World of Everbies

-the Project

This document serves as a collection of our thoughts and reflections that might provide helpful insight to the people who inspect this creation of Android programming.

Overview

World of Everbies has been a very interesting project in our eyes due to the way we worked throughout the project. We had never used the scrum method of working before and although we did learn how to use it, it still took some time getting used to and finding the right “flow” throughout the project. This led to that we lost some time learning the scrum way and the scrum way of planning. But as time went by we found ourselves more and more at home with scrum. So as a conclusion of scrum is that we think that although we didn’t lose that much time in learning it and we thought it was an effective way of working. It still took some time getting used to it and thereby held us back in some degree. We also experienced some trouble with coding in android due to that we never had done it before. Therefore we had to read up on android coding before we could begin with the actual project and although we didn’t experience too much of a problem it still took extra time and in a certain degree created frustration. So we thought that would have been good to have some form of introduction to android instead of only how you set up a project.

Tests

Well, well. The JUnit tests. The nemesis of smooth project planning.
It all started when we one day, like any other, arrived at our workspace and began working in the same manner as the day before. Then it began, we decided to write some tests due to the mandatory nature of these. Unsuspicious of what was to come we initially tried to write standard JUnit tests as if we didn’t do android programming at all, oh silly us. Even though it soon became apparent that they wouldn’t work our spirit was still up with the mindset that we could copy most of the code to a correct file, being an Android JUnit file instead.

Now we studied up on what differs Android JUnit from the standard JUnit and followed all the instructions as well as we could. Though the test still wouldn’t work. The issue now was that the tests seem to run fine, but the data wasn’t collected at the end. And after some fixing and trixing they stopped working altogether. From this point we tried many different approaches, none of which presented a working solution. Neither our group supervisor nor the teacher understood

what the problem was or how to solve it.

In the end we had spent 3-5 days of work trying to get JUnit to work but had to resolve to writing new “ugly”-tests, that just print results in the log, and manual tests that are tests describing how we test features manually.

These “failed” tests are available for inspection at Github at user: Karl-Agnes, repo: WoETest.

Branches

During the fourth week of this period we experienced a problem with git that greatly set us back for that sprint. It began when we decided to start working with branches. We created one branch for each part of the sprint and started working. It worked fine at the start, but when we were finished and were ready to merge the branches into the master branch we notice that every branch had it’s own set of problems which prevented it from merging. So this took a substantial amount of time and prevented us from continuing on the sprint plan.

Our reflections of this problem is that we didn’t have enough knowledge of branches before we started working with it. Which in turn led to that we probably pushed and pulled incorrectly and thereby our problems grew until we had no choice but to deal with it.

Races

When we started out we only made the races into description to keep track of the different images for each specific race. But when the program grew and we implemented more features we thought it to be a good idea to make each race more specific. Thereby we created a class for each class that extended the abstract class Race in order to be able to make each race even more unique. Though upon closer inspection it turned out to be just a class filled with private variables and getters to fetch them. Therefore we wanted to redesign the “.race” package to just be classes with public static variables but after some experimentation we ran into the issue of inheritance of static variables, and from that we had to revert back to the previous concept.

Refactored Occupation

At first we let the Everbie hold it's own thread that ticked down the occupied time per second, this resulted in that we couldn't access the log class without circular dependencies. To solve this we moved the Occupation thread to it's own class, from where it could tamper with both the Log and Everbie. However we later discovered that the time does not tick down if the hardware's display is turned off and the hardware is unplugged from a computer. In order to solve this issue we first needed to do extensive testing to isolate the problem, being made extremely annoying due to the fact that we lost access to the LogCat when the android device was disconnected from the computer. So after a long time spent doing manual tests and rubber duck-programming, we solved the problem by saving the currentTime at the start of the "occupation"(work or training) and check this value against the currentTime upon resume(). Though further refactoring, implementing a database, opened up the possibility to save the startTime upon every save and when loading just check the same way against the currentTime. Also in the midst of these changes we moved the "Occupation"-class into Use.java to make the filesystem a little more slim-looking.

Graphics and artwork

Creating and editing art is always a time consuming process, this is also true for the process of finding images to use for all evolution steps for each race and for the backgrounds in the menus. all in all this has cost us somewhere around 10-15 hours since we first made a full set of images that we later replaced with images we found online. these images were edited by our graphics designer. The images were scaled, in some cases redrawn and all images were given alpha channels.

Files and Documentation

Regarding the miscellaneous documents that are to be handed in alongside the code, our group feels that the course has been a little too unorganised in terms of informing the students of what's required of them. For instance, we were sure that RAD and SDD only had to do with the groups that did waterfall. Though the same day as everything was supposed to be handed in, we found a speadsheet displaying what scrumming and waterfall groups were to do. And to our suprise, there were an SDD there. And we never really got a good explanation of what the Project Plan was supposed to contain.

Backlog

We didn’t have time to finish all of the backlog. But we came a long way and we are in the end happy with our progress. We also knew from the day we that we planned the backlog that we weren’t going to be able to complete everything but that we would try to do as much as possible. But what we didn’t know was how much extra time some parts of the backlog would take. Some things took extremely much longer time than expected and we had to spend much more time than we had originally predicted. Thereby our planning of some weeks were almost completely destroyed and we had to put more efforts on bugfixing and regaining lost hours.