CSCB07 - Android Studio

Android & Components

Android

- Android is an platform comprising three components
    - An operating system
    - A framework for developing applications
    - Devices that run the Android operating system and the applications created for it

Android SDK

- A collection of libraries and tools that are needed for developing Android applications

Android Studio

- IDE for Android application development

Android App Basics & Layout

An Android app is a collection of screens, and each screen is comprised of a layout and an activity

- Layout: describes the appearance of a screen (written in XML)
- Activity: responsible for managing user interaction with the screen (written in java)

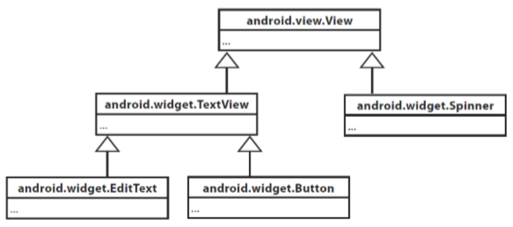
Folder structure

- Manifest file
    - Defines the structure and metadata of an application, its components, and its requirements
    - Stored in the root of its project hierarchy as an XML file
- Java files
    - Activity code (+ any additional code [classes/runtime code])
- Resource files
    - Non-Java files used in application: layout XMLs, images, String constants, etc.
    - Resources are maintained in sub-directories of the app/resdirectory (res/layout, res/values, etc.)
    - A resource can be accessed in the code using its resource ID (e.g. R.layout.activity_main)
        - Android uses R.java to keep track of the resources used within the app
- Gradle scripts
    - Build system, used to compile and deploy application
    - Part of folder structure
    - Has libraries/dependencies for application, specifies SDK

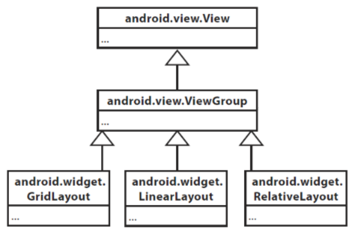
Random Specifics from Lec 9

View

- Most GUI components are instances of the Viewclass or one of its subclasses (e.g Button, EditText, ImageView, etc.)



- Common GUI components include TextView, EditText, Button, Switch, Spinner, Toast
- View Group
  - A special type of view that can contain other views
  - A layout is a type of view group


android.view.View
...
android.view.ViewGroup
...
android.widget.GridLayout
...
android.widget.LinearLayout
...
android.widget.RelativeLayout
...

Intents

- An intent is an object that can be used to bind activities together at runtime
  - If one activity wants to start a second activity, it does it by sending an intent to Android. Android will start the second activity and pass it the intent.
- Data can be passed between activities using intent extra
  e.g. intent.putExtra("message", value);

Data storage options

File System

- Store data in arbitrary file, when data is unstructured
- idk man just use FIS or whatever, we didn't even use it for the project
- Android's file system consists of six main partitions
  - /boot - Android kernel
  - /system - Android OS
  - /recovery - Backup files
  - /data - User data
  - /cache - Frequently used data
  - /misc - Settings
- Reading/writing data to a file on internal storage can be done using openFileInput(), openFileOutput()

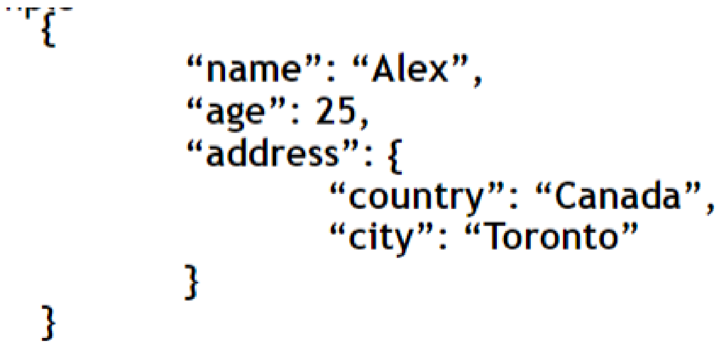
Shared preferences

- Suitable for simple data that could be stored as key/value pairs
- A SharedPreferences object refers to a file containing key/value pairs and provides methods to read and write them
- Creating/accessing shared preference files can be done using: getPreferences(), getSharedPreferences()

Database

- Data is structures/complex (e.g. student information in Excel sheet)
- SQLite (Structured Query Language Lite):
  - Essentially a library that allows offline interaction with a database file stored locally (used when the application accesses local data only)
  - Relational database (might prevent adding components where there isn't a match)
  - Serverless, zero-configuration, file-based
- Firebase Realtime Database:
  - Google's pretty solution for holding a JSON file online and having a library to interact with it
  - Cloud-hosted

    - Employs data sync (meaning all of your clients receive any changes to the database)
    - "NoSQL" (a fancy term for saying it's using JSON instead of a relational database :^])
    - The Firebase SDK provides many classes and methods to store and sync data. (e.g. DatabaseReference, DataSnapshot, ValueEventListener)
    - “Wait, who the heck is JSON?”
        - JavaScript Object Notation
        - Language independent (so ignore the JS portion)
        - Supported by many programming languages
        - Uses readable text to represent data in the form of key/value pairs



```
{
        “name”: “Alex”,
        “age”: 25,
        “address”: {
                “country”: “Canada”,
                “city”: “Toronto”
        }
}
```

Model-View-Presenter

- Some design pattern we got hit with at the last minute- I mean, an architectural design pattern that results in code that is easier to test (especially in the context of mobile apps)
- Consists of 3 components:
    - Model (Data, file/database, goal is to manage the data)
    - View (UI, different components that user interacts with)
    - Presenter (Main component, holds the business logic)
- Main idea of this design is to make a presenter that is *independent* of the type of Model or View attached to it (e.g. it shouldn't use some method that is Android or Firebase specific)

Testing & Testing Classes

Local and Instrumented Tests:

- Local unit tests:
    - Run on the machine's local JVM
    - Do not depend on the Android framework
    - Unit tests have the most number of tests and are least costly/complex
    - e.g. JUnit
- Instrumented tests
    - Run on an actual device or an emulator
    - Usually used for integration and UI tests
    - Integration and UI tests have the least number of tests but are most costly/complex
    - e.g. Espresso

JUnit

- Writing unit tests

Mockito

- Mocking framework for Java
- Allows for writing tests using the Android API without actually using it
- Features include creating mocks, stubbing, verifying behaviour, creating spies (that monitor certain objects)
- "Mock?"
  - A mock is software component that is used to replace the real component during testing
  - Test logic of function without regard for external influence
  - Eliminate failure from external dependencies
  - Could be used to represent components that have not yet been implemented, speed up testing, reduce the cost, avoid unrecoverable actions, etc.

Roboelectric

- Running tests that involve the Android framework without an emulator or device

Espresso

- Writing UI tests