University of Liège
Master in computer engineering
Academic year 2015-2016

Object-oriented programming on mobile devices

Prof. L. Mathy

AGILE : The Game !

Laurent Vanosmael
S114351
Sylvain Dazy
S090802
May 6 2016

1 Introduction

This document is the report of our application "AGILE the game". This application has been developed for android tablet under android for the course "Mobile programming" given by the professor Laurent Mathy. For many of us, it was the first time we create an android application.

Our application is a game in order to learn with fun. The section Game Manual will explain you the game. What is the goal of the application ? How to play ? will be answered in this section. Some screen shots will also be presented.

The next section, Scheme of the implementation, will present you how the game has been coded and how it has been structured.

The section Details and justification will explain you our choices and why do we choice. It will also say what minimum android version is required.

Finally, the lasts sections will presents some idea to go further or to improve our application and we will finish with a short conclusion.

2 Game Manual

2.1 Description

In "AGILE the game", you are a project manager who has to create an application for a client. The goal of the game is to realized all tasks to be done for the application within your given budget.

You have to manage your team by recruiting new programmers or by fired them. In order to realized the tasks, you have to assigned your programmers to tasks each month. Programmers have different skills. You should assign task to the programmers with the best skill for this given task.

The difficulty of the game is that you have to pay your programmers each month but you also have to realized all tasks with a given budget.

By realizing tasks, your team will improve their skills and be more efficient. However, the more experience you programmers gain, the more their salary will raise. An inexperienced coders will have a lower salary than an experienced one.

You win the game if all tasks are done and you have still budget. On the other hand, you loose the game if there is no budget left and there is still task to be done.
If you win the game, a score is calculated from the budget you have left and based on the number of month it takes you to finish the application.

