MultiQuiz App (Part 2 of 3)

In this second part of our MultiQuiz app, we will update our app to handle multiple questions, and properly support device rotation. We will also alter the way that hints work, and change the Reset button to a Submit button.

Getting Started

First, make a backup copy of your completed Project 1A, in case there are any questions about your submission (or if you find you want to restart Project 1B from the beginning). The best way to make a backup is to select File | Export | Export to Zip File and select a safe name/location for the backup file (but please don't place the archive within your project directory). The archive will include all your code, assets, resources, and other related source materials, yet exclude any libraries and other artifacts that can be generated or built directly from the sources. To review the state of the system at that time and/or start again, just unzip the zip file into any convenient folder, then use File | Open and navigate to that folder.

Next, you must get some new updates from the remote Git repository that you configured when starting Project 1A. To do this, select Git | Update Project and choose the option to Merge incoming changes.

**Update**: After the merge, please access the Main Menu (four horizontal lines) then select File | Sync Project with Gradle Files (or just click the little elephant-with-arrow icon near the upper-right) as the current version of Android Studio may not do the sync for you automatically.

These updates provide the following changes specific to Project 1B:

* **build.gradle.kts** (Module :app)
  + This configuration file has been updated according to BNRG Listing 4.1 to support view models
* **Question.kt**
  + A new data class, as part of the Model, where each instance holds a single question and a list of its possible Answer objects
  + Please note that you must incorporate this class as-is within your solution

Please use the above files as provided, without making any changes to them.

Project Structure and BNRG Notes

For this part of the project, all mutable state information must be stored in a ViewModel, as described in BNRG Chapter 4, in order to retain state across device rotation. Note that the use of techniques from earlier textbook editions will result in a major score deduction, so be sure to confirm that you're using the **5th Edition**! Please also note that "Saving Data Across Process Death" is neither required nor expected for this assignment; you're encouraged to read this section, but please don't implement any of this for Project 1B.

MultiQuiz Questions and Answers

You must create four **original** multiple-choice questions, on any topics of your choice, with four **original** answer options each. Please don't include any questions from the our course materials, and don't use questions from any lists you've found online. Each question should be reasonably short (generally one or two lines) and each answer option should be much shorter (generally one or two words).

To assist with eventual testing, please arrange your answer options such that the first answer option is correct for the first question, the second answer option is correct for the second question, the third answer option is correct for the third question, and the fourth answer option is correct for the fourth question.

Layout

The overall layout should be the same as the layout of Project 1A. When the application is initially launched, the first question should appear, along with its associated answer options. The lifeline (50:50) button should be changed to a Hint button, and the Reset button should be changed to a Submit button.

The IDs for your buttons must be as follows:

* answer\_0\_button
* answer\_1\_button
* answer\_2\_button
* answer\_3\_button
* hint\_button
* submit\_button

If your answer buttons aren't named as above, this is an excellent time to change them. These are necessary for testing in the upcoming project.

Behavior

The answer buttons behave exactly as they did in Project 1A for each question.

The new Hint button behavior is as follows:

* The Hint button begins enabled
* When the Hint button is clicked, a single answer button is randomly chosen among the currently enabled incorrect answers
  + The chosen button becomes disabled and deselected
    - If there are no remaining enabled incorrect answers, the Hint button itself becomes disabled

The new Submit button behavior is as follows:

* The Submit button is only enabled when an answer button is currently selected
* When the Submit button is clicked, the next question and its associated answer buttons are displayed
  + The enabled/selected state of all other buttons should be as they were when that question was most recently submitted
    - If that question has never been submitted, the defaults should be used
  + Upon submitting the final question, the first question and its associate answer buttons should be displayed again

A short video demonstrating the expected behavior has been posted to Piazza.

All of the user-facing strings, including each question text and all button labels, must be stored only in the strings.xml file.

Functional Programming

While developing the app, be sure to use Kotlin's functional programming features, rather than an imperative programming style. In this part of the project, the random and let functions can be useful, in addition to the functions from Project 1A.