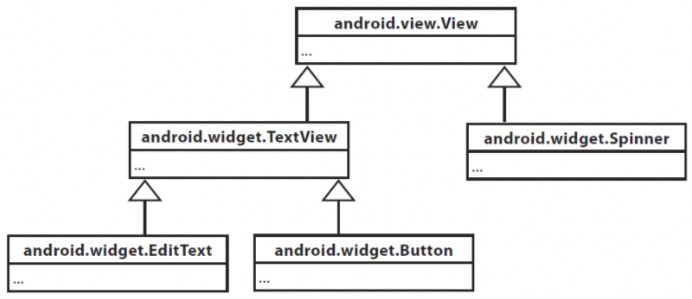
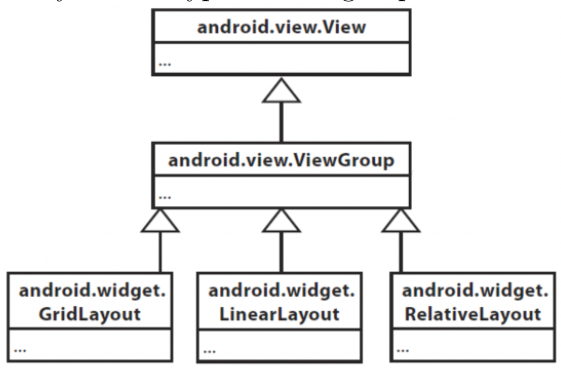
Introduction to Android

- Android
  - Android is a platform comprising of 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
  - 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
    - Resource files
    - Java file
    - Gradle scripts
- The Manifest file
  - It 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
- Resources and resource IDs
  - Resources are maintained in sub-directories of the `app/res` directory
    - `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
- View
  - Most GUI components are instances of the **View** class or one of its subclasses
    - e.g. Button, EditText, ImageView, etc.



- 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
...

- Common GUI compnents
  - TextView
  - Switch
  - EditText
  - Spinner
  - Button
  - Toast

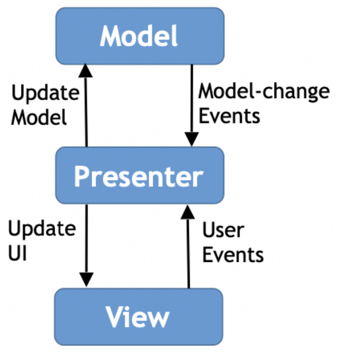
- Intents
  - An intent is an object that can be used to bind activities together at runtime
    - If one activity wants to start a second activity, it does it by sending an intent to Android. Android will start the second activity and pass it the intent
  - Data can be passed between activities using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

Android - Storing Data

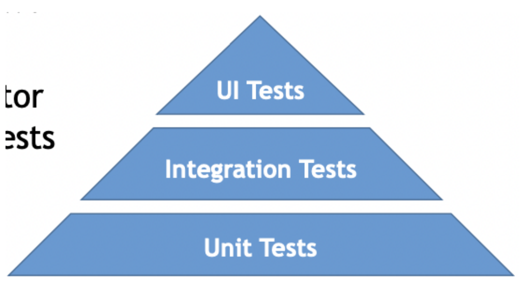
- Data storage options
  - File system
  - Shared preferences
  - Databases
    - e.g. SQLite, Firebase Realtime Database
- File system
  - Android's file system consists of six main partitions
    - `/boot`
    - `/recovery`
    - `/cache`
    - `/system`
    - `/data`
    - `/misc`
  - Reading/writing 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()`
- SQLite
  - Relational database
  - Zero-configuration
  - Widely used
  - Serverless
  - File-based
- Firebase Realtime Database
  - Cloud-hosted
  - Employs data synchronization
    - Every time data changes, all connected clients automatically receive updates
  - NoSQL
    - Data is stored as JSON
  - The Firebase SDK provides many classes and methods to store and sync data. E.g.
    - `DatabaseReference`
    - `ValueEventListener`
    - `DataSnapshot`
- JSON
  - **J**ava**S**cript **O**bject **N**otation
  - Language-independent
  - Supported by many programming languages
  - Uses readable text to represent data in the form of key/value pairs
  - Example

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

Android - Testing



- Model-View-Presenter
  - An architectural design pattern that results in code that is easier to test
  - It consists of three components:
    1. Model (Data)
    2. View (UI)
    3. Presenter (Business logic)



- Local and Instrumented Tests
  - Local unit tests
    - Run on the machine's local JVM
    - Do not depend on the Android framework
  - Instrumented tests
    - Run on an actual device or an emulator
    - Usually used for integration and UI tests
- Commonly used tools
  - JUnit
    - Writing unit tests
  - Mockito
    - Creating dummy (mock) objects to facilitate testing a component in isolation
  - Roboelectric
    - Running tests that involve the Android framework without an emulator or a device
  - Espresso
    - Writing UI tests
- Mock Objects
  - A mock is software component that is used to replace the "real" component during testing
  - Mock objects could be used to:
    - Represent components that have not yet been implemented
    - Speed up testing
    - Reduce the cost
    - Avoid unrecoverable actions
    - Etc.
- Mockito
  - A mocking framework for Java
  - Features include:
    - Creating mocks
    - Stubbing
    - Verifying behavior