University of Edinburgh

Software Engineering Large Pracitcal

Part 3: Grabble Documentation

Author:
s1317642 Kendeas Theofanous

January 25, 2017

1 Introduction

The Grabble Documentation describes the implementation of the game based on the previously-supplied design, with a few changes based on the feedback and suggestions specified in the previous parts of the course. Algorithms and data structures used for the core functions of the implementation are described. Sections are included in the Documentation stating clearly which parts of my design have not been realised in the implementation, as well as additional features of my implementation where were not described in my design. Details on the use of version control systems to manage my project's source code and the types of testing applied on my implementation are also included.

2 Software Architecture

2.1 Changes to Activities

The implementation of the game was not feasible with only the four activities stated in the Design Document. Therefore, I have added two additional activities, handling signing up of the user and the creation of words. Further additions and changes were applied to the previously explained activities.

2.1.1 MainActivity

MainActivity, being the login page of the application, needed a Sign Up button for the first time a user would try to log in and play the game. When the user clicks that button, they are directed to the SignUpActivity. Once the user decides on their log in information, this button is renamed to "Change pass", such that the user can change their password. When clicked, they are first prompted to enter their previous password and then they will continue to the SignUpActivity again to change their log in information. I provided another option for the user to log into the game, without having to write a username and password all the time. I have added a Google Sign In button, which is a sign in feature with authentication. When the user clicks that, they are prompted to choose from their Google email accounts to sign into the game. Once they do that, they proceed to the game. I have enabled the automatic Google sign in feature, i.e. the user's account is saved the first time and they can proceed to the game by simply clicking the Google sign in later on. I have also added a Disconnect button, in case the user decides to disconnect their Google account from the application.

2.1.2 SignUpActivity

I have implemented a simple sign up activity where the user enters their preferred credentials. Once the user clicks the Sign Up button, they are prompted to confirm that they indeed want those to be their credentials. If the user clicks yes, their credentials are stored directly on the device's internal storage, creating a file private to the application. These credentials are checked in the MainActivity to confirm the user is using the correct log in information.

2.1.3 SplashActivity

This activity was implemented as stated in the Design Document.

2.1.4 MapActivity

As suggested from the feedback received, the application will not ask permission to access the calendar of the user's device in order to load the map. Instead, I imported the Calendar Java Library and used the included methods. I added a *Play* button, which when clicked starts the MakeWordActivity. I also added two buttons to pause and resume the music played throughout the activity. These buttons are one on top of the other and when clicked will swap visibility state indicating the state of the music playing.

2.1.5 CollectionOfLettersActivity

This activity's layout is changed completely. It contains the buttons stated in the Design Document. The Letters button displays a dialogue with the letters the user has collected so far in their game time, but not limited to the day game time as stated in my Design Document as that makes it harder and less fun for the user. The Dictionary button is now part of an Additional Bonus Feature explained in the aforementioned section. The History button was kept the same. I added a Score indicator at the top of the activity's layout, showing the sum of the scores of the words the user has created in their game time. I created *Image Assets* for all 26 letters of the alphabet resembling Scrabble tiles and added them on the layout with each letter's individual value.

2.1.6 MakeWordActivity

MakeWordActivity is the activity where the user creates the words by using the letters they collected. At the top I put a title Value. I created a large TextView containing seven smaller TextViews, all in LinearLayouts with a rectangular shape which can be found in the *res/drawable* folder. I added the "boxes" in a List to make it easy to process what is written in them and do further processing. I also created a special "keyboard" using ImageViews of the Image Assets of the letters mentioned above. I associated each ImageView with the letters using a HashMap, such that when the ImageView is clicked, the specific letter appears in an empty TextView. I have a function for adding the letters to the TextViews, called *populateBox(String letter)*, which checks in order from left to right (0 to 6 index in the List) which is the next empty TextView and sets its text to the letter clicked. The boxes can only have one letter as text. Once the user clicks on an ImageView, the amount of collected letters associated to that ImageView is reduced by one. The user is prompted when they run out of a letter, if they click on its ImageView. At the bottom I added three buttons. A Calculate button, will add the values of the letters and display the total value of the letters below the Value title. A Clear button will clear all the boxes from their content and set the Value to 0. The Save button saves the word created. *See section 3.4.*

2.2 Additional Features

I made a slight change when using the music player for the song I mentioned in the Design Document. In order to reduce battery consumption of the application and give more options to the user, I removed the MusicService class and created a MusicPlayer instance in the MapActivity where it's the only activity music will be played through.

I removed the vibration after receiving feedback from Part 1, suggesting that

due to the large number of markers, vibrations are intrusive. Therefore, I decided to make all markers invisible when read at the start of MapActivity and when the user is at a distance of 25m from them they become visible.

