnable(String author)
      {
         this.author = author;
      }

      public void setShouldStop()
      {
         shouldStop = true;
      }

      public void run()
      {
         String books;

         try
         {
            URL url = new URL("https://pennyjson.onrender.com" +
                              "/searchresults?author=" + author);
            HttpsURLConnection con =
               (HttpsURLConnection) url.openConnection();

            con.setRequestMethod("GET");
            con.setDoInput(true);
            con.connect();
            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



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)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   tools:context=".MainActivity">

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical"
      tools:layout_editor_absoluteX="16dp"
      tools:layout_editor_absoluteY="2dp">

      <EditText
         android:id="@+id/editText"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:hint="Enter an author"
         android:inputType="text" />

      <TextView
         android:id="@+id/textView"
         android:layout_width="match_parent"
         android:layout_height="wrap_content" />
   </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```


PennyAndroid/MainActivity.java (Page 1 of 3)

```
//---------------------------------------------------------------------------
// MainActivity.java
// 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
{
   private EditText editText;
   private TextView textView;
   private AuthorSearchRunnable authorSearchRunnable = null;
   private Timer timer = null;

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

   private class AuthorSearchRunnable implements Runnable
   {
      private String author;
      private boolean shouldStop = false;

      AuthorSearchRunnable(String author)
      {
         this.author = author;
      }

      public void setShouldStop()
      {
         shouldStop = true;
      }

      public void run()
      {
         String books;

         try
         {
            URL url = new URL("https://pennyjson.onrender.com" +
                              "/searchresults?author=" + author);
            HttpsURLConnection con =
               (HttpsURLConnection) url.openConnection();

            con.setRequestMethod("GET");
            con.setDoInput(true);
            con.connect();
            int responseCode = con.getResponseCode();
```

PennyAndroid/MainActivity.java (Page 2 of 3)

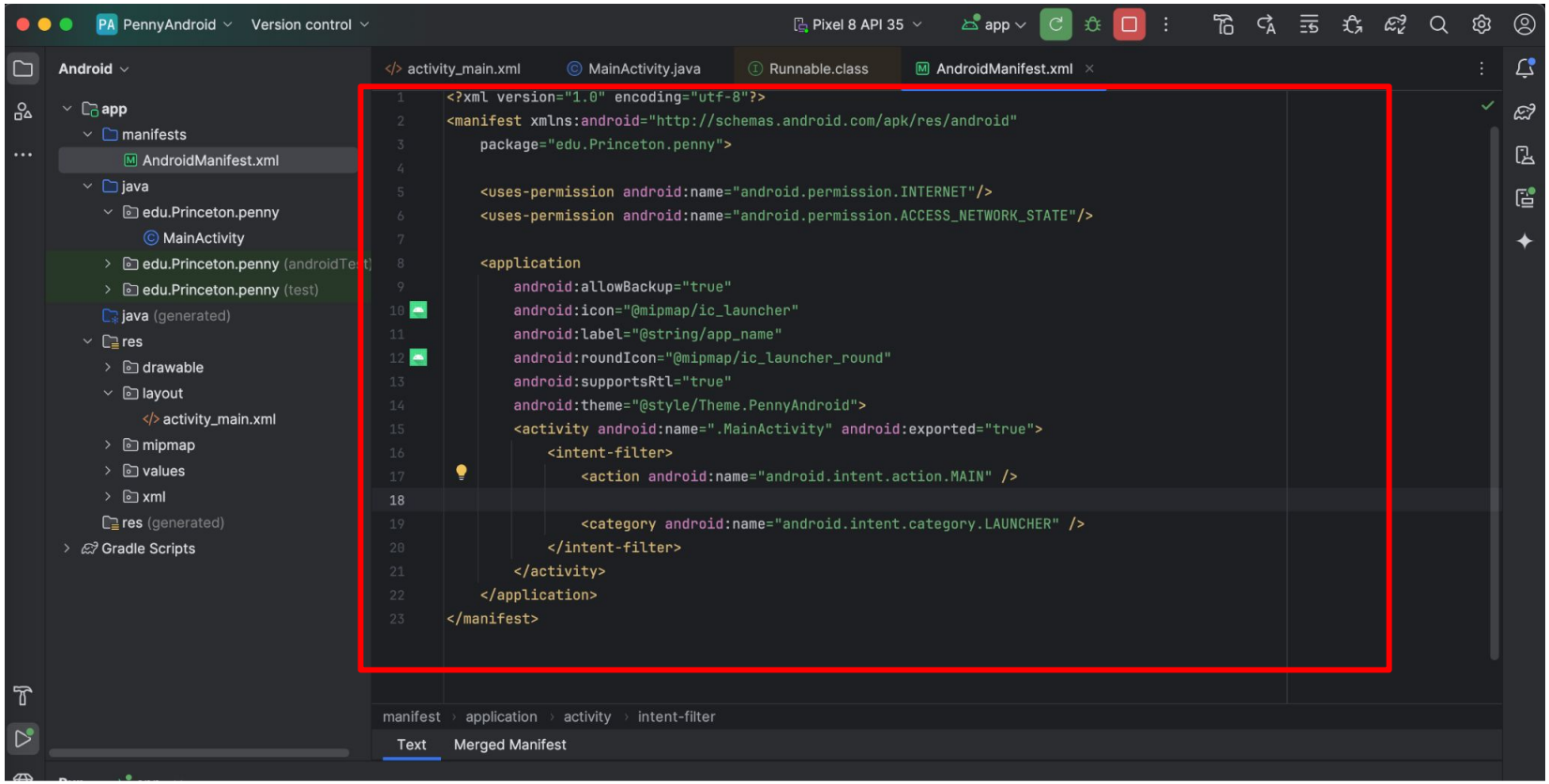
```
66:             if (responseCode != HttpsURLConnection.HTTP_OK)
67:                throw new IOException(
68:                   "HTTP error code: " + responseCode);
69:
70:             // Read the response, a JSON document containing books,
71:             // from the server.
72:             BufferedReader 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:
108:   private class MyTimerTask extends TimerTask
109:   {
110:      public void run()
111:      {
112:         String author = editText.getText().toString();
113:
114:         if (authorSearchRunnable != null)
115:            authorSearchRunnable.setShouldStop();
116:
117:         authorSearchRunnable = new AuthorSearchRunnable(author);
118:         Thread authorSearchThread = new Thread(authorSearchRunnable);
119:         authorSearchThread.start();
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:
132:      public void beforeTextChanged(CharSequence s, int start,
133:                                    int count, int after)
134:      {
135:      }
136:
137:      public void afterTextChanged(Editable s)
138:      {
139:         if (timer != null)
140:            timer.cancel();
141:         timer = new Timer();
142:         timer.schedule(new MyTimerTask(), 500);
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



PennyAndroid/AndroidManifest.xml (Page 1 of 1)

