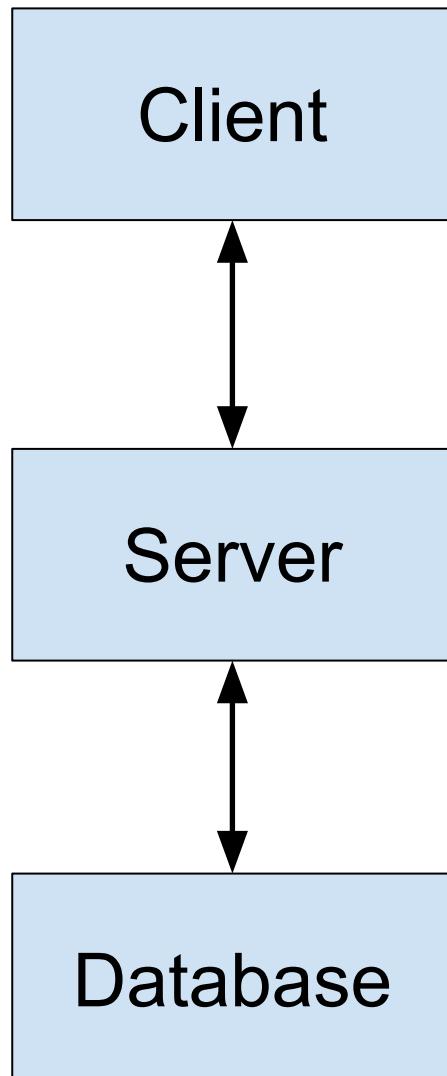


Client-Side Options (Part 2)

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
$
```

Multithreaded Programming

- See Spawning1.java

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

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
$
```

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	✓	

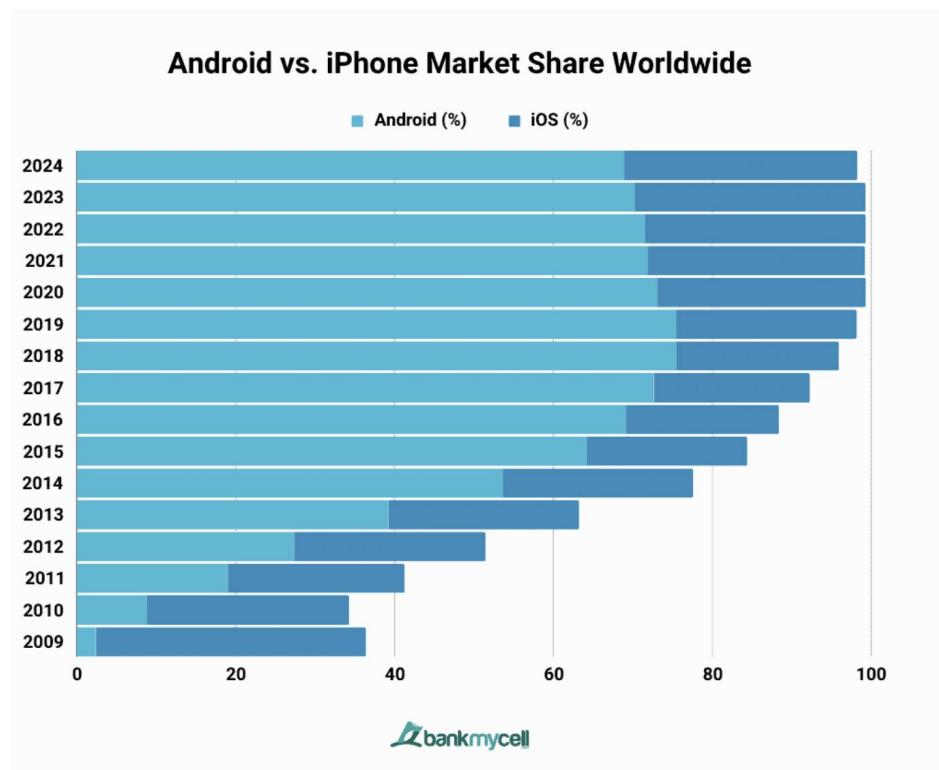
* May need multiple native mobile apps

<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

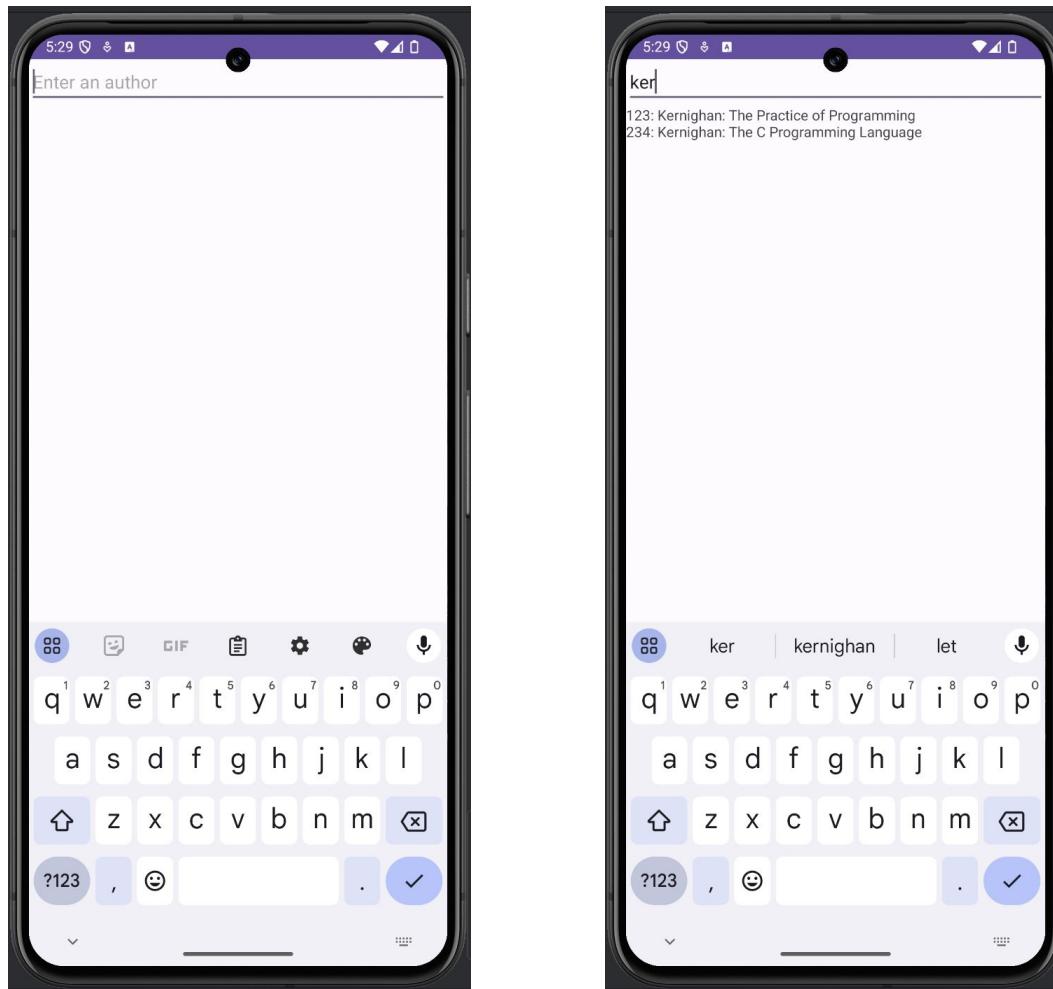
- Aside: Function def expressions
- Aside: Multithreaded programming
- Mobile programming
- **PennyAndroid app: defining**
- PennyAndroid app: running
- Pennylos app: defining
- Pennylos app: running

PennyAndroid App: Defining

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

PennyAndroid App: Defining

The
goal:



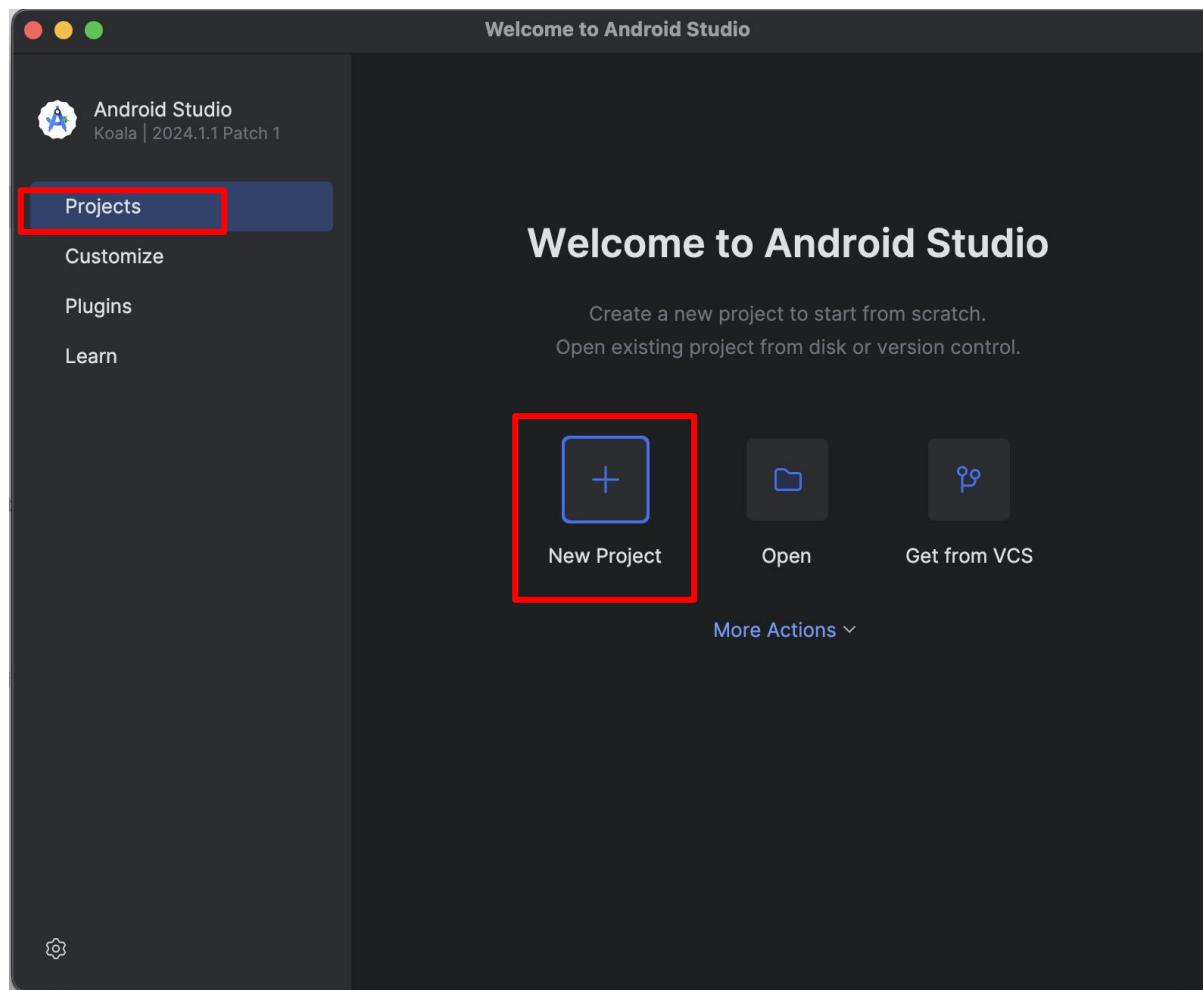
An Android client for the PennyJson server

