CS478: Software Development for Mobile Platforms

Project #5

Due time: 9:00 pm on 5/1/2018
Submit using Blackboard web site
Total points: 100
Instructor: Ugo Buy

TAs: Vinay Manchundiya and Tathagatha Ganguly

In this project you are to code two Android apps that help users find out how much money the federal goverment of the US had on hand on any given working day in the year 2017. The web site *treasury.io* provides this data. However, for your convenience your client, Mr. Oogawh Booey, has extracted the relevant data for the year 2017 and the first few months of 2018. The data is stored in a text file called *treasury-io.txt*. Each line in the file has data for a given working day. Each line contains the following items, separated by commas, for a given day:

1. Year (either 2017 or 2018)
2. Month (1—12)
3. Day (1—31)
4. Day of week (e.g., "Monday")
5. Amount held by US at day open in millions of US dollars
6. Amount held by US at day close in millions of US dollars

In summary, for this project you must create an SQLite database that formats the data in the text file in a single table. You must also define a bound service called *BalanceService* exposing an API for creating and querying the database.

The first app acts as a client. The main activity in the this app must define fields for creating and querying a database. Upon pressing a first button the activity will issue an API call for creating a database to the service. If the database is created successfully, a user can press a second button to view the amount of money held by the US in each day of a prespecified range. In addition the UI of your app will contain three fields for specifying a date range (e.g., the month and day for the starting day of the date range, and the total number of working days in the range). The year is assumed to be 2017, but see also the section on extra credit below. Upon pressing the second button, your main activity will check that the specified date range is valid. If successful, the activity will issue an API call for obtaining the specified data from the service. The data will be displayed in a list view contained in a second activity. The user can return to the main activity by pressing the "back" button on the device.

The second app defines a bound service called *BalanceService* that manages the database as well as database queries. The API exposed by *BalanceService* consists of the following two remote methods:

1. *createDatabase() : Boolean* — This zero-argument method reads the text file and creates the SQLite database used for subsequent querying. It returns *true* or *false* depending on whether the database was successfully created or not.

2. *dailyCash(int, int, int, int) : DailyCash[]* — This method takes as input 4 integers: a day, a month, a year, and a number of working days. The first 3 integers denote a date in years 2017 or 2018. (See extra credit below.) The last integer denotes a number of working days between 1 and 30. If successful, this method returns an array of *DailyCash* instances containing the requested data. Otherwise the method returns a length-zero array. If the database had not been created before this method is called, the method returns a zero-length array.

In addition to the bound service the second app defines a main activity whose only goal is to display text explaining the purpose of the app.

Extra Credit. If your app handles both the entire 2017 year and the additional data for 2018 contained in the text file, you will receive up to 20 extra credit points. If you go for extra credit, you must add a UI field for specifying the year of the starting day of the date range in the first app.

Implementation notes. You are provided the class definition for *DailyCash*. The class implements the Android Java interface *Parcelable*, making it suitable for interprocess communication. You must use an AIDL spec to expose the service's functionality to the clients. Make sure that the service's code is thread-safe; multiple clients could be bound to the service at the same time. As your implementation platform, use a Pixel 2 XL virtual device running the usual Android version (API 25—Nougat 7.1). Design your client app layout in such a way that it will display best in landscape mode. You are not required to provide backward compatibility with previous Android versions or to support device reconfigurations.

Hints. Download the definition of class *DailyCash* and the text file from the assignment page on Blackboard. Review the slides, recorded lectures, and programming examples on services and SQLite in order to manage the bound service as well as database creation and querying.

You must work alone on this project. Submit a zip archive containing two root directories; each directory contains the full Android Studio repository of the corresponding app. No late submissions will be accepted.