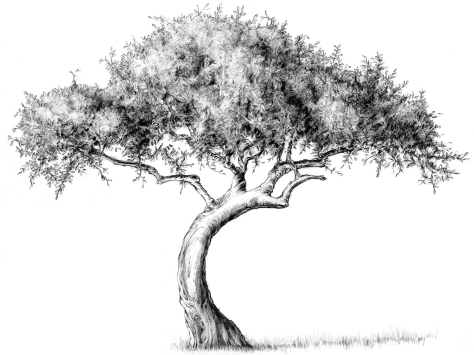
Android Dev Series: Family Tree App

Part One(?)

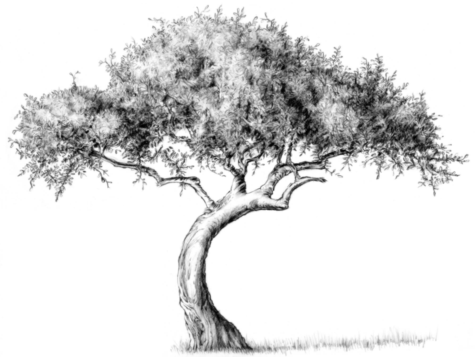
Project Specifications

- Make an app that someone can use to keep track of their family tree
- Requirements:
    - Add members
    - Delete members
    - Keep info about members
    - Search for members
    - Include a view that shows an actual family tree
    - Use fancy animations



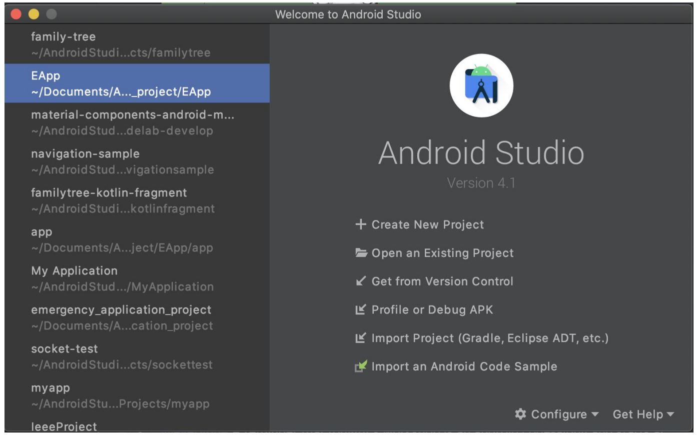
Project Specifications

- Make an app that someone can use to keep track of their family tree
- Requirements:
    - Add members
    - Delete members
    - Keep info about members
    - Search for members
    - Include a view that shows an actual family tree
    - Use fancy animations



Setup

1. Open Android Studio
2. If you have the welcome screen, click “Create New Project”
   a. Otherwise, click File > New > New Project…
3. Select “Empty Activity” and click “Next”
4. Give your app a name
5. Make sure the selected language is Java
6. Set the minimum SDK to “API 21: Android 5.0 (Lollipop)”
7. Click “Finish”


Welcome to Android Studio

family-tree
~/AndroidStudi...cts/familytree
EApp
~/Documents/A..._project/EApp
material-components-android-m...
~/AndroidStud...delab-develop
navigation-sample
~/AndroidStud...vigationsample
familytree-kotlin-fragment
~/AndroidStudi...kotlinfragment
app
~/Documents/A...ject/EApp/app
My Application
~/AndroidStud.../MyApplication
emergency_application_project
~/Documents/A...cation_project
socket-test
~/AndroidStudi...cts/sockettest
myapp
~/AndroidStu...Projects/myapp
leeeProject

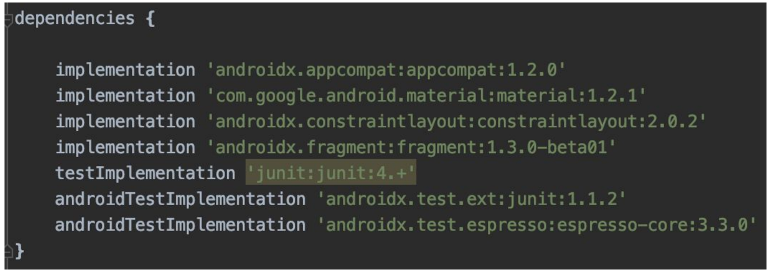
Android Studio
Version 4.1

Setup

1. Setup a virtual device
    a. Tools > AVD Manager > Create Virtual Device…
    b. Select “Pixel 3a XL” and click “Next”
    c. Select “R, API Level 30” as the system image and click “Next”
        i. You will have to download the image first if you haven’t already
    d. Click “Finish”
2. Setup the SDK
    a. Tools > SDK Manager
    b. Check API levels 21 through 30 and click “OK”
        i. This will install any of the SDKs that you checked that aren’t installed yet