When the player grabs a letter, the Placemark disappears from the map and therefore the user has a clearer view of the map and which Placemarks are avaiable to get a letter from.

Showing the History, i.e. the words the user has created and their score and displayed in descending order is something I added.

The use of the Dictionary was improved. The user can click on the button, but they only have access to it if their Score is greater than 500. If this is the case, the user is prompted that by accessing the Dictionary they will lose 500 points from the Score and they can only access it once. When in the Dictionary dialog, they can browse as much as they want but when they leave the dialog by clicking the back button, they can no longer access it (unless at the expense of 500 more points).

I also decided to not allow the user to create duplicates of the words they already created. The reason for this is to make the game more interesting and make it slightly harder and more creative to play.

The Instructions dialog in the SplashActivity is also an option the application provides.

Another trivial thing to mention is that I force the orientation of all activities of the application to be portrait. In case the user has auto-rotation activated while playing the game and rotates their phone at some point they would force landscape orientation which I do not support in my implementation. By only allowing portrait orientation, I get rid of this possibility.

3 Algorithms and Data Structures

Throughout the application, I use Internal Storage to store any necessary data needed between activities and also data that needs to be loaded after the application is closed and opened again.

3.1 Data structures

I used a HashMap associating the "grabbed" letters and the number of them grabbed, a HashMap associating each letter with its value, a HashMap associating the words created and their values, a List of strings as the Dictionary, a List of markers as the markers loaded on the map.

3.2 Download and parsing of the daily Grabble letter map from the server

I use a List to add all the markers loaded either from a file or from the KML layer from the URL.

3.2.1 Write to file

I use a file to write in all the markers when MapActivity is paused and destroyed. I use *FileOutputStream* with Private mode. I go over the List of markers and write on the file the latitude, longitude and snippet(letter) of the marker split by commas, each on a new line.

3.2.2 Decide what to load as markers

A decision on what markers to load on the map needs to be made every time the user opens the application, since some letters are taken by the user and should be missing from the map throughout the day. A new KML layer though needs to be loaded according to the day of the week. In the *onMapReady()* function of the MapActivity, I choose the KML layer to be loaded by deciding on the URL to load according to the current day of the week using functions from the Calendar Java library. I don't load it straight away though. I check the date the file was last modified. If the file was last modified on a different day as the current day, the application goes through the KML layer and gets the Placemarks, adding the markers on the map and in the List of markers. Otherwise, the user already opened the application on the current day and so modified the contents of the map. Therefore, the markers should be read from the file.

3.2.3 Read from file

I use *FileInputStream* to read the contents of the file. I go through the lines of the file and parse the String by splitting it from the commas. I take the latitude, longitude and snippet, create a Marker, add it on the map and in the List of markers. All markers are initially set to be invisible.

3.3 Detection of the letters which can be grabbed at the user's current location

Every time the location changes, the method *handleNewLocation* is called and, going through the list of all the markers, the distance between each marker and the location of the user is calculated. If that distance is less than 25m, the marker becomes visible otherwise it becomes invisible. This way, I handle both cases of being in the range and exiting the range of 25m from a marker. When the marker becomes visible, then the user can click on it and grab the letter, adding it to the HashMap of collected letters.

3.4 Creating words and lookup of words in the Grabble dictionary

As mentioned in the MakeWordActivity, letters are written on the boxes viewed. Whenever the user clicks the Save button, all letters that appear on the boxes are concatenated together to form the word they entered. Then, I check whether that word exists in the Dictionary. If it exists, I check whether the user had already created that word before because I do not allow duplicates of the words. If they hadn't, a dialog pops to prompt the user to decide whether they want to save the word or not. In case the word doesn't follow these rules, the user is prompted accordingly.

4 Version Control System

I extensively used *Git* throughout the implementation of the application. I archived the whole project, in order to control all changes made in all directories of the project. The Java files, Manifest, layout files as well as raw files are kept in different directories and I wanted to know when I made changes and what changes I made, for backtrack purposes and to go back to a previous version of my work in case I broke something in the code. I also kept a good log when committing my changes and pushed them online to BitBucket.

5 Testing

For testing purposes, I mostly used a physical device with Android version 5.1 to collect letters from the Central Area and test the functionalities of the application in real-time. I also used the emulator, i.e. the Nexus 5 API 22 virtual device to make sure there were no issues.

5.1 Instrumented Tests

I formulated some basic tests, to check the existence of the layout objects of every activity, ensuring there were no null objects. Also, I tested some of the buttons I implemented that use an Intent to go to other activities.

5.2 JUnit Tests

I check a few of my methods using JUnit Tests, ensuring they work properly and have the right outcome when used. Also, testing them ensures they are not broken when further functions are added in the same class.