CMPS 312 Mobile Application Development

LAB 10 - Handlers AsyncTask and Threading

Objective

Android enforces a **worst case reaction time** of applications. If an activity does not react within 5 seconds to user input, the Android system displays an Application not responding (ANR) dialog. From this dialog the user can choose to stop the application.

In this Lab, you will learn how to use threads in Android. You will learn how not Block UI thread and how to access the Android UI toolkit from outside the UI thread. The Lab will introduce you the following Android threading concepts.

- Threads and Runnable
- Handlers: How to define them and use them to modify the UI using Messages
- AsyncTask
  - onPreExecute()
  - doInBackground(Param...)
  - publishProgress(Progress)
  - onProgressUpdate(Progress)

Overview

In the lab will be creating the following application shown in Figure 1. [**SHOW DEMO**]

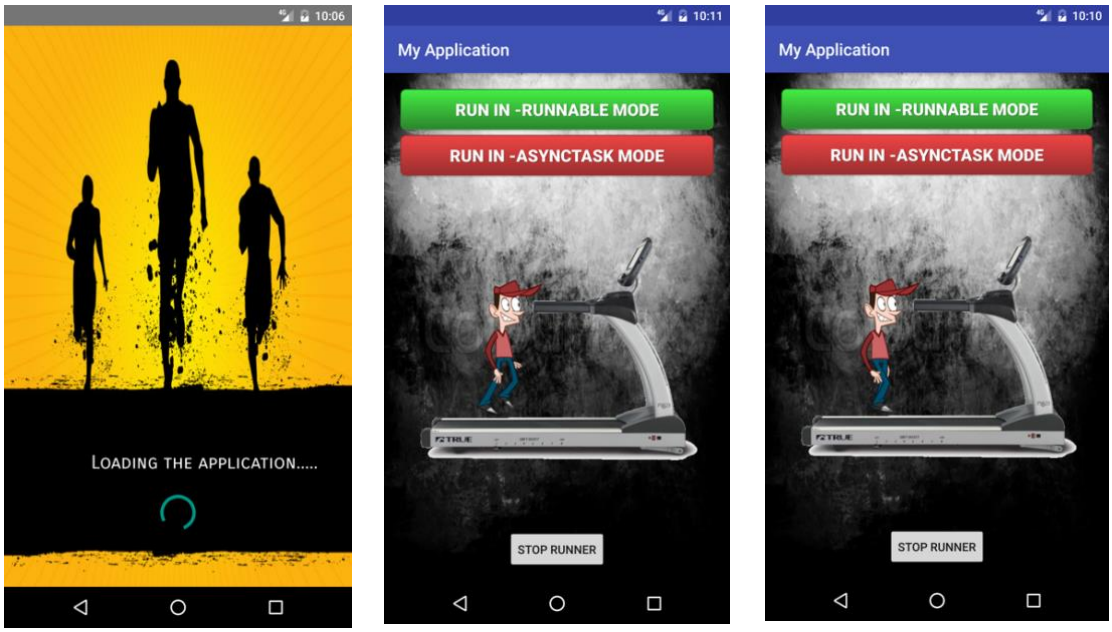

Figure 1: The Running APP screens

Application Description

You need to create an application named "**Runners**" with the following package name "**qa.edu.qu.cmps312.runners" .** The application will have two screens. The first screen called **Splash Screen** and a Second Screen called **Main Screen**. When the application starts, it should show the **Splash Screen for five seconds** and then move the user to the **Main Screen,** without them clicking anything. On the **Main Screen** the user is presented with two buttons that make the shown character on the treadmill in Figure 3 to animate(run). When the user clicks the first button, you should animate the character with the use of **Thread, Runnable, and Handler** classes. And, when the user clicks the second button you should animate the character with the use of **AsyncTask**. If the User presses on the Stop button then you should stop the running thread which in turn stops the character from running.

NOTE: Before starting, download the skeleton code of the Layouts from the BlackBoard**.**

PART A
THE SPLASH SCREEN

Instruction

1. Create an activity and call it **Splash Screen.**
2. Use the provided skeleton code for the layout of the activity.
3. The **Splash Screen** should stay on the screen for about **five Seconds** and then **navigate** the User to the **Main Screen** of the application which is shown in Figure 3.

NOTE: In the Splash Screen you are required to use **Handlers** with **PostDelayed Method**, **Thread** and **Runnable** classes to make the application transition from the splash screen to the Main screen.

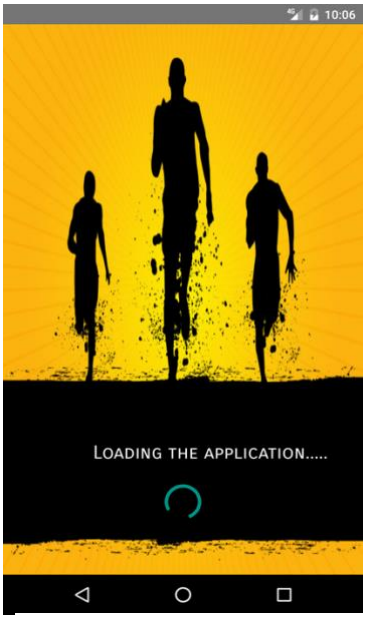

Figure 2: Splash Screen

PART B
THE MAIN SCREEN

After the Splash Screen of the application, you need to create the Main Screen of the application as shown in Figure 3. The code of the layout for the Main Screen is given however, you are required to create the controller class (Activity).

Instruction

1. Create the **Main Screen** Activity. If you already have a Main Activity just rename it to **MainScreen**.
2. Use the Skeleton code to create the Layout of the activity.
3. In the skeleton code layout, you will find three buttons named "**RUN IN RUNNABLE MODE"**, "**RUN IN ASYNC TASK MODE"** and **"STOP RUNNER"**
4. Your task is to make the first two button animate the character on treadmill by loading the different sprites of the character from the drawable's folder that is provided in the skeleton code.
5. When the user clicks on the "**RUN IN RUNNABLE MODE"** button you need to use **Thread, Runnable and Handler classes with Message** to animate the character.
6. On the other hand, when the user clicks on the "**RUN IN ASYNCTASK MODE"** you should animate the character with the use of **AsyncTask** class. You are required to override the necessary Methods of the AsyncTask class to update the UI in each loop.
7. If the user clicks on the **STOP RUNNER,** then the runner should stop running by stopping the running thread.
   Hint: Use a Boolean inside the Thread loop that checks if it should continue running.

Figure 3 : Main Screen