```
<?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>
```

blank (Page 1 of 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

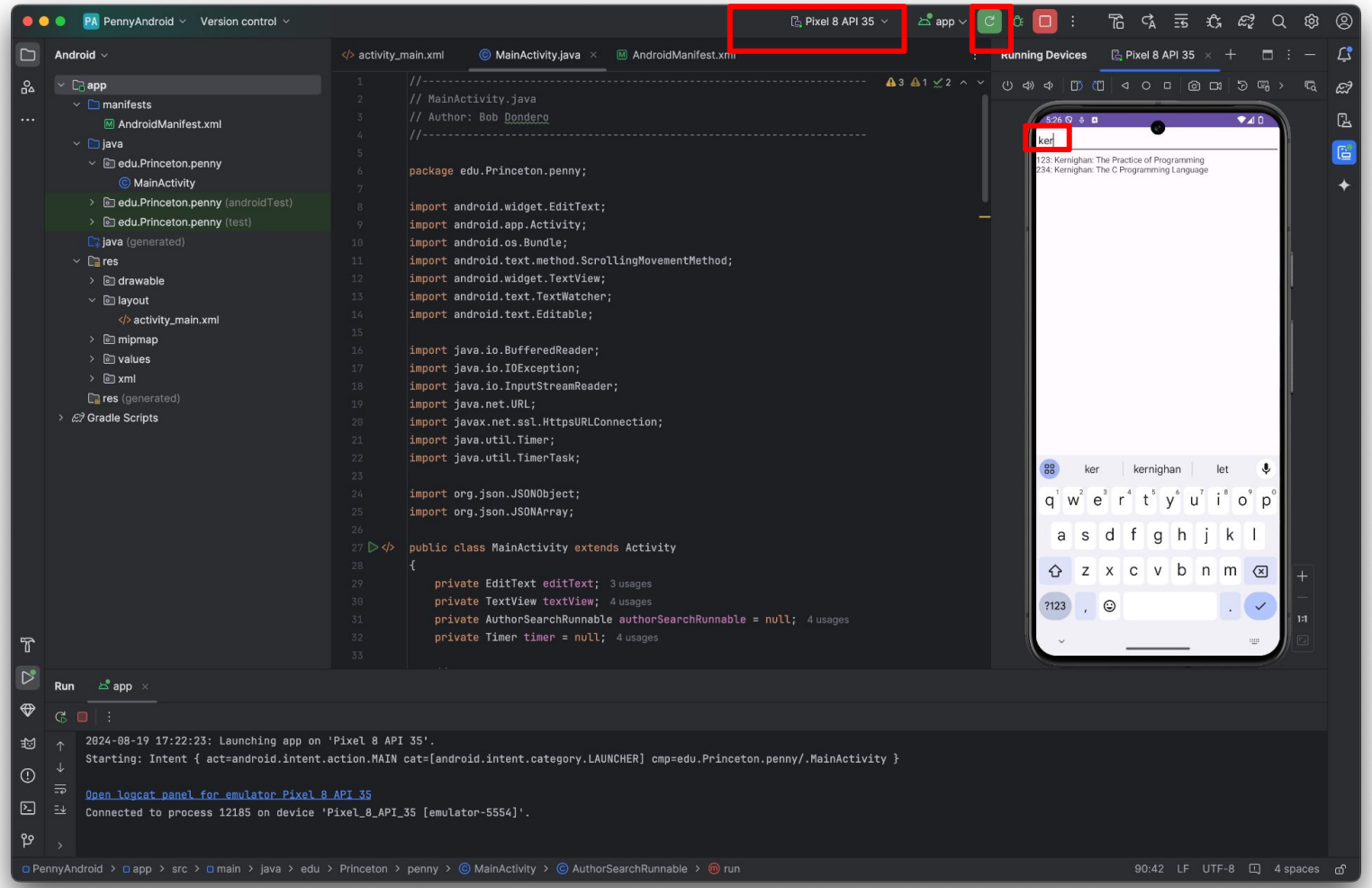
- To run on an Android emulator…

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!

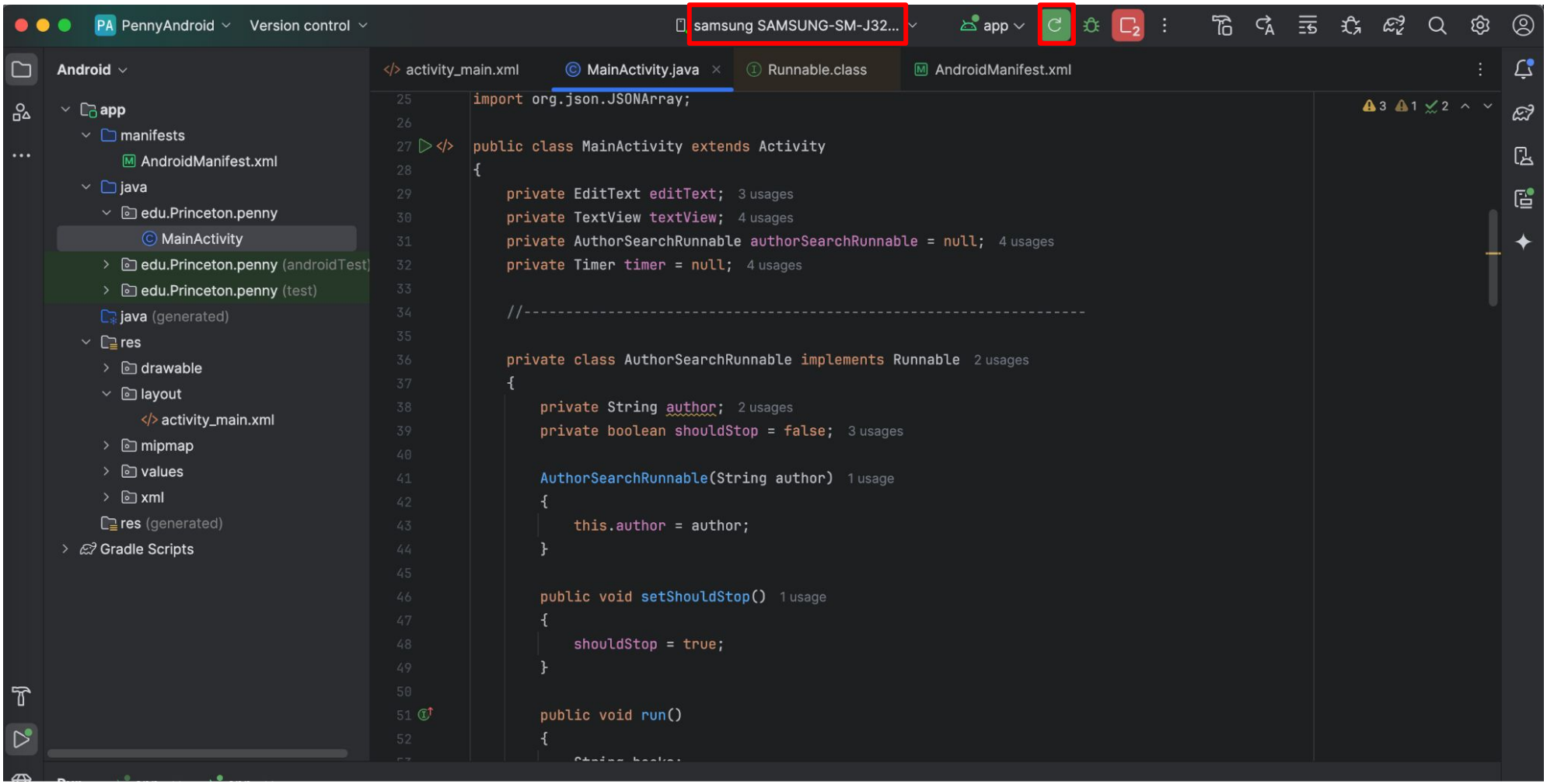


PennyAndroid App: Running

- 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!



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

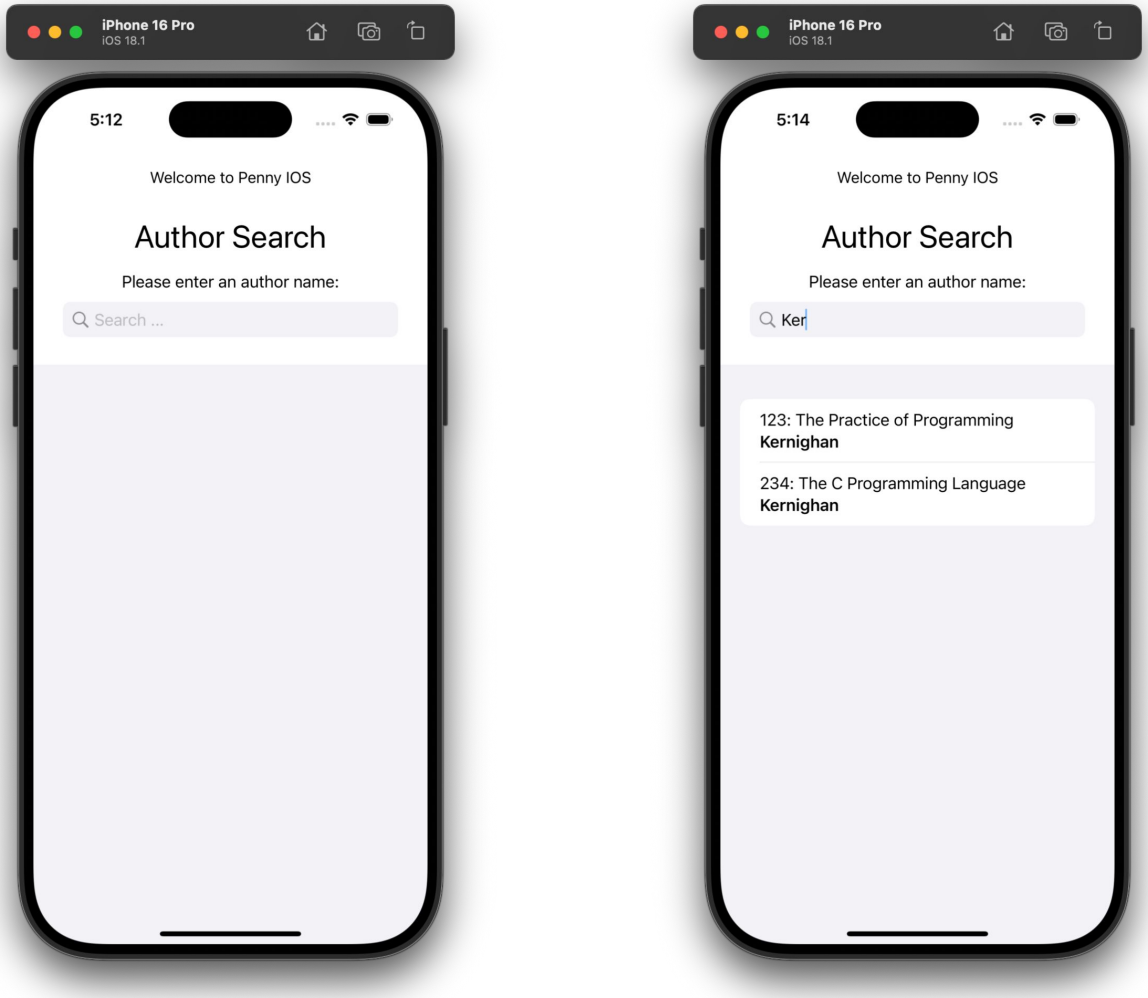
Thanks to
Katie DiPaola (‘26)...

PennyAndroid App: Defining

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

Pennylos App: Defining

