ITIS/ITCS 4180/5180 Mobile Application Development
Homework 5
Date Posted: 03/01/2012 at 11:00am
Due Date: 03/11/2012 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 5 (200 Points)

In this assignment you will develop a Trivia quiz. Trivia questions will be downloaded as an XML file from a remote server. You will get familiar with parsing XML and generating the proper user interface for showing each parsed question. This project is composed of three Activities, which include: **Main**, **Trivia**, and **Stats** Activities.

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.

Part A: Main Activity (80 Points)

The **Main** activity is responsible for the loading of the trivia contents (questions and answers) included in the XML web service located at the below address: http://liisp.uncc.edu/~mshehab/api/trivia.xml. The content of the XML file should be retrieved by establishing a HTTP connection to the service. The trivia.xml file contains a **questions** element as the root element, which contains a number of **question** elements. An example question element is as follows:

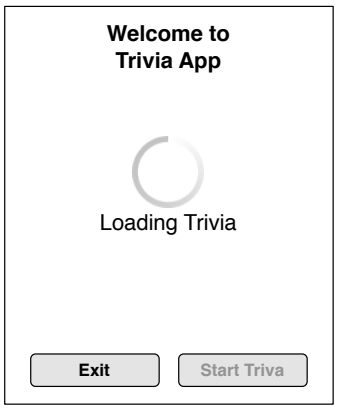
```
<questions>
    <question id="1">
        <text>Who is the first President of the United States of America?</text>
        <image>http://liisp.uncc.edu/~mshehab/api/figures/georgewashington.png</image>
        <choices>
            <choice answer="true">George Washington</choice>
            <choice>Thomas Jefferson</choice>
            <choice>James Monroe</choice>
            <choice>John Adams</choice>
        </choices>
    </question>
........
</questions>
```

A **question** element will contain a **text** element, and a **choices** element. The **choices** element has a number of **choice** elements (number of choices is variable). The choice the holds the correct answer will contain an **answer** attribute set to true. For example, in the above question the answer is the choice containing "George Washington". A **question** element can also have a **image** element that should be displayed to a user when viewing the question. A **image** element will hold a URL to an image on a remote server.

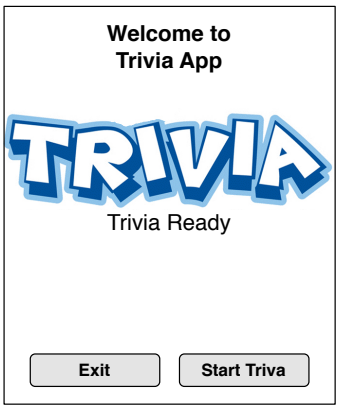
To parse the XML, you should use one of the methods described in class (SAX, or Pull). All the parsing and HTTP connections should be performed by a worker thread or an AsyncTask and should not be performed by the main thread. When a question is parsed from the XML, it should be mapped to a **Question** object. Your **Question** class must

implement the Serializable interface (or you can use Parcelable interface). If you decide to create an **Answer** class make sure it also implements Serializable. You should keep track of all questions in a List, as you will use them to show the questions to the user later on.

The below figure shows the wireframe of the **Main** activity. Note that when the activity is created it should retrieve and parse the trivia questions. The activity progress should be presented as shown in Figure 1(a). Note that the "Start Trivia" button is disabled while the loading and parsing are being performed. Figure 1(b) shows the activity after the loading and parsing are completed. Note that the "Start Trivia" button is enabled. The parsing should generate a list of questions. Clicking the "Start Trivia" button should start the **Trivia** activity. Clicking the "Exit" button should exit the application.