2.2 Screen-shot gallery

FIGURE 1 – Manager View

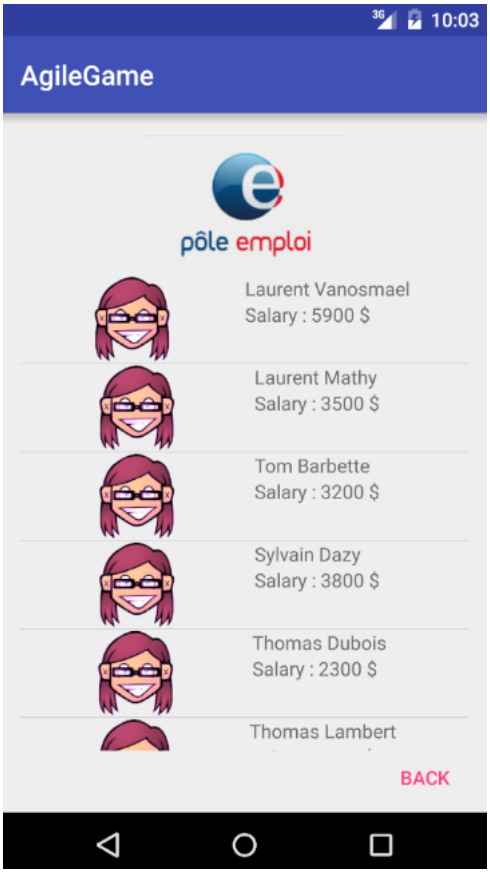

FIGURE 2 – Job market view

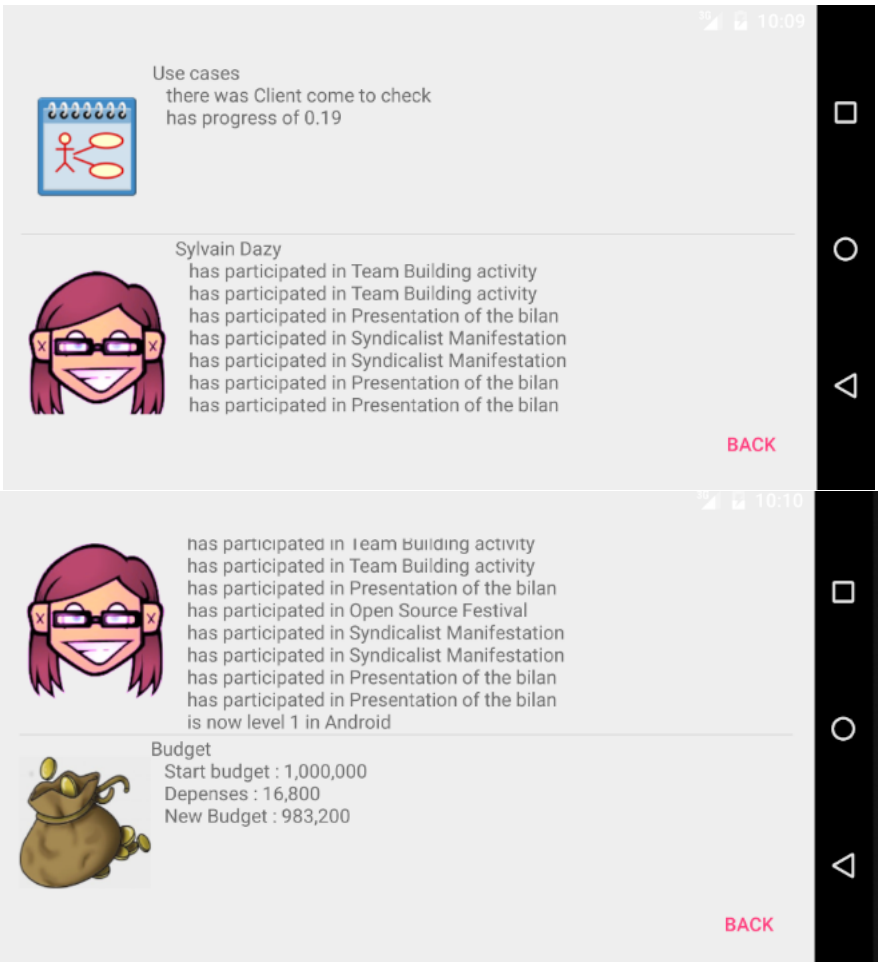

FIGURE 3 – Summary view

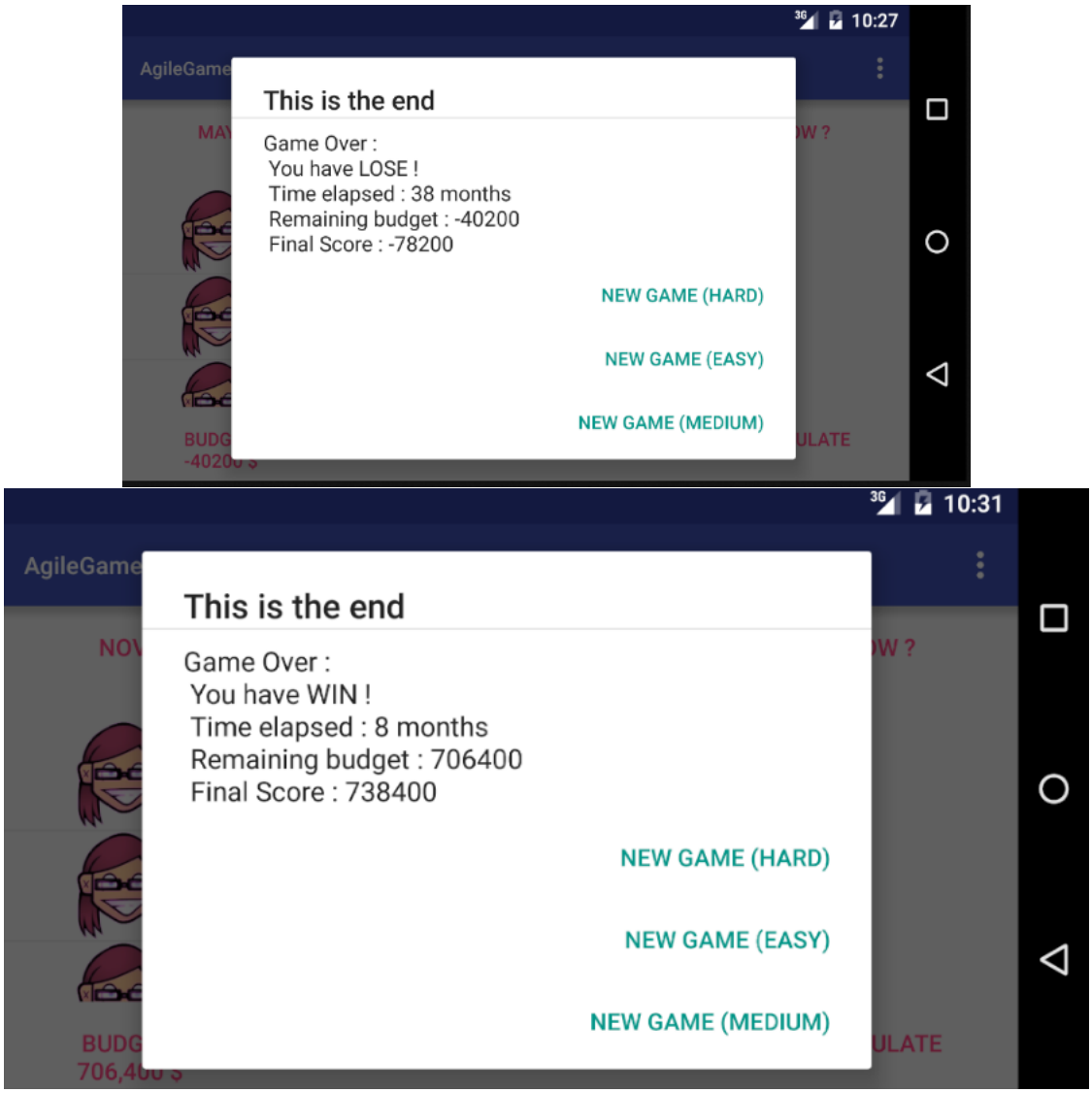

FIGURE 4 – Game Over

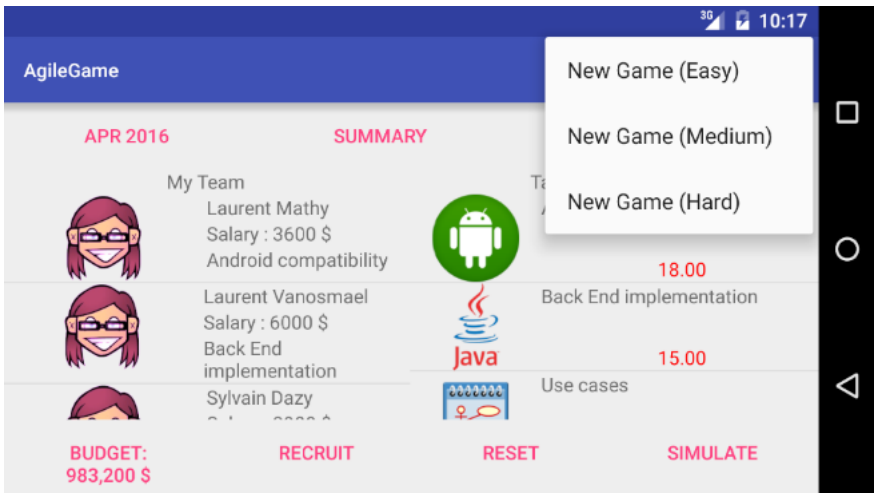

FIGURE 5 – New Game

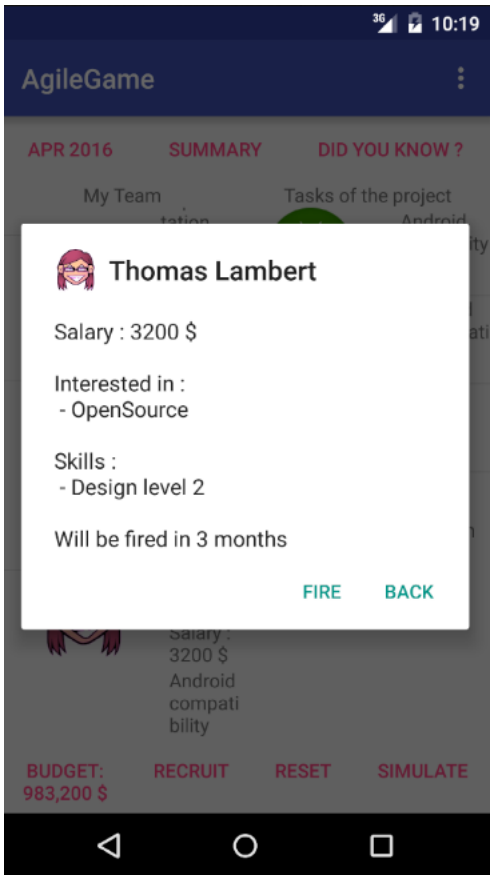

FIGURE 6 – Programmer's information

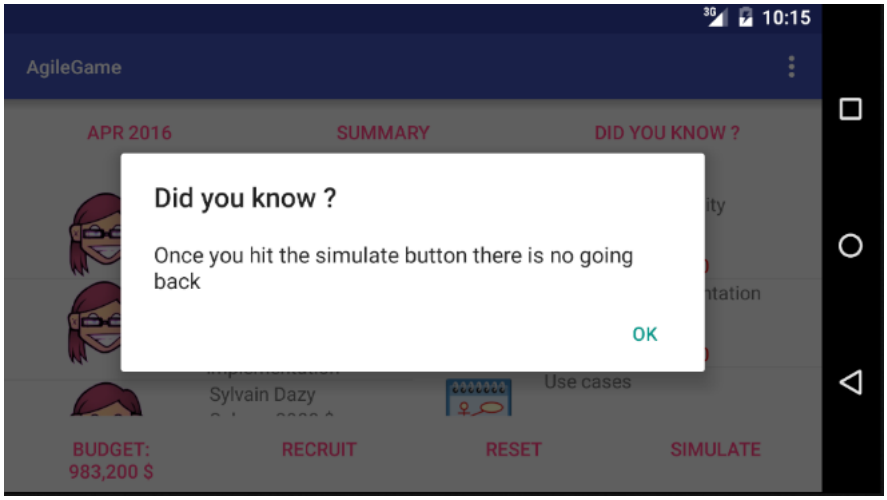

FIGURE 7 – Did you know example

2.3 How to play

When you start the game, you begin with the manager view as can be seen in figure 1. In this view, you see your team of programmers and all tasks you have to realized.

At the top, you have 3 buttons. The first one is the current date. If you click on it, you go to the calendar view (see later). The second one is called "Summary", you can have a summary of what have been done during the last month (will be explained later). Finally, the last one is "Did you know ?". Click on it and a pop-up windows will appear to know some interesting fact about the AGILE method or the game. Try it. See an example in figure 7.

In order to assign a team member to a task, you have to drag his profile picture and drop it on the task you want to assign. Then you will see under his name the work he is assigned to for the next month.

You can have more information on your programmers by clicking on their name and salary area. An example is shown in figure 6. A pop-up will appear in the center of the screen. Name, salary, interests and skills with his level will be presented. You also have 2 buttons to go back to the manager view or to fire the programmer. When you fire a programmer, it is not immediately effective as it is in real life due to prior notice.

If all tasks or all programmers cannot be displayed, you can scroll these lists.

At bottom, you have 4 areas. The first one is you current budget. The second one is the "Recruit" button. Click on it to go to the job market in order to hire new programmers. The button "Reset", un-assign all your programmers from their assigned task. When you have finished, click on the button "Simulate" to simulate the month and see the summary of what happened.