The goal:



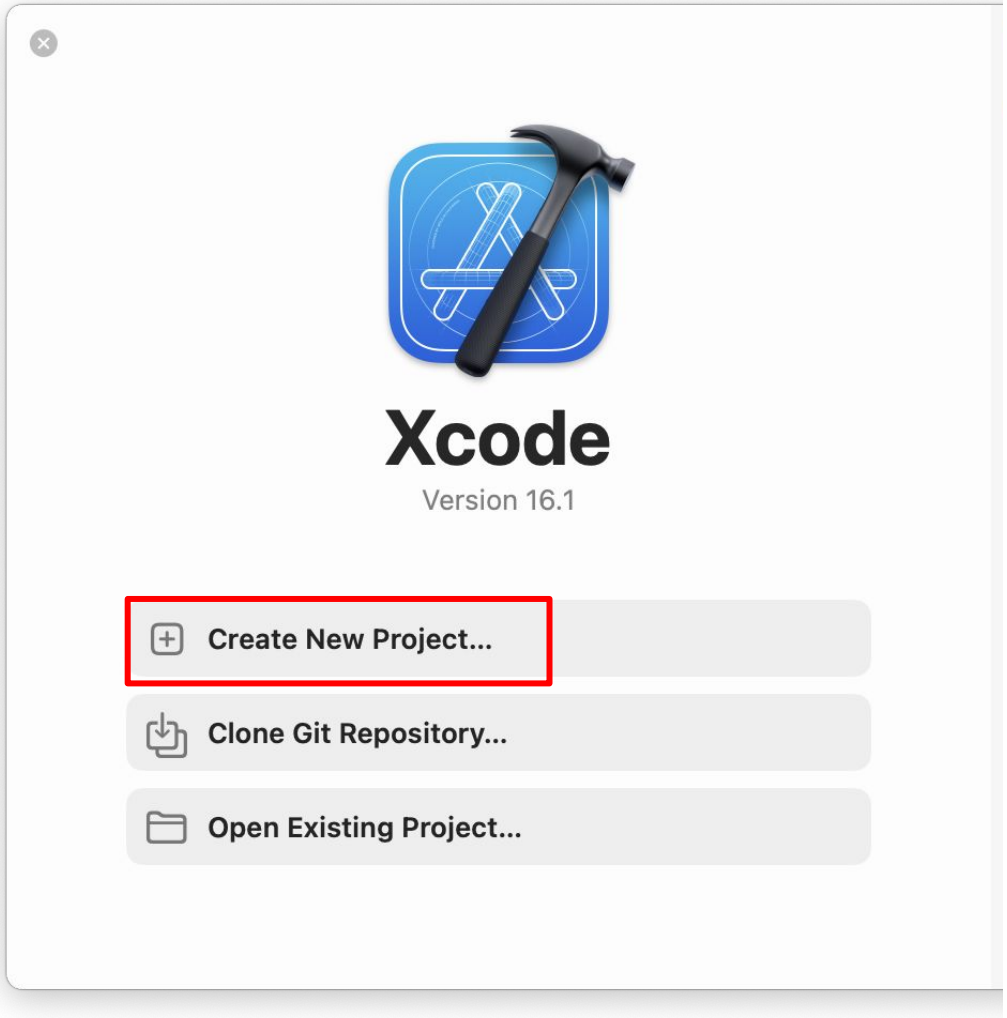
An iOS client for the PennyJson server

Pennylos App: Defining

- Download and install ***XCode***

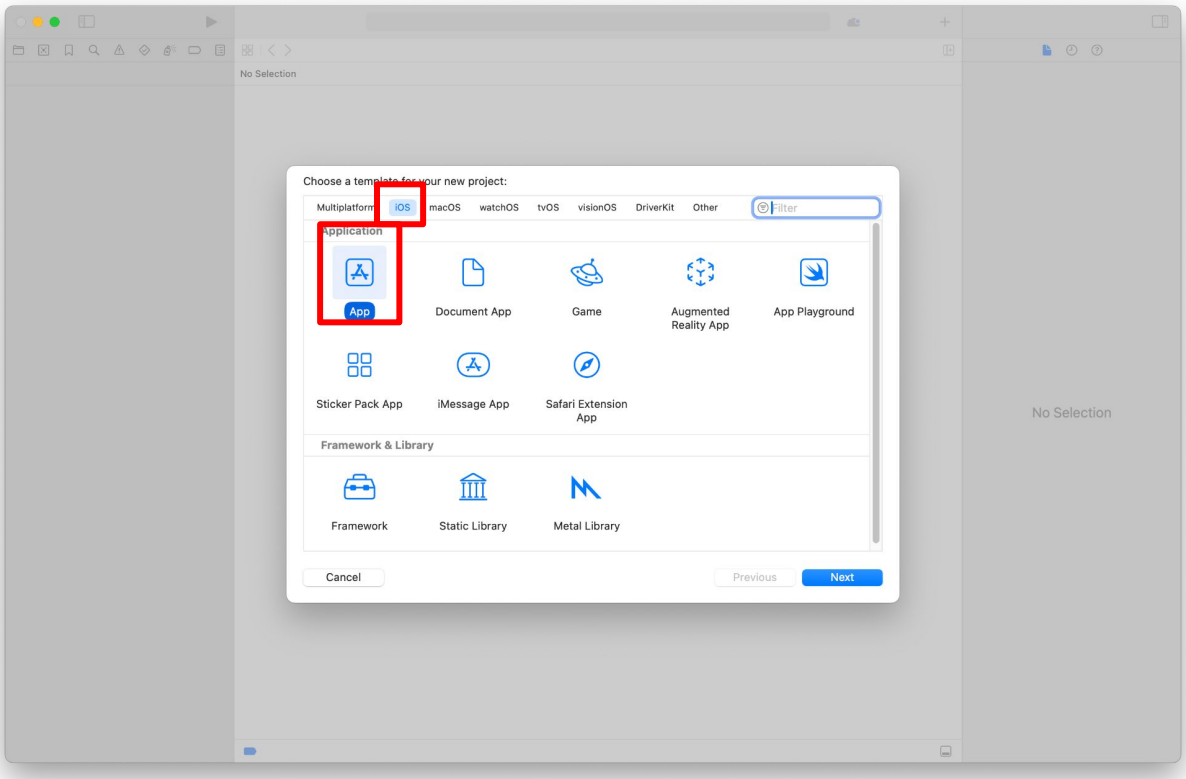
Pennylos App: Defining

Launch XCode; select *Create New Project…*



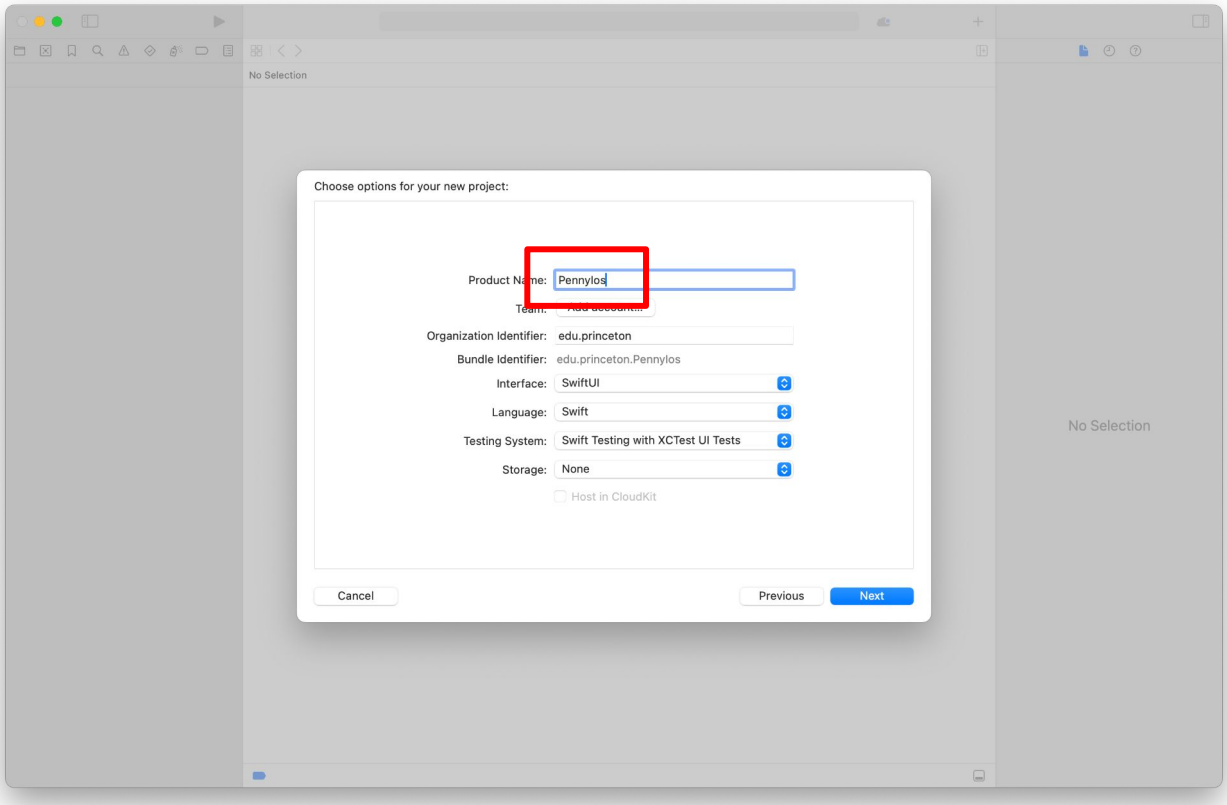
Pennylos App: Defining

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



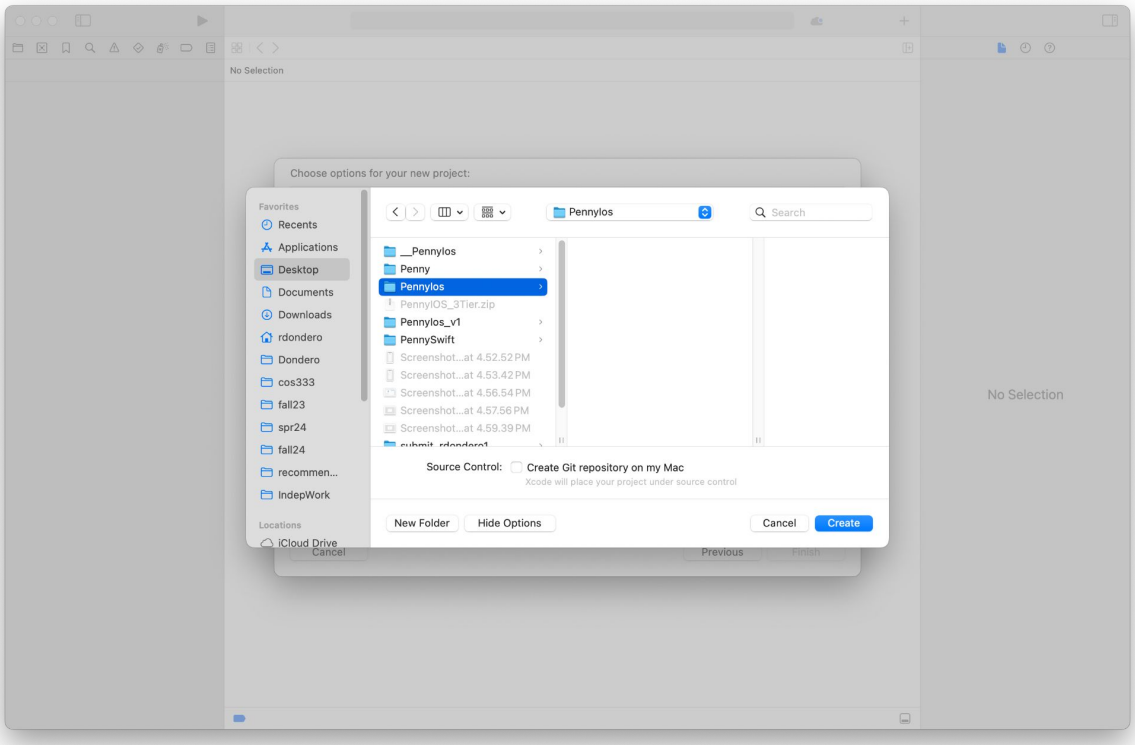
Pennylos App: Defining

For Product *Name* enter `PennyIos`; click on *Next*



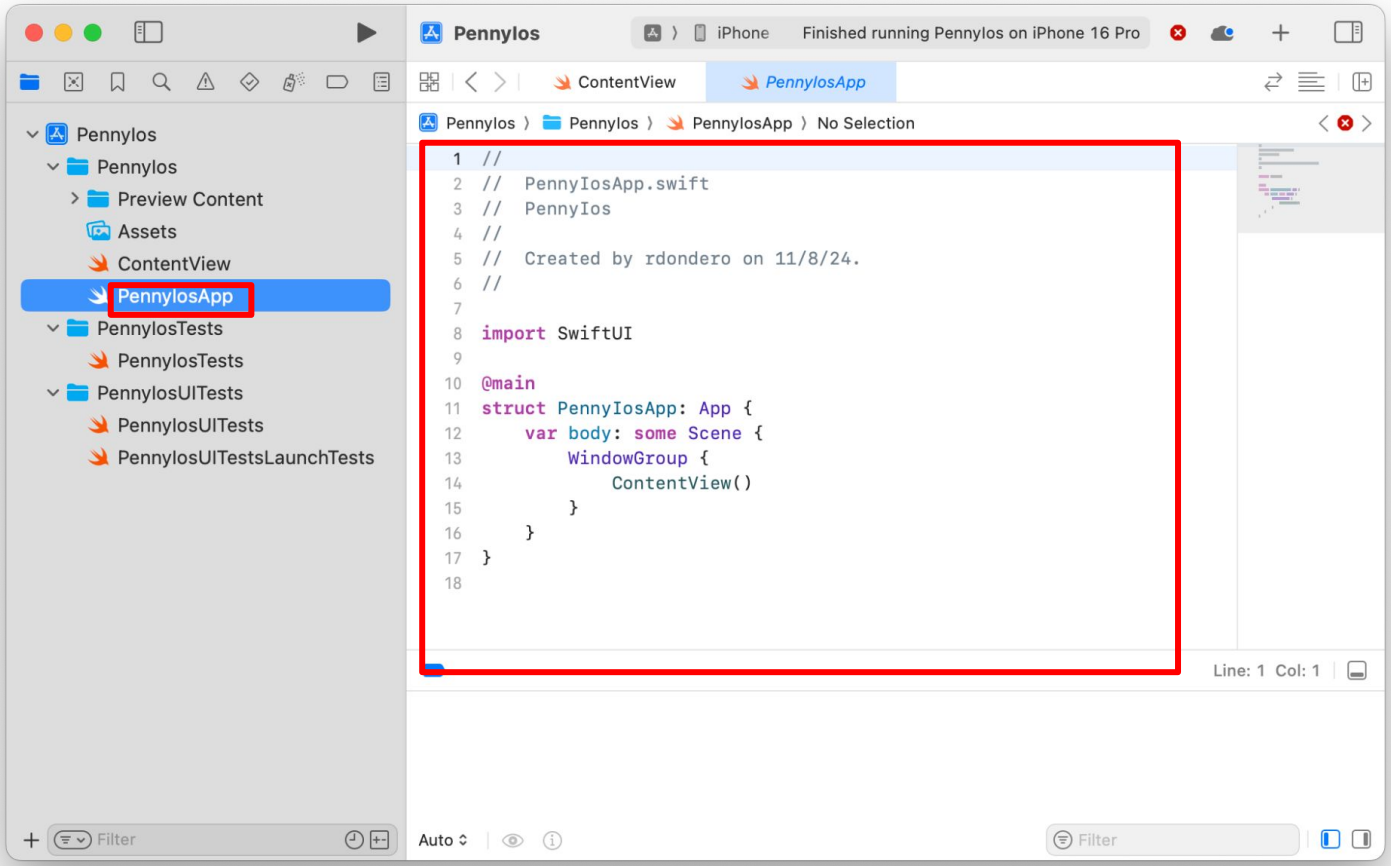
Pennylos App: Defining

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



