Mobile Application Design and Development

CS 5520

Adrienne Slaughter

Fall 2021

Northeastern University
College of Computer and Information Science

Class Today

- Presentation
- Review ToDo App
- Introduce helpful new ideas: mapping data to the UI

OMG!! How are we supposed to build this???

- Saving the data throughout lifecycle of app, without persisting
- Data structure to hold data
- Versioning:
  - Gradle plugin version
  - Uploading aab &target 28, google console said target 30
- Desiging activities: which path to choose?
  - Using provided AndroidStudio template (uses fragments), or just use Activities
- How to maintain data: singleton
  - On orientation change, had concurrent change exception (bundles) (solved by cloning)
- onActivityForResult
- Putting data a bundle, strings: couldn't put the Task object in the bundle
  - Parcelable
- Listview; tags: dropdown menu; -- how to populate the listview with data
- Cardview

Your Process

- Started with UI
  - Created Activity/Layout to reflect the UI
  - Data next
    - Singleton

My Process

- Create a ToDo class and a repo
- Create the list, in a simple way:
  - Custom UI component to display a ToDo
  - Modify the layout by adding instances of this UI component
    - **Problem: I put it in onCreate(), which means it doesn't update when a new todo is added to the**
- Create a fragment to show a ToDo
  - Enable create/edit
- Got the flow down– a sketch of the solution. Left the details for later.

Challenges (Milestone 1)

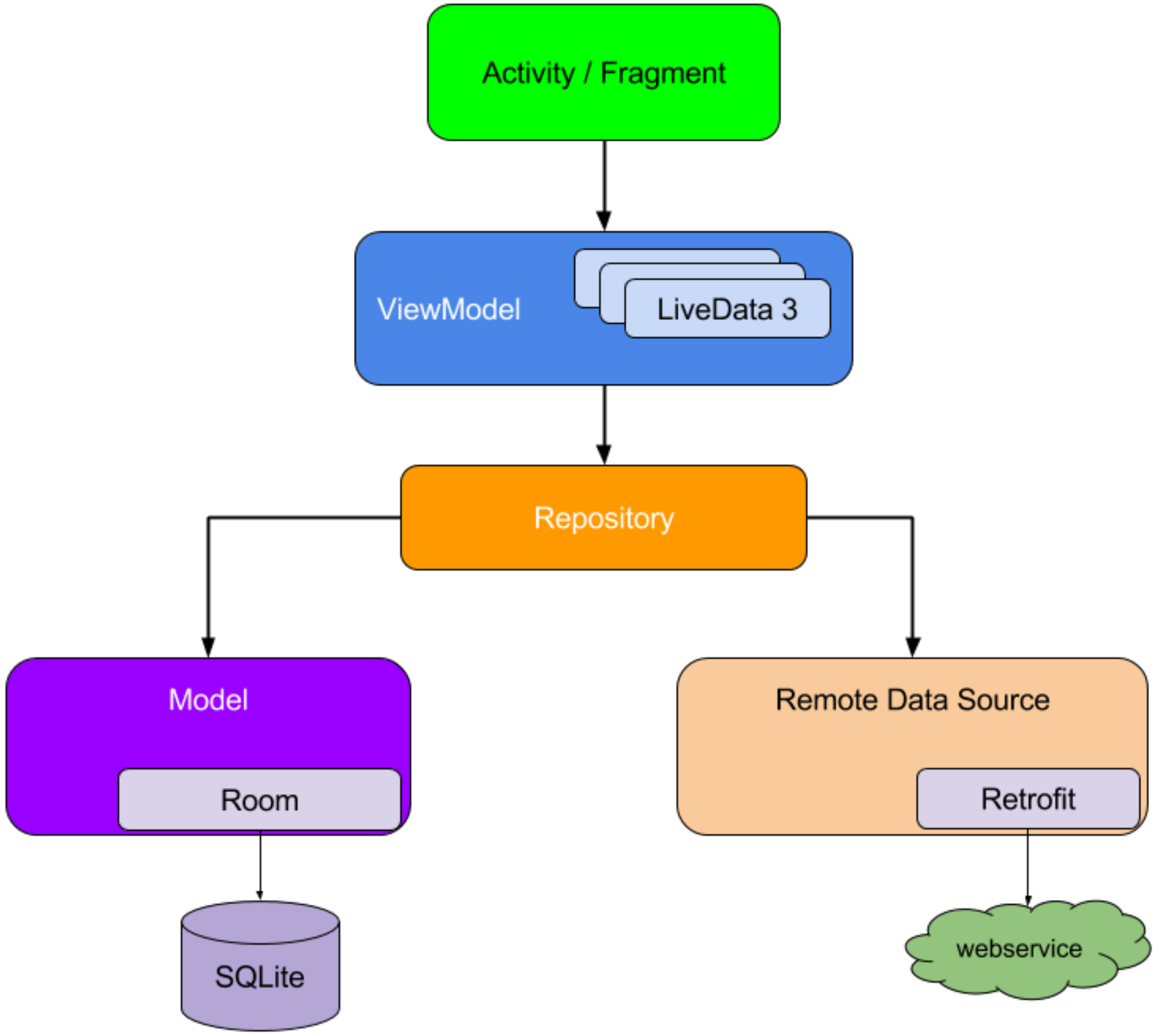
- How to structure the app
- Consistently showing data
- Sharing data among app components
- Do we REALLY have to instantiate a new thing for each thing on the screen?

