Mobile Application Project 1

Robin Karmakar API level 8 Used my original old droid

This project was a great dive into the Android world. For me, it was an unprecedented legitimate test of modularity, encapsulation, polymorphism, and other fundamental concepts in software and software engineering. At the first introduction of the project the concepts seemed simple enough. However once development started it was far more complex than I had imagined. The build began with first brainstorming a way to design a project. Java provides easier tools to open and build connections than C, but I knew the same concepts would be implemented from Network Centric Programming. It begins with obtaining a URL and opening a connection. After establishing this connection one can begin to choose the destination of the downloaded file and open file streams, and I/O streams. This was my download function and is the required component of this application.

After this, some time was devoted to learning how the Android application is read by the system. The keyword is Activity, which my main class will extend. Much of the tools Activity offers I came to know about through the lifecycle demonstration. The onCreate function initializes all the necessary fields used for this app. And the onClick function kicks off the app.

The AsynTask is probably the most important piece of this app, it contains the contents of the download function in the doBackground method. Utilizing the AsyncTask rather than a regular thread is far more convenient to the developer. As the App runs the user, is notified of all the general events occurring. Therefore for external communication to the user there must be internal communication between the UI thread and the background thread. This called for the use of a handler.

Messages sent to the handler are sent to a toast function. So App is able to part (i), it downloads the file in /Pictures/My Pictures/(nameoffile). Another thing is it can take a specified url input(part of (ii, “…meaning that the user can enter a url…” and if the user clicks the home button after the download starts it finishes the download anyhow. Clicking the back button however cancels the download. The user is notified through toast messages when wifi is available and when its not(which is the last bullet point of part iii). It implements all of part iv. This can be tested as such – once clicking download and clicking on home and then turning off wifi(doing all this really quickly) part of the file has been downloaded, and if the user switches back to the app he/she can see the percentage of how much of the complete file was downloaded.

The problem I faced with this app is being able to stop and restart the Asynctask so that reading or writing to the file would not cause a FileNotFoundException. The toast process would be “wifi lost” then “file error(due to exception in read/write)”. However after countless days and hours trying for a solution it was to no vein. I tried many implementations of wait and notify with synchronized(either of the two threads) lock of both the UI thread and AsyncTask. I also tried putting a Thread.sleep(100) before and after the outStream.write(…), in the while loop. This was the closest I could get to a solution, because it would not toast “file error” if I turned wifi off. However after turning wifi on it would toast “wifi found” and then “File error”. So somehow an exception is occurring. I also tried for checking for a network connection in my handler function and also my onStop() state. Nothing worked. I realized too late I could use DownloadManager to automate this task.