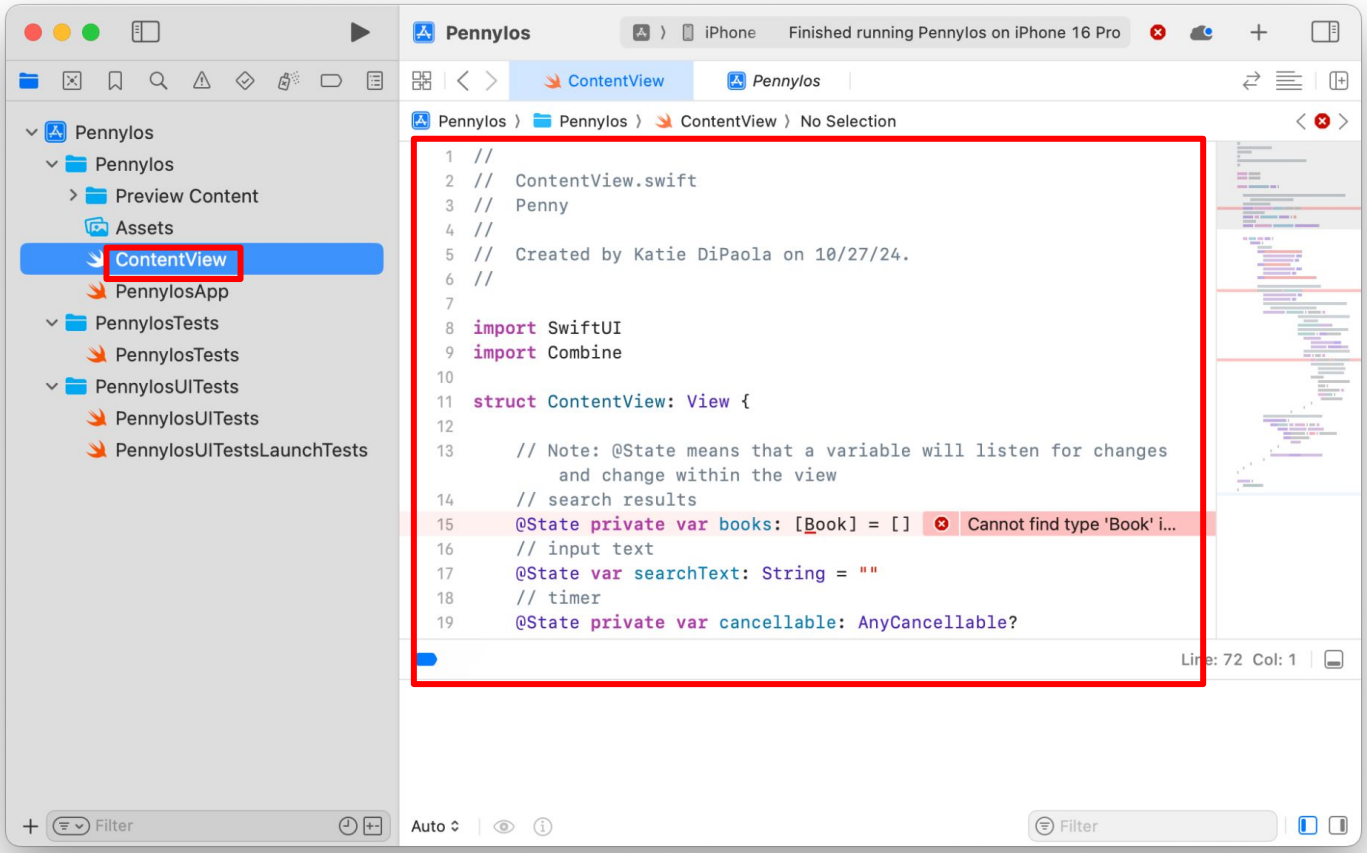
Pennylos App: Defining

Click on *PennylosApp*; examine the generated code



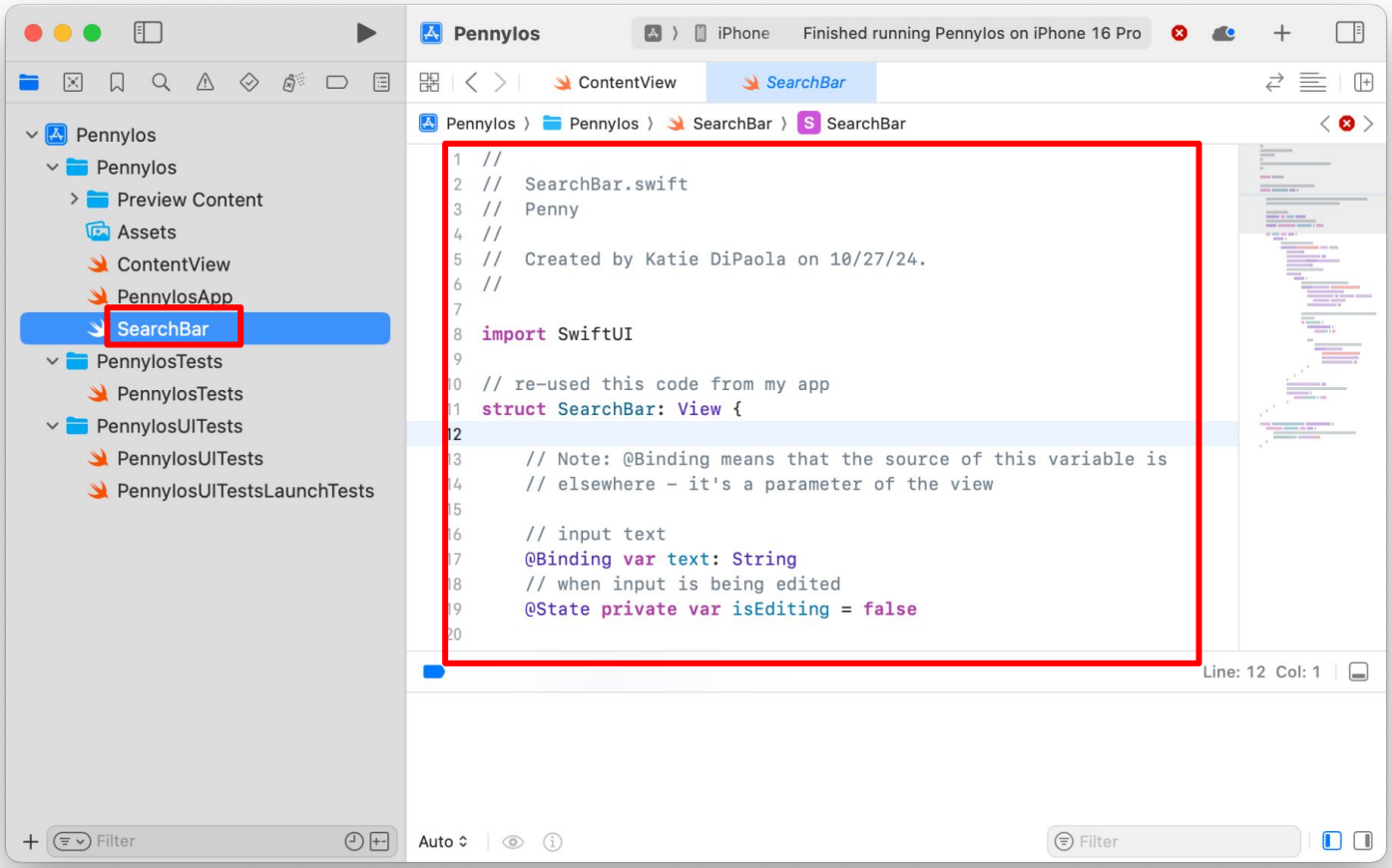
Pennylos App: Defining

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



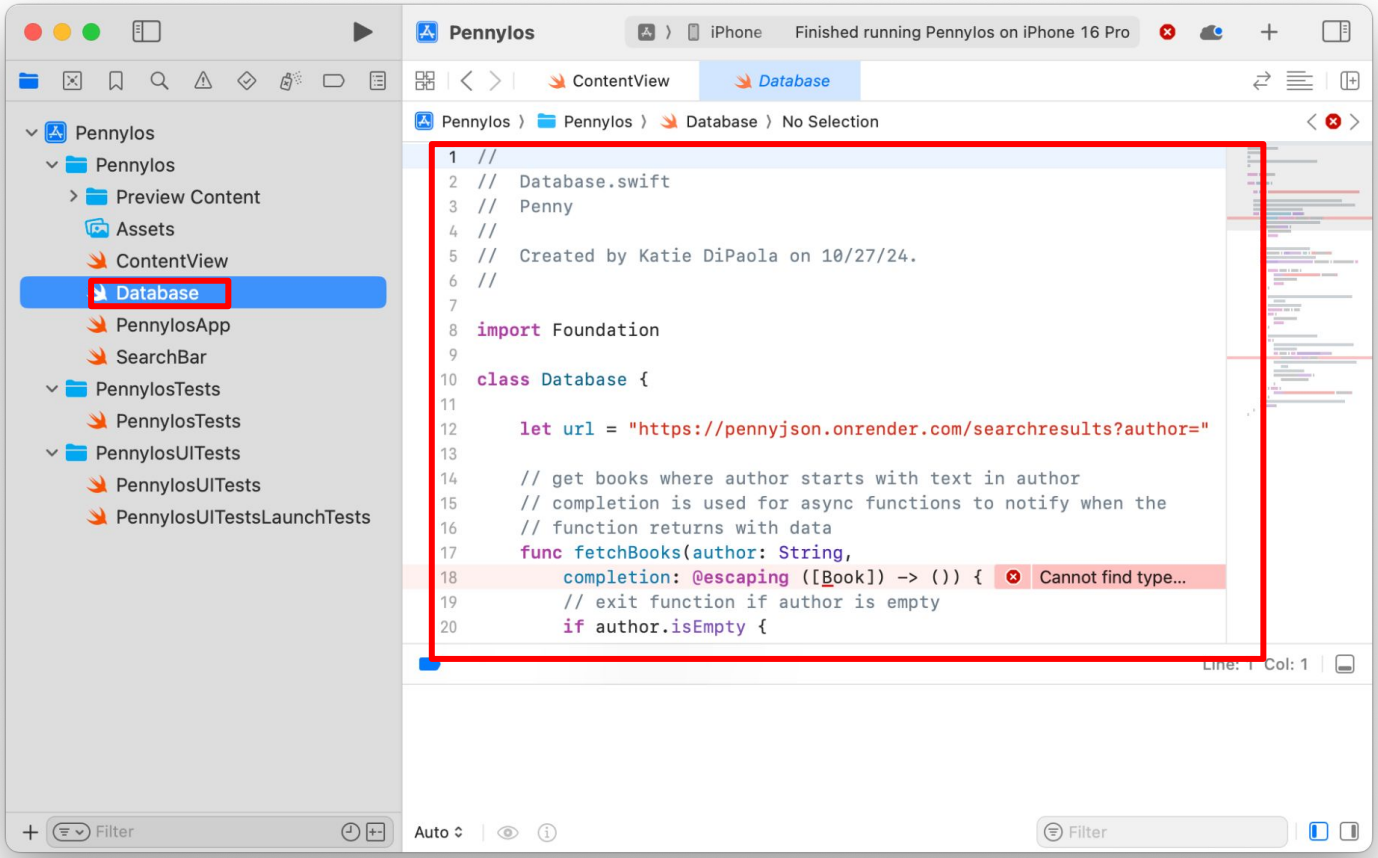
Pennylos App: Defining

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



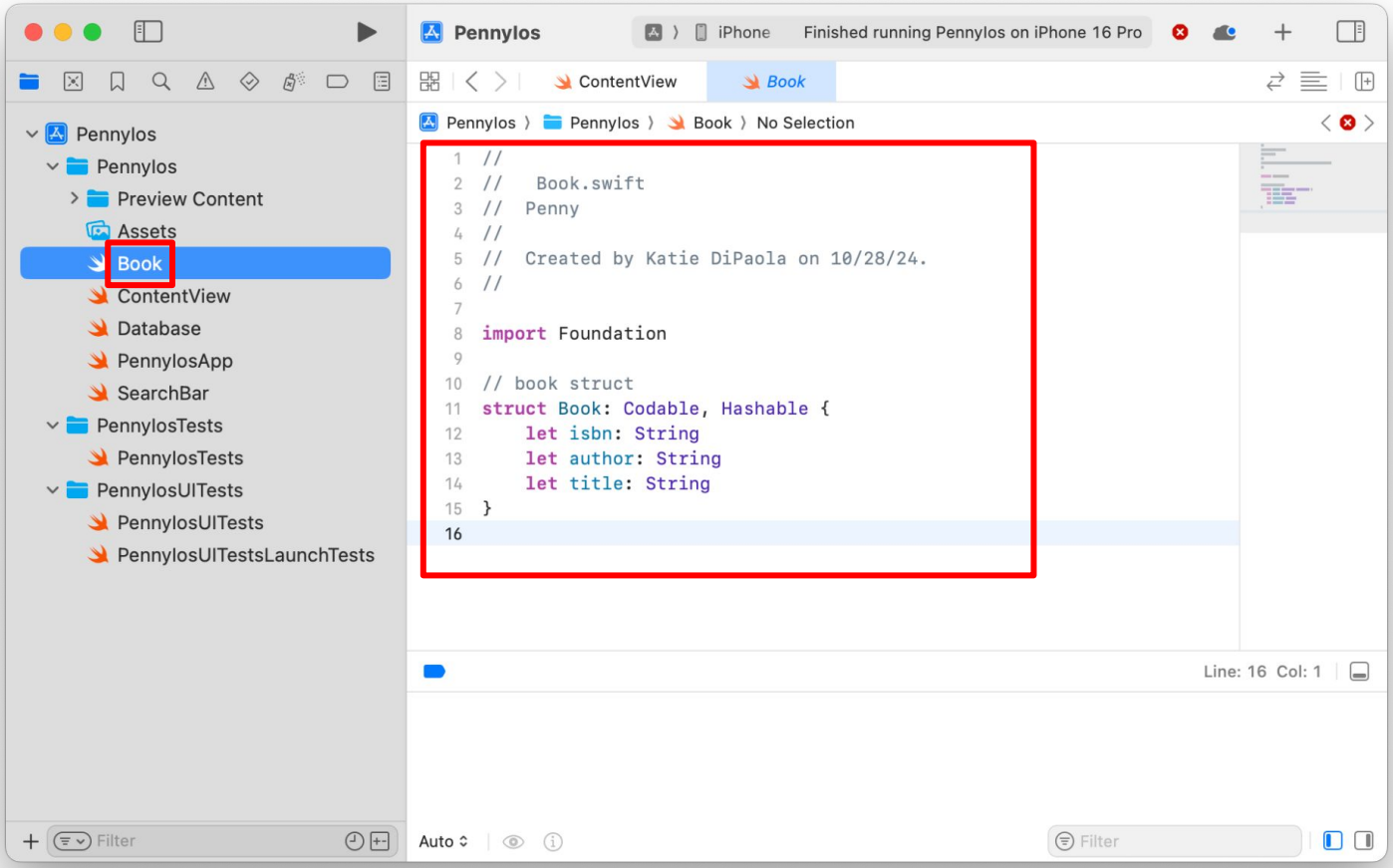
Pennylos App: Defining

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



Pennylos App: Defining

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



PennyIos/PennyIosApp.swift (Page 1 of 1)