In the screen of the job market, see figure 2, you can click on programmer to get more information and to hire them (same pop-up as in the manager view). Do not forget to scroll the list to see all unemployed coders available in the job market.

In the summary view (figure 3), you see a list of all events that happen during the month, with each tasks and each programmers.
You see that some task has progressed, others can be canceled, etc. See events sections.
In the same way, your programmers has participate to some random event for example an open source conference or a presentation of the balance sheet (see event section to know more about events). We also display if the programmer has levelled up and what amount of work he did on his task. At the end of the list, you have the summary of your budget spending. Click on the back button to get back to the manager view (you can get back to this summary view by the way of the Summary button).

2.4 Events

At the end of the month (i.e. when the player hit the simulate button), the progress of the different tasks are evaluated. The progress depend of the level of the skills of the programmer but also of the different events that have happened during the past month. Those events can have a positive effect (e.g. Company Team Building) or a negative effect (e.g. participate in a union strike). They can make the programmer gain experience (e.g. Oracle Conference) or just making it loose his time (e.g. Salon de l'auto).

The event can also affect a particular feature like "Crash of the server" or "We do not need this anymore".

It is easy to add events to our game, you just have to add a string in the string.xml file, and add a line in the event builder, in the case that a special effect is needed for your event, you just have to inherit the Feature or Programmer Event and override the effect method.

2.5 Difficulty

There are 3 levels of difficulty : the easy, the medium and the hard ones.

When you launch the game the first time, you begin at the easy level. But you can change by clicking

on the settings button (figure 5). At the end of the game, you can restart with a new difficulty (figure 4). At the hard difficulty, you have much more tasks to realized. In this configuration, you really have to assign a task to a good programmers and to recruit good programmers. It is very hard challenge.

3 Scheme of implementation

We have developed our application by separating the logic of the game and the view and android activities.
The concept of programmers, events (of the game), tasks, feature, application, etc are in the game logic whereas activities (manager, pole_emploi, summary), list adaptors, are not.

3.1 Model side

Here is the diagram of the model side of our application, it can be seen on the diagram that all class are connected somehow to the GameLogic class

