Analysis and Design Document

Student: Barabás Hunor

Group: 30432

Revision History

Date	Version	Description	Author
<04/04/18>	<0.1>	First Iteration	Barabas Hunor

Table of Contents

I. Project Specification

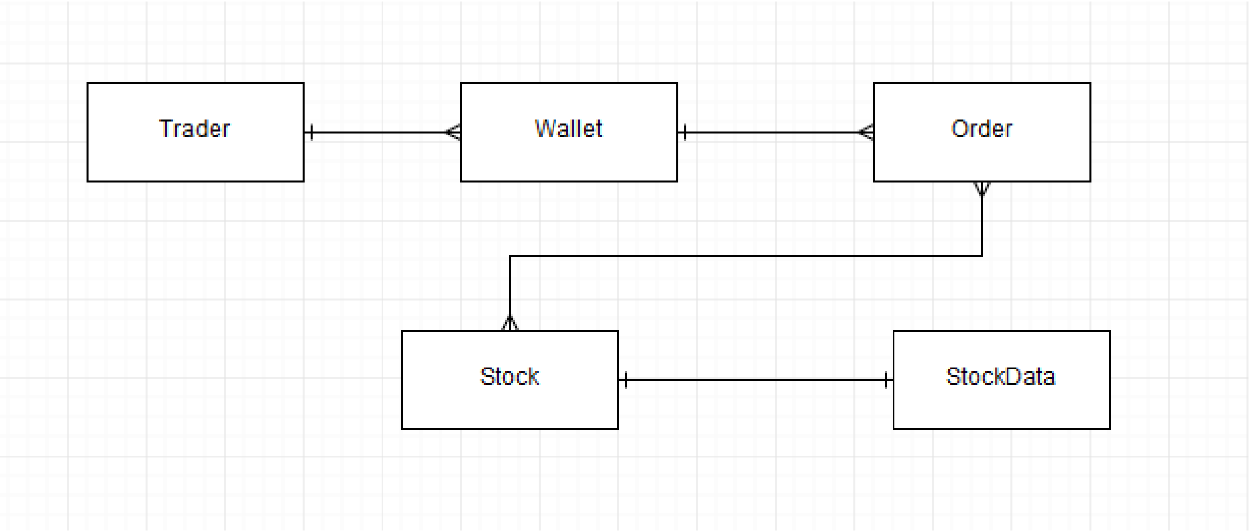
Stock Market Simulator used to teach newbies to trade.

II. Elaboration – Iteration 1.1

Domain Model

The domain is somewhat similar to Assignment 1 in that it contains the models required to differentiate users: I'll have a Login model with information such as username, password and role. I'll have a User Model with basic user information and balance.

The difference will be in that we have the concept of buying and selling here, so I need to model these accordingly. In addition, modelling Prices and important details such as OpenPrice and ClosingPrice are also necessary.



Architectural Design

Conceptual Architecture

For the purpose of this project I will use the Layer Architectural Pattern. If I were to use real-time data this would not be enough in itself, as I would need a mechanism to update my database every second. Thankfully, due to the nature of a simulator, I can use past prices of certain stocks which I will store in a database. The application will access, process and show that data to the user through a traditional 3 layer architecture:

Data Access

Pretty self-explanatory, this layer will have the responsibility of accessing data in our database. That can be anything ranging from prices for a certain stock and certain date, user wallet data, login information etc. This layer will mostly contain the above mentioned models and data access objects for each and every one of them.

Business Logic

Input validation will take place here so to make sure that the data going through fulfills our criterias.

Presentation Layer

The user interface will be built in this layer. Users will be able to buy/sell stocks with imaginary money. They will have the option to follow price charts and export reports which detail their balance changes over the past x days.

