Jacob Taggart
Week 4 Day 3

Research

1. What is the difference in a started service and a bound service?

   A started service requires the developer to start the service in the activity when necessary. The service can then potentially run in the background indefinitely including after the point the component that started the service is destroyed. Bound is different in multiple ways. Bound services are called when a component binds to one and then a client/server relationship is started allowing the two to send and receive requests and data. Bound services also allow for communication across processes via Interprocess Communication (IPC). Bound services terminate when the component unbinds from it (unless startService was used to begin the service).

2. Explain how map, flatmap, and range works in RxJava.

   Map, in RxJava, is used to transform items from observables via mapping the items to specific functions. Thus, something like getting the results from an API call and mapping them to a RecyclerView is a good example of what Map would be used for. Flatmap, on the other hand, maps observables to other observables. Thus, operations are mapped to other operations. A good example of this would be getting a profile from the API on Github and then from there the Flatmap would (flat)map the profile to the user name, which could then (flat)map to the repositories of said username. Range allows for the developer to specify the breadth of what is wanting to be retrieved. For example, to get a certain scope of results from an API call which returns object after object the developer could use range to specify the exact location and amount of objects to return.

3. What are the differences in RxJava 1 and 2?

   One major difference is that RxJava2 uses what is called the ReativeStream SPI which is a standard in Java that treats asynchronous data processing in a streaming pattern which uses backpressure control. Because of this, Flowables were introduced in RxJava2 and are an observer that have backpressure control. RxJava1 did not have a built in way to deal with backpressure control. RxJava2 was completely written from scratch and because of its new architecture, dependent libraries like Retrofit had to be updated as well. RxJava2 was rewritten in a way that greatly improves the efficiency of performance as well as memory usage.

4. Define what a observer/ subscriber relation is.

   The observer/subscriber pattern is a way in Java/Android to have potentially several observers listening for the same observable. By subscribing to a specific observable, an observer knows exactly what event/observable is going to trigger its own action. Until an observer subscribes to an observable, the observer won't have any way to carry anything out. As said previously, a wide range of observers can subscribe to a single observable if so desired.

5. Define what a client/server relation is.

   In terms of services and IPC, the client/server relationships are the communication and relative affect the client process and service process have with/on each other. This mainly deals with sending information to and from connected services and clients which is known as Interprocess Communication.