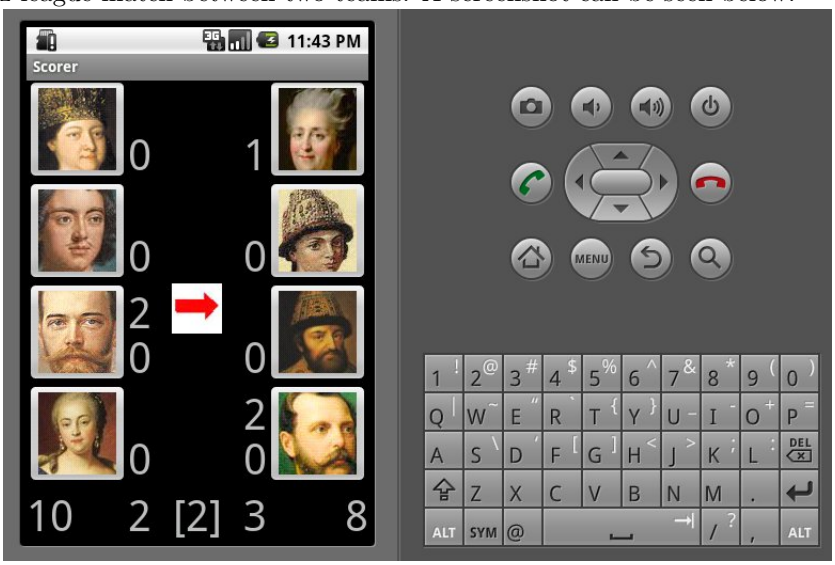
U08928 (Mobile Software)
Coursework for 2011

March 3, 2011

The task

You are to implement an app designed to assist in the score-keeping of a pub quiz league match between two teams. A screenshot can be seen below.



Rules of the Quiz

There are four players in each team, and they sit facing each other, as indicated above. The question master (who is also the user of the app) sits in a position corresponding to the bottom of the screen. There are eight rounds. In each round, each of the players is asked a question in turn. They may answer it themselves for two points, or if they are not sure, they can ask one of their team mates to answer it for one point. If they take too long to answer, as timed with the question master's watch, they are penalised by getting only one point

instead of two. If either they or their team mate gives the wrong answer, a player on the opposing team may answer for one point.

The asking of questions in turn is done alternately, starting with the two players nearest to the question master for each round. By agreement, one team (Team A) goes first in Rounds 1 to 4 and the other team (Team B) goes second in those rounds. Then Team B goes first in Rounds 5 to 8 and Team A goes second in those rounds. The team score for each round is the sum of the individual scores. The team score for each match is the sum of the scores for each round. The team with the greater score when all eight rounds have been played is declared the winner. If both teams have the same score, the match is declared a draw.

Explanation of screenshot

The scores shown are a snapshot of one round. Each player has two individual scores shown. The upper score is either 2 or blank, according to whether the player got his/her own question right for two points. The lower score is the number of questions (s)he got right for one point. These are both modified by clicking the player images, as explained below. The arrow indicates whose question it is. At the bottom, the round number is shown in square brackets. To the left and right of that, the team scores for that round are shown. On the left and right edges of the screen, the team scores for the whole match so far are shown.

Requirements

The quiz

- implement the rules of the quiz as given in the explanation above

The players

- each player should already exist in the question master's contacts list and their names and photos imported from there

Response to button clicks

- when the user presses the image of a player whose turn it is to answer a question, as indicated by the arrow, a dialog will pop up, displaying the name of the player, the player's team, and the question for that player. The question master will then be able to award him two points, one points or no points according to whether (s)he answers the question correctly within ten seconds, answers the question correctly within more than ten seconds, or fails to answer it at all.