PennyAndroid App: Defining

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

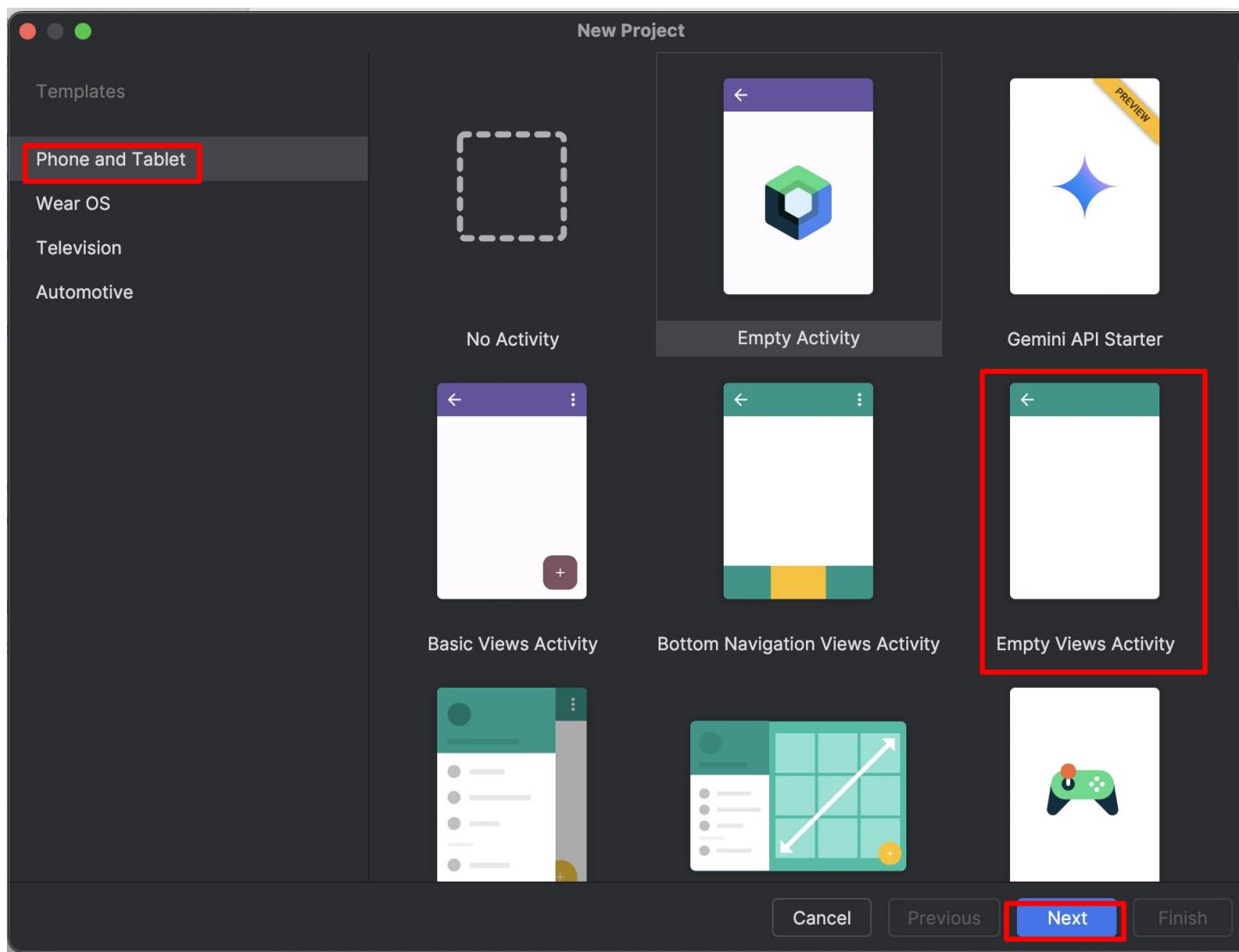
PennyAndroid App: Defining

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



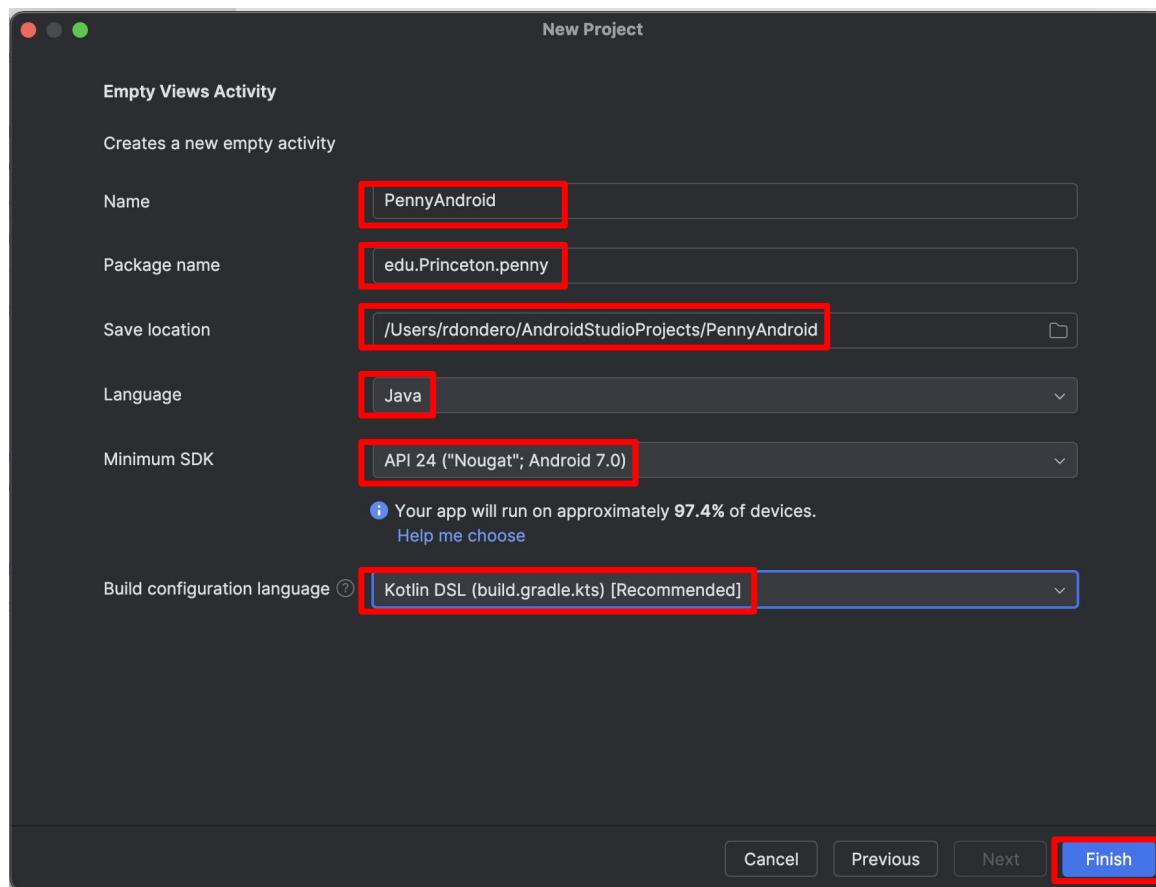
PennyAndroid App: Defining

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



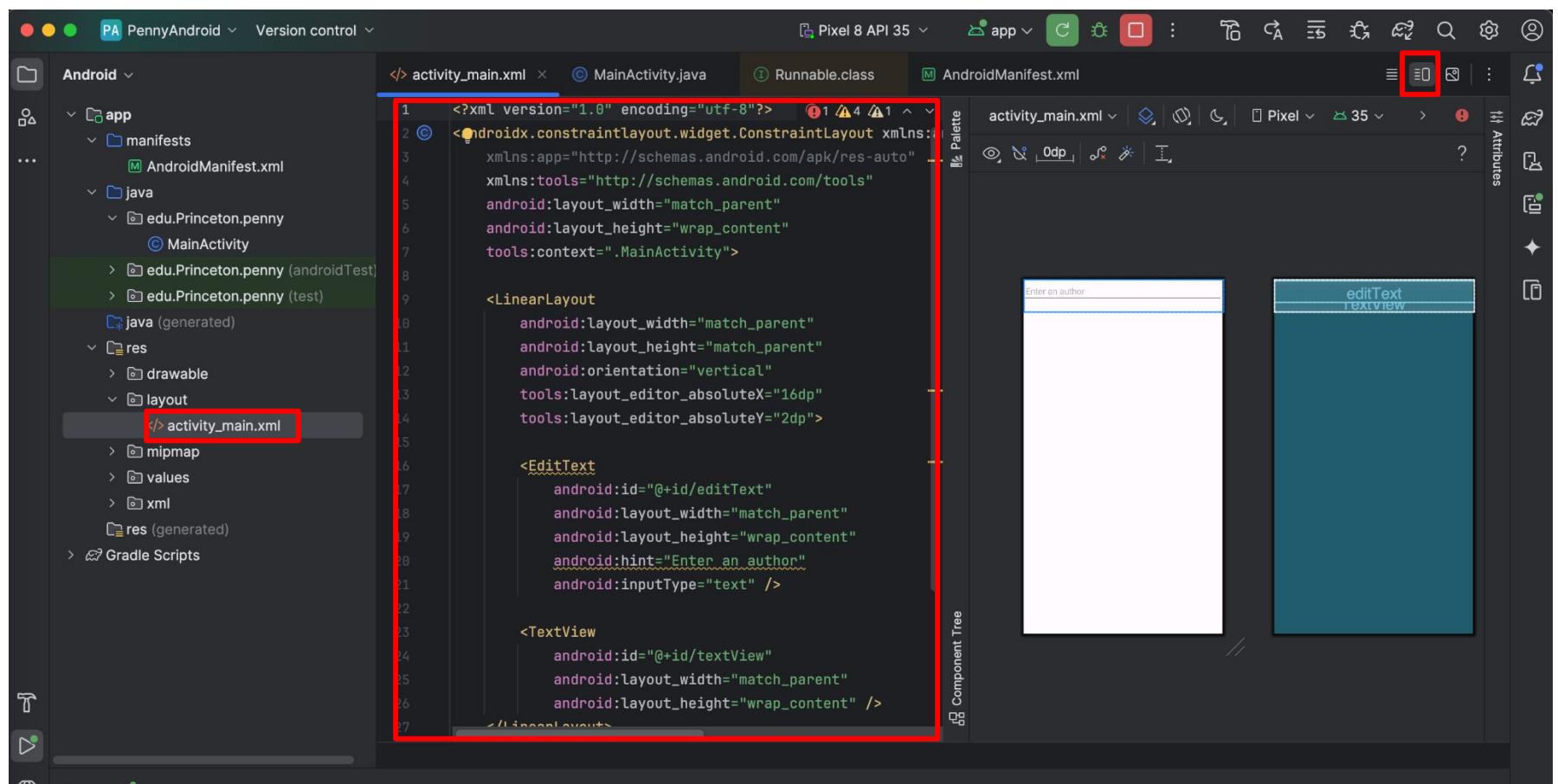
PennyAndroid App: Defining

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

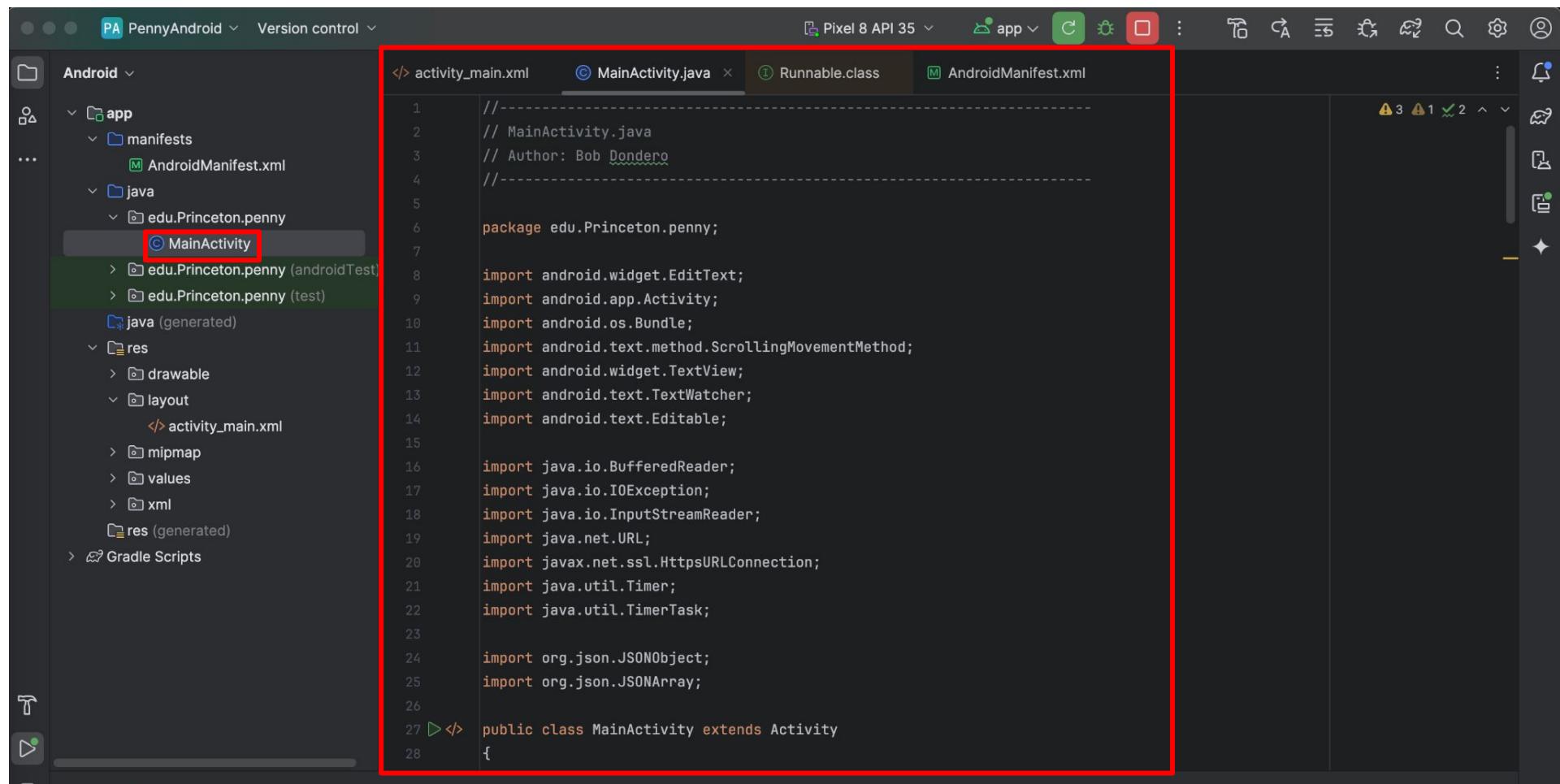


PennyAndroid App: Defining

- 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

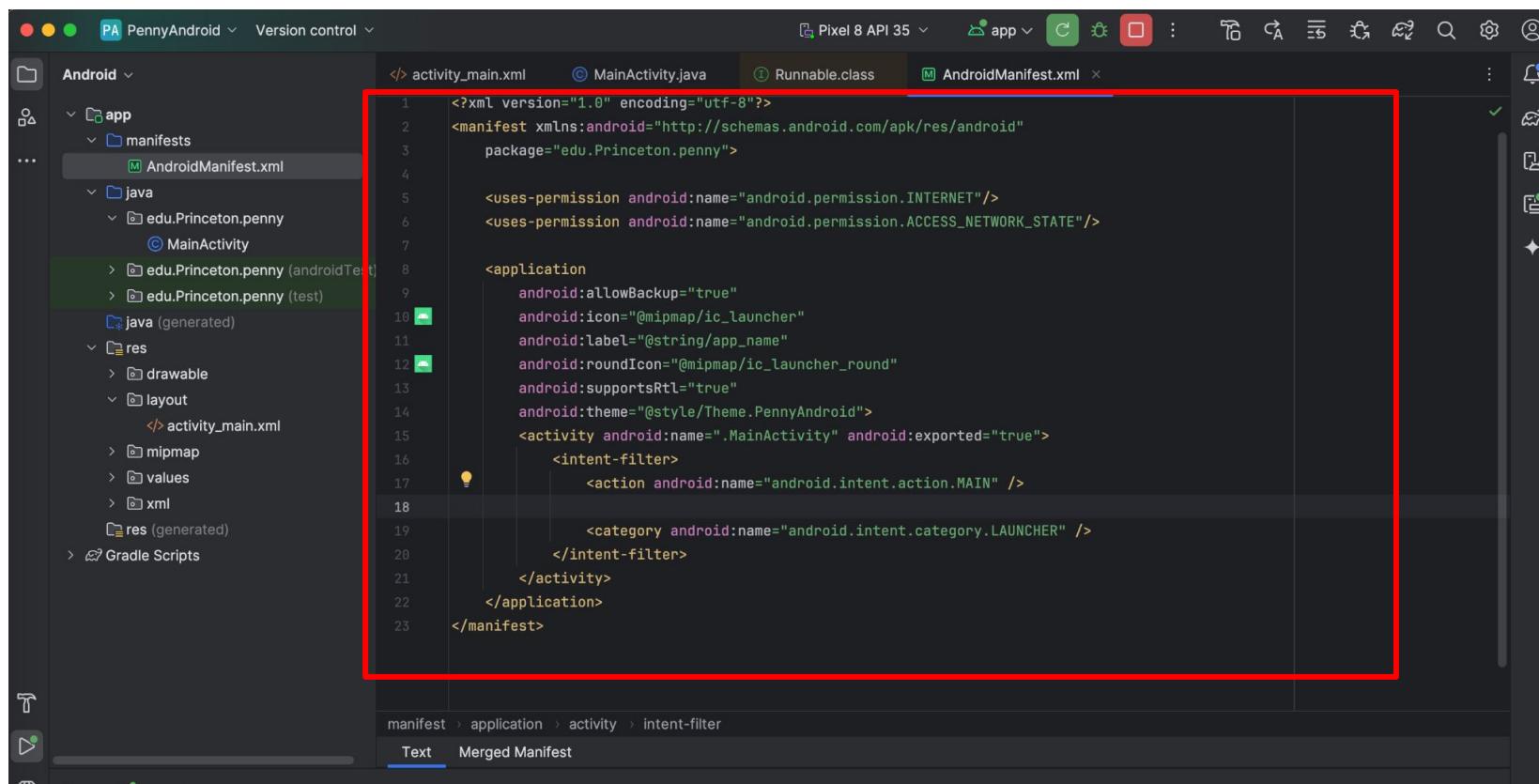
- **MainActivity.java**

- 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 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 project 'PennyAndroid' open. The left sidebar displays the project structure, including the 'app' module which contains 'manifests' and 'AndroidManifest.xml'. The right side shows the code editor with the 'AndroidManifest.xml' file open. The XML code defines the application's manifest, specifying permissions for the Internet and Network State, and defining the main activity with its intent filter and launcher category.

```
<?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>
```

