ITIS/ITCS 4180/5180 Mobile Application Development
Homework 4
Date Posted: 02/20/2013 at 02:00am
Due Date: 02/27/2013 at 11:55pm

Basic Instructions:

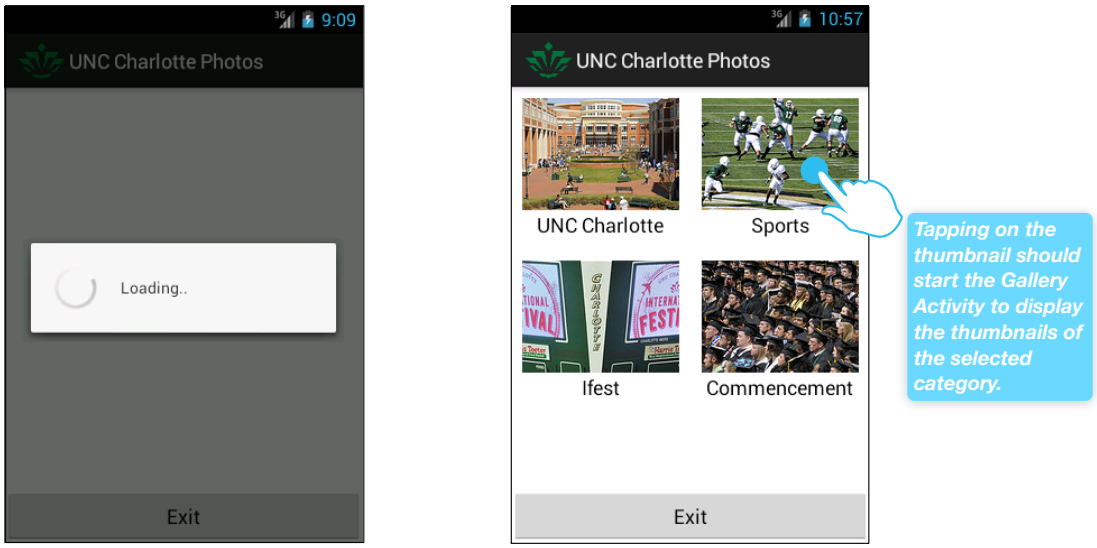
1. In every file submitted you MUST place the following comments:
   a. Assignment #.
   b. File Name.
   c. Full name of all students in your group.
2. Each group should submit only one assignment. Only the group leader is supposed to submit the assignment on behalf of all the other group members.
3. 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 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 Android 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.
6. Submission details:
   a. When you submit the assignment, compress your exported Android project into a single zip file. The format of compressed file name is HW#.zip
   b. You should submit the assignment through Moodle: Submit the zip file.
7. **Failure to follow the above instructions will result in point deductions.**

Homework 4 (100 Points)

In this assignment you will develop a simple UNC Charlotte photo gallery application for android. The app will enable the user to pick from four different photo sets, display contents of each set, and show full sized specific photos. The app is composed of 3 activities, namely **MainActivity**, **GalleryActivity**, and **ImageViewerActivity**. The provided support files includes arrays of photo and thumbnail urls for the different photo categories. In this assignment you will get familiar with multi-activity communication using Intents, and multi-threading.

Notes:

1. The recommended Android Virtual Device (AVD) should have a Target **at least** Android 4.1, API Level 16, and should display correctly on 3.2" QVGA (ADP2) (320x480: mdpi).
2. All image downloading should be performed using Threads (or AsyncTask) and your code should not block the main thread.



