MOBILE APPLICATION DEVELOPMENT

ANDROID (2017)

LECTURE 13: GLOBAL APPLICATION STATE

GLOBAL APPLICATION STATE

- Most applications of any significant size deal with large amounts of data.
- Most large applications are also comprised of multiple Activities and Fragments.
- Passing data between Activities and Fragments using intents, listener chains, and properties can get very cumbersome when dealing with large numbers of classes and large amounts of data.
- Activities and Fragments are also frequently destroyed, meaning that the mechanisms for passing data around the application need to tolerate this.
- Rather than directly pass data from point to point throughout the lifecycle of an application, it can be much more straightforward to manage some state globally.

WHEN AND HOW TO USE GLOBAL STATE

- Consider using some kind of global state when you have data that must be referenced in a number of otherwise unconnected places within your app, and/or when you have multiple parts of the application modifying the same data.
  - Do not use global data simply to break encapsulation of data within classes.
  - Do not (generally) use global data if it will only be referenced in a single place.
- If global data is desirable, implement it according to a commonly agreed-upon structure. Good candidates for global data representation (depending on the use case desired) are shared preferences, data files, databases, and singletons.

SHARED PREFERENCES

- Android represents application-specific preferences as key/value pairs which map Strings to basic data types such as Ints and other Strings.
- The SharedPreferences class serves as the interface between the programmer and the application's preferences.
  - Can use a SharedPreferences.Editor to define a series of edits to the application's preferences before calling `commit()` to write the edits to disk.
  - For simple kinds of data that are modified infrequently, SharedPreferences can be an effective way to manage global state. The overhead of using the class can, however, present a problem when dealing with large amounts of data or making frequent edits.

DATA FILES

```
<manifest ...>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
</manifest>
```

- As with most software platforms, Android allows the programmer to write arbitrary data to local storage.
- Can write pure data (bytes), text, or even objects to the filesystem.
  - Use a BufferedOutputStream or similar class to write raw bytes to a file.
  - Use a BufferedWriter wrapping a FileWriter (or similar) to write text to a file.
  - Use an ObjectOutputStream to write an instance of a Serializable class to a file.
- This is a good way to write lots of data to single files, but is somewhat slow and cumbersome to have to use over and over throughout an application.

DATABASES

- Databases (either local or remote) are a great way to store lots of data that can be structured and queried efficiently.
- While databases require lots of overhead to get set up and use, they provide a number of reliability guarantees and have extensive optimizations built in.
  - Can use specialized classes to read/write data to various database implementations.
  - Can also interact with databases via API calls to remote servers.
- Databases are difficult to use cleanly without extra code wrapping them, but provide some of the best storage and retrieval options for application data.

SINGLETONS

- For a convenient, customized interface to global data, the programmer can use `object` declarations to create singletons that can be accessed anywhere in an app.
- Singletons can represent persistent and non-persistent data with any interface the programmer chooses.
  - Can wrap any of the previous methods of storing data in a singleton for easier use, and if data persistence is not needed singletons can act as regular objects with no backing component in the filesystem.
  - Singletons are one of the 'cleanest' ways to interface with global data.
- Singletons may introduce some problems into applications, including retain cycles.