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

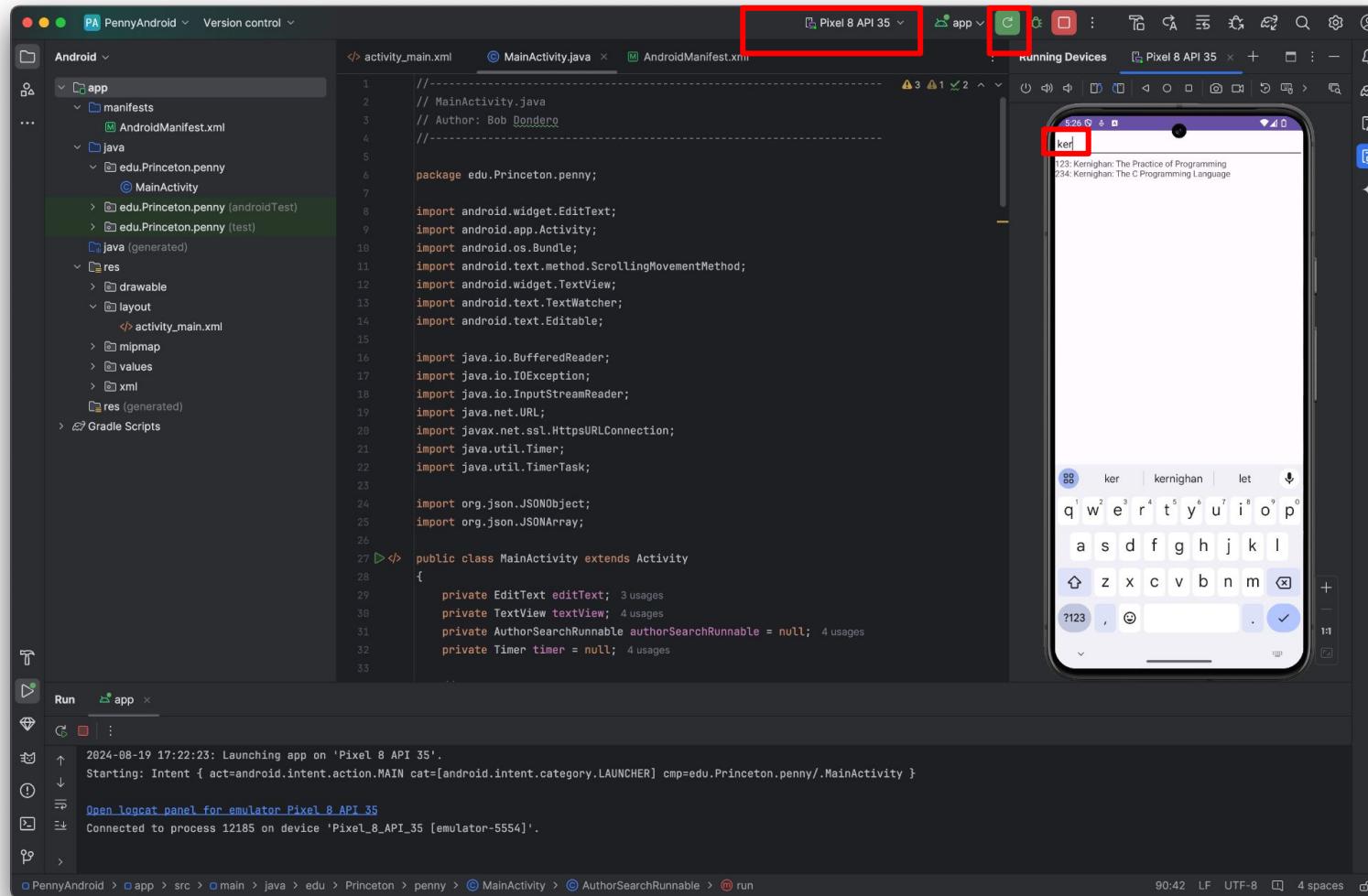
- To run on an Android emulator...

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 dropdown for "Version control".
- Run Configuration:** The dropdown shows "samsung SAMSUNG-SM-J32...". A red box highlights this dropdown, and a green box highlights the "Run" button (a green triangle icon) in the toolbar.
- Project Structure:** The left sidebar shows the project structure under "Android".
 - app:** Contains "manifests" (AndroidManifest.xml), "java" (MainActivity.java, edu.Princeton.penny), "res" (drawable, layout, activity_main.xml, mipmap, values, xml), and "res (generated)".
 - Gradle Scripts:** Shows build.gradle and settings.gradle.
- Main Activity Code:** The main content area displays the code for MainActivity.java.

```
25 import org.json.JSONArray;
26
27 public class MainActivity extends Activity
28 {
29     private EditText editText; 3 usages
30     private TextView textView; 4 usages
31     private AuthorSearchRunnable authorSearchRunnable = null; 4 usages
32     private Timer timer = null; 4 usages
33
34     //-----
35
36     private class AuthorSearchRunnable implements Runnable 2 usages
37     {
38         private String author; 2 usages
39         private boolean shouldStop = false; 3 usages
40
41         AuthorSearchRunnable(String author) 1 usage
42         {
43             this.author = author;
44         }
45
46         public void setShouldStop() 1 usage
47         {
48             shouldStop = true;
49         }
50
51         public void run()
52         {
53             String[] results = null;
```

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

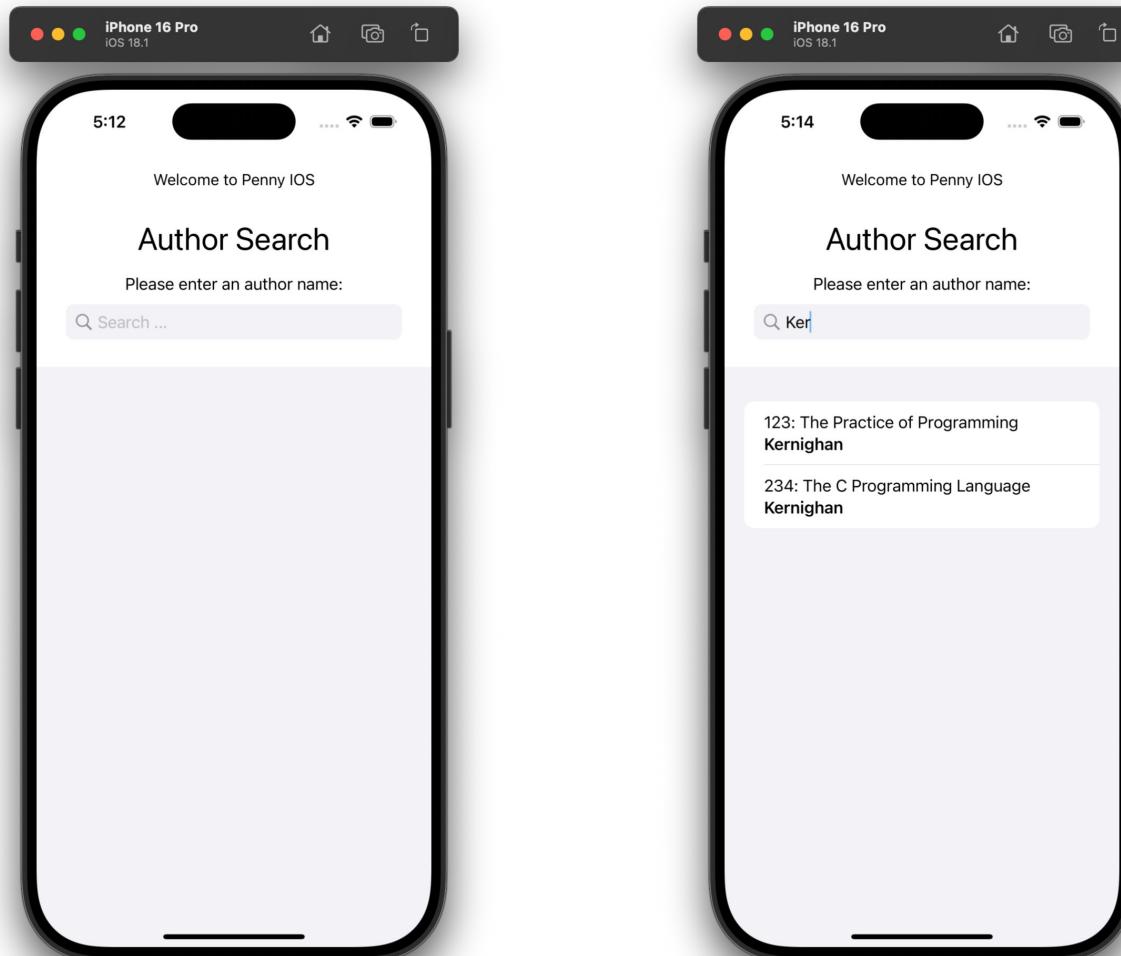
Thanks to
Katie DiPaola ('26)...

