ITIS/ITCS 4180/5180 Mobile Application Development
Homework 4
Date Posted: 02/20/2012 at 3:00am
Due Date: 02/27/2012 at 5:00pm

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. Please download the support files provided with this assignment and use them when implementing your project.
4. **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) The first time you export, 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) When you submit the assignment, compress your exported Android project into a single zip file. The format of compressed file name is:
        i. HW#.zip
    b) You should submit the assignment through Moodle:
        i. Submit the zip file.
6. **Failure to follow the above instructions will result in point deductions.**

Homework 4 (100 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.

Part A: Loading and Parsing Trivia Questions (50 Points)

The first task you will perform is to load an XML file named **trivia.xml** which is located at http://cci-webdev.uncc.edu/~mshehab/api/trivia.xml. The root element is a **questions** element, and contains a number of **question** elements. An example question element is as follows:

```
<question id="11">
  <questiontext>The above flag is for which country?</questiontext>
  <questionimage>http://cci-webdev.uncc.edu/~mshehab/api/figures/argentina.png</questionimage>
  <choices>
    <choice>Chili</choice>
    <choice>Mexico</choice>
    <choice>Argentina</choice>
    <choice>Spain</choice>
  </choices>
  <answer>2</answer>
</question>
```

That is, a **question** will have a **questiontext**, a number of **choice** elements (number is variable), and a provided correct **answer**. The correct answer indicates the index of the correct answer and starts from 0. A **question** element can also have a **questionimage** element that should be shown to a user when viewing the question. A **questionimage** 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 (DOM, SAX, or Pull). When a question is parsed from the XML, it should be mapped to a **Question** object. Your **Question** class must extend the provided abstract class **IQuestion** (In support files). You'll want to keep track of all questions in a List, as you will use them to show the questions to the user later on.

Once you finish parsing and loading all questions into a List, you should show a **Welcome** activity to the user. This activity should have a "Start Trivia" button that begins the series of trivia questions. It should also inform the user that he has 4 minutes to finish all trivia questions. You will have a countdown timer in the Questions activity that monitors the start and end of the 4 minutes.

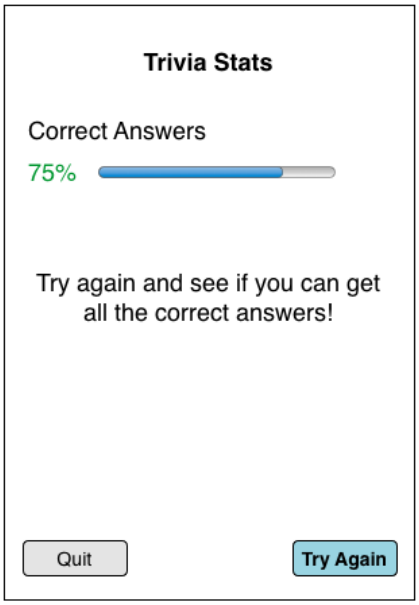
Part B: Questions Activity (40 Points)

When a user starts the trivia he is shown a new **Questions** activity. This activity should be able to show a question to the user, whether a question contains a **questionimage** element or not. That is, it should detect the existence of an image within a question and show it appropriately (use the **hasImage** method of your **Question** class).

As seen in the figure, the questions activity shows a question number, a countdown timer, a question text, and the set of answer options. It also shows an image if one exists for the current question. Users can answer the question or skip to the next one. A skipped question should be counted as incorrect. The user can also choose to quit the trivia and go back to the "Welcome" 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 **Questions** activity to display the next question. Do not use a separate activity for each question, but instead update the layout of the **Questions** activity to show the new question.

Notice that the countdown timer will 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, then upon clicking next the user should be sent to Stats activity.



Part C: Stats and Evaluation (10 Points)

Stats Activity: This activity shows the user the percentage of correct answers they have got. The stats shown will reflect whatever progress the user has made, whether they finished all questions or not. It also allows them to try the trivia again or go back to the "Welcome" activity.