MVC

To complement the Layered Architecture I will use the Model-View-Controller design pattern. This is composed of the 3 parts that give it’s name:

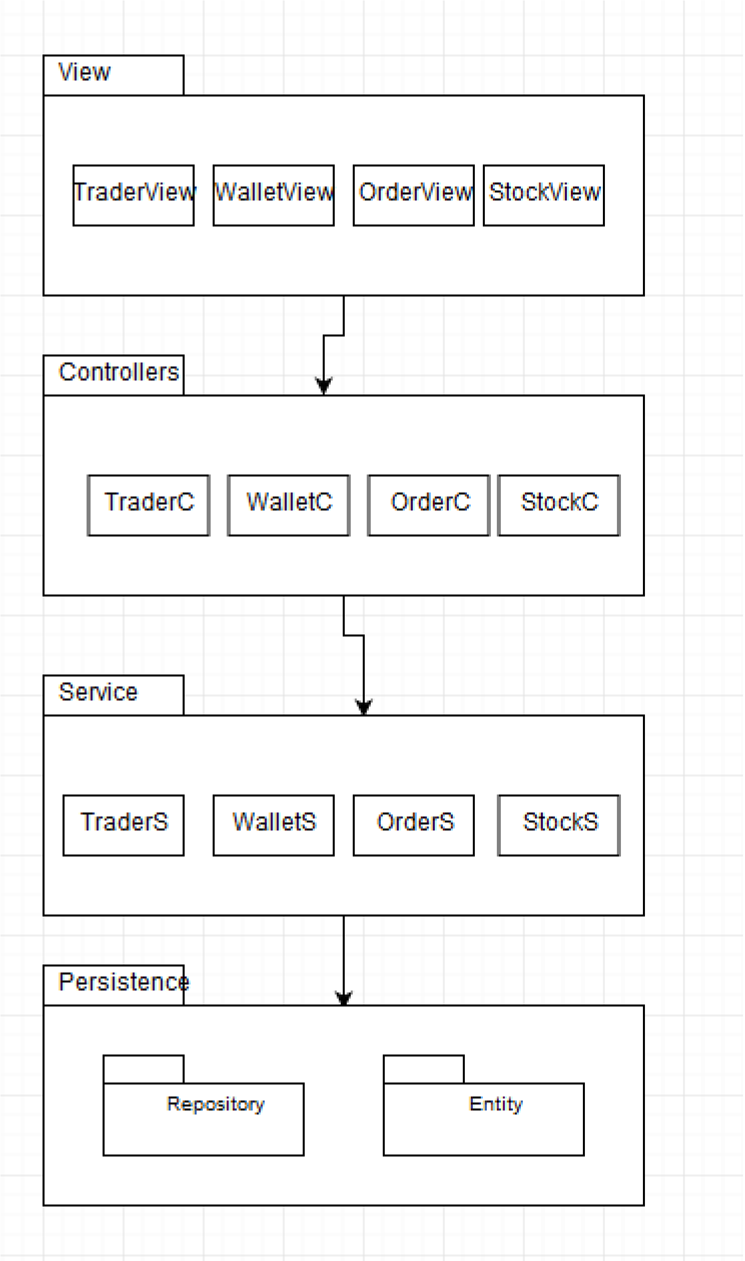
Model – Lowest level, responsible for data maintenance.

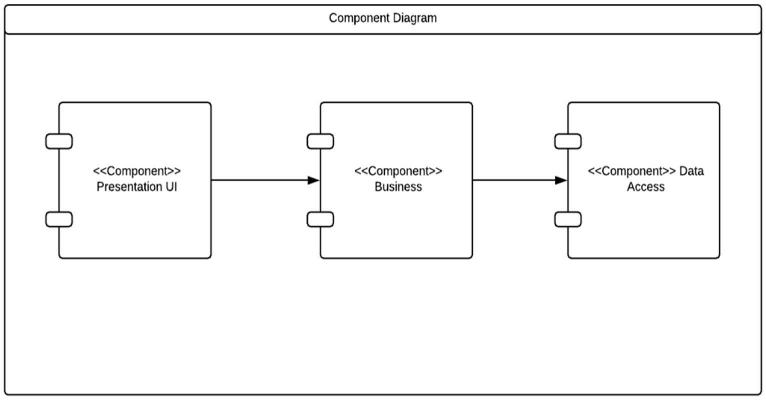
View – Responsible for displaying data on a user interface