PennyAndroid App: Defining

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

Pennylos App: Defining

The
goal:



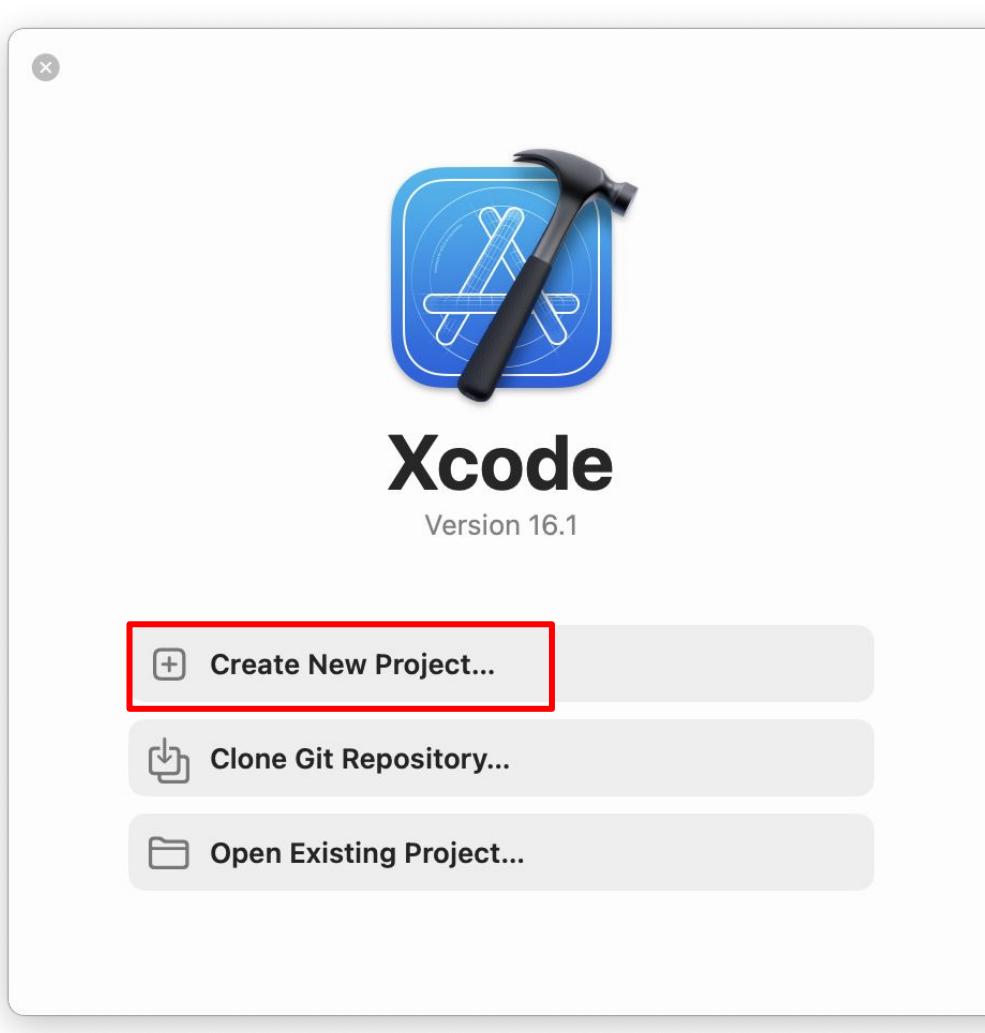
An iOS client for the PennyJson server

Pennylos App: Defining

- Download and install **XCode**

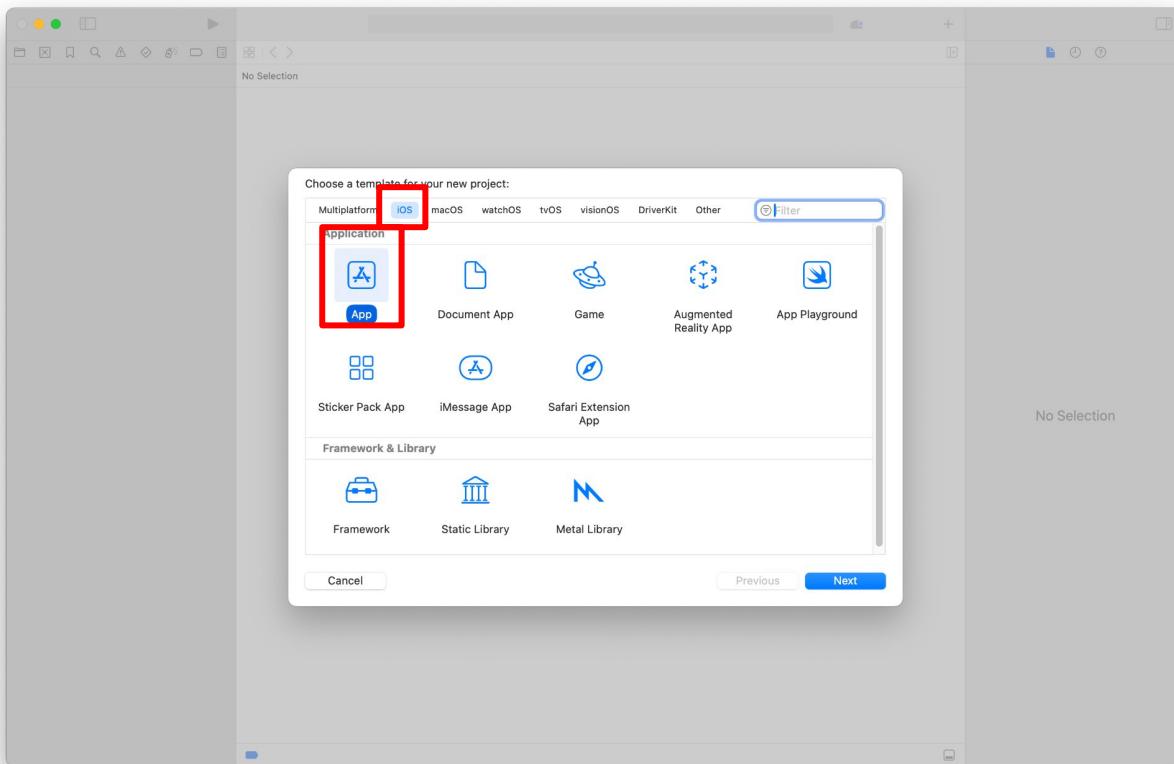
Pennylos App: Defining

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



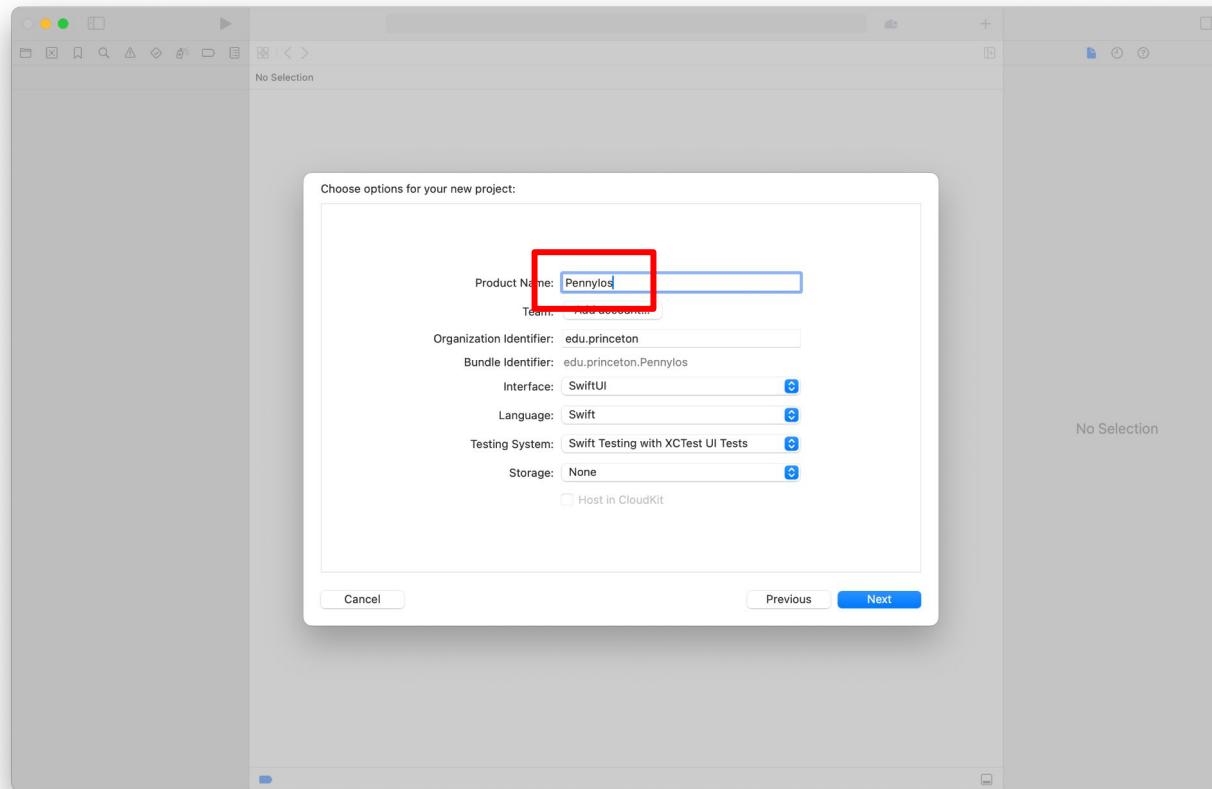
Pennylos App: Defining

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



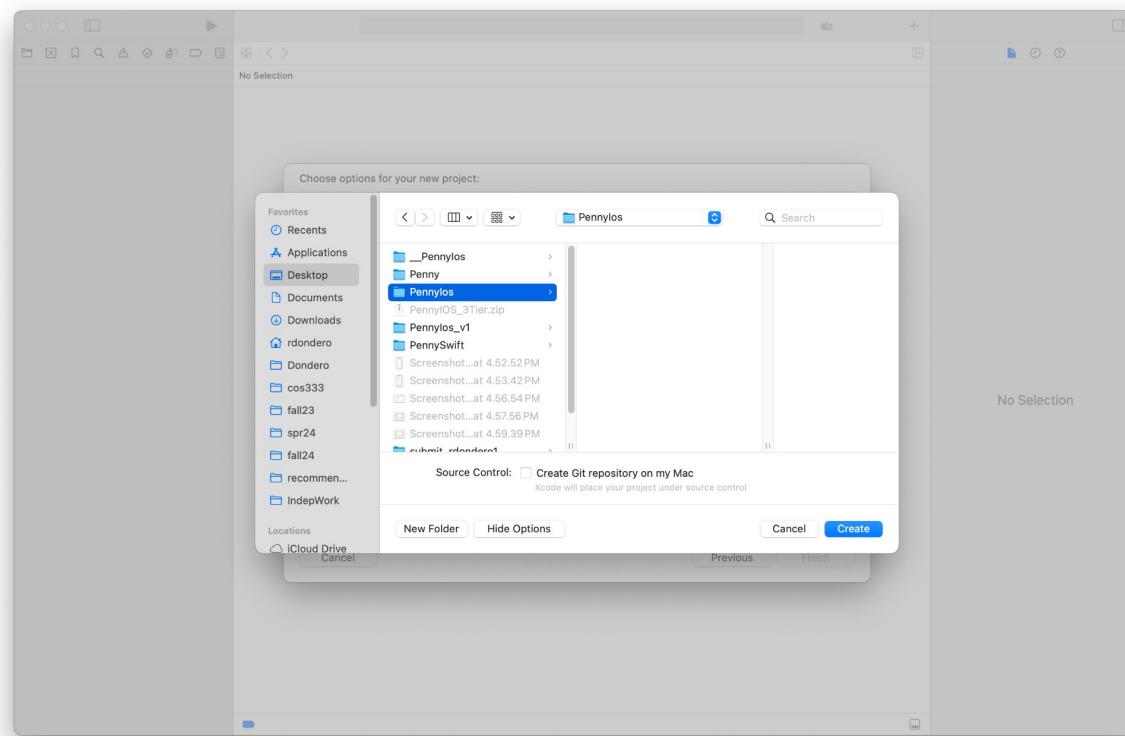
Pennylos App: Defining

For Product Name enter PennyIos; click on Next



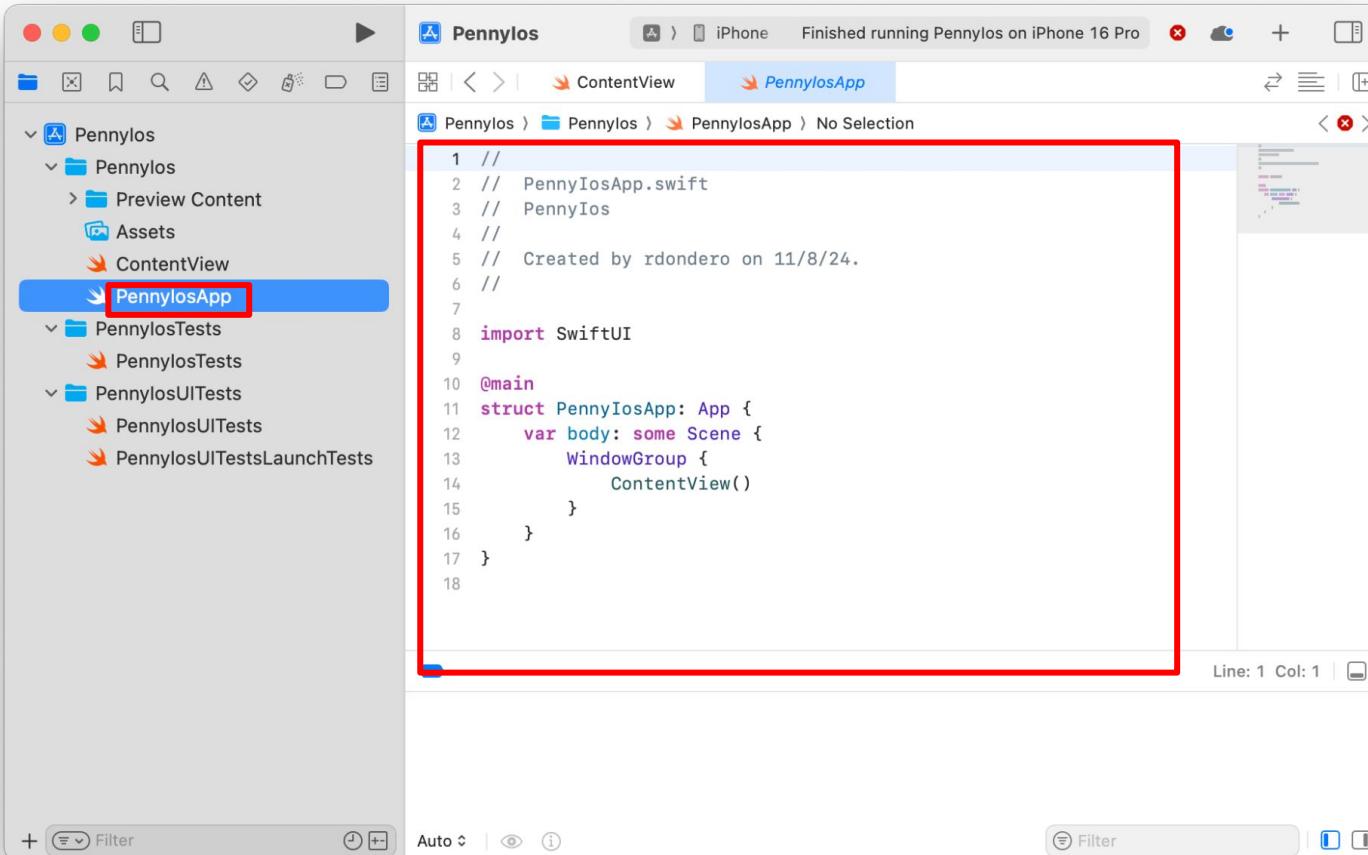
Pennylos App: Defining

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:

- Pennylos (selected)
- Pennylos (group)
- Preview Content
- Assets
- ContentView
