ITIS/ITCS 4180/5180 Mobile Application Development
Homework 2
Date Posted: 06/03/2014 at 14:45
Due Date: 06/08/2014 at 23:55

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 2 (100 Points)

In this assignment you will develop a Tic-Tac-Toe game for Android. You will get familiar with event listeners and how to handle Android Intents. For this project you will have 3 activities, a Game activity, a ScoreBoard activity, and a GameCredits activity. You will design the game for 2-player mode, i.e. 2 human players play against each other, rather than one playing against the computer.

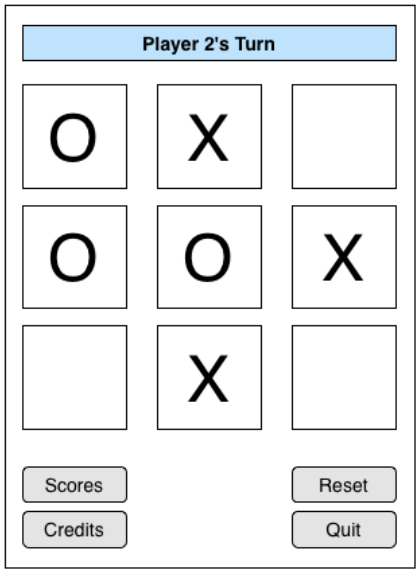
Notes:

1. The recommended Android Virtual Device (AVD) should have minimum SDK version set to **14** and target SDK at least 17. The app should display correctly on 3.2" QVGA (ADP2) (320x480: mdpi).

Part A: Game Activity (65 Points)

The **Game** activity contains the main elements of the Tic-Tac-Toe game. The Figure shows a wireframe of the game's interface which includes:

1. A *Player Turn* indicator, which shows the player who currently has the turn to play (Player 1 or Player 2).
2. There are nine slots as indicated in the figure. The **X** and **O** figures are provided (use the provided **letterx.png** and **lettero.png**) and should be represented as instances of **ImageView** with appropriate event listeners. At the start of the game, these slots should be set to empty (Use the **empty.png** from provided support files). When a player clicks a slot, the image is changed to either an **X** or **O** based on the player. Assume that Player 1 is **X** and Player 2 is **O**.



3. The players take turns until one of the players succeeds in occupying 3 slots horizontally, vertically or diagonally. Each time a player makes a move, your code should verify if there is a winner. A simple way to do so is to assign a **Tag** to an **ImageView** when it is clicked. This tag will represent the Player who clicked it (player1 or player2). If there exists three equal Tags in a row (*vertically, horizontally, or diagonally*) then the player for that **Tag** is declared a winner.
4. When a winner is detected or when there is a draw, you should display a message congratulating the winning player via **ScoreBoard** activity. In case it is a draw, indicate there is no winner.
5. The *Quit* button should quit the game and exits the application.
6. The *Reset* button should reset the game display to its initial state. The game score statistics should not be reset.
7. The *Scores* button should show the **ScoreBoard** activity. To do so, you will use

Android's Intents (Remember to register the **ScoreBoard** activity in the **AndroidManifest.xml** file). You can use an explicit intent to show the **ScoreBoard** activity.

8. The *Credits* button should show a **Credits** activity. You should use an Implicit Intent to show the **Credits** activity. The **Credits** activity will simply show the names of your project group members. It also has a "Back to Game" button to return to the **Game** activity.

Part B: ScoreBoard Activity (30 Points)

The **ScoreBoard** activity shows the overall scores for players 1 and 2.

1. The ScoreBoard activity is displayed if a user clicks the *Scores* button, or when a winner is detected, or when draw is detected. A wireframe of the **ScoreBoard** interface, the activity should indicate the total number of games played so far, the wining stats of each player and a congratulation or draw message.
2. The overall player scores can either be managed by the **Game** activity or by the **ScoreBoard** activity. If you choose to manage them in the **Game** activity, then you will always have to pass the scores to the **ScoreBoard** via the **putExtra()** method. In addition the **Game** activity should provide the **ScoreBoard** activity with the required information to display the correct congratulations message or draw message. If you choose to manage them within the **ScoreBoard** activity, then you should save the scores whenever the **ScoreBoard** activity is dismissed. To achieve this, you can investigate using the **SharedPreferences** class (**See Hints**). You should override the **onPause** and **onResume** Activity methods within the **ScoreBoard** class to save (when pausing the activity) and retrieve (when resuming the activity) the score values.
3. When a winner is detected, the overall scores need to be updated (In the **Game** activity or the **ScoreBoard** activity, depending on your choice). You should then show the **ScoreBoard** activity and send it information indicating who is the winner and the current updated overall scores (if the scores are managed by the **Game** activity). If the scores are managed by the **ScoreBoard**, then simply pass the winner. The **ScoreBoard** should then update the overall scores and update the scores in the **SharedPreferences**.
4. If the **ScoreBoard** is started via the "Scores" button in the **Game** activity, it simply shows the current overall score stats for Players 1 and 2. In this case the **ScoreBoard activity** should not display a congratulation message.
5. The **ScoreBoard** activity should also have a button with a title "Back To Game" which finishes the **ScoreBoard** activity and takes you back to the **Game** activity.

Part C: GameCredits Activity (5 Points)

The **GameCredits** activity shows the names of the project group members. It also has a "Back to Game" button to return to the **Game** activity. The **GameCredits** activity should be started using an implicit Intent object.

Hints

ImageView Dimensions:

The provided **dimens.xml** file is a resource file that we will use to contain all component dimensions. For example, in this project we use it to store the **ImageView** dimensions. In other projects, you could store a button's width and height. The file should be in the project's **res/values** folder.

Storing values into Preferences:

```
SharedPreferences preferences = getPreferences(MODE_PRIVATE);
SharedPreferences.Editor editor = preferences.edit();
editor.putInt("some_score", 2);
editor.commit();
```

Retrieving values from Preferences:

```
SharedPreferences preferences = getPreferences(MODE_PRIVATE);
int someScore = preferences.getInt(("some_score", 0); //0 is our default value
```