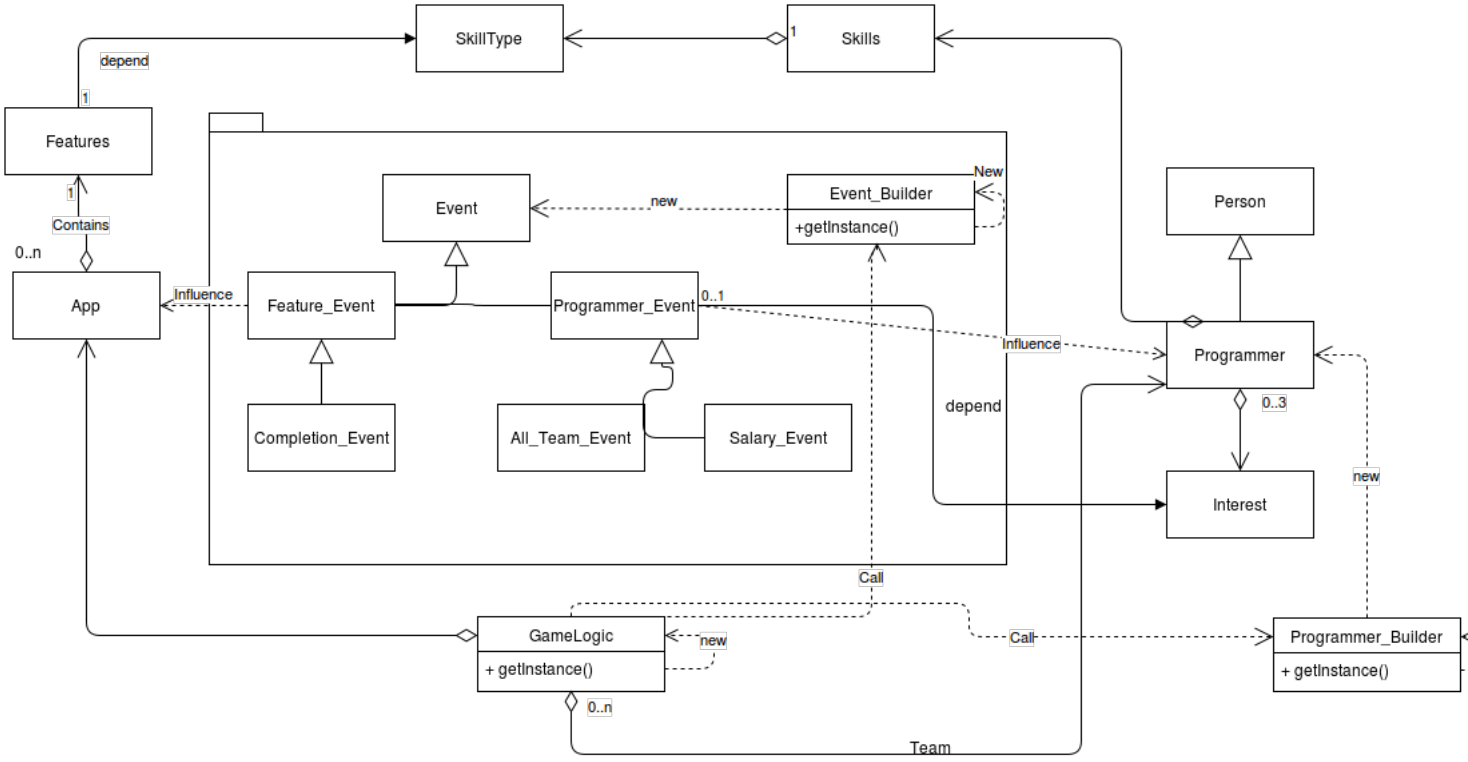

FIGURE 8 – Model side

3.2 View side

Here is the diagram of the view side, (i.e. activities, fragment and adapters), all those class have an ancestor that is part of the android library or the compat library. All the activities get their data from the GameLogic class.

