Team Juventus

Banking System Phase VI - Integration Defect Report

Course Code: CSCI 3060/SOFE 3980U - Software Quality Assurance
Course Instructor: Dr. Jeremy Bradbury

Members:
Dominick Mancini - **100517944**
Mohamad Vedut - **100535400**
Alexander Wheadon - **100514985**

Report Due Date: **April 11th, 2016**

Introduction

For this phase of the project, all teams/groups were requested to make significant changes to who they are working with. This involved moving onto individuals whom you've never worked with before as per required for this phase. Each new group was requested to adopt and combine the Bank Account Front End from one group member and the the Back End from another member.

The frontend for this phase was largely the product off all three members putting their frontends together into one that worked, as all three were missing varying degrees of functionality.

The backend, on the other hand was taken from that created group Polandball without much modification required at all.

Purpose

We required to create a daily script of our application that does many major features. Our daily script consists of running our Front End over a number of Bank transaction sessions and saving the output Bank Account Transactions File for each session in a separate file. Our application should then concatenate in a separate Bank Account Transaction File and be able to merge the Bank Account Transaction File into a Merged one. The Daily script is then running our Back End with the merged Bank Accounts Transaction File as an input.

After this we to develop a Weekly script which runs the Daily script 5 separate times, simulating 5 days and in turn the week. This is a simulation of a full week of the Banking System at work. The different days run transaction sessions, the Current Bank Accounts File as well as the Master Bank Accounts File of each day should be outputs of the previous. The application begins with an empty Current Bank Accounts File and an empty Master one.

Our Solution

We have created a solution what is run through Unix shell scripts where all files are written out and read as text files by the Front End and Back End of our application.

After running our integrated application we have the following Integration Defect Report. This table demonstrates and shows all of the issues that we had over the course of the integration.

Error	Outcome of Error	How Was It Fixed
Front - End	Front-End was not checking if the account was disabled, however the Back End did.	Implementing the checking functionality for disabled accounts.
Front - End	Front-End did not write any transactions for a file or any transaction structs.	Added the functionality of writing any transactions for a file as well as any transaction structs throughout the application
Back- End	Originally the Back-End was merging files to itself, which did not meet the requirements of this phase.	Fixed the merging feature so that the program simply receives a merged file as an argument instead of creating it.

Table 1: This is a report of defects or errors that occurred during integration between the Back End and the Front End. The error was mentioned as well as how the error was fixed to meet the requirements of this phase.

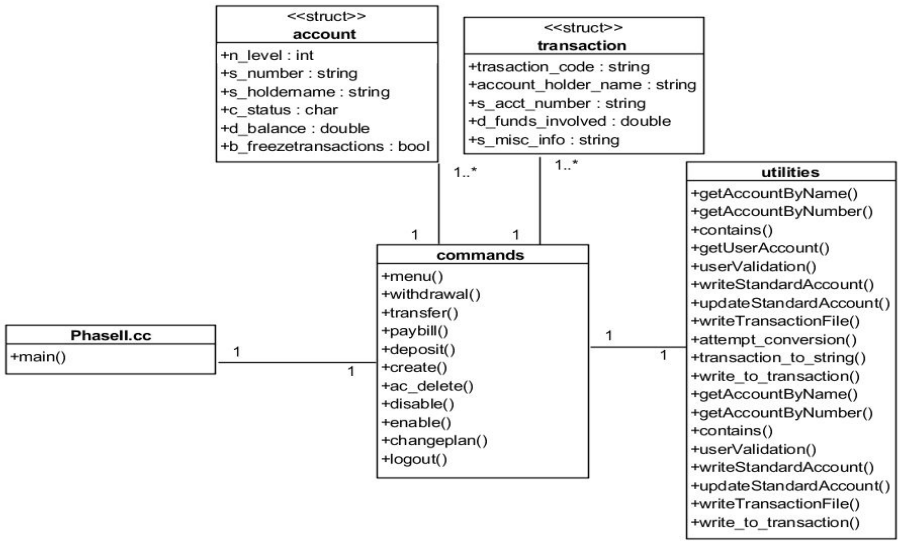

Figure 1: This UML diagram is final structure of our Front End which puts into account transactions, commands and utilities throughout all accounts.

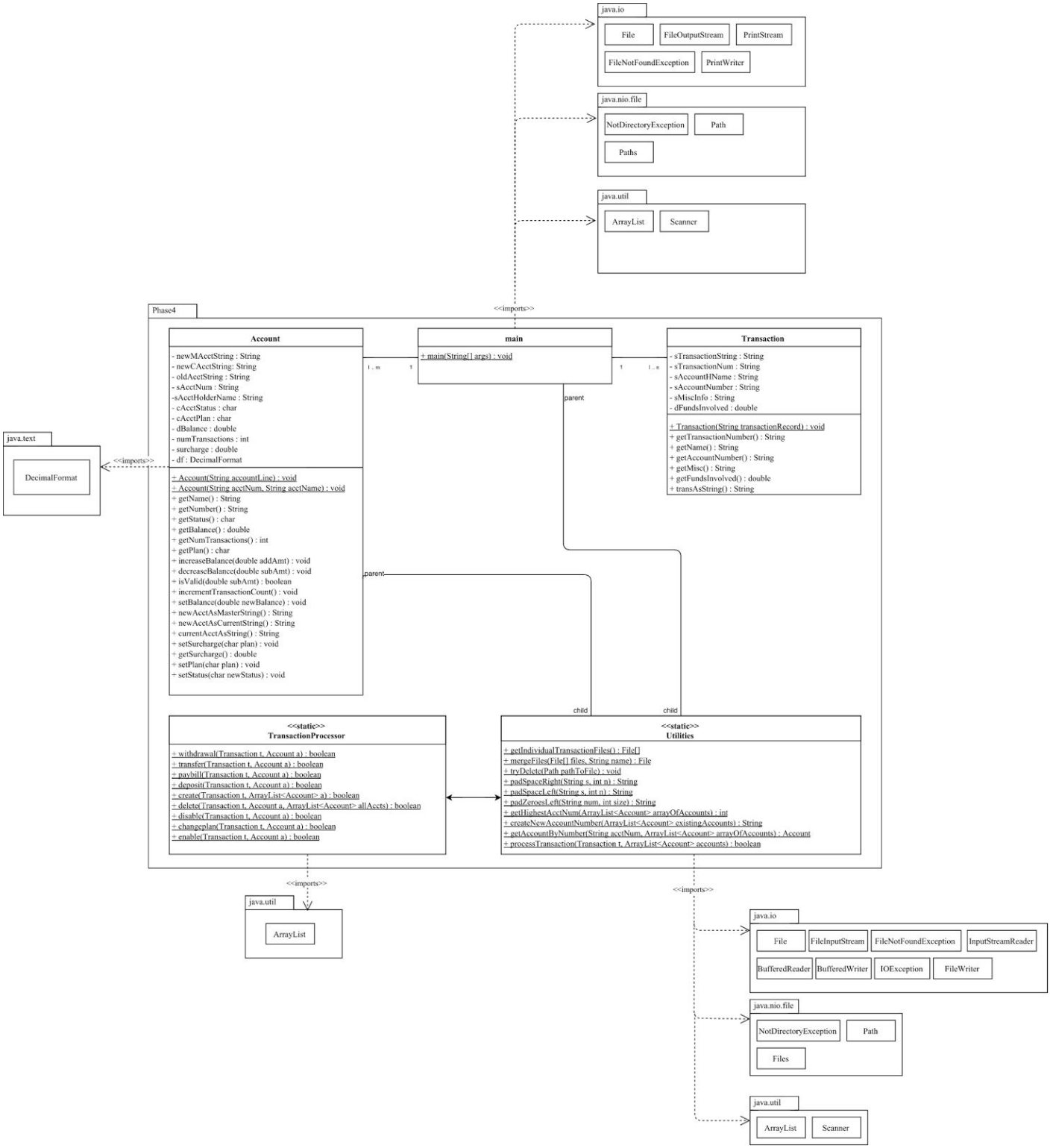

Figure 2: This is the current architecture of Back End which includes Current & Master account transactions, as well as Merged accounts.

Conclusion

In conclusion, our group has successfully adopted an application with a Front End and Back End for all Banking Transactions in all types of accounts required for this project. We have successfully simulated Daily and Weekly transactions working with all types of Day-to-Day Banking Systems.

As part of this phase, we have integrated different Front and Back End code of different group members and delivered a product which meets all the requirements of this application.

As part of this report, you will find Daily and Weekly scripts running that have been implemented according to the specification and handles all the required files. You will also find the Merging feature of the Front End and Back End that runs through Bank Account Transactions and provides them in correct format as specified in the requirements.

Our solution is not only unique and scalable but meets all requirements for all phases throughout the course of this project. We have successfully deployed this application and our findings have been clear throughout this whole process of integration. Software Quality Assurance is an essential part of Software Engineering & Computer Science. It is clear now why this project was not only a handful in terms of requirements but also simulated the real life experience of developing and deploying software while taking into account the testing that is required of any application.