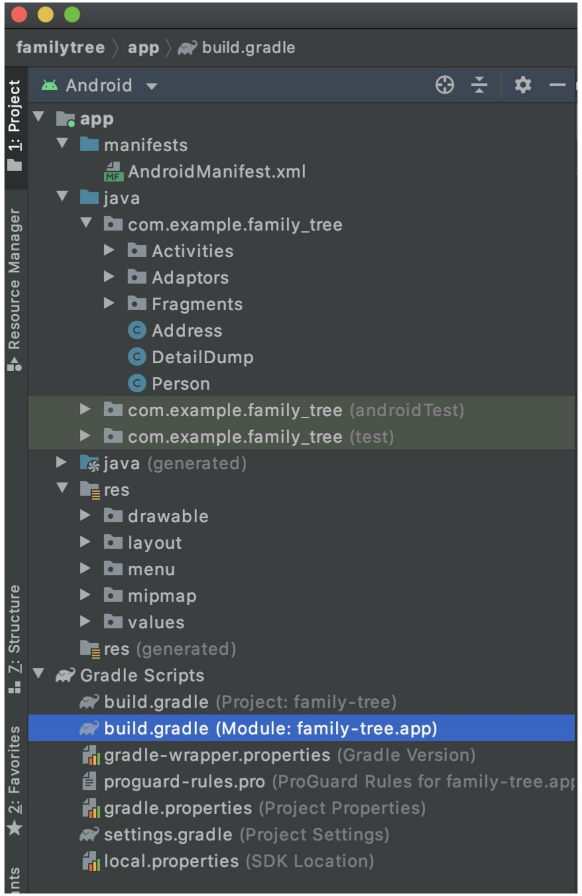
Setup

1. Navigate to build.gradle (Module: your-app-name.app)
2. State our dependencies
   a. Add the following within dependencies { … }:
   b. implementation 'androidx.fragment:fragment:1.3.0-beta01'



```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.2.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.2'
    implementation 'androidx.fragment:fragment:1.3.0-beta01'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}
```



```
familytree 〉 app 〉 build.gradle
Android
▼ app
  ▼ manifests
      AndroidManifest.xml
  ▼ java
    ▼ com.example.family_tree
      ▶ Activities
      ▶ Adaptors
      ▶ Fragments
        Address
        DetailDump
        Person
    ▶ com.example.family_tree (androidTest)
    ▶ com.example.family_tree (test)
  ▶ java (generated)
  ▼ res
    ▶ drawable
    ▶ layout
    ▶ menu
    ▶ mipmap
    ▶ values
    res (generated)
▼ Gradle Scripts
    build.gradle (Project: family-tree)
    build.gradle (Module: family-tree.app)
    gradle-wrapper.properties (Gradle Version)
    proguard-rules.pro (ProGuard Rules for family-tree.app
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

Sections

1. Setup our layouts
2. Setup our

But first...

1. Open the folder “res” in the navigation menu
2. Click on the drawable directory
    a. Click File > New > Vector Asset
    b. Click “Clip Art” and select an icon
    c. Click “Next” and then “Finish”
    d. Now there should be a new xml file in the drawable folder
3. Repeat step 2 for an add, menu, back, search, info, settings, and overflow icon

activity_main.xml (finally!)

1. Add code and run
2. Leave out SearchView for now
3. Explore the possibilities of floating action button and bottom app bar

app_bar_menu.xml

- Click on the “res” folder
  - Click File > New > Android Resource Directory
  - Under “Resource Type” select “menu”
  - Click “OK”
- Click on the menu folder we just created
  - Click File > New > Menu Resource File
  - Name the file “app_bar_menu”
- This puts some icons on the bottom app bar

fragment_bottomsheet.xml

- Add code
- Configure “bottom_nav_drawer_menu.xml” in the menu folder

MainActivity.java

- Add reference to bottom app bar