(a) Main Activity While Loading (b) Main Activity After Loading

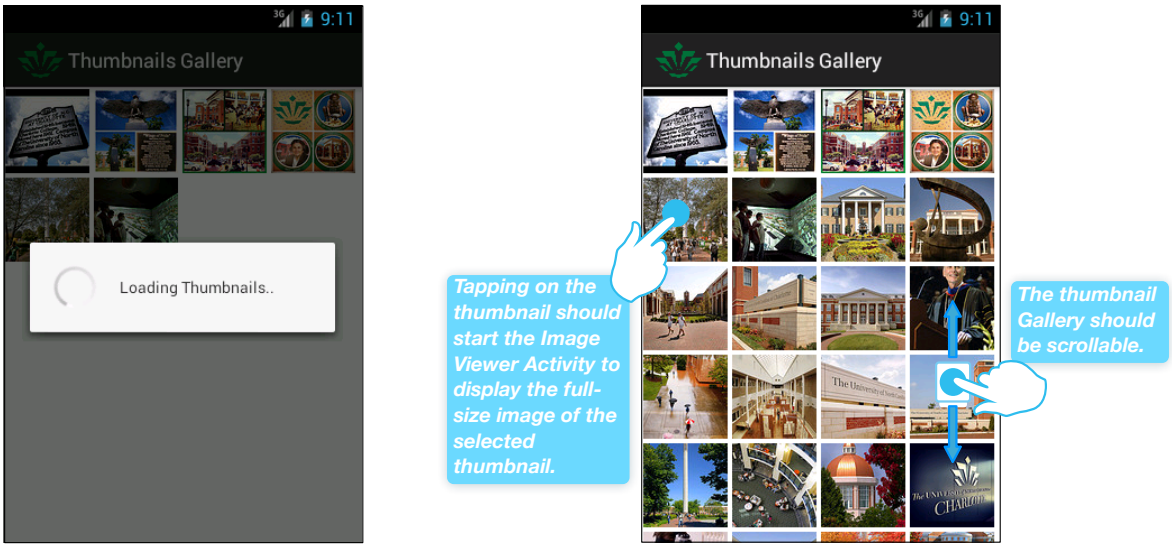
Figure 1, Main Activity Screen Shot

Part 1: MainActivity (20 Points)

The **MainActivity** is responsible for the loading the thumbnails for the different photo categories. There are four categories, namely "UNC Charlotte", "Sports", "Ifest", and "Commencement". The category label strings are in the provided `strings.xml` file. Also the category thumbnail urls are provided in the `strings.xml` file. For example, the "UNC Charlotte" thumbnail is under the `uncc_main_thumb` string in the `strings.xml` file. The interface should be created to match the user interface (UI) presented in Figure 1. The implementation requirements include:

1. Your application should have an application launcher icon, please select your launcher icon to represent your app. Set the application title to "Photo Gallery".

2. The string values used for the text labels, and button labels should be read from the `strings.xml` file and should not be hardwired in the layout file. The provided `strings.xml` file contains the required strings.
3. Use a thread pool (or AsyncTask) to download the thumbnails of the four photo categories. While the thumbnail images are being downloaded display a ProgressDialog indicating that the thumbnail images are being loaded, see Figure1(a). Do not use the main threads to download the thumbnails.
4. The ProgressDialog should not be cancelable. The ProgressDialog should be dismissed when all the main category thumbnails are downloaded and displayed.
5. Display the downloaded thumbnails and their corresponding text labels. See Figure1(b). Clicking on a category thumbnail should setup the required intent to start the **GalleryActivity** and to pass it the required gallery information. For example, clicking on the "Sports" category thumbnail should start the GalleryActivity to display all the "Sports" thumbnail images. Note: there `strings.xml` file includes arrays that contain urls of the thumbnails and photos for the different categories. For example, `uncc_thumbs` string array in the `strings.xml` file contains the urls for the "UNC Charlotte" thumbnail images. The `uncc_photos` string array in the `strings.xml` file contains the urls for the full size images for the thumbnail images in the `uncc_thumbs` array. Hint: Using the Intent Extras the GalleryActivity should be sent the IDs of the thumbnail array and photo array to be used.
6. Tapping on the "Exit" button should finish and exit the MainActivity.



(a) Loading Gallery Thumbnails (b) Loaded Gallery Thumbnails

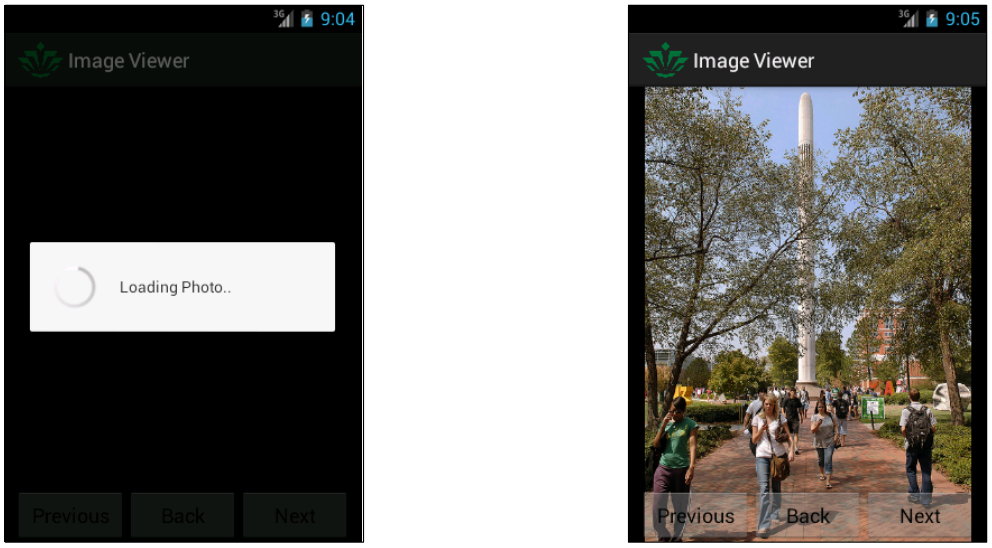
Figure 2, GalleryActivity Screen Shot

Part 2: GalleryActivity (40 Points)

