# Super Simple Stock

Overview

As shown in the following diagram, the Super Simple Stock Service has been distributed in three layers. The first layer represents the entities of the service which has an abstract class called Stock and implements the common information exchanged by the Stock entity and two extensions, one for each type of stock (Common and preferred). The extended Stocks implement the dividendYeldFunction since this function has a different behavior depending on the type of stock. The second entity is Trade which represents the transactions per Stock. The next layer is the data access layer; there are two data access objects, StockDao to access Stock data and TradeDao to access transaction data. The last layer is the Service layer with the core functionality.
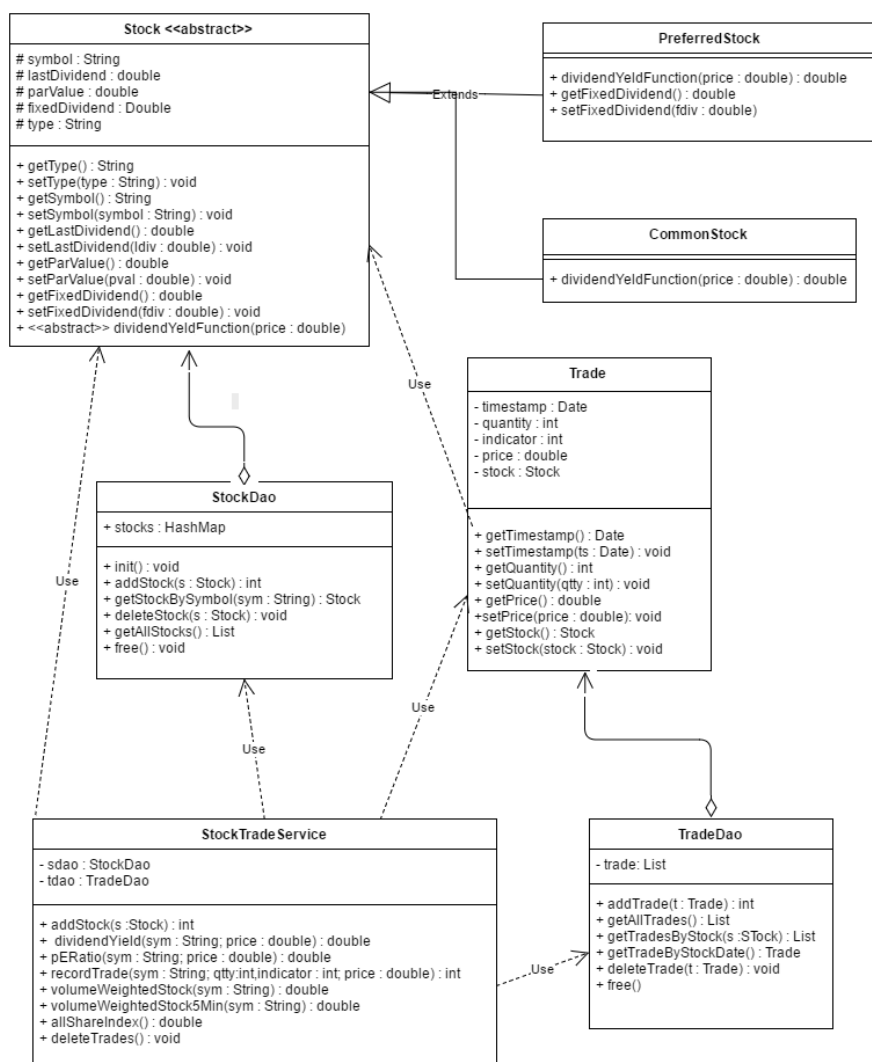
Core Service

The core functionality has been provided as methods of the StockTradeService class as follows:

- **addStock**: Adds a new Stock. This method is used for testing purpose.
- **dividendYeld**: Calls the dividendYeldFunction of a given Stock with a certain price to return the dividend yield for each type of Stock.
- **pERatio**: This methods returns the pERatio of a given Stock and a certain price.
- **recordTrade**: Records a trade given the Stock Symbol and parameters of the Trade.
- **volumeWeightedStock**: Not a part of the core functionality, used by the allShareIndex method. Calculates the Volume Weighted Stock for al trades of a given stock. Returns 1 if no Trades available for the given Stock in order to avoid 0 value of the geometric mean.
- **volumeWeightedStock**: Part of the core functionality. Calculates the Volume Weighted Stock for a given Stock based on past 5 minute trades. Returns 1 if no trades available for the given Stock in order to avoid 0 value of the geometric mean.
- **allShareIndex**: Calculates the GBCE All Share Index based on loaded Stocks and Trade data.
- deleteTrades: Used for testing purpose, deletes all inserted Trades.

Test

As shown in the following image TradeDao has a dependency towards StockDao and StockTradeService has a dependency towards StockDao and TradeDao. Based on the dependency graph the first module to be tested is StokcDao followed by TradeDao and StockTradeService.
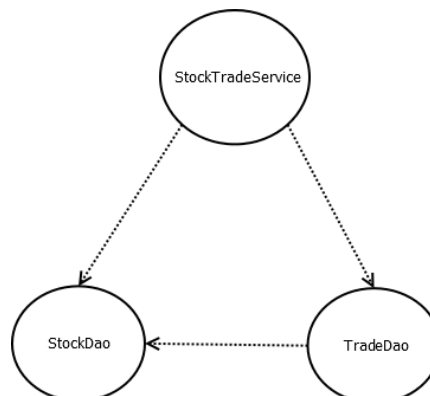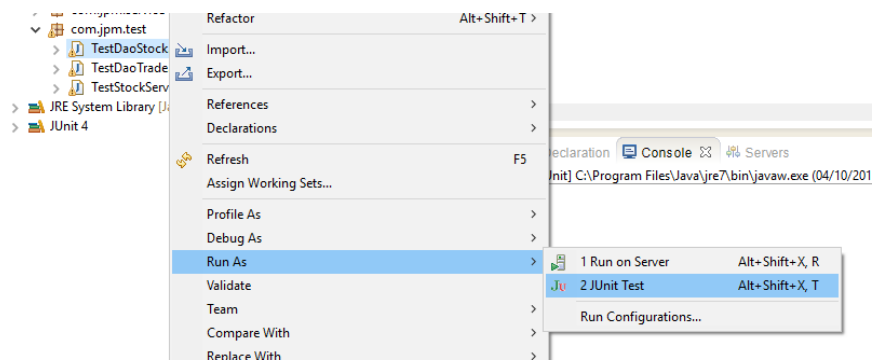
Finished after 0,039 seconds

Runs: 11/11   ⊠ Errors: 0   ⊠ Failures: 0

∨ com.jpm.test.TestStockService [Runner: JUnit 4] (0,00
      testVWSP5Min (0,000 s)
      testVWSP (0,000 s)
      testAllShareIndex (0,000 s)
      testDividendYeldCommonErr (0,000 s)
      testDividendYeldPreferred (0,000 s)
      testVWSPNoTrades (0,000 s)
      testPERCommonErr (0,000 s)
      testPERPreferred (0,000 s)
      testDividendYeldPreferredErr (0,000 s)
      testPERCommon (0,000 s)
      testDividendYeldCommon (0,000 s)
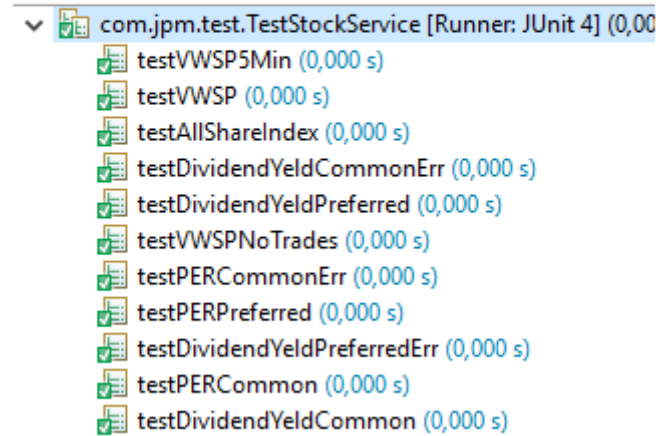
TestDaoStock

Test Case	Test flow
testInsertCommon	Inserts a Common Stock and then tests whether it exists.
testInsertedPreferred	Inserts a Preferred Stock and then tests whether it exists.
testAllStocks	Inserts 3 Stocks and checks the size the List returned by getAllStocks
testDeleteStocks	Inserts 3 Stocks deletes 1 Stock and checks the size of the List returned by getAllStocks

TestDaoTrade

Test Case	