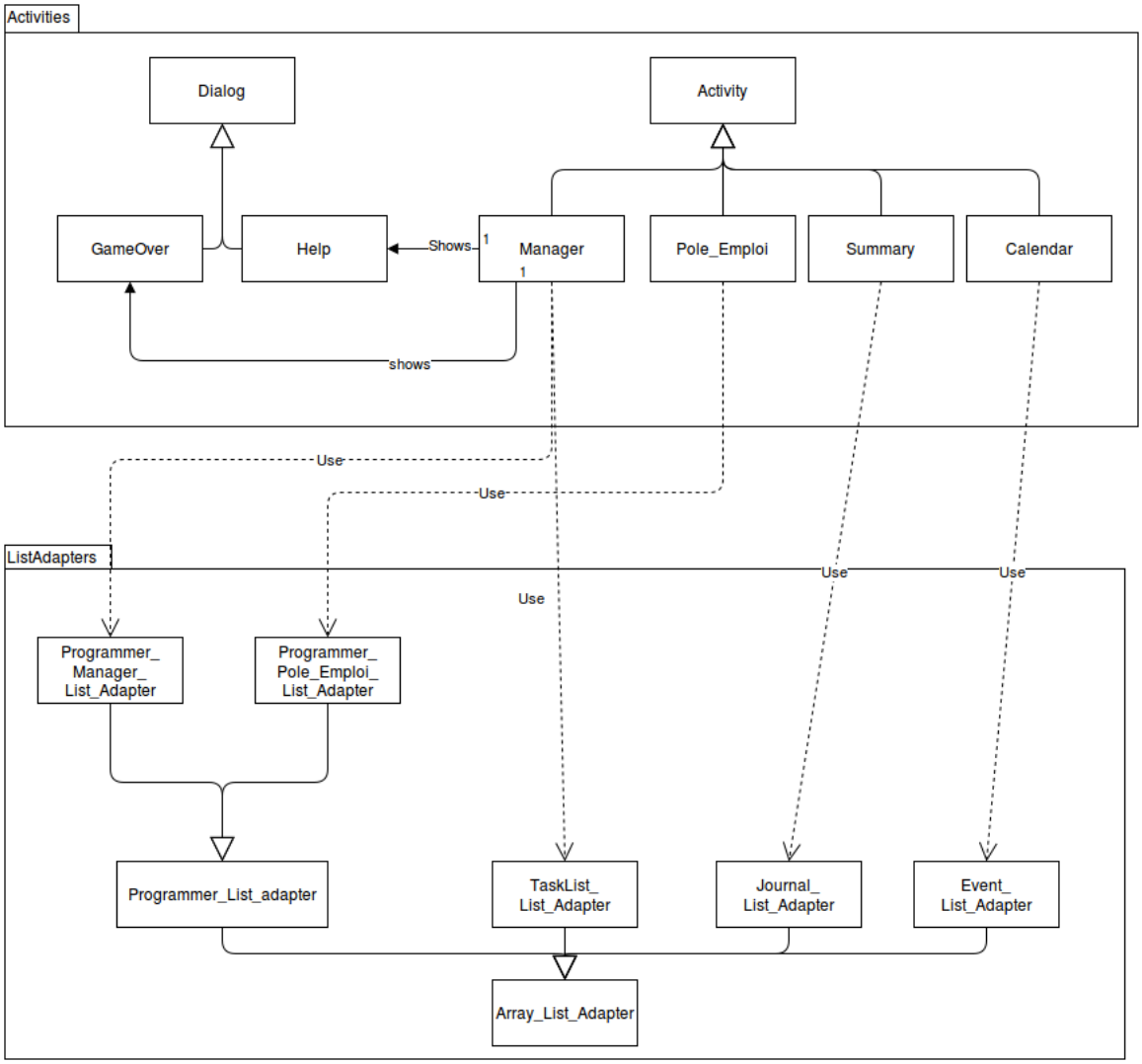

FIGURE 9 – View

It can bee seen on the graph that all adapters are specially designed for a specific activity. But it is interesting to notice that the Pole_Emploi and the Manager have both to show a list of programmer, the only difference is the actions that can be done with them.
We have thus take advantage of the oriented object aspect of JAVA and android an create an abstract class to avoid copy paste.

4 Details and justifications

4.1 Version of Android

The application has a minimumSDK of 15. This SDK covers 97% of the users of he Google Play store according to AndroidStudio. It is a good trade off between performances (because we can use appcompat for mechanisms implemented in futur release of android) and reachability because there is not much chance than someone want download with the Ice cream sandwich from 2011 android version.