1(a) While Loading and Parsing



1(b) After Loading and Parsing

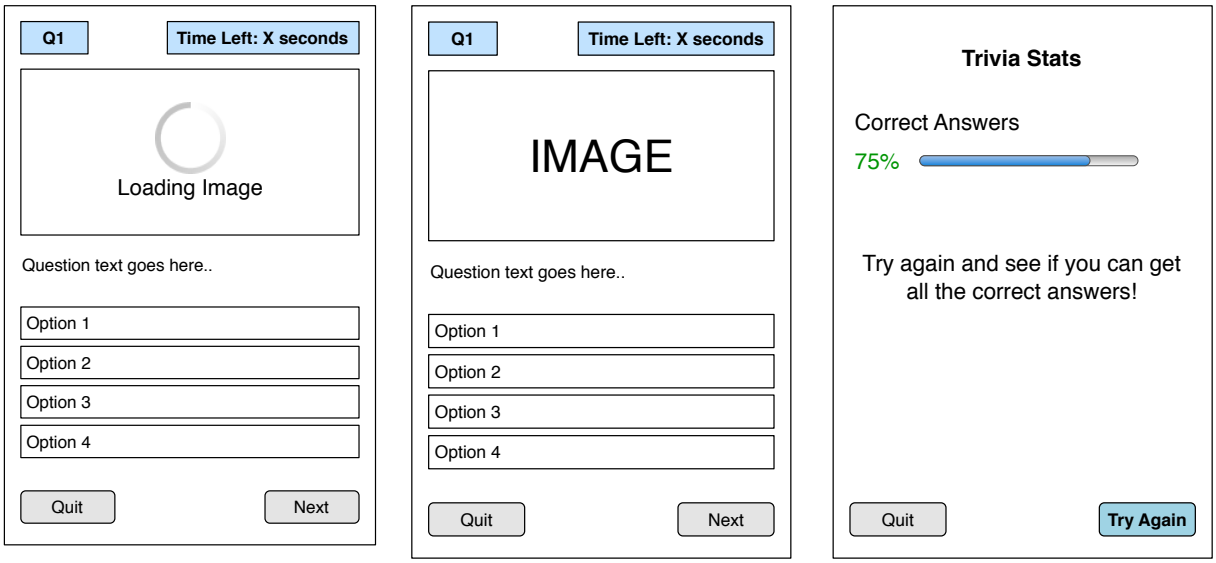
Part B: Trivia Activity (100 Points)

When the **Trivia** activity is instantiated by the **Main** activity it will receive a list of the trivia questions. Figures 2(a) and 2(b), show the wireframe of the Trivia activity. The activity shows the question number, a countdown timer, a question text, and the set of answer options. The Trivia activity shows an image if one exists for the current question. If the current question has an image, then the image should be downloaded from the specified image url indicated in the question using a separate thread (or AsyncTask) and not using the main thread. Your activity should ensure that the downloaded image is displayed only when it's question is the currently displayed question and not when other questions are displayed. While the question image is still loading you should display an activity progress indicating the image is loading, as indicated in Figure 2(a).

The user should have at most 4 minutes to finish all the trivia questions. The countdown timer monitors the start and end of the 4 minutes. Users can answer the question or skip to the next one. A skipped question should be counted as an incorrect answer. If the user clicks the "Quit" button the activity is finished and the user is sent back to the **Main** activity.

When the user answers a question, you should detect whether the selected answer was correct or not, and keep track the number of correctly answered questions. Then you should update the **Trivia** activity to display the next question. Do not use a separate activity for each question, but instead update the layout of the **Trivia** activity to show the new question. Note that the number of choices for each question varies, so the views representing the choices should be dynamically generated in your code and should not be statically created in the layout xml.

The countdown timer should start once the user starts the first question. If the countdown timer reaches 0 before user answers all the questions, it should be assumed that the remaining questions were answered incorrectly, then the user should be sent directly to the "Stats" activity. (Check the Android's **CountDownTimer** class). If the user manages to answer all the questions within the allotted time (4 minutes), then upon clicking next the user should be sent to **Stats** activity.



2(a) Trivia Activity (Image Loading)

2(b) Trivia Activity (Loaded Image)

2(c) Stats Activity

Part C: Stats Activity (20 Points)

Figure 2(c) shows the wireframe for the Stats activity. This activity shows the user the percentage correctly answered trivia questions. Clicking the "Quit" button should send the user to the **Main** activity. Clicking the "Try Again" button should send the user back to the **Trivia** activity and should redisplay the first trivia question to enable the user to retry the trivia from the first question.