```
1: //
2: //  PennyIosApp.swift
3: //  PennyIos
4: //
5: //  Created by rdondero 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)

```swift
//
//  ContentView.swift
//  Penny
//
//  Created by Katie DiPaola on 10/27/24.
//

import SwiftUI
import Combine

struct ContentView: View {

    // Note: @State means that a variable will listen for changes and
    // change within the view search results
    @State private var books: [Book] = []
    // input text
    @State var searchText: String = ""
    // timer
    @State private var cancellable: AnyCancellable?


    var body: some View {
        VStack {
            // header
            Text("Welcome to Penny IOS")
                .font(.system(size: 16))
                .padding(.vertical, 20)
            Text("Author Search")
                .font(.system(size: 32))
                .padding(.bottom, 10)
            Text("Please enter an author name:")

            // searchBar view from SearchBar.swift
            SearchBar(text: $searchText)
                .padding(.horizontal, 20)
                .padding(.bottom, 20)
                // when text is entered (aka input changes), delay and
                // then fetch from database
                .onChange(of: searchText) { newValue in
                    // Cancel the previous timer if it exists
                    cancellable?.cancel()
                    // Start a new timer for delay
                    cancellable = Just(newValue)
                        .delay(for: .milliseconds(500),
                            scheduler: RunLoop.main)
                        // fetch books from database
                        .sink { text in
                            let database = Database()
                            // handle completion (aka await return from
                            // fetchBooks)
                            database.fetchBooks(author: text) {
                                fetched_books in
                                self.books = fetched_books
                            }
                        }
                }
                // display result of current search
                NavigationView {
                    List(books, id: \.self) { book in
                        VStack (alignment: .leading){
                            Text(book.isbn + ": " + book.title)
                            Text(book.author)
                                .bold()
                        }
                    }
```

