Coinz

Software Design Decisions
&
Project Planning

Pieris Kalligeros s1607754

1. Introduction

In the following document I describe the design decisions I have made for my implementation of the practical coursework, including the selection of one of the two programming language options (and the reasons of choice), the aspects of the game beyond the high-level specification and some bonus feature options, of which I will attempt to include as many as possible in my implementation. Some of the design decisions might change in the progress of my implementation, which I will underline in the 2nd coursework's report.
Moreover, following the design decisions, a sample plan is outlined for the implementation, which I intend to follow, in order to render my design into a real app prior to the deadline on the 14th of December.

2. Design Decisions

2.1. Programming Language Selection

I have decided that Kotlin is the suitable programming language for my implementation. The important reasons of choice, besides the learning opportunity for a new programming language, are described below:
Reasons of choice:

1. Assuming that the implementation will be given to a professional developing team for further development, the Kotlin programming language provides a range of experienced-programmer features that Java doesn't, which can prove very useful for a professional team wishing to extend the software's functionality. Some of this features are ***operator overloading***(the ability to overload an operator symbol with a function of choice for a specified type), ***extension functions and properties*** (adding a function/property to some sub-class without having to inherit it from its super-class or use a Design Pattern manoeuvre), ***object expressions*** (a method to express single-object classes to implement the useful Singleton design pattern), ***function types with receiver types***(similar to Haskell's type signatures for functions, with the addition of the receiver notation, declaring that the function is callable for an object of the receiver type) and more…
2. Kotlin provides some features that save beginners a significant amount of work and errors. The most solid examples are ***smart casting*** and ***null-safety***, which provide automatic type inference for variables and the avoidance of NullPointerException, respectively. These are tools that will help hand in a safe and clear code to the development team.
3. Kotlin can refer to Java code which can be auto-translated to Kotlin code.
4. Kotlin is a newly-created and recently recognized programming language (especially for Android development - by Google). This provides certainty that the game will be **future-proof**, ensuring its potential scalability and reproducibility in the near future, when Java might become obsolete.

2.2. Implementation decisions:

The following points specify the implementation decisions that we must make as designers given the initial underspecified design of the game:

2.2.1 Login/account credentials

Each user/account holder (1:1 correspondence between users and user accounts, as specified), will be able to create an account (sign-up to the game), using a *unique* username and a password of their choice. A new user will also provide an *unmodifiable* e-mail address, used for when he/she forgets his/her password. The credentials of the user will be stored in a database using the Firebase service. The user will then be able to login to the application using the username and password that were entered during sign-up.

2.2.2. Gameplay rules

2.2.2.a. Sessions per day

Each user will be allowed 1 session of the map per day per mode (see *bonus features* below). This, in spite of restricting the gameplay and some features (e.g. collection of more coins in a later time of the day), will make the game fairer and more competitive as players will only have one chance to collect as much coins as they can. This boundary also reduces the risk of addiction to the game and regulates the time of gameplay to allow free time for other activities (such as studying).

2.2.2.b. Expiration of coins

Coins (and extra coins kept in wallet) expire in the end of the current calendar day, so the users must cash in the collected coins to the bank before the day ends (and optionally send spare change to another person before the wallet coins expire).

2.2.2.c. Conversion of coins to GOLD

The amount of selected coins to cash in to the bank will be auto-converted to the GOLD currency, so as to avoid addiction to “gambling” with the currencies’ value each day.

2.2.2.d. Spare Change

Spare change received/sent will be also auto-converted to GOLD before arrival to the receiver’s bank.

Moreover, the received spare change will be automatically cashed into the recipient’s bank, and whether the recipient has already cashed in the corresponding coins that they received will not matter.

2.3. Bonus Features considerations

The following are ideas of bonus features that I would like to implement for the game. The expected number of implemented features in the final implementation is two, but I will try to implement as many of these features as possible. The bonus features are listed below:

2.3.1. Modes of play

Modes of play are new, different instances of the game, which are similar to the original game but with each one having their own twists and goals.

2.3.1.a Challenge Mode

In Challenge Mode, a user/player can challenge another user/player (by entering their username into a challenge search box) into an additional session of the current day's map, ***betting*** an amount of GOLD as the prize of the winner.

- The maximum betting amount is limited to the highest GOLD amount that **both** players have in their banks (that is, the minimum account balance of the two players' accounts).
- The recipient of the challenge can either accept or refuse the challenge.
- In Challenge Mode, each of the two players has a 5-minute period to collect as many coins as they can.
- The winner is the player who has collected the coins **worth** the **highest GOLD** value.
- The winner of the challenge will win the betting pot, which will be *deducted* by the loser's bank and *added* to the winner's bank.
- As stated in the design decision **2.2.2.a. ,** each player has a limit of one session per mode per day, so any user can only play one match of challenge mode per day (either challenging another player or accepting a challenge from another player)\

2.3.1.b. Coin Fever Mode

In any Coin Fever game, (1 daily as in **2.2.2.a.**) a player races against a 3-minute timer to collect 10 or more coins.

- If the player makes it to the 10 coins (or more) before the time ends, then the collected coins are converted to GOLD and added to the player's bank.
- If the player however collects 9 or less coins, those coins will be tossed and the player will exit Coin Fever without any profit.

2.3.2. Fair-play Fine

If a player collects more than 25 coins in Classic Mode (the original game as specified in the coursework) and decides not to send **any** spare change to another player, then the extra coins are tossed and in the case that he/she receives spare change from another player, a 75% fine will be applied on any amount of spare change received. This is to encourage players to play fairly and with good intention towards other players, avoiding greed and unfair competition.

2.3.3. Leader board

A player leader board showing the player names (usernames) sorted in descending order of GOLD in bank.

2.3.4. Show/Hide unwanted currency of choice on map

Buttons allowing the user/player to hide or show a chosen currency's coins on the map. This is to help the player focus on a specific group of target coins of a particular currency.

2.4 Project Week-to-Week Timetable

Week 2:

- Make a thorough read of the coursework specification.

Week 3:

- Complete Coursework 1's tasks and submit the Coursework 1 plan document.

Week 4:

- Setup of online service backends (e.g. Firebase,MapBox).
- Start-screen of application (Splash and Login Screens).
- Attempt to finish account sign-up/log-in implementation.

Week 5:

- Automated tests for sign-up/log-in, testing that the services work.
- Design main menu screen.
- Start implementation of Map downloading and persistent storage of application data.

Week 6:

- Finish implementation and start testing map downloading and app data storage.
- Start to implement and design Map display on screen, with appropriate markers for coins.

Week 7:

- Finish Map implementation and design and test its functionality.
- Start implementation of Coin collection into wallet when within distance (location services).

Week 8:

- Test Coin collection within distance and appropriate wallet transactions.
- Start implementation of cashing-in coins and sending spare change to another player (involves bank account objects and methods for each player)

Week 9:

- Finish and test implementation of cashing-in coins and sending spare change.
- Complete any unfinished tasks.
- Start writing report on included/excluded design features in implementation and produce screenshots of all completed implementations for the report.

Week 10:

- Start of implementation and layout design of Bonus Features.

Week 11:

- Further implementation and layout design of Bonus Features.
- Start automated tests for the Bonus Features.

Week 12:

- Further tests on Bonus Features and re-testing of all previous functions of the application to check consistency without interference with the bonus features implementations.
- Screenshots of the bonus features in the report.

Week 13:

- Final checks on application, its design and all its functions.
- Finalization of report.
- Submission of Coursework 2.

----------END OF DOCUMENT----------