Challenges (Milestone 1)

- How to structure the app
  - App Architecture
- Consistently showing data
  - Data Binding
- Sharing data among app components
  - View Model
  - Live Data, Mutable Live Data
- Do we REALLY have to instantiate a new thing for each thing on the screen?
  - RecyclerView and Adapter

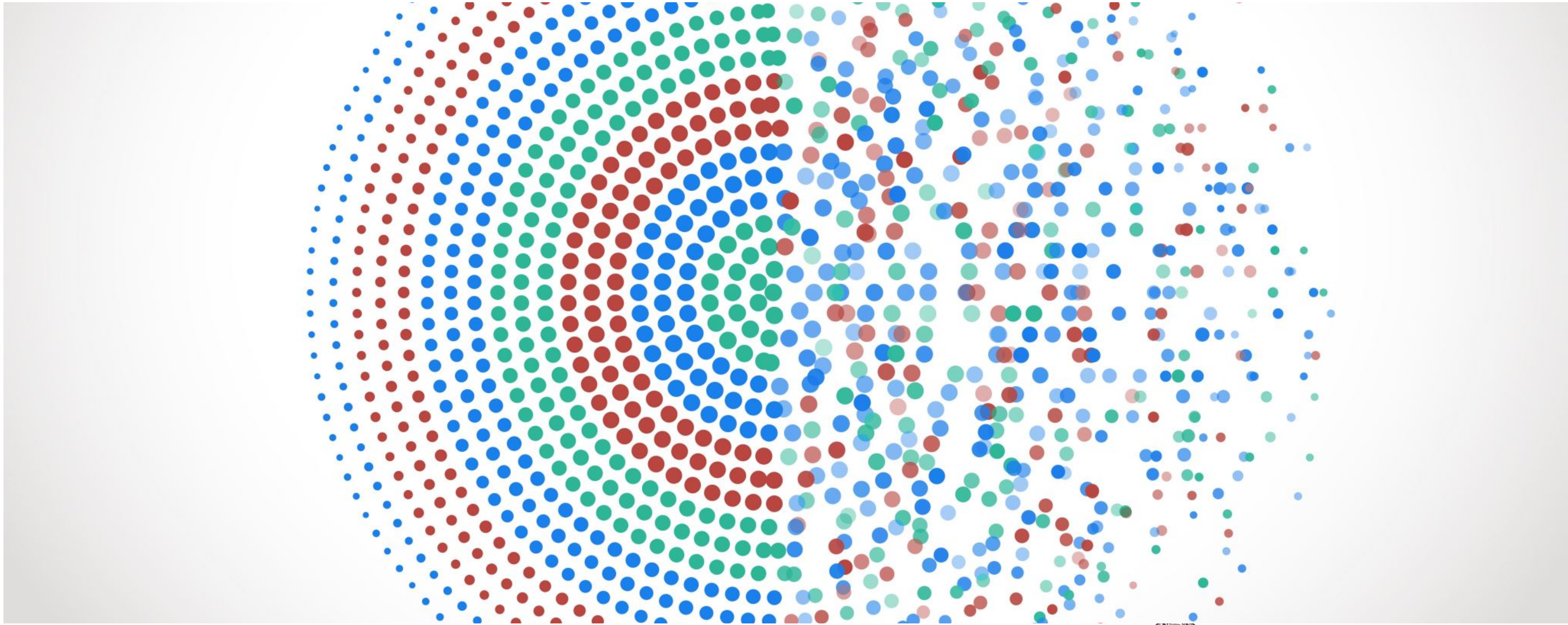
Challenges (Milestone 2)

- Persisting data
  - We won’t have a ton of time to go through all of this today…



https://developer.android.com/jetpack/guide

Jetpack Magic



Jetpack

- A bunch of tools to make the really tedious parts of creating Android apps easier.