PennyIos/ContentView.swift (Page 2 of 2)

```swift
                        .background(Color(.systemGray6))
                }
        }
    }
}

#Preview {
    ContentView()
}
```

PennyIos/SearchBar.swift (Page 1 of 2)

```
//
//  SearchBar.swift
//  Penny
//
//  Created by Katie DiPaola on 10/27/24.
//

import SwiftUI

// re-used this code from my app
struct SearchBar: View {

    // Note: @Binding means that the source of this variable is
    // elsewhere - it's a parameter of the view

    // input text
    @Binding var text: String
    // when input is being edited
    @State private var isEditing = false

    var body: some View {
        HStack {
            // input text field
            TextField("Search ...", text: $text)
                .padding(7)
                .padding(.horizontal, 25)
                .background(Color(.systemGray6))
                .cornerRadius(8)
                // make it look pretty
                .overlay(
                    HStack {
                        // uses a build-in IOS image
                        Image(systemName: "magnifyingglass")
                            .foregroundColor(.gray)
                            .frame(minWidth: 0, maxWidth: .infinity,
                               alignment: .leading)
                            .padding(.leading, 8)

                        // provide functionality to delete full user
                        // input
                        if isEditing {
                            Button(action: {
                                self.text = ""

                            }) {
                                // uses a build-in IOS image
                                Image(systemName:
                                    "multiply.circle.fill")
                                    .foregroundColor(.gray)
                                    .padding(.trailing, 8)
                            }
                        }
                    }
                )
                .padding(.horizontal, 10)
                // track when user is editing input
                .onTapGesture {
                    self.isEditing = true
                }
        }
    }
}

struct SearchBar_Previews: PreviewProvider {
    static var previews: some View {
```

