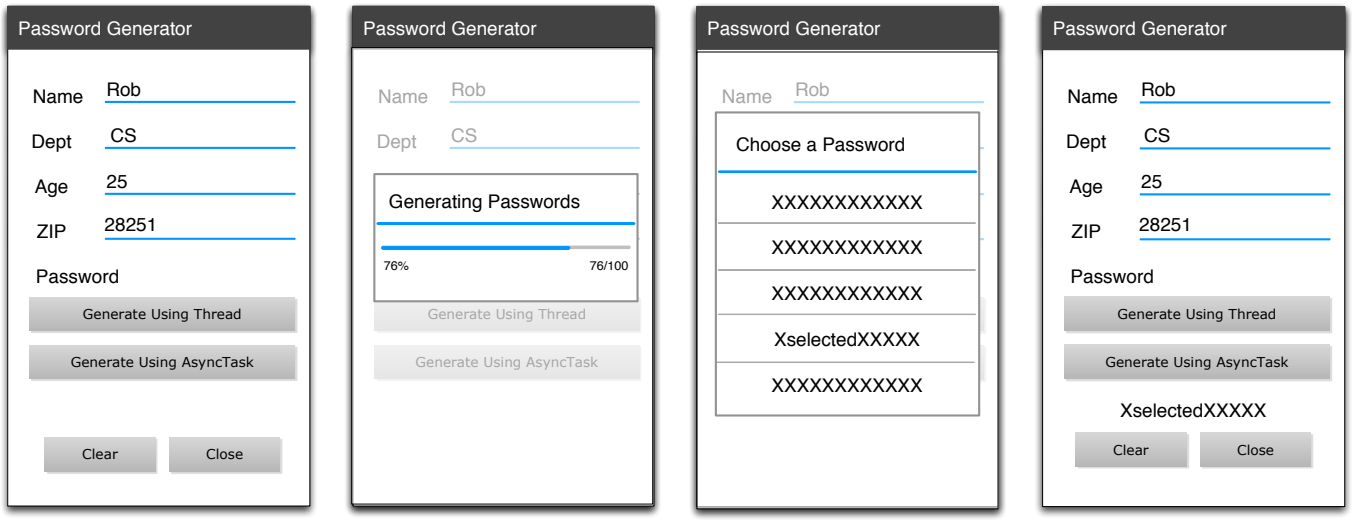
ITIS/ITCS 5180 Mobile Application Development
In Class Assignment 4

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. Export your Android project and create a zip file which includes all the project folder and any required libraries.
5. Submission details:
   a. Only a single group member is required to submit on canvas for each group.
   b. The file name is very important and should follow the following format: **Group#_InClass04.zip**
   c. 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 Assignment 4 (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 puts their Name, Dept, Age, and ZIP, and the app generates a strong password from using the inputs. There are two options for generating passwords: one using Threads, and another one is using AsyncTasks.



(a) Main Screen (b) Progress Dialog (c) Alert Dialog to choose a password (d) Selected a Password

Figure 1: Application Wireframes

Using Threads and Handlers (70 points)

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

1. You are provided with a Util class that contains a static method getPassword(String name, String dept, int age, int zip). This method is expected to take 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.
2. You should create a thread pool of size 2, and use the pool to execute your created threads.
3. You should put four InputTexts to take four inputs: Name, Dept, Age, and ZIP. All four inputs are necessary. Use appropriate input validations to take care of the followings:
    1. All inputs are mandatory.
    2. Name should not have any number.
    3. Age must be a positive integer.

    4. ZIP must be a 5 digit positive number.
4. There will be two buttons. First, a button to generate passwords using Threads. And second, a button to generate passwords using AsyncTasks.
5. Tapping the “Generate Using Thread” button should start the execution of a background thread and return the **list of five passwords** by using the getPassword() method. The getPassword() method will run 5 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 ProgressBar indicating the progress, see Figure1(b).
6. 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.
7. The ProgressBar should not be cancelable. The ProgressBar should be dismissed after all the getPassword() calls are completed. The list of passwords should be displayed using an alert dialog as shown in Figure 1(c).
8. Tapping one of the passwords generated should dismiss the Dialog and display the selected password as shown in Figure 1(d).
9. Clicking on Clear button should reset the app and take you to the empty form, see Figure 1(a).

Part 2: Using AsyncTask (30 points)

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

1. Tapping the “Generate Using AsyncTask” button should start the execution similar to Part 1 but using Async Tasks.