ITIS/ITCS 4180/5180 Mobile Application Development
In Class Assignment 4

Basic Instructions:

---

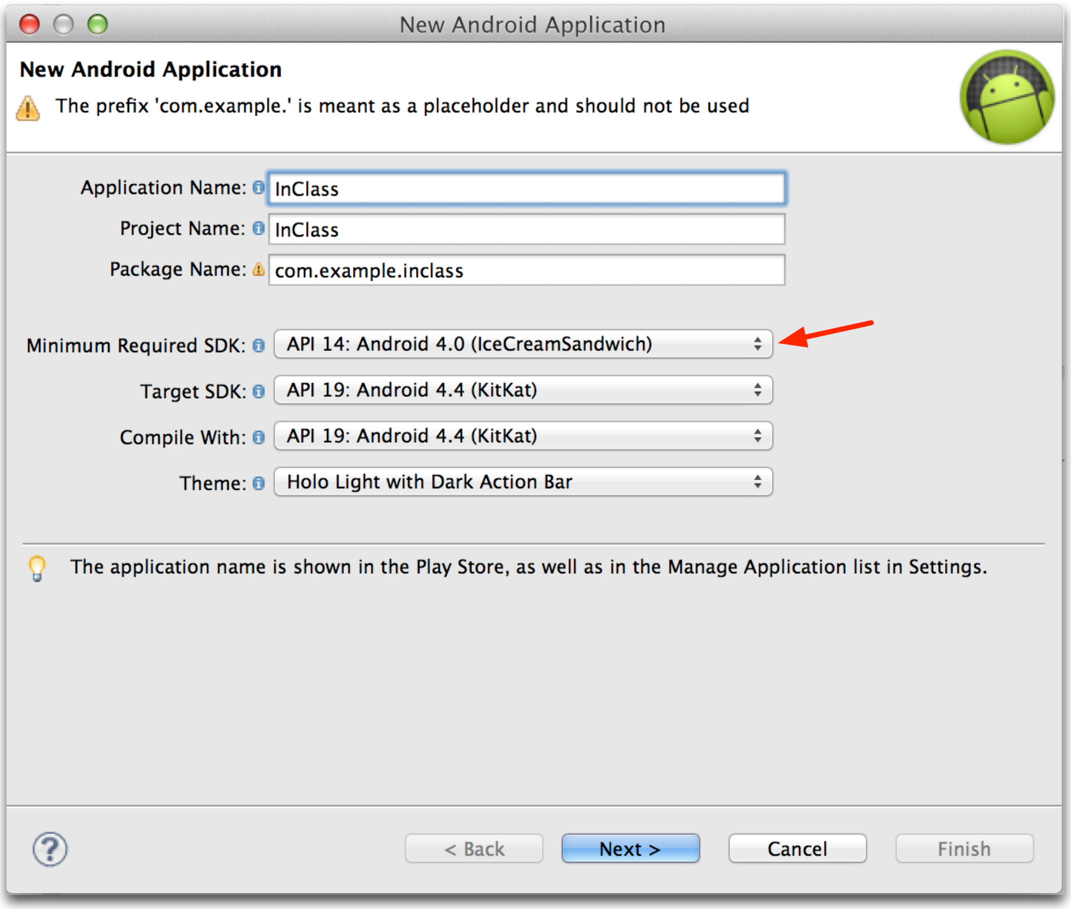
1. In every file submitted you MUST place the following comments:
   a. Assignment #.
   b. File Name.
   c. Full name of all students in your group.
2. Each student in the group is required to submit the assignment on moodle.
3. Please download the support files provided with this assignment and use them when implementing your project.
4. **Export your project as follows:**
   a. From eclipse, choose *"Export…"* from the File menu.
   b. From the Export window, choose *General* then *File System*. Click *Next*.
   c. Make sure that your project for this assignment is selected. Make sure that all of its subfolders are also selected.
   d. Choose the location you want to save the exported project directory to. For example, your *Desktop* or *Documents* folder.
   e. When exporting make sure you select *Create directory structure for files*.
   f. Click Finish, and then go to the directory you exported the project to. Make sure the exported directory contains all necessary files, such as the .java and resource files.
5. Submission details:
   a. All the group members should submit the same zip file.
   b. The file name is very important and should follow the following format: **Group#_InClass04.zip** For example, Group2A_InClass04.zip
   c. You should submit the assignment through Moodle: Submit the zip file.
6. **Failure to follow the above instructions will result in point deductions.**

<u>**In Class Assignment 4 (100 Points)**</u>

In this assignment you will get familiar with Android concurrency models.