PennyIos/SearchBar.swift (Page 2 of 2)

```
        // view paramter (@Binding variable from above!)
        SearchBar(text: .constant(""))
    }
}
```

PennyIos/Database.swift (Page 1 of 1)

```swift
//
//  Database.swift
//  Penny
//
//  Created by Katie DiPaola on 10/27/24.
//

import Foundation

class Database {

    let url = "https://pennyjson.onrender.com/searchresults?author="

    // get books where author starts with text in author
    // completion is used for async functions to notify when the
    // function returns with data
    func fetchBooks(author: String,
        completion: @escaping ([Book]) -> ()) {
        // exit function if author is empty
        if author.isEmpty {
            completion([])
            return
        }

        // add author to database url
        let queryUrl = URL(string: url + "\(author)")!
        // make asynchronous http request
        URLSession.shared.dataTask(with: queryUrl) { data,_,error in

            if let error = error {
                print("Error fetching books: \(error)")
                completion([])
                return
            }

            // guard lets us check if an optional value exists without
            // throwing an error
            guard let data = data
            else {
                completion([])
                return
            }

            // otherwise handle json data
            do {
                // decode into list of Book objects (defined in
                // Book.swift)
                let books = try JSONDecoder().decode(
                    [Book].self, from: data)
                // completion on the main thread (UI updates must occur
                // on main thread)
                DispatchQueue.main.async {
                    completion(books)
                }
            } catch {
                print("Error decoding JSON response: \(error)")
            }
        // start task (send http request we defined above)
        }.resume()
    }
}
```

PennyIos/Book.swift (Page 1 of 1)

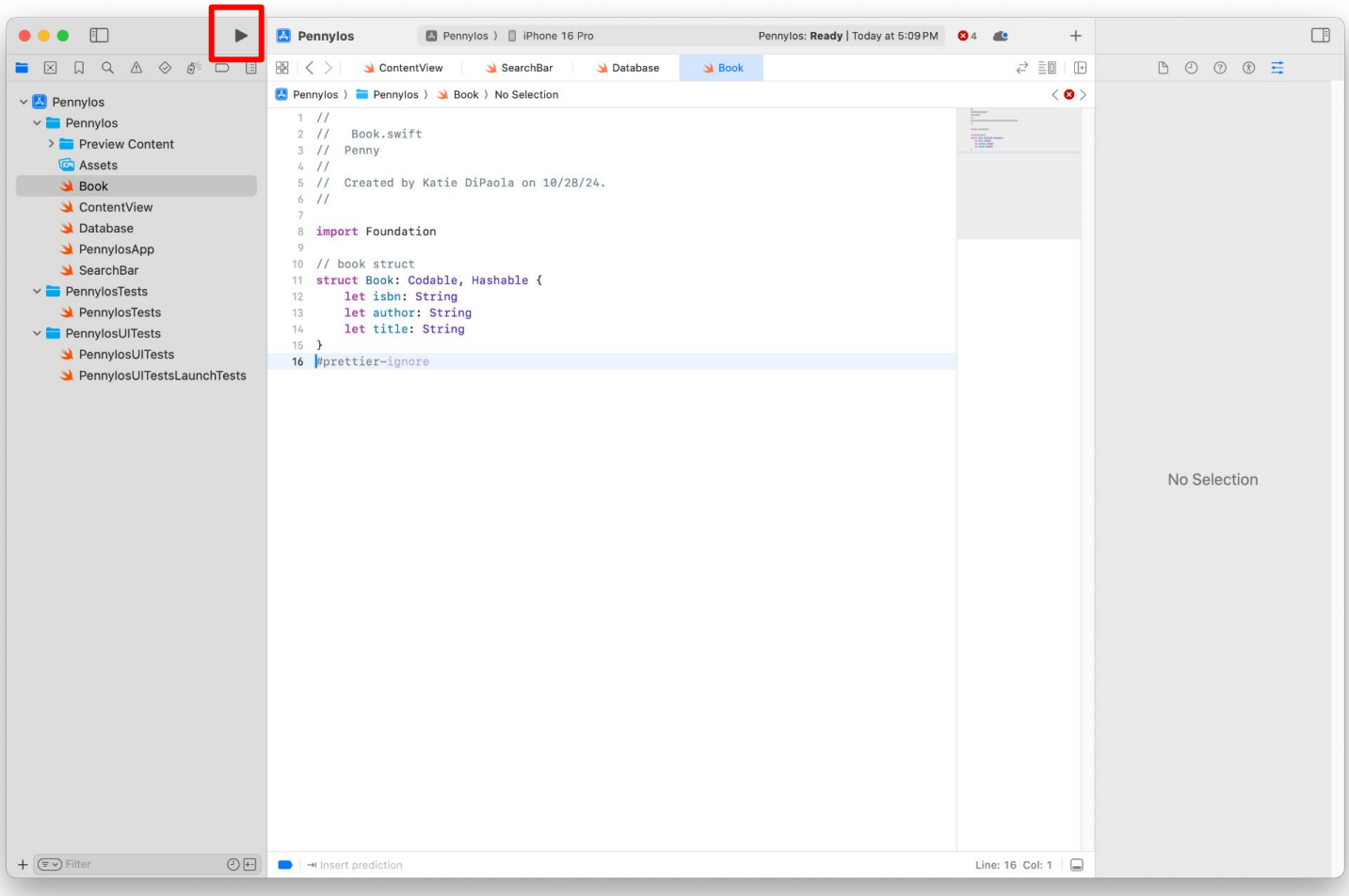
```swift
//
//   Book.swift
//  Penny
//
//  Created by Katie DiPaola on 10/28/24.
//

import Foundation

// book struct
struct Book: Codable, Hashable {
    let isbn: String
    let author: String
    let title: String
}
```