4.2 Patterns used

4.2.1 Observer

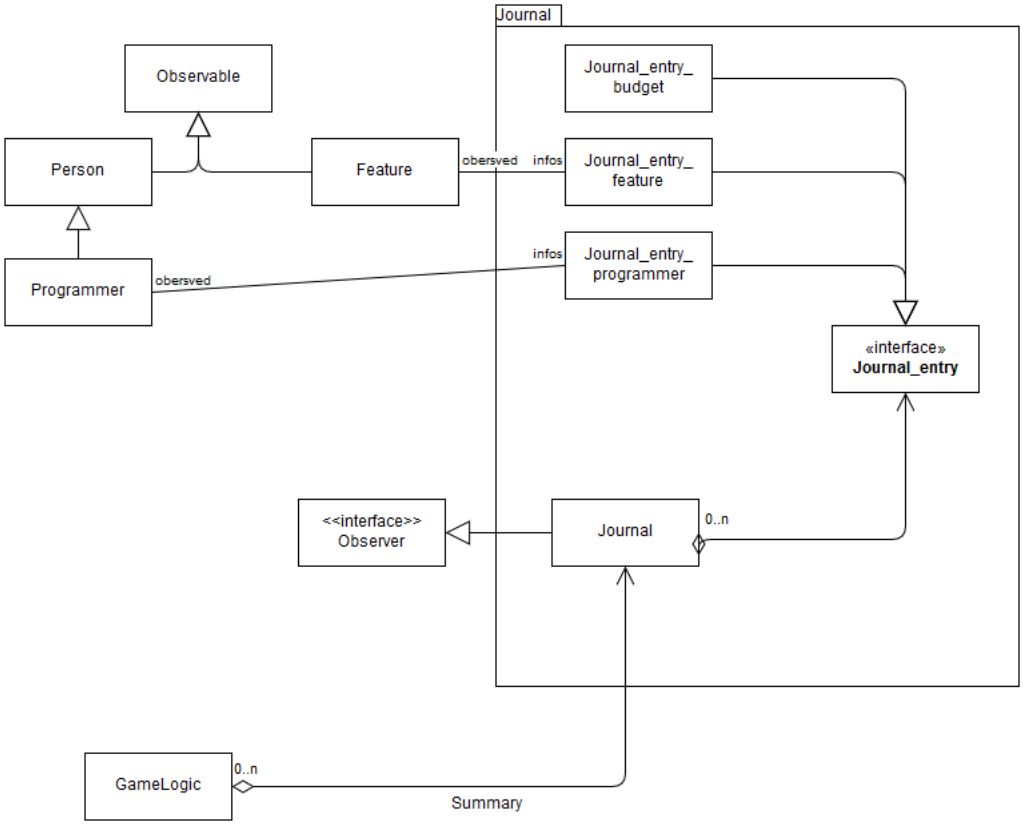

FIGURE 10 – Observer pattern

As shown on the graph Fig 10 the diagram the observer pattern is implemented on the side of the model system, it is used to provide a summary of what happens when the Simulate button is hit.

What is done on the model side, the observable side, is notify the observers if a programmer gain a level in a skill, if a feature progress (and thus if a programmer works), if some event affect someone and if some money is spend.

On the Journal side, the observer side, each time some of the previously described action occurs the journal object will register it by either instantiate a new object for the programmer, feature or budget, either by modifying the existing one. Those objects are store in different Hashtables because it should be easy and quick to get one object and modify it or to add one. But to order the different Journal entries when they are displayed. Those entries are first put all together in the same ArrayList.

One journal object represent one month (i.e. : one turn) of the game, those information are displayed in the Summary activity.

4.2.2 Builder

As it can be seen on 8 we have use the Builder pattern for two different type of objects. Firstly for programmer : It allow us to control the capacities (skills, interest, number) of the programmers on the job market without making complicated constructors into the Programmer class. Secondly for the different events : It allow us to control the instantiation of event and their effect compared to their id (their id refers to some string that should be in accord with the effect). The random part of the events and the programmers are also put in the builder because once it is instantiate it should not be modified.

4.2.3 Singleton

The two builders from the previous section are implemented as Singleton because we have to be sure that all the object they instantiate are following the same rules and the same quotas.
The GameLogic class is also implemented as a Singleton because it represent the state of the game, and only one game can be running at the same time.
Moreover having the GameLogic implemented as a Singleton allow us to class it really easily on the view side, without having to pass it as a parameter. Thus the GameLogic can also be seen as a controller making the link between the models and the views.

5 Optional features not implemented

Some improvement could be done in the future. The list below show some of them that were not implemented due to lack of time, out of scope, or just because we had the idea after the final deadline.

— Adding a deadline parameter that could make loose the game
— Adding more features (in-game tasks), events, persons, skills,...
— Linking with Facebook and use the name (and pictures) of your friend in-game
— Allow programmers to take holidays
— Use a server to publish your score
— Save and restore the game
— Show previous summaries and not only the last one
— A French/Spanish/Russian/... version

6 Known bugs and issues

We do not know any bugs however we cannot be sure that our application is bug free.

7 Conclusion

This project lead us from an idea to a concrete application. We also learn how to create Android application.

The game we have created is basic and do not really meet its goal to really ẗeachÄGILE but we beleive that it can be somehow fun and challenging to play.

Advanced features can be added easily in further versions of the application.

Finally, we really enjoy this project.