- PennylosApp** (highlighted with a red box)
- PennylosTests
- PennylosUITests
- PennylosUITestsLaunchTests

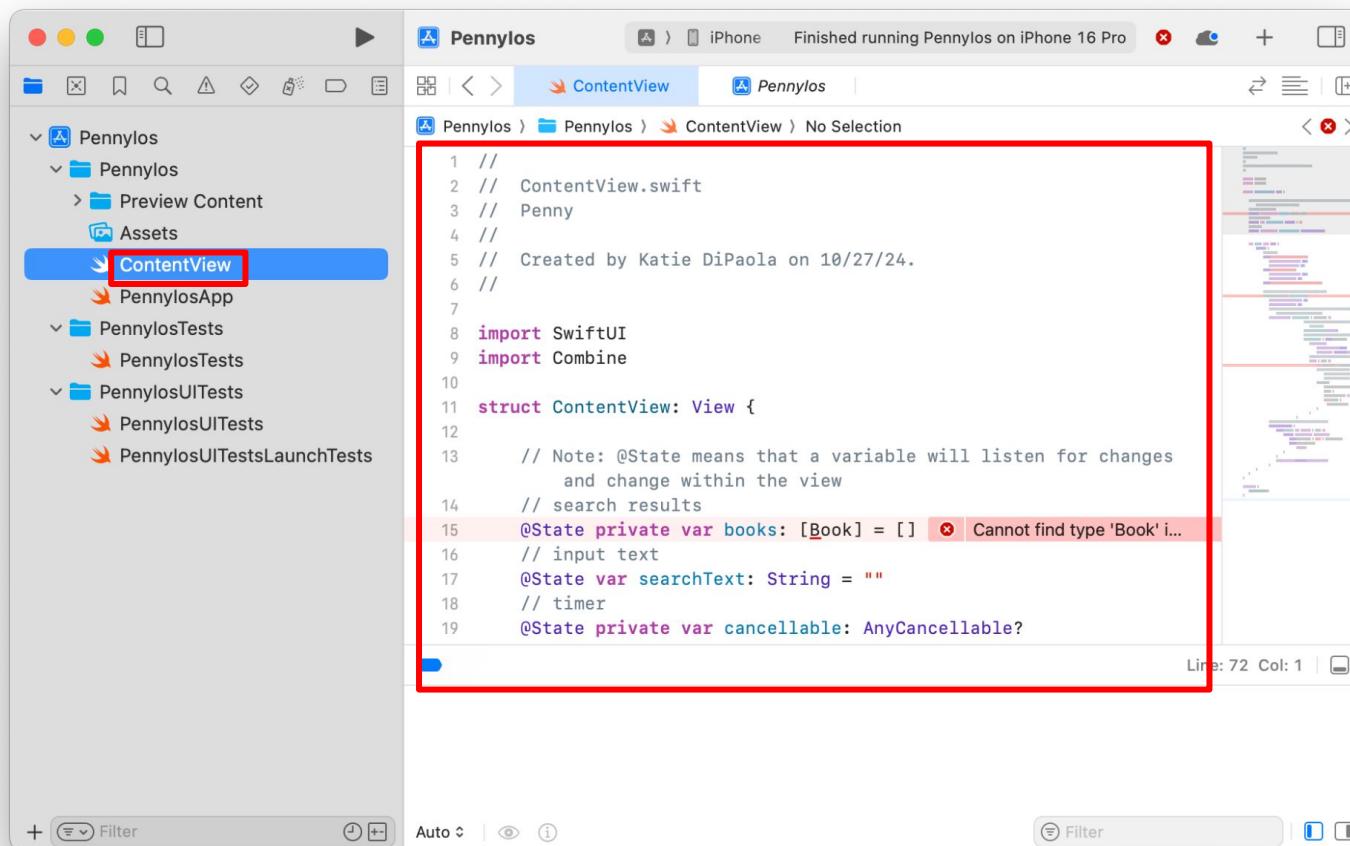
The main editor area shows the generated `PennylosApp.swift` file:

```
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 box highlights the entire code block in the editor.

Pennylos App: Defining

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



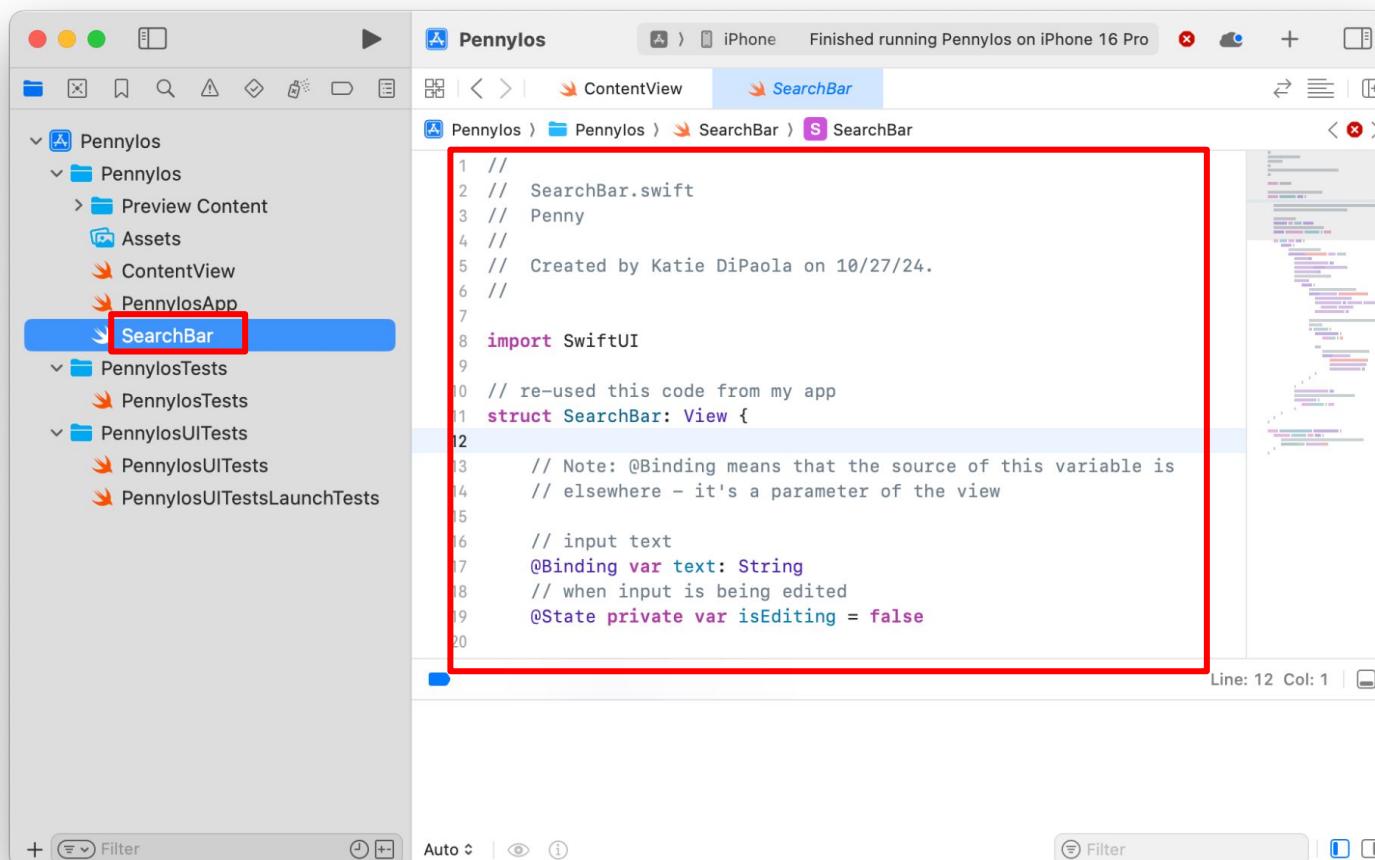
The screenshot shows the Xcode interface with the project 'Pennylos' open. The left sidebar displays the project structure, including 'ContentView' which is selected and highlighted with a blue background. The main editor area shows the 'ContentView.swift' file content. A red box highlights the entire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