- when the user presses the image of a player whose turn it is **not** to answer a question, that player will be awarded one point for answering correctly a question that has been passed to him, either by a team mate or an opposing player.
- when somebody has been awarded a point for answering a question, the arrow moves on to the next player. If the player concerned was the last player, the arrow will begin again from the bottom of the screen, and the next round will start. If there is no next round, the arrow will not move and a toast will be displayed announcing the winner.

Menu options

Menu options should be implemented for the main activity that

- allow the game to be restarted at any point. This can be used to start a new game once the current game has ended.
- starts a new activity in which the user can enter the name of each team, select the players for that team from the contacts lists, and indicate which side should go first. This can only be called at the start of the match.
- starts a new activity that shows the individual player scores for the match as a whole and the scores by round, including the total score so far. This can be called at any point in the match: at the start, in the middle or at the end of the match.

Other requirements

- at the end of a round in which the four members of a team have got all of their own questions right, a toast should be displayed proclaiming that the team have scored a "team full house" on their questions. Two such toasts should be displayed if both teams achieve this.
- similarly, at the end of a match in which one player has got all eight of his own questions right, a toast should be displayed proclaiming that he has scored an "individual full house", and up to eight such toasts can be displayed.
- the questions for the match should be stored in a database and made available through a Content Provider

Marking

To obtain a pass mark of 40%, you should:

- implement the basic functionality of the quiz

- have the questions hard-coded into the app
- have the choice of who goes first, the team names, and the player names are hard-coded as string resources so there is no need for the activity that sets them
- similarly, the photos can also be hardcoded in the app
- have a reasonable level of design, testing and documentation

To obtain an A grade, you should:

- meet the requirements for a pass
- implement all the requirements of the application, in particular the requirements for a Content Provider for the questions and selecting the players from the contacts list
- design and implement the app bearing in mind the Android goals of Performace, Responsiveness and Seamlessness
- have a high level of design, implementation, testing and documentation

Guidance for design

Design should be good not only in the object-oriented sense but in the procedural sense too. For example, one likely practice that will contribute to a low quality mark is writing eight copies of the same code, one for each player. Documentation of design can be given either as Javadoc or as a UML class diagram. Either way, the correctness with which you have applied the documentation technique in question will be assessed. If you use UML, all methods should nevertheless start with a comment explaining what they do rather than how they do it. All variables and methods should be wisely named following the standard Java naming conventions.

Guidance for testing

Quality of testing will be marked according to how well you have partitioned the task of testing into test cases. After that we will check that you have a test or tests corresponding to the test case you identified.

Further Advice

- Doing the practical for Week 4 will help you with RelativeLayout and it will also help you practice writing appropriate code for two-dimensional arrays of buttons. This is a good starting point. Other practicals will help you with other aspects of the GUI components, with intents and with Content Providers.

- See `HelloListView.java` in the week 3 lecture for an example of how to create a toast. On some emulators, it appears to be necessary to call the method show on the Toast object thus created.
- If your app crashes when you run it, this will be most likely be the result of a `NullPointerException`, an `ArrayIndexOutOfBoundsException` or a failure to obtain the right permission for a Content Provider. If you check your LogCat window, you will probably find the log is explicit about what has occurred and it will give you the sort of stack trace you are used to. The Week 5 practical was based on this.
- You might find it useful to write a method that given a button that has just been clicked, works out which player it is for.
- The parameter to the method `onCreateDialog` can be used to pass information about which player the dialog is for.
- You can build a list of contacts of your own for testing by going to the Contacts and adding them one by one. Pictures can be added to by using the web browser to google for images and long clicking on these to save them.
- You are free to use any image or images you want for the arrow.
- Sample quiz questions can be found at the URL:
  `http://www.sfquiz.org.uk/quizpage.html`

Submission details including deadline

You will need to submit your work by email to `ibayley@brookes.ac.uk` and `frmitchell@brookes.ac.uk` and into the Turing Building boxes by 5pm on April 29th.

We will arrange a viva in the following week, week 12, for you to discuss your solution.