Agenda

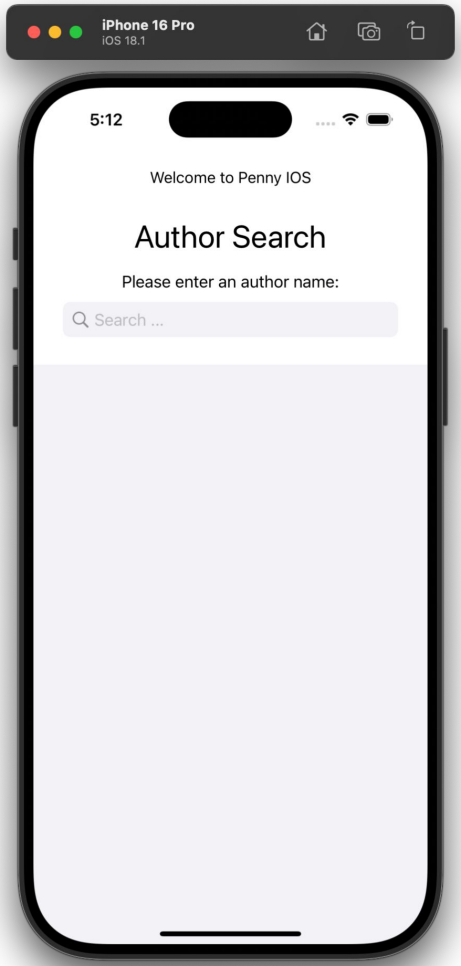
- Aside: Function def expressions
- Aside: Multithreaded programming
- Mobile programming
- PennyAndroid app: defining
- PennyAndroid app: running
- Pennylos app: defining
- **Pennylos app: running**

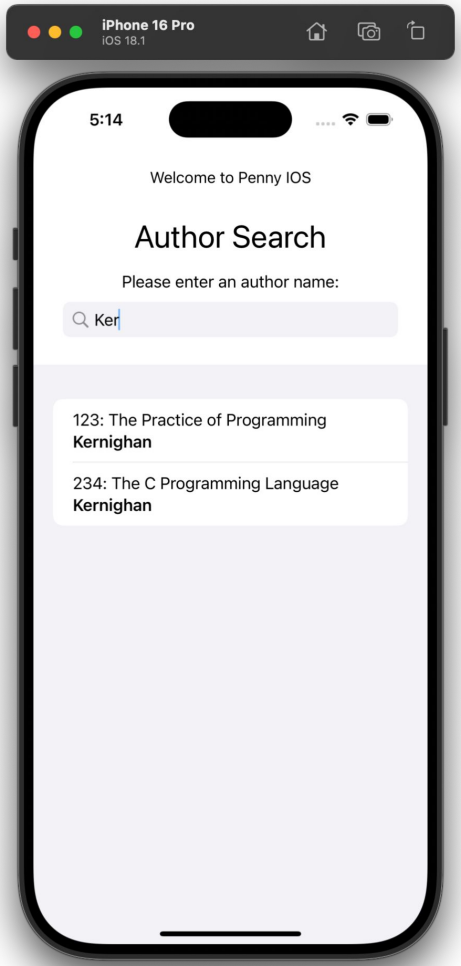
Pennylos App: Running

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)

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

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

import threading

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

class BankAcct:

    def __init__(self):
        self._balance = 0

    def get_balance(self):
        return self._balance

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

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

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

class DepositThread (threading.Thread):

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

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

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

class WithdrawThread (threading.Thread):

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

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

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

def main():

    bank_acct = BankAcct()

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

    deposit_thread.start()
```

Threads/race.py (Page 2 of 2)

```
    withdraw_thread.start()

    deposit_thread.join()
    withdraw_thread.join()

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

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

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

Threads/Race.java (Page 1 of 2)

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

Threads/Race.java (Page 2 of 2)

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

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)

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

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

import threading

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

class BankAcct:

    def __init__(self):
        self._balance = 0

    def get_balance(self):
        return self._balance

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

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

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

class DepositThread (threading.Thread):

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

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

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

class WithdrawThread (threading.Thread):

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

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

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

def main():

```

Threads/lockinuser.py (Page 2 of 2)

```
    bank_acct = BankAcct()
    bank_acct_lock = threading.RLock()

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

    deposit_thread.start()
    withdraw_thread.start()

    deposit_thread.join()
    withdraw_thread.join()

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

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

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

Threads/LockInUser.java (Page 1 of 2)

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

class Bank
{
   private int balance = 0;

   public int getBalance()
   {
      return balance;
   }

   public void deposit(int amount)
   {
      int temp = balance;
      temp += amount;
      System.out.println(temp);
      balance = temp;
   }

   public void withdraw(int amount)
   {
      int temp = balance;
      temp -= amount;
      System.out.println(temp);
      balance = temp;
   }
}

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

class DepositThread extends Thread
{
   private Bank bank;

   public DepositThread(Bank bank)
   {
      this.bank = bank;
   }

   public void run()
   {
      for (int i = 0; i < 10; i++)
         synchronized (bank)
         {
            bank.deposit(1);
         }
   }
}

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

class WithdrawThread extends Thread
{
   private Bank bank;

   public WithdrawThread(Bank bank)
   {
      this.bank = bank;
   }

   public void run()
   {
```

Threads/LockInUser.java (Page 2 of 2)

```
      for (int i = 0; i < 5; i++)
         synchronized (bank)
         {
            bank.withdraw(2);
         }
   }
}

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

public class LockInUser
{
   public static void main(String[] args)
   {
      Bank bank = new Bank();

      Thread depositThread = new DepositThread(bank);
      Thread withdrawThread = new WithdrawThread(bank);

      depositThread.start();
      withdrawThread.start();

      try
      {
         depositThread.join();
         withdrawThread.join();
      }
      catch (InterruptedException e)
      {
         Thread.currentThread().interrupt();
      }

      System.out.println("Final balance: " + bank.getBalance());
   }
}
```

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)

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

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

import threading

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

class BankAcct:

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

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

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

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

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

class DepositThread (threading.Thread):

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

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

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

class WithdrawThread (threading.Thread):

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

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

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

Threads/lockinresource.py (Page 2 of 2)

```

def main():

    bank_acct = BankAcct()

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

    deposit_thread.start()
    withdraw_thread.start()

    deposit_thread.join()
    withdraw_thread.join()

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

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

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

Threads/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)

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

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

import threading

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

class BankAcct:

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

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

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

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

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

class DepositThread (threading.Thread):

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

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

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

class WithdrawThread (threading.Thread):

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

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

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

Threads/conditions.py (Page 2 of 2)

```

def main():
    bank_acct = BankAcct()

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

    deposit_thread.start()
    withdraw_thread.start()

    deposit_thread.join()
    withdraw_thread.join()

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

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

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

Threads/Conditions.java (Page 1 of 2)

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

class Bank
{
   private int balance = 0;

   public synchronized int getBalance()
   {
      return balance;
   }

   public synchronized void deposit(int amount)
   {
      balance += amount;
      System.out.println(balance);
      notifyAll();
   }

   public synchronized void withdraw(int amount)
   {
      try
      {
         while (balance < amount)
            wait();
         balance -= amount;
         System.out.println(balance);
      }
      catch (InterruptedException e)
      {
         Thread.currentThread().interrupt();
      }
   }
}

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

class DepositThread extends Thread
{
   private Bank bank;

   public DepositThread(Bank bank)
   {
      this.bank = bank;
   }

   public void run()
   {
      for (int i = 0; i < 10; i++)
         bank.deposit(1);
   }
}

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

class WithdrawThread extends Thread
{
   private Bank bank;

   public WithdrawThread(Bank bank)
   {
      this.bank = bank;
   }
```

Threads/Conditions.java (Page 2 of 2)

```

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

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

public class Conditions
{
   public static void main(String[] args)
   {
      Bank bank = new Bank();

      Thread depositThread = new DepositThread(bank);
      Thread withdrawThread = new WithdrawThread(bank);

      depositThread.start();
      withdrawThread.start();

      try
      {
         depositThread.join();
         withdrawThread.join();
      }
      catch (InterruptedException e)
      {
         Thread.currentThread().interrupt();
      }

      System.out.println("Final balance: " + bank.getBalance());
   }
}
```

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**!