Engineering Design for Embedded Systems: Assignment 8 (version 2, typos)

Due Date: March 11, 2013

This assignment is an Android programming assignment, integrating skills that you have seen on previous assignments. You will create an app to display flashcards, and a test suite for the app. It is similar to Assignment 4, but more sophisticated.

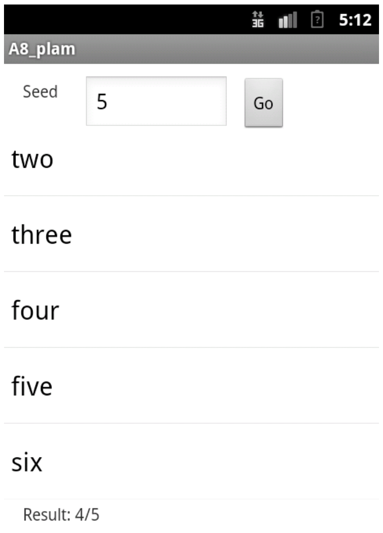
Handin Requirements. Please hand in this assignment in the `a8/` subdirectory of your main user directory on the ECE 155 repository. Your package name should be `ca.uwaterloo.ece155.a8`. As always, hand in the `java` files in your `src` and `test` directories, your `R.java` file for the app, and the `apk` files for your test and app.

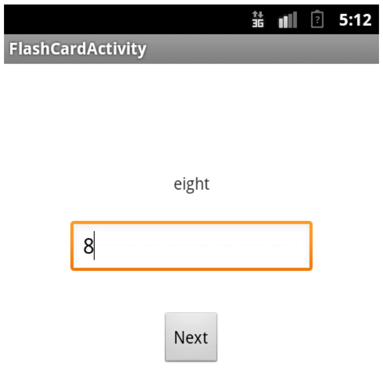
Please do not use any global variables except for a static field containing a `CardDatabase` object in your `MainActivity`.

If I find any cases of plagiarism on this assignment, I will apply the standard penalties and report the case to the Associate Dean, as per Policy 71.

1 Activity Structure

Create two activities for this assignment: `MainActivity` and `FlashCardActivity`. `MainActivity` accepts a seed for the random number generator and displays a list of cards, as well as the result of the previous play. The `FlashCardActivity` displays the front of a card and waits for the user to type in some text, which it compares to the back of the card. Your two Activity classes must communicate: the `MainActivity` says which cards to load, while the `FlashCardActivity` reports the result. You must use `Intents`, not global variables, to communicate between the Activities.





2 Helper Class (2 points total)

Implementation. [**1 point**] Implement a class called `CardDatabase` to store the flash card contents. It should contain the following methods:

```
public int getN();
public String getFront(int card);
public String getBack(int card);
public void setFront(int card, String front);
public void setBack(int card, String back);
public List<String> getFrontSet();
public List<String> getBackSet();
```

You can implement this class however you want. However, do not make any fields of the `CardDatabase` class visible: that would be poor design. If you only make the methods visible, then you can swap out your implementation with a better implementation.

Default values. To facilitate testing, use the numbers from 1 to 10, written out in English ("one", "two", etc.) as the card fronts and the digits 1, 2, ... as the card backs.

Bonus mark. You can get 1 bonus mark by storing the list of cards in a file on the phone's filesystem. However, I'm not going to explain how to do that. If you do this, put a `README` file in your `a8` directory and explain 1) how to use your code and 2) how your code works.

Unit Tests [1 point]. Write 3 unit tests for your `CardDatabase` implementation. The tests must be distinct and meaningful.

3 MainActivity Layout and Implementation (3 points total)

Next, populate your `MainActivity`. I recommend the following layout: at top level, create a `LinearLayout` with vertical orientation. Inside that layout, create a second `LinearLayout` with horizontal orientation. This layout must contain three views: a `TextView`, an `EditText`, and a `Button`. Next, in the top-level layout, add a `ListView` and a `TextView`.

To facilitate testing, the ID of the first `EditText` must be `seed` and the ID of the `Button` must be `go`. Also, the ID of the second `TextView` (below the `ListView`) must be `result`.

Implementation. [3 points: 1 per method] Implement three methods: `onCreate`, `onClick`, and `onActivityResult`. The `onCreate` method must populate the `ListView` with the set of card fronts. The click listener must also add a click listener to the `go` button.

Your click listener initiates the testing activity by creating a random number generator (RNG) with the provided seed and requesting integers from it. To create a RNG:

```
Random r = new Random(seed);
```

Then, to request a random number between 0 and N, call:

```
int i = r.nextInt(N);
```

You probably want to use an array to communicate the sequence of cards to the `FlashCardActivity`. Put it on the `Intent` as an extra. `int[]` may be easiest. Remember that you'll also want to start the `FlashCardActivity` with the `startActivityForResult()` call.

Finally, when the `MainActivity` gets a result back from the `FlashCardActivity`, it must post the result in the `TextView` labelled `result`. Please use the following format:

```
Result: c/t
```

where c is the number of correct responses while t is the number of total responses.

4 JUnit App Tests (1 point total)

Test Implementation. [0.5 points per test case] As in Assignment 6, implement an Android Application Test project. Write 2 test cases: one where you supply the seed `17` and verify that feeding 5 correct answers to the app produces the correct result in the `result` on the `MainActivity`; and one where giving the seed `42` and feeding 2/5 correct answers produces the correct `result`.

5 FlashCardActivity Layout and Implementation (4 points)

The final part of the main implementation is to implement the `FlashCardActivity`. (Note how I've set up the assignment to encourage test-driven development.)

Layout. This time, you can use a `RelativeLayout` and place the constituent `View` objects manually. The `TextView` with the flashcard front must have ID `prompt`. The `EditText` with the flashcard back must have ID `answer`. The `Button` to proceed to the next flashcard must have ID `next`.

Implementation. [**4 points**] Your `FlashCardActivity` must display the next flash card in the sequence that the `MainActivity` provided and wait for the answer (using a click listener on the `next` button). If the answer is correct, it must increment the number of correct answers. Your click listener must display a `Toast` giving the user feedback about her response. If the user has completed the sequence, then your `FlashCardActivity` must return a result to the `MainActivity`.

You must maintain the number of correct answers without using a global variable. You may use an `Intent` to start a new `FlashCardActivity` upon a click of `next`.

6 Bonuses: Online Card Editing; multimedia

If you implement any bonus functionality, please create a `README` file and put it in your `a8` directory.

For a bonus mark, implement a listener on the `ListView` which brings the user to a new `Activity` (not described here). That `Activity` should allow the user to edit the card in the `CardDatabase`.

For another bonus mark, implement multimedia cards. You could have a picture as a prompt, for instance. If you're feeling ambitious, figure out how to use the speech recognition libraries and recognize the user response. Both of these suggestions are worth 1 mark each.