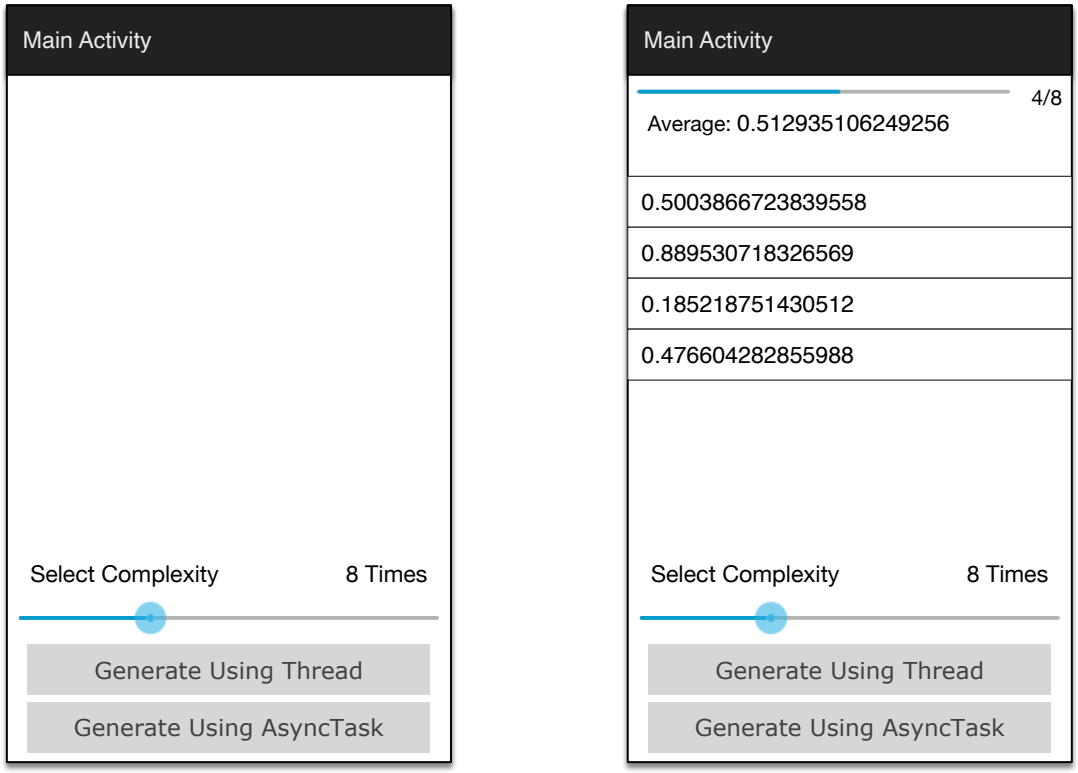
Mobile Application Development
In Class Assignment 6

Basic Instructions:

1. In every file submitted you MUST place the following comments:
   a. Assignment #.
   b. File Name.
   c. Full name of the student.
2. Each group is required to submit the assignment on Canvas.
3. Please download the support files which include a Java project to be used for this assignment.
4. **Submit Codes:**
   a. Zip all the project folder to be submitted on canvas.
5. Submission details:
   a. The file name is very important and should follow the following format:
      Group#_InClass06.zip
   b. You should submit the assignment through Canvas: Submit the zip file.
6. **Failure to follow the above instructions will result in point deductions.**

In Class 06 (100 points)

In this assignment you will get familiar with Android concurrency models. This application is composed of a single activity: Main Activity.



(a) Launching Screen (b) Showing Progress

Figure 1, App Wireframe

Part 1 (60 Points): Using AsyncTasks

The interface should be created to match the user interface (UI) presented in Figure 1. Perform the following tasks:

1. You are provided with a HeavyWork class that contains a static method getNumber(). This method takes a long time to execute and should be executed in the background. Import the provided Java file by simply dragging the file into the src folder under your project package.
2. Your task is to use an AsyncTask to execute this method in a background thread. Do not use the main thread to generate the number. The UI should only be manipulated by the main thread.
3. Complexity SeekBar
   a. Use a SeekBar to set the complexity of the heavy work. The selected **number** defines the number of times you will execute the getNumber() method in the background thread.
   b. The SeekBar maximum should be set to 20.
   c. The TextView showing the selected complexity number which is displayed to the

right of the “Select Complexity” label, this number should be updated whenever the user moves the SeekBar.

4. Tapping on the “Generate Using AsyncTask” button should start the execution of a background AsyncTask in order to compute a list of numbers by repeatedly calling the getNumber() method based on the selected complexity.
   a. The AsyncTask should:
      i. Receive the complexity number as input.
      ii. Should report progress as an integer which is the progress or count of numbers generated so far.
      iii. Should return an ArrayList of doubles to hold all the generated numbers.
   b. For example, if the complexity was set to 5, the getNumber() method should be called 5 times in the background thread, and return a list of 5 numbers.
   c. Progress updates: for each retrieved number, the progress should be reported using the progress bar at the top of the screen, the current average should be recomputed, and the ListView should include all the numbers retrieved so far including the newly retrieved number. See Figure 1(b).
   d. Make sure all the UI updates are performed on the main thread.

Part 2 (40 Points): Using Threads and Handlers

This part is similar to Part 1, but you should use threads and handlers to implement the same functionality provided by Part 1. Perform the following tasks:

1. You should create a thread pool of size 2. Use the thread pool to execute the created thread.
2. Tapping on the “Generate Using Thread” button should start the execution of a background thread and get the list of numbers (based on the selected complexity) returned by the getNumber() method, as done in Part 1.
3. To be able to exchange messages between the child thread and the main thread use the Handler class. Either use messaging or setup a runnable message.