Data Binding

- Bind UI components to data sources
- When using data binding, Android Studio creates a “Binding” class that we can use to hold the data and access UI
- Create a “data class” that holds the data we want to be used in the UI
- From the code (Java), populate the data model
- In the layout (xml), name the data and specify which fields show on which components

https://developer.android.com/topic/libraries/data-binding

ViewModel

- What we use as the base for our “data class”
- Android provides infrastructure to map an instance of the ViewModel to an Activity and/or Fragment
  - This allows us to ensure that an Activity and a Fragment can access the same instance of the ViewModel without tightly coupling the Activity and Fragment
  - Also handles the complexity of managing data across Activity/Fragment lifecycle/config changes

- https://play.google.com/store/apps/details?id=edu.neu.slaughter.sampleapp

- https://play.google.com/apps/testing/edu.neu.slaughter.sampleapp

LiveData

- The ViewModel holds data that can be displayed in a UI component
- LiveData allows UI components to update displayed data when data in the ViewModel changes
- Also allows ”stuff” to respond to changes in the data by “observe”

MutableLiveData

- Generally speaking, we like to ensure data is immutable.
- But sometimes we do want to modify the data:
  - We have a UI component that displays a data record to be modified. MutableLiveData allows the data to be updated as well, and those changes can be observed by multiple parties to respond.

Review code

- Listing the todos:
  - Main Activity, ToDoListFragment, ToDoItemView
- Creating a new Todo:
  - ToDoFragment
    - Binding, ViewModel
- Create a new ToDoListFragment: RecyclerView

Types of Storage

- Shared Preferences
  - Store private primitive data in key-value pairs.
- Internal Storage
  - Store private data on the device memory.
- External Storage
  - Store public data on the shared external storage.
- SQLite Databases
  - Store structured data in a private database.
- Network Connection
  - Store data on the web with your own network server.

Saving data

- Saving data obviously essential for any sophisticated program

- New mobile challenge: save data efficiently
  - Despite app assassin
  - Despite installs/uninstalls
  - Despite data corruption
  - Despite unreliable Internet connections

Saving data: When?

- Between Activities
- Custom view state
- App information

Saving data

Your options (most complex apps use all)

- Temporary
  - Intents
  - Application/singleton pattern
  - Bundles
- Long-term
  - Shared Preferences
  - Local files
  - Local database (SQLite)
  - Remote database

Saving data between Activities

- Intents and extras (preferred but not always realistic)
- Singleton pattern
- Shared preferences
- Saving complex objects

Intents and extras

- Set string/value StringExtra information
- Send to new Activity when started
- Possible for Activity to return information as well

Singleton pattern

```java
public class Globals{
   private static Globals instance;

   // Global variable
   private int data;

   // Restrict the constructor from being instantiated
   private Globals(){}

   public void setData(int d){
     this.data=d;
   }
   public int getData(){
     return this.data;
   }

   public static synchronized Globals getInstance(){
     if(instance==null){
       instance=new Globals();
     }
     return instance;
   }
}
```

```java
Globals g = Globals.getInstance();
g.setData(100);

....
int data=g.getData();
```

Extending Application class

```java
public class Globals extends Application{
   private int data=200;

   public int getData(){
     return this.data;
   }

   public void setData(int d){
     this.data=d;
   }
}
```

```xml
<application
   android:name=".Globals"
   .... />
```

```java
Globals g = (Globals)getApplication();
int data=g.getData();
```

Caveats with singletons

- Your entire process can be killed
- Sometimes static variables bound to activities are uninitialized ***even when they have been initialized***
- Android docs seem to encourage singleton: ***"There is normally no need to subclass Application. In most situation, static singletons can provide the same functionality in a more modular way."***

Caveats with singletons

If application process is killed (inevitable if app in background), singleton will be recreated (resetting defaults)

- Hard to reproduce this issue for testing
- Solution:
  Save data using SharedPreference or DB or files and reinit variables as needed

Saving other info

- Application preferences
- UI selections
- Data entry
- Important timing information
- ?

Shared Preferences

- Simple, lightweight key/value pair (or name/value pair NVP) mechanism
- Shared among application components running in the same application context
- Support primitive types: Boolean, string, float, long and integer

Using Shared Preferences

Shared across an application's components, but are not available to other applications

- Caveat: they can be made available to multiple processes within same application, but be careful
- Stored as XML in the protected application directory on main memory
  (usually /data/data/[package name]/shared_prefs/[SP Name].xml)