CS478: Software Development for Mobile Platforms

Project #5

Due time: 11:59 pm on 12/6/2017
Submit using Blackboard web site
Total points: 100
Instructor: Ugo Buy

TAs: Siddhesh Chavan and Kunal Shah

You are to code two Android apps. The first app, named *FedCash*, takes as input a year or a date range from an interactive user. This app uses a second app, named *TreasuryServ*, to access a database maintained by the US Department of the Treasury. The database is found at URL http://treasury.io/, which supports most networking protocols, such as JSON and HTTP Clients. In order not to block its user interface during network operations, *TreasuryServ* defines a bound service in order to access the database. The service in *TreasuryServ* defines a simple API, described below, for use by *FedCash* and other apps. Upon receiving an API call from *FedCash*, *TreasuryServ* appropriately creates and formats a query to be forwarded to the *treasury.io* site. After it receives the the site's response to the query, *TreasuryServ* will return the requested information back to the *FedCash* app as the value returned from the API call. For obvious reasons, *FedCash* must use a worker thread when making API calls on *TreasuryServ*.

The API exposed by *TreasuryServ* consists of the following three remote methods:

1. *monthlyCash(int) : List(Integer)* — This method takes as input an integer denoting a year between 2006 and 2016 inclusive. It returns a list of 12 Integer values denoting the amount of cash the US Government had on hand at the opening of the first working day of each month of the input year.

2. *dailyCash(int, int, int, int) : List(Integer)* — 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 the years 2006–2016. The last integer denotes a number of working days between 5 and 25. This method returns a list of integers denoting the cash the government had on hand at the opening of the input date and subsequent working days, as specified by the fourth input parameter.

3. *yearlyAvg(int) : int* — This method takes as input an integer. It returns as output the average of the cash on hand at the opening of each working day in the input year.

In addition to the service, *TreasuryServ* defines an activity whose sole purpose is to display the status of the service. The service could either be (1) not yet bound, (2) bound to one or more clients but idle, (3) bound and running an API method, and (4) destroyed. Notice that a destroyed service could be bound again, if a request comes from a client.

App *FedCash* has two activities. The first activity allows the interactive user to specify a request to be forwarded to *TreasuryServ*. Define appropriate fields (widgets) allowing a user to select one of the three APIs supported by the service in *TreasuryServ* and the corresponding input parameters. An additional widget forwards a request to the service. If *FedCash* was not bound to the service, it will bind to the service at this point. Finally, this activity contains a widget that will unbind *FedCash* from the service. The second activity

consists of two fragments to be displayed side-by-side. This activity is opened by selecting an appropriate button in the first activity. The left fragment contains a list of all the calls made on the service. When a user selects one of the calls, the right fragment will display the result returned by the corresponding API call on the service contained in *TreasuryServ*.

Implementation notes. 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 XL virtual device running the usual Android version (API 25—Nougat). 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. Use class *AndroidHttpClient* to forward a request to the *treasury.io* site. Method *execute()* takes as input a query as a URL string and an instance of the *ResponseProcessor* class. Finally, use class *URLEncoder* to create the query portion of the URL sent to the HTTP client.

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.