ITIS/ITCS 4180/5180 Mobile Application Development
In Class Assignment 3

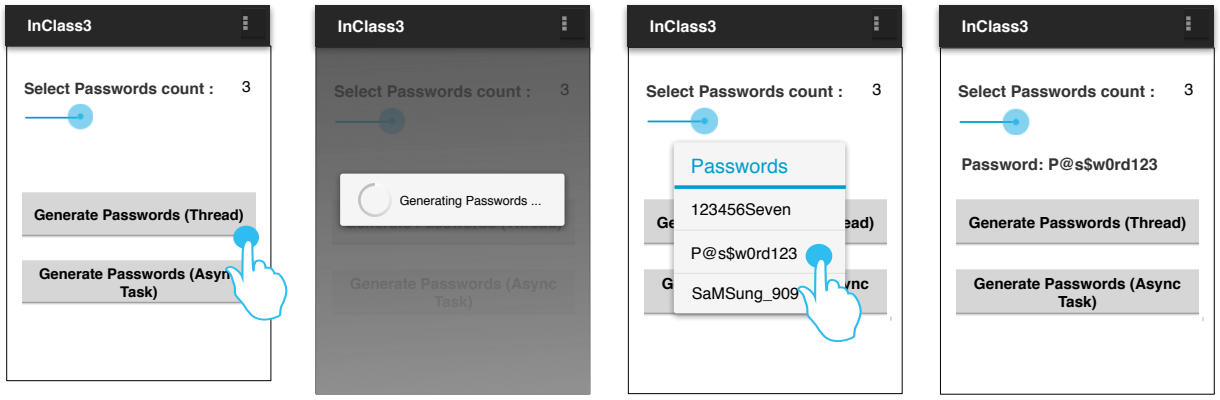
Basic Instructions:

---

1. In every file submitted you MUST place the following comments:
   a. Assignment #.
   b. File Name.
   c. Full name of all students in your group.
2. Each group should submit only one assignment. Only the group leader is supposed to submit the assignment on behalf of all the other group members.
3. Your assignment will be graded for functional requirements and efficiency of your submitted solution. You will loose points if your code is not efficient, does unnecessary processing or blocks the UI thread.
4. Please download the support files provided with this assignment and use them when implementing your project.
5. Export your Android project and create a zip file which includes all the project folder and any required libraries.
6. Submission details:
   a. Only a single group member is required to submit on moodle for each group.
   b. The file name is very important and should follow the following format: **Group#_InClass03.zip**
   c. You should submit the assignment through Moodle: Submit the zip file.
7. **Failure to follow the above instructions will result in point deductions.**

In Class Assignment 3 (100 Points)

In this assignment you will get familiar with Android concurrency models. The app is a password generator app to help the user to choose strong passwords. The User selects how many passwords they want the app to generate and then chooses one of them. This application is composed of a single activity, namely the Main Activity.



(a) Main Screen (b) Progress Dialog (c) Passwords List (d) Password selected

Figure 1, App Main Screen

Part 1 (60 Points): Using AsyncTasks

The interface should be created to match the user interface (UI) presented in Figure 1. You will be using layout files, and strings.xml to create the user interface. Perform the following tasks:

1. Create a new android project called “In Class 3”.
2. You are provided with a Util class file that contains a static method getPassword(). This method takes a long time to execute and returns a random String password. Import the provided Java file by simply dragging the file into the src folder under your project package in Android Studio.
3. Your task is to use an AsyncTask to execute this method in a background thread. Do not use the main thread to generate passwords. The UI should be manipulated by the only main thread.
4. Use a SeekBar to set the number of passwords to be generated. The SeekBar maximum should be set to 10. Also note, the TextView showing the selected count number which is displayed to the right of the “Select Passwords count” label, this number should be updated whenever the user moves the SeekBar. The selected **count** defines the number of times you will execute the getPassword() method in the background thread.
5. Tapping on the “Generate Password AsyncTask” button should start the execution of a background AsyncTask and generate all the passwords and saves them into a list. For example, if the complexity was set to 5, the getPassword() method will run five times in the background thread, and return 5 passwords. The list of these 5 passwords should be returned to the main thread and displayed in the AlertDialog.

While the passwords are being generated, display a ProgressDialog indicating the progress, see Figure1(b). The Async input parameter should be set to Integer, the progress parameter to Integer, and the result parameter to ArrayList<String>.

6. The ProgressDialog should not be cancelable. The ProgressDialog should be dismissed after all the getPassword() calls are completed (in onPostExecute). The list of passwords should be displayed using an alert dialog as shown in Figure 1(c).
7. Tapping one of the passwords generated should display the selected passwords and dismiss the Dialog, see Figure 1(d).

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 the selected size/2. Unless the size is one then the size is one. Use the thread pool to execute the created thread.
2. Tapping on the “Generate Password Thread” button should start the execution of a background thread and return the list of all passwords (based on the selected count) by using the getPassword() 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.