The **GalleryActivity** is started by the MainActivity when the user selects one of the photo categories. The GalleryActivity should display all the thumbnail photos of the selected photo category. For example, while viewing the MainActivity user taps on the

"UNC Charlotte" photo category in the MainActivity, this should start the GalleryActivity to display all the thumbnail photos of the "UNC Charlotte" category. Note that, the thumbnails photo urls for each category are included the provided `strings.xml` file. Below are the main implementation requirements:

1. Use a thread pool (or AsyncTask) to download the thumbnail images for the selected category. While the thumbnail images are being downloaded display a ProgressDialog indicating that the thumbnail images are being loaded, see Figure2(a). Do not use the main threads to download the thumbnail images.
2. The ProgressDialog should not be cancelable. The ProgressDialog should be dismissed when all the thumbnails are downloaded and displayed.
3. You should use a GridLayout to hold the thumbnail image ImageViews. The user should be able to scroll up/down to view all displayed thumbnail images. See Figure2(b).
4. Clicking on a thumbnail image should start the ImageViewerActivity and should pass it the required image and gallery information. Setup the required onClick event handlers. For example, clicking on the Belk Tower thumbnail should start the ImageViewerActivity to display the full sized image of the Belk Tower. Note: the `strings.xml` file includes arrays that contain urls of the thumbnails and photos of the different categories. Hint: The Gallary Activity should use Intent Extras to pass the the ImageViewerActivity the ID of the photo array, and the array index in the photos array of selected photo that should be displayed.
5. Tapping on the device's back button should finish the GalleryActivity.



(a) Loading Full Size Image (b) Full Size Image Loaded

Figure 3, ImageViewerActivity Screen Shot

Part 3: ImageViewerActivity (40 Points)

The **ImageViewerActivity** is started by the GalleryActivity when the user taps on one of the thumbnail images. The ImageViewerActivity should display the full size photo of the

selected thumbnail. For example, clicking on the Belk Tower thumbnail should start the ImageViewerActivity to display the full sized image of the Belk Tower. Note that, the thumbnail and photo urls for each category are in the provided `strings.xml` file. The implementation requirements include:

1. Use a thread pool (or AsyncTask) to download the full sized photo. While the photo is being downloaded display a ProgressDialog indicating that the photo is being loaded, see Figure 3(a). Do not use the main thread to download any photos.
2. The ProgressDialog should not be cancelable. The ProgressDialog should be dismissed after the photo is downloaded and displayed.
3. You should use a FrameLayout to hold the full sized photo ImageView and the three buttons at the bottom of the layout, see Figure 3(b). Hint: For the full sized photo ImageView, make sure to set the `android:scaleType` attribute to "centerInside".
4. Tapping on the "Next" button should display the next full sized photo in this category. For example, if the currently displayed image has an index value of 5 in photo url array, then tapping on "Next" button should display the image with the index value 6. If the index value of the currently displayed image is N-1, then tapping the "Next" button should rollover and display image 0, where N is the size of the array holding the image urls for the selected category.
5. Tapping on the "Previous" button should display the previous full sized photo in this category. For example, if the currently displayed image has an index value of 5, then tapping on "Previous" button should display the image with the index value 4. If the index value of the currently displayed image is 0, then tapping the "Previous" button should rollover and display image N-1, where N is the size of the array holding the image urls for the selected category.
6. For every full sized image being displayed a Thread pool (or AsyncTask) should be used to download the full sized image. Also the ProgressDialog should be displayed to indicate that the photo is being loaded.
7. Tapping on the "Back" button should finish and destroy the ImageViewerActivity, and should go back to the GalleryActivity.

Part 4: Bonus (20 Points)

Use the finger swipe direction on the screen to decide which full size image to be displayed next. If the user swipes from left to right, this should be equivalent to tapping on the "Next" button. Similarly, if the user swipes from right to left, this should be equivalent to the tapping on the "Previous" button.