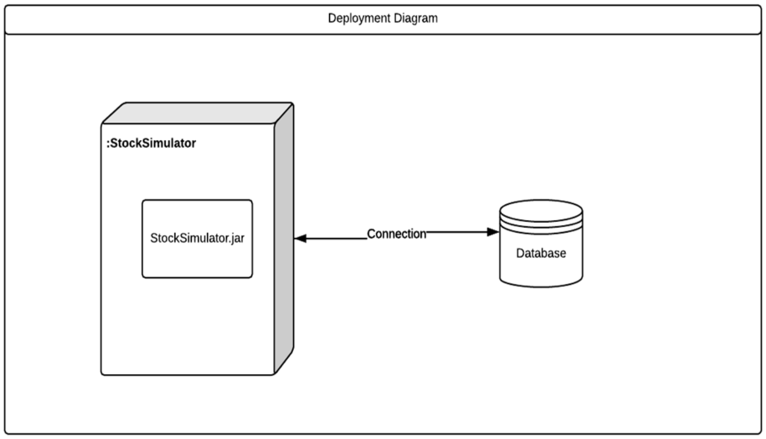
Controller – Controls interactions of the above two.

Package Design

Component and Deployment Diagrams


View
TraderView
WalletView
OrderView
StockView
Controllers
TraderC
WalletC
OrderC
StockC
Service
TraderS
WalletS
OrderS
StockS
Persistence
Repository
Entity

