

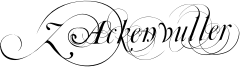
Reactive stock game

Congratulations! You have been selected as one of the contestants for an amazing job offer at our new Faros investment bank! A job that will ensure you an endless supply of pizza, Friday night parties at the bowling alley, a workfloor donkey, and much more.

To get this job, you will have to prove that you are the best of the group. You will be doing this through following assignment. We wish you best of luck, and hope you’ll become part of our amazing development team.

Kind regards,

Zacharias Ackenvuller



## Your assignment

Your assignment is to create a reactive system which will automatically buy, and sell shares of a company. By buying the shares low, and selling them again when they’re high, you can make us a hefty profit.

In order to do this, we have provided you a base project, which has a number of TODO’s in it for you to complete. In the end you will have created an application that uses websocket in a resilient way to receive messages from the server regarding the stock price, then use your program’s internal event bus to propagate this to a service which will decide to buy, sell, or do nothing, and will then use the REST service of the server to do your order.

## The online stock broker server

The online stock broker service offers both a Websocket Rervice, and a REST service.

The Websocket Service is run on two different ports. Be aware that these are not very reliable..at all. During your assignment, you will have to ensure your code is resilient enough to deal with this. Only one port will have a running Websocket service on it at the same time, so you will have to able switch from one to the other automatically.

The REST service is always run on the same port. It offers you a way to register yourself into the system, and to place orders.

Specifics on the server info can be found in your ServerInfo.java class.

## Provided code

The provided code has a number of easy to follow TODO’s to complete. In case you have any questions, don’t mind asking our bank’s CEO’s who will be walking around to answer your questions. However, we are confident that most things should be pretty clear to you.

The code is separated in 3 different parts.

The ClientStart.java class can be used to start the client, and will be the starting point of your assignment as well. You can run it like any regular Java program, Vertx only requires its maveb dependency to run.

The shared package contains all the value objects that can be sent From and To the server. Some of these are nested in a clear way. The TODO’s should be clear on what to use in which situation. These classes should stay as-is.

The client package contains the rest of your TODO’s. It has 3 main components of the system in it, and two files for constants. The use of the two latter ones will become clear during the exercise.

The 3 main components of the system are the WebsocketRouter, the BuyDecisionService, and the RestRouter. The WebsocketRouter will be used to receive messages from the websocket services of the stock broker service. The BuyDecisionService will be used for your business logic which decides on whether to buy, sell, or do nothing based on the current situation. Finally the RestRouter will be used to send REST calls to the provided REST service on the Stock Broker server.

The codec package will be used to be able to put objects on your program’s internal message bus, and can be ignored.