- Add reference to floating action button
- Create the inner class BottomNavigationDrawerFragment for the menu icon
- Setup the bottom app bar
- Setup the floating action button
- Run the app

HomeFragment.java

Creating HomeFragment.java

1. Click on java > com.example.your_app_name in the file navigator
2. Click File > New > Fragment > Fragment (Blank)
3. Name this fragment “HomeFragment”
4. Make sure the language is Java
5. Click “Finish”
6. Open fragment_home.xml

fragment_home.xml

- Add code
- Switch to MainActivity.java

Update MainActivity.java

1. getSupportFragmentManager().beginTransaction().add(R.id.*host_fragment*, new HomeFragment(), HOME_FRAG_TAG).commit();
2. Switch to HomeFragment.java

HomeFragment.java

- Make sure it’s inflating in onCreateView()

Pause... activities vs. fragments

What's the difference?

Good question!

Activities
- Good to use as a box that holds all of your fragments
- You can swap fragments in and out
- "God activity"

Fragments
- Android's solution to creating reusable user interfaces
- Makes things easier when you try to make UIs for tablets and phones
- Think of them as a self-contained widgets
- ...but oftentimes they need to communicate with activities or other fragments

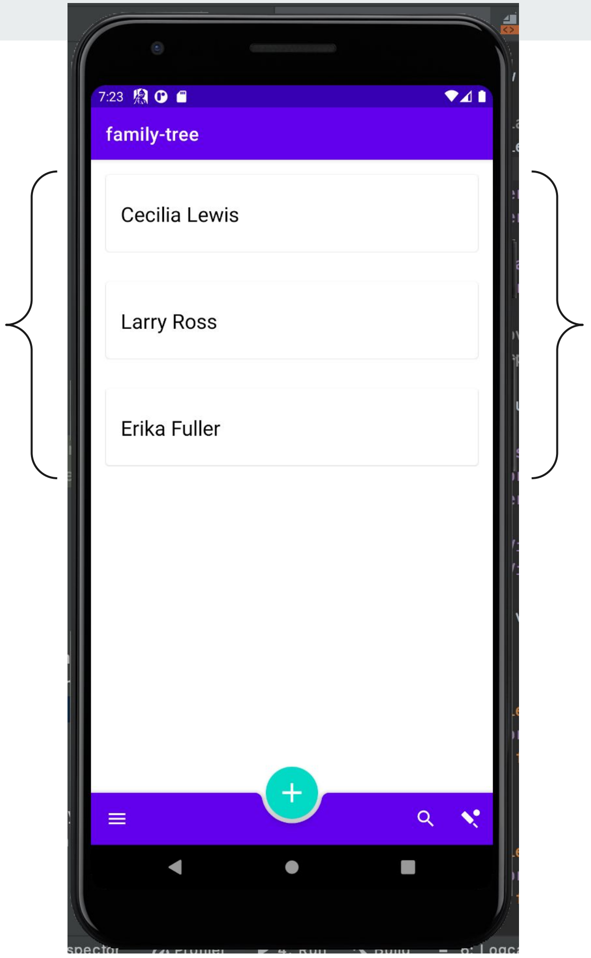
Adding a list of names

adaptor_person_item.xml

- First things first

RecyclerView

- A container that manages a list for us
- Add this to fragment_home.xml
- Update HomeFragment.java, specifically add references and update onCreateView()
- Create Person.java
- Create Address.java
- Create DetailDump.java
- Create PersonAdaptor.java
    - Recycler view really only manages how the user scrolls and animations, stuff like that
    - The adaptor is what will actually manage the items, tell the recycler view when to refresh, etc.



PersonAdaptor.java

- PersonAdaptor extends RecyclerView.Adaptor<PersonAdaptor.MyViewHolder>
- Needs these references: mDataset, mainActivity
- Make inner class MyViewHolder extends RecyclerView.ViewHolder
    - Add references and bind() method
- Add constructor
- Implement onCreateViewHolder
- Implement onBindViewHolder
- Implement getItemCount

A ViewHolder is like the row in the list

That's all for part 1!