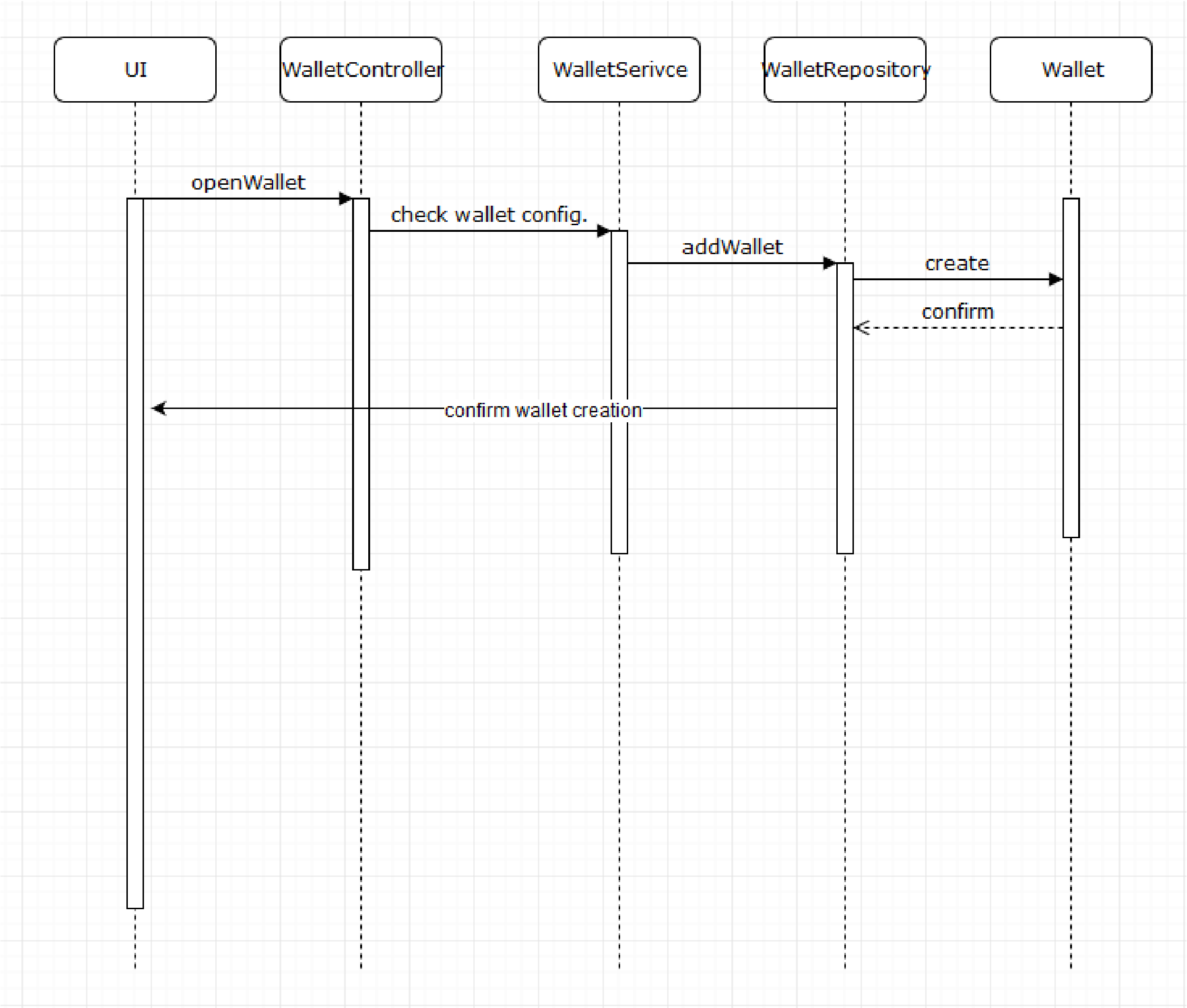

Component Diagram
<<Component>> Presentation UI
<<Component>> Business
<<Component>> Data Access