Important App Requirements:

1. Create a new android project called “In Class 4”.
2. The required Android Virtual Device (AVD) should have **minimum SDK version set to 14 and target SDK at least 19**. The app should display correctly on Nexus 5. Your assignment will not be graded if it does not meet these requirements, and you will not be granted any points on your submission.
3. You will be using layout files, and strings.xml to create the required user interfaces. The layout XML file can be modified through the raw xml, or through the GUI tools provided within eclipse.



New Android Application

New Android Application

The prefix 'com.example.' is meant as a placeholder and should not be used

Application Name: InClass

Project Name: InClass

Package Name: com.example.inclass

Minimum Required SDK: API 14: Android 4.0 (IceCreamSandwich)

Target SDK: API 19: Android 4.4 (KitKat)

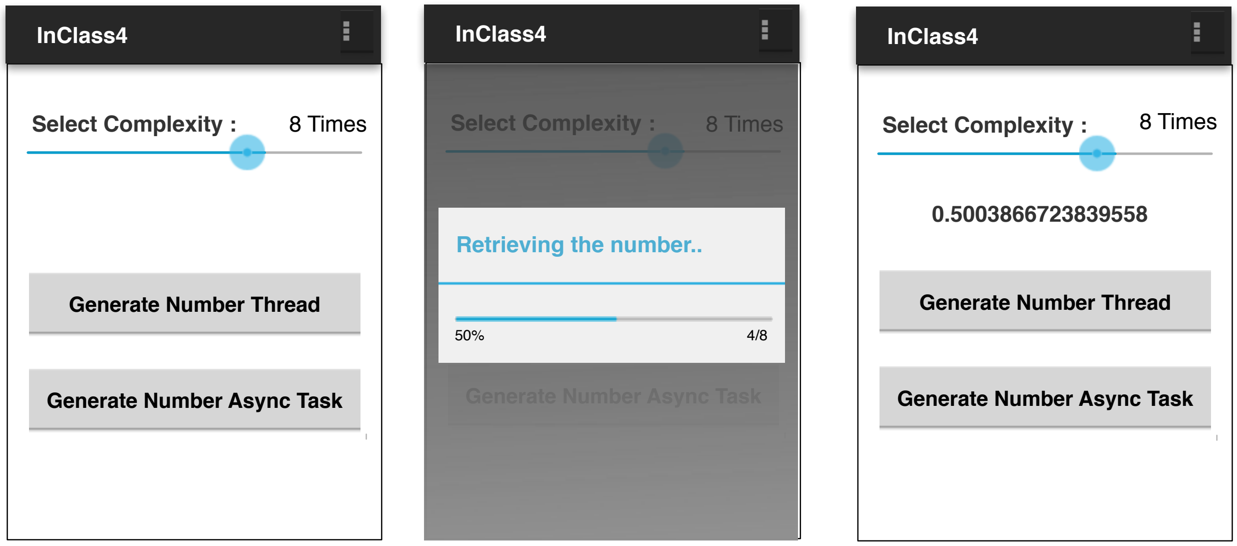
Compile With: API 19: Android 4.4 (KitKat)

Theme: Holo Light with Dark Action Bar

The application name is shown in the Play Store, as well as in the Manage Application list in Settings.

Fig 1. Choosing Minimum Required SDK to 14

This application is composed of a single activity: Main Activity.



(a) Launching Screen (b) Progress Dialog (c) Displaying Number

Figure 2, App User Interface

Part 1 (60 Points): Using AsyncTasks

The interface should be created to match the user interface (UI) presented in Figure 2. 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 4".
2. You are provided with a HeavyWork class that contains a static method getNumber(). This method takes a long time to execute. Import the provided Java file by simply dragging the file into the src folder under your project package.
3. 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 be manipulated by the only main thread.
4. Use a SeekBar to set the complexity of the heavy work. The SeekBar maximum should be set to 10. Also note, 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. The selected **number** defines the number of times you will execute the getNumber() method in the background thread.
5. Tapping on the "Generate Number AsyncTask" button should start the execution of a background AsyncTask and compute the average of all the numbers returned by the getNumber() method. For example, if the complexity was set to 5, the getNumber() method will run five times in the background thread, and return 5 numbers. The average of these 5 numbers should be computed and displayed in the TextView. While this average number is being generated, display a ProgressDialog indicating the progress, see Figure2(b). For example, if the complexity is set to 5, then the

progress should be 0%, then 20%, then 40% and so on.

6. The ProgressDialog should not be cancelable. The ProgressDialog should be dismissed automatically after all the getNumber() calls are completed and the average number generated and displayed in the TextView, see Figure 2(c).

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 Number Thread” button should start the execution of a background thread and compute the average of all the 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.