Engineering Design with Embedded Systems:
Assignment 10: Refactoring with Blackjack App; Robotium Testing

March 18, 2013

Prepared by Devin Binnie
Due Date: March 25th, 2013

Learning Goals

The goals of this assignment include:

- Demonstrating an ability to refactor code.
- Practicing your code writing and code review skills in Java and Android.
- Extending your knowledge of JUnit testing using Robotium.

Refactoring (7 Marks)

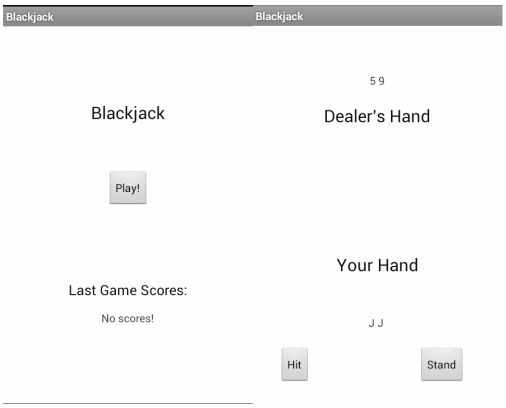

Figure 1: The A10 Blackjack App.

We've provided an Android implementation of a blackjack game. If you don't know what blackjack is, look it up. The rules used by this app are the same as standard blackjack except:

- There is no betting, split or double down.
- The Ace (A) is always equal to 1.
- If there is a tie, dealer wins.

In this implementation, you (the player) face a dealer, controlled by the computer. You can hit first. If you choose to stand, the dealer will look at the your visible cards (that is, all but the first card, left to right), add 10 to the cards' value, and hit until the dealer card value is higher.

In the code, there are 10 'code smells' which I've deliberately put in. Refactor them out. Fix 6; you will get 1 mark for each smell. Note: You should try and find more than 6: if you find something that isn't a code smell, I can't give you any more than a thumbs up.

You will also receive 1 mark if the refactored code behaves exactly like the provided code.

Robotium Testing: 3 Marks

Robotium is a testing framework built on top of JUnit and Android for performing black-box testing on Android applications. It allows for multiple activities to be tested at once. It is easy to use, and I'll be talking about it in this week's tutorials. You can find the Robotium website here:

```
https://code.google.com/p/robotium
```

The website contains some straight-forward tutorials on how to use Robotium to test your application. I suggest you read them to understand this technology. For this assignment, assume that you will be testing an external APK file.

You will need to write 3 JUnit tests using Robotium. They are worth 1 mark each. These are the cases you will be testing:

- If the player goes over 21, player loses.
- On the dealer's turn, it should take the player's visible cards, add 10 and hit until the dealer's card value is greater, or has gone bust. (You may want to run this a few times to make sure it works. Alternatively, you can temporarily edit the code, and provide a custom seed (like in A8) for the random number generator.)
- The program must record wins and losses, and display them when the user hits 'Quit'.

You may want to steal some of the code that we provided to write these tests.

Deliverables

Your directory structure upon submission must match the following structure. Make sure that all files are named correctly. **There will be no exceptions.** Note that 'userid' means 'dwbinnie' not '2037xxxx'. If you have extra files, don't worry, but try and avoid uploading them.

The project you are given contains many uses of 'userid'. Make sure you find and change all of them to your userid so that it can be submitted (and marked) properly. These instances exist in AndroidManifest.xml, your .java files and some directories.

Make sure you also don't change any variable names, or Android View names. Also do not add or remove classes. Hint: This might help you in Part 2!

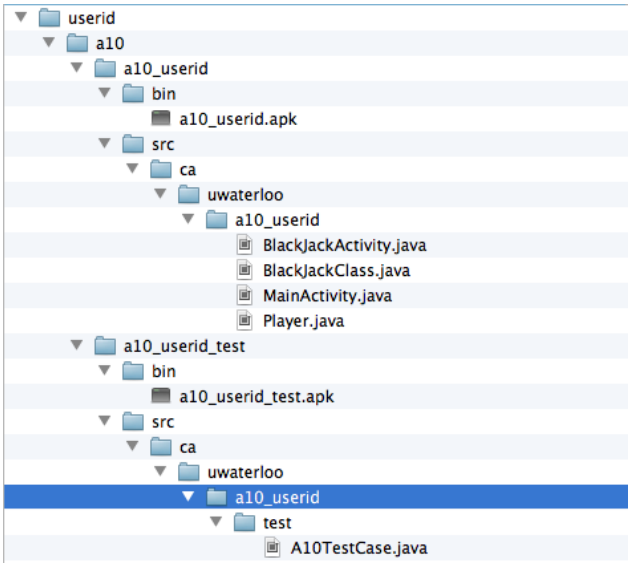

Figure 2: The directory structure of a successful A10 submission.

Bonuses!

This assignment contains 3 bonus levels, as follows:

- 1 mark—Implement a method to store the wins and losses to the SDCard of the phone.
- 2 marks—Fix the Ace problem (the initial specification says that Ace is worth 1; fix it so that it is equal to 1 or 11, whichever is more advantageous for the holder of the Ace).
- 4 marks—Implement a card-counting algorithm for the dealer (ask us for details).

To get full marks, you must do a bonus level as well as all bonus levels below it—if you want marks for bonus level 3, you must also do bonus level 1 and 2. Also, please note that the bonus marks do not stack. (For instance, if you do bonus level 3, you get +4 bonus marks, not +7).

To submit these bonus assignments, create new projects for each bonus assignment you complete (you can use our provided code, of course), and put them in your `a10/` folder with the folder name '`a10_userid_bonus_level`' (for example, `a10/a10_dwbinnie_bonus_3`).