Deployment Diagram
:StockSimulator
StockSimulator.jar
Connection
Database

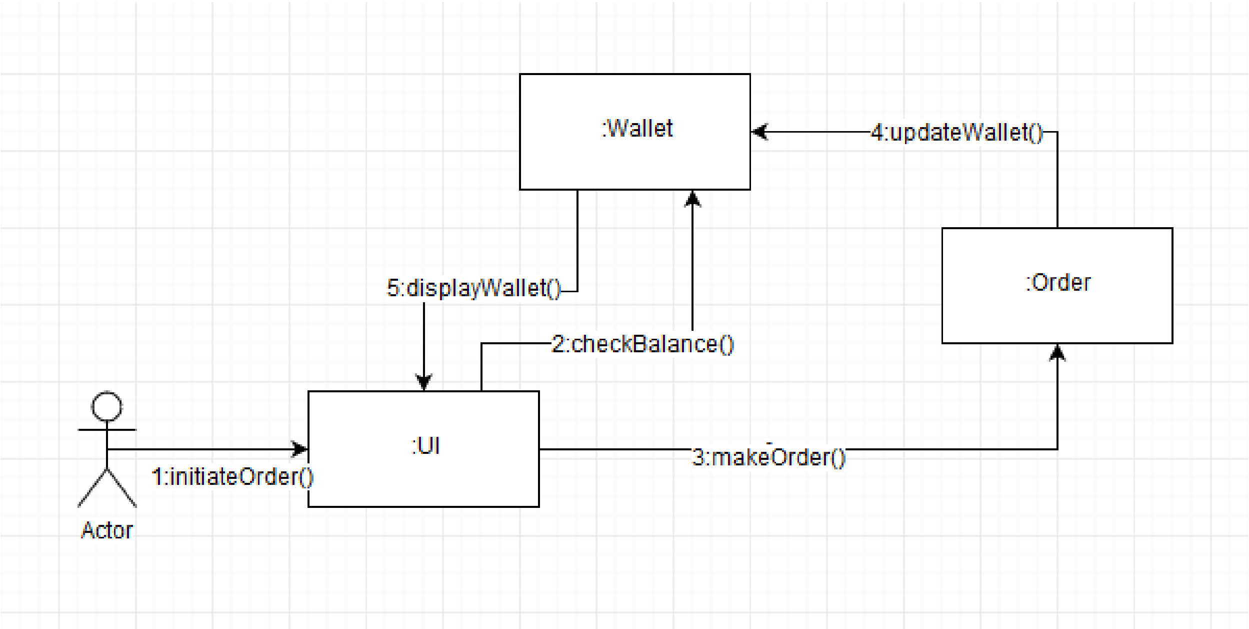
I. Elaboration – Iteration 1.2

1. Design Model

1.1 Dynamic Behavior



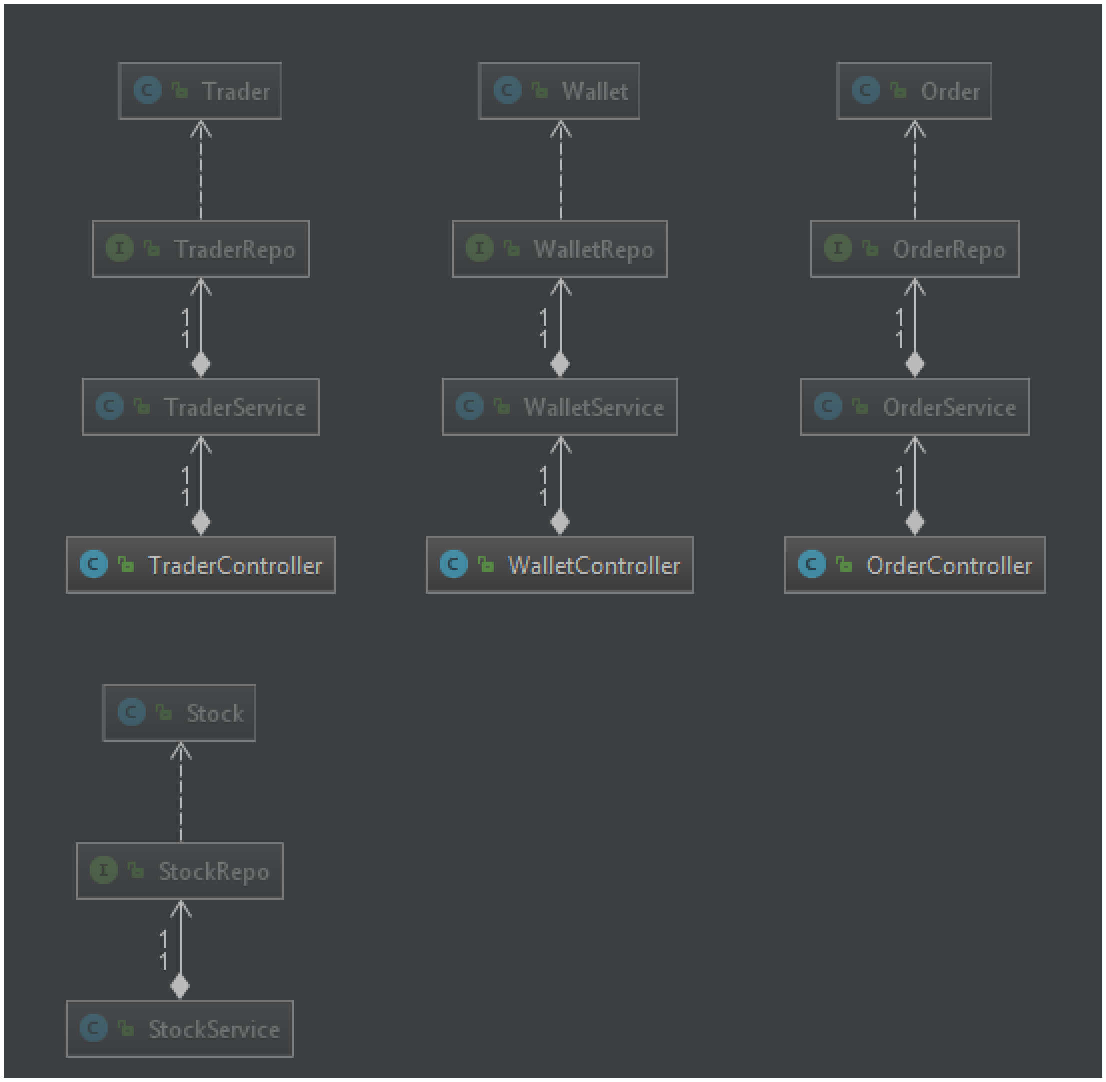
1.1.1 Opening a trading wallet – sequence diagram



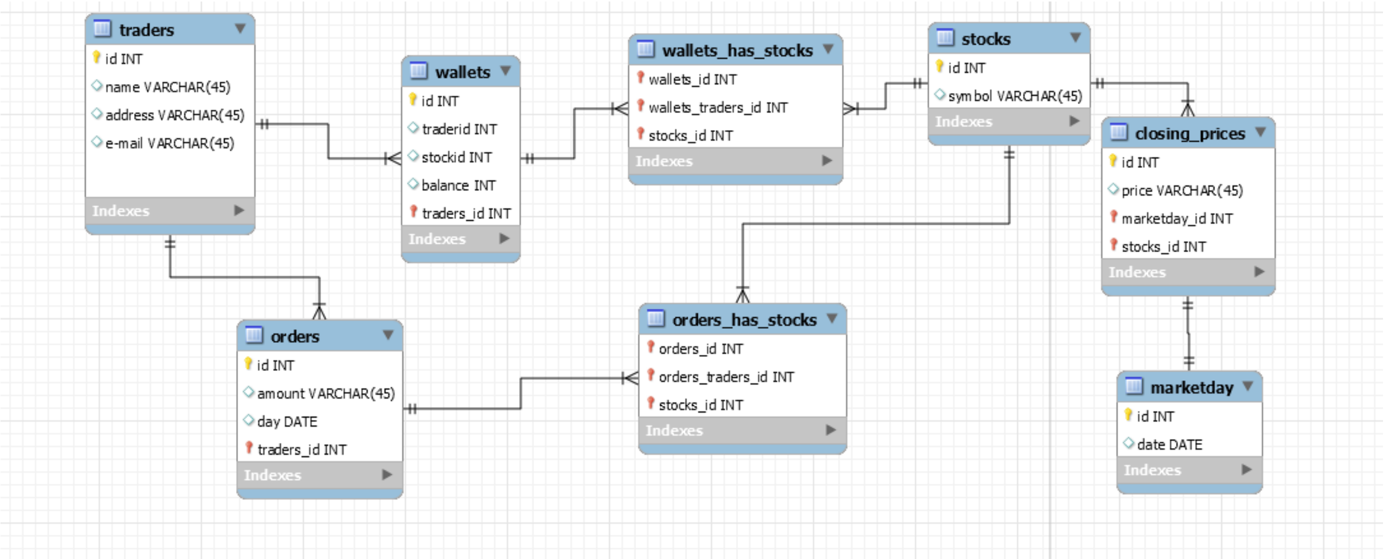
1.1.2 Making an Order

1.2 Class Design

My class design will reflect a traditional Spring MVC class design.


Trader
Wallet
Order
TraderRepo
WalletRepo
OrderRepo
1
1
TraderService
WalletService
OrderService
1
1
TraderController
WalletController
OrderController
Stock
StockRepo
1
1
StockService

2. Data Model


traders
id INT
name VARCHAR(45)
address VARCHAR(45)
e-mail VARCHAR(45)
Indexes
wallets
id INT
traderid INT
stockid INT
balance INT
traders_id INT
Indexes
wallets_has_stocks
wallets_id INT
wallets_traders_id INT
stocks_id INT
Indexes
stocks
id INT
symbol VARCHAR(45)
Indexes
closing_prices
id INT
price VARCHAR(45)
marketday_id INT
stocks_id INT
Indexes
orders
id INT
amount VARCHAR(45)
day DATE
traders_id INT
Indexes
orders_has_stocks
orders_id INT
orders_traders_id INT
stocks_id INT
Indexes
marketday
id INT
date DATE
Indexes

3. Unit Testing

[Present the used testing methods and the associated test case scenarios.]

I. Elaboration – Iteration 2

1. Architectural Design Refinement

[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]

2. Design Model Refinement

[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]

I. Construction and Transition

1. System Testing

[Describe how you applied integration testing and present the associated test case scenarios.]

2. Future improvements

[Present future improvements for the system]

I. Bibliography