```
// ContentView.swift
// Pennylos
// 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] = [] Cannot find type 'Book' i...
    // input text
    @State var searchText: String = ""
    // timer
    @State private var cancellable: AnyCancellable?
}
```

Pennylos App: Defining

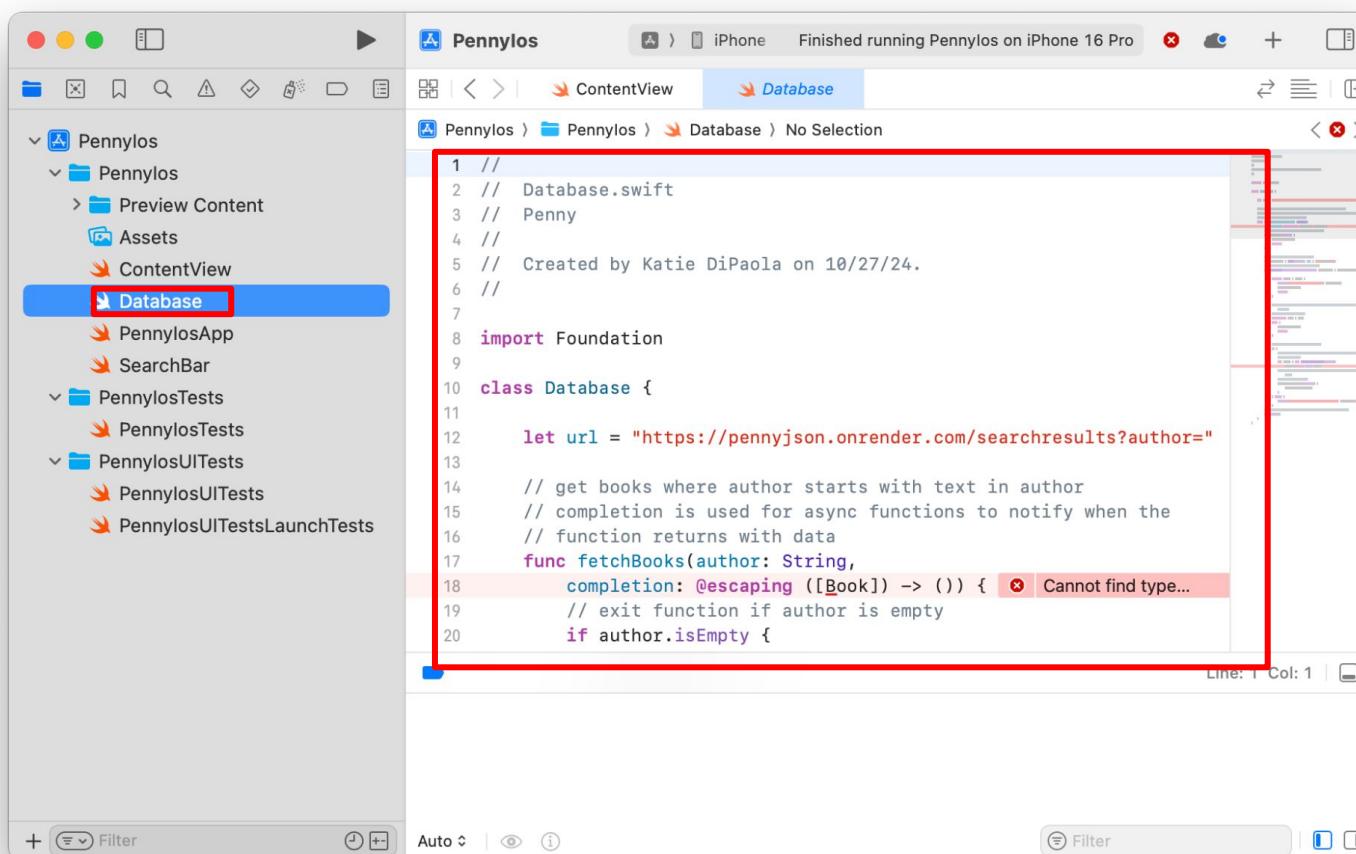
Create new file *SearchBar*; copy/paste the given code



```
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
```

Pennylos App: Defining

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



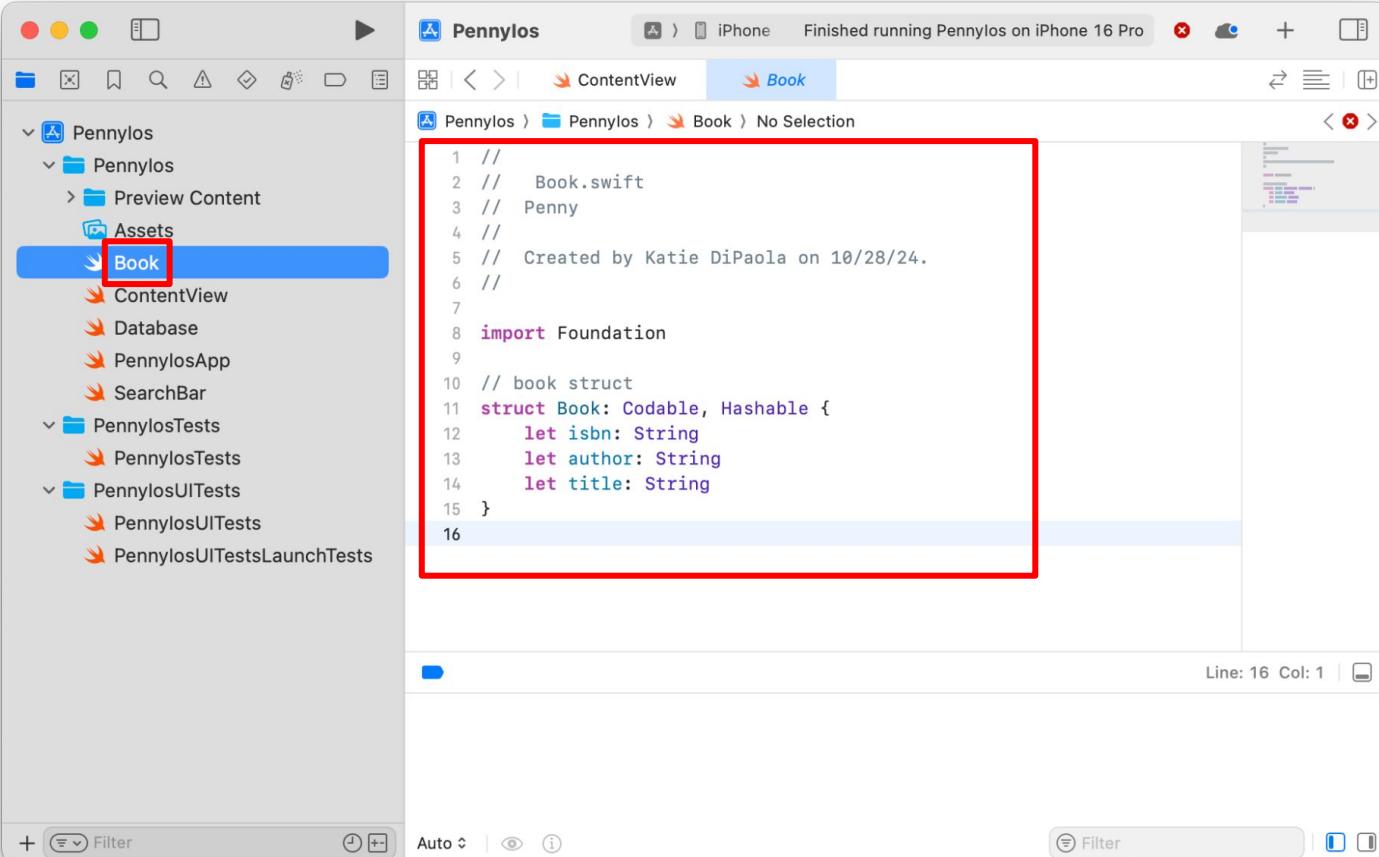
The screenshot shows the Xcode interface with the project 'Pennylos' open. The left sidebar displays the project structure, including 'Pennylos', 'Assets', 'ContentView', 'Database' (which is selected and highlighted with a blue bar), 'PennylosTests', and 'PennylosUITests'. The main editor area shows a Swift file named 'Database.swift' with the following code:

```
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 rectangular box highlights the entire code block in the editor. A tooltip 'Cannot find type...' appears over the word 'completion' in the code.

Pennylos App: Defining

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



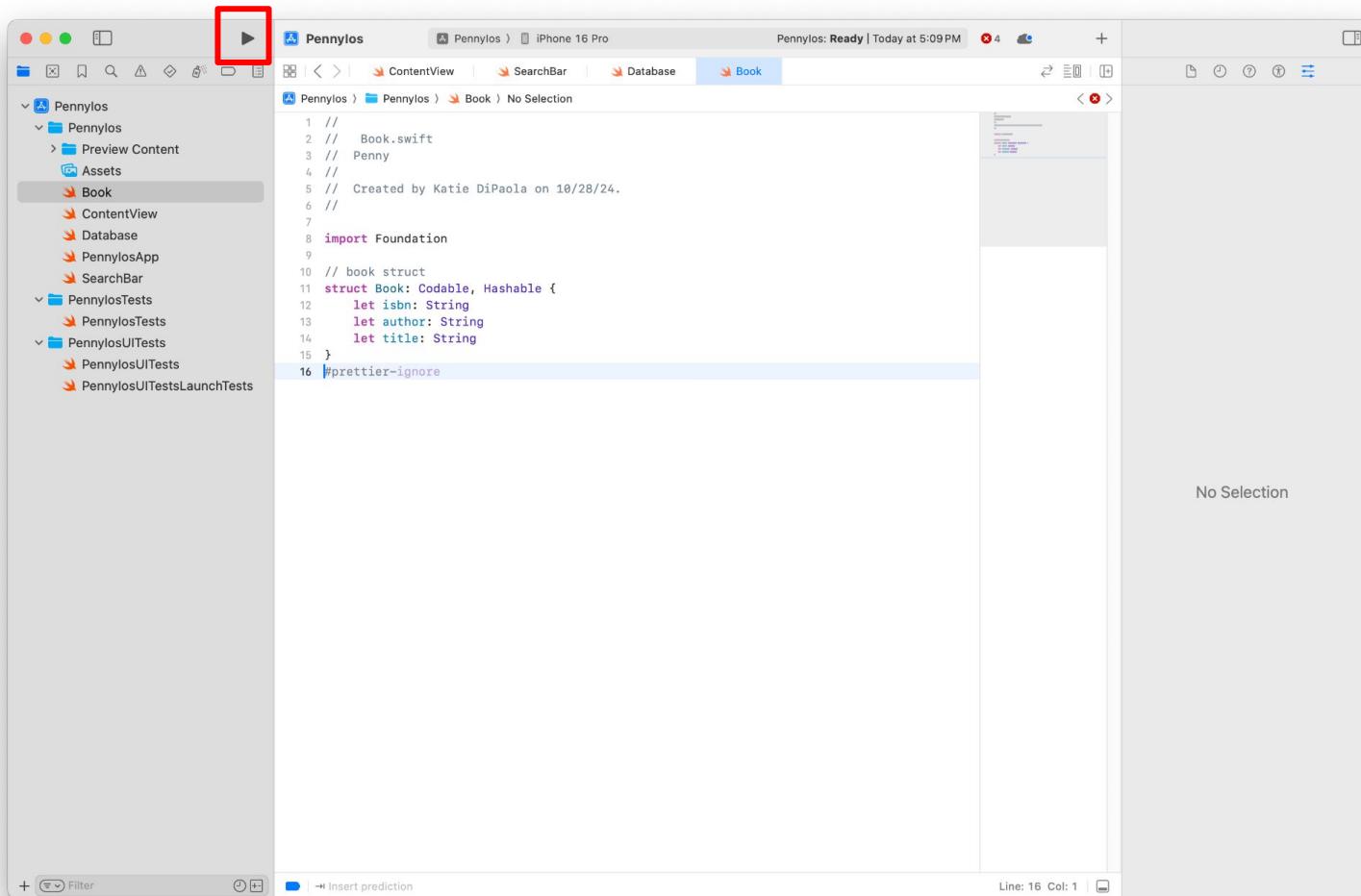
```
1 //  
2 //  Book.swift  
3 //  Penny  
4 //  
5 //  Created by Katie DiPaola on 10/28/24.  
6 //  
7  
8 import Foundation  
9  
10 // book struct  
11 struct Book: Codable, Hashable {  
12     let isbn: String  
13     let author: String  
14     let title: String  
15 }  
16
```

Agenda

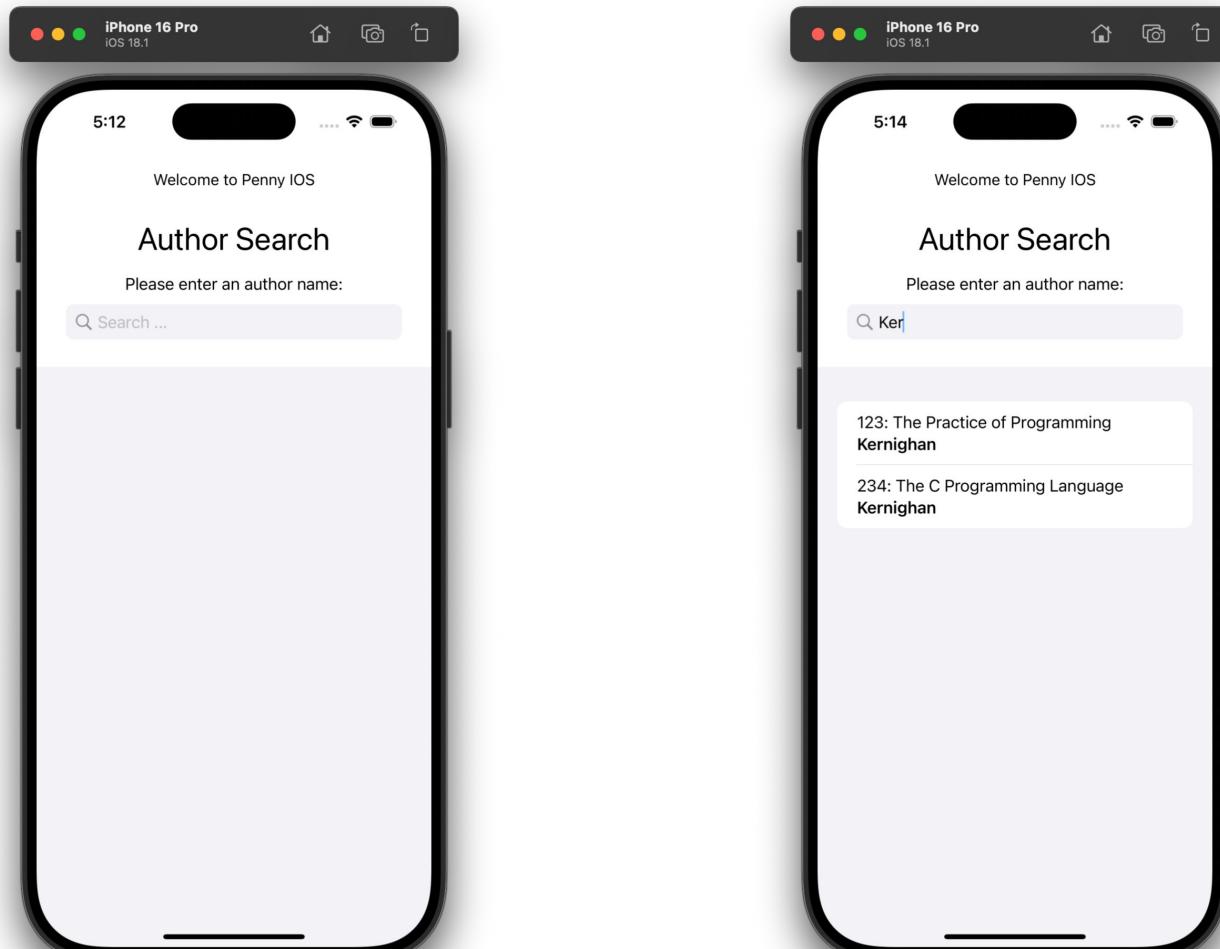
- Aside: Function def expressions
- Aside: Multithreaded programming
- Mobile programming
- PennyAndroid app: defining
- PennyAndroid app: running
- Pennylos app: defining
- **Pennylos app: running**

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  
$
```

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
$
```

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  
$
```

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  
$
```

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!