Master in Information and Computer Sciences
Prof Mr. Jacques Klein

Principles of Software Development

Final Report

Iva Toshkova and Vendel Tóth
Principles of Software Development
2014-2015

Summary

I. Introduction

This report concerns the final project for the course Principles of Software Development, for the University of Luxembourg. The main idea of the project is to create an application for android-type devices (smartphones), named City Rally. We did the project by using Google Maps API, and creating 3 mini games for the challenges. We tested it on an Android virtual machine. There are a total of 7 places to go to, therefore each puzzle to be solved twice (the final location has no puzzle). The puzzles are : a Tic-tac-toe game played against an AI; a Memory game where the size of the game and the places of the cards are random; and a Millionnaire game where you have to answer 4 out of 5 questions correctly using the helplines proposed. We used Eclipse for coding. There are a total of 4 packages in the app; activity where all the activities are saved; communication where all the communication between the database and the activities take place; database where we declare the database, tables, columns and its syntax to be used in communication; and item which feautures some items used in the app, such as the definition of a “Card” to be used in the memory game.

II. City Rally : the app

1. Classes

A. Activity Classes

Before telling you about the individual classes, there are a couple of things that are true for all classes :
-When calling the database, we call the worker class to do it in the background in order to not freeze the app while the app is loading or saving.
-There are a couple of functions which have the same name and called for every activity :

initUIFields : initializes the layout fields
linkNavigationButtons : creates onclicklisteners for the buttons
initDatabase : (only for classes which use the database) initializes the database, sometimes loads the contents of it
makeText : creates a toaster message (This app relies heavily on toasters as we found it was the simplest and most convenient way to return a result from something

a/ Main Activity
This is the activity when the app starts. Here we have nothing special : a background image showing Luxembourg, and a couple of buttons, which link you to other activities : Play game launches either How to play activity or GoogleMapAppActivity depending if you already have a save file or not. Delete local database simply deletes your save game and you'll start off from the beginning again. (This is good if you want to replay the game, or if you want to show other people the app and start over).

b/ GoogleMapApp Activity
This is the main interface of the game. Here you have the google maps API, that shows where you are (with the “man with the map” icon, where you have to go with the “puzzle” icon, and a line between you two); and the buttons that link you to other activities : “Solve a puzzle” links you to one of the 3 puzzle activities depending on which place you are at; “What's here” calls InfoActivity to display information about the place; and “Show hints” calls the activity that show the hints you've already discovered from the puzzle games.

When this activity is launched, it loads a save game or starts a new game depending on the save file; initializes google maps and draws the Puzzle icon. Please note that your location will update only when you start moving, therefore when you start the app and aren't moving, only the puzzle icon where you have to go will be visible.

c/ How to play activity
This activity is launched automatically if no save file was found; or if the user clicks on the how to play button in the main menu.

It simply displays the instructions to play the game and the only button to be clicked launches the main interface.

d/ Info Activity
This activity is called from the GoogleMapAppActivity, and displays simply information about the place you have to visit currently. GoogleMapAppActivity sends information to it regarding which current place to show, using a Bundle.

e/ PuzzleMillionnaireActivity
This puzzle plays a game of "Who wants to be a millionnaire" where you have 3 helplines in total to help you answer 4 out of 5 questions correctly. Note that with the helpline"Ask the public", the computer randomly generates an answer (since we have no public) : 7/10 times it will tell you the correct answers (so you never know for sure...).

f/ PuzzleMemory Activity
This puzzle lets you play a game of memory; where you have to solve a puzzle in a number of turns (depending on the size of the puzzle). The size of the puzzle is random from 4*4 to 6*6 and the location of the cards are also random.

g/ Puzzle TicTacToe Activity
This puzzle lets you play a game of tic-tac-toe against the computer. The computer is really good and therefore there's only one way to win : Start from the left-upper corner, then the right-down corner, then the left-down corner and you can win by choosing one of the two solutions.

h/ Hints Activity
This activity is called from GoogleMapApp activity, and shows you the current hints you've already discovered from the puzzles. This is where you can solve the final question as well and win the game. The color of the words of the hints change as you discover more and more hints.

i/ TestActivity
This activity is called from the MainActivity and lets you test each feauture separately (to see if they work on their own). There's also a new feauture not included in the main game : TestDatabaseActivity

j/ DatabaseTestActivity
This activity lets you add any words you want to the database. This is made purely for debugging reasons (to show that the database works on its own); and is not included in the main game.

B. Communication Classes

These classes are about the communication between the database and the activities.

a/ OpenHelper
This class implements the following functions : create, flushTable, onCreate (calls create), onUpgrade; This activity creates the database, creates the tables, deletes them and upgrades them.

b/NumberPoints database
This class specifies the nature of the NumberPoints table in the database : it has a Primary key, a String text and an Integer. The integer is not used; it is only there if we want to continue in this project and

implement other puzzles where we might need an integer : for example, a High Score implementation of a specific puzzle, or a Score number for the hints, ...

c/ SQLiteNumberPointsDataSource
This is the class that the activities call upon if they want to use the database. Its functions are : Open (open database), Close (close database), DeleteTable (delete a table), CreateNumberPoints (create a new NumberPoints (see earlier) type input in the database), cursorToNumberPoints (used by this class to retrieve the information of the cursor), deleteNumberPoints (delete a row in the NumberPoints table), and getAllNumberPoints (which returns an array List of <NumberPoints> containing all the elements of the database. We didn't implement a function which returns only one element in the database because we don't use it in the database.

C. Database Classes

The SQLite database has been done very generically (every portion can be reused later if we want to add more things to the App later on, such as MySQL (online database), or other databases).

a/ SQLiteTableInterface and SQLiteColumnInterface
These are Interfaces for SQLiteTable and SQLiteColumns and just specify the type of functions they have to have.

b/ SQLiteColumn and SQLiteTAble
These classes implement the interface functions and create Columns and Tables, validating the syntax of the queries and checking for errors in the syntax (and using SQLiteVerificator)

c/ SQLiteVerificator
This class verifies if everything is correct, without errors. SQLiteColumn and SQLiteTable use this class for verifying usual errors in the syntax of the queries.

d/ EmptyTableException
This is a custom exception class thrown when SQLite has an empty table.

D. Item Classes

a/ Card
This class is the basic representation of a card, for the memory puzzle game. It has x, y, the coordinate (position) of the card on the game; and the button for it.

b/ NumberPoints
This is the basic representation of NumberPoints, which is only a class with a String and an Int , to be saved in the database in the NumberPointsTable.

2. Content of the drawable folder

Here we have all the images we used to create the app ; which are :

-All the card images for the memory game
-The background for a couple of activities
-The main background (the City of Luxembourg as seen in MainActivity)
-x and o for the tic-tac-toe puzzle game
-different icons for the location markers (where you are and where you have to go)
-icons for a couple of puzzles

3. The android manifest

The android manifest had to be modified in order to install the Google API. We had to add the following permissions :

```
"com.pearsonarticles.googlemapapp.permission.MAPS_RECEIVE"
"android.permission.INTERNET"
"android.permission.WRITE_EXTERNAL_STORAGE"
"com.google.android.providers.gsf.permission.READ_GSERVICES"
"android.permission.ACCESS_NETWORK_STATE"
"android.permission.ACCESS_COARSE_LOCATION"
"android.permission.ACCESS_FINE_LOCATION"
```

Get the gms version (which for us is 6171000)

```
android:value="6171000
```

And get a Google Account API key :

```
"com.google.android.maps.v2.API_KEY"
```

4. Layout

The layout has been created to work on a virtual device running on a computer with a resolution of 1980x1020, therefore on a smaller computer some positions for the buttons and/or text may be slightly misplaced.

5. Unit tests

By clicking on the Test button in the main menu, you can already test every feauture individually; from saving and loading to the database to playing the puzzles and loading google maps.

Here are the steps to have a successful run of the program :
-Launch an Android virtual Machine running on Google API or launch it on your android smartphone of version 4.4 or higher with network connection and localisation enabled.
-After launching the program, in the main menu click on Play and it'll go to the Tutorial screen because there is no saved data in the database. Click on Lets Play! A toaster message will inform you that Google maps is loading, is loaded; the Database is loading and the Database is loaded. You can see a puzzle icon next to Campus Kirchberg on Google maps. That is where you have to go. If you put in a coordinate on the DDMS or start moving with your smartphone; the app will update your location and you'll be able to see where you are with a special icon; with a line drawn where you have to go. There are a total of 7 places (and 6 puzzles; the last one has no puzzle) with coordinates :

```
(49.626183, 6.158648), (49.633579, 6.173556), (49.611307, 6.126848), (49.627371,
6.145309), (49.618798, 6.142430), (49.617422, 6.127421), (49.611512, 6.122113)};
```

Put in the coordinates next to Campus Kirchberg (close vicinity – with a difference of 0.005 lat/long) or to its exact place in the DDMS; or move there with the smartphone.
A toaster will appear informing you that you have arrived and the puzzle has been enabled. Click on

Solve new Puzzle. You will be asked 5 questions, for which the answers are : A, D, B, A, C. We had 5/5 score. Once the final question has been answered, you are taken back to the Google maps with a Toaster message informing you “Well done, you have discovered a new hint!” and “Hint saved in database”. The next location is in Utopolis and you can see the puzzle icon has been taken there and the puzzle button is disabled. Move there and the puzzle button will be enabled again (the toaster notifying you when you have arrived), Click on Solve new puzzle. You will have a memory game (which is random so can't tell you the solution); for us we had a 6*4 memory game which had to be solved in less than 48 moves. We did it in 29, after which the puzzle quits and takes you back to google maps where the toaster informs you that you have discovered another hint. Go to the next location; and this time it'll be a Tic-tac-toe game against the computer. The only way to win against the computer is by putting top left; bottom right; bottom left; bottom middle. After you win; you are taken back to google maps and this goes on until you have discovered 6 hints. Once you discover 6 hints; go to Show hints and the question to be answered will be visible and should look like `"What kind of sport did Luxembourg win recently the World cup of?"`: To which the answer is “babyfoot” ; you can also enter it with uppercases since it ignores cases. Click on submit and the Toaster will inform you “Congratulations, you won the game!”; and the main interface activity finishes which leads you back to the tutorial activity or the main menu depending if you did it in one go or not. Click exit and the game exits.

III. How to play / possible errors

Once in the main interface (GoogleMapAppActivity), you are proposed a marker (Puzzle icon) to go to. The “Solve a puzzle” button is not enabled unless you are in the very close proximity of the place. Every time you move it updates your location marker (Man with a map icon) and updates the line between you and the puzzle. On the top right corner you can see the place you are supposed to go to (they are all famous places from Luxembourg), and you can check what's there by clicking the button below it : it will give you a brief summary of thatt place's interesting points. Once you have arrived to the destination, the “Solve a puzzle” becomes clear; you have to solve this puzzle in order to get the next marker and advance in the game. Every time you solve a puzzle, you get a word or two for the final question. The aim of the game is to answer the final question correctly, however you can most likely only know the purpose of the final question once you've visited all or most of the places. If you answer to the final question correctly, you win the game. There are in total 3 puzzles to be solved and 7 places to go to. Once you solve a puzzle, the marker updates itself and the Solve a puzzle button becomes disabled again.

Possible errors while trying to launch might be :
-the android GMS version for us is 6171000 but it is possible that you have a different version (in that case Logcat will tell you which version was expected, once the API crashes).
-For the Google API key, we used a key from our eclipse account, which most probably won't work if you try to launch on your computer. This is because in order to launch Google maps on a virtual device, you need to have Google account give you a special key, unique for your application on the computer and we need to get a new one every time we try to install it on a different computer.
-Google maps not showing : this might be an error of the virtual device not correctly installed or its version not supported by the application (We ran it on version 4.4)

IV. Conclusion

In this project we did a lot of things and learnt lots of important android-related principles, as well as Java principles, such as bundles and intents to be passed to other activities; SQLite databases, calling and saving in the database, ”Worker” class , how to install Google maps API, and use it, how to create simple-graphical games; and most importantly how to organize a project so as to not get lost in the code and have it structurized.

We didn't include GPS in the Google Maps API (We only drew lines from source to destination), because, working with Eclipse, the GPS library is independent from Google Maps. We would have to draw all the lines ourselves for every instruction from the GPS library, onto Google Maps; furthermore we don't know if the person is in a car, or on foot; which would require a lot of time to code and we don't think that was the objective of this project.

There is some javadoc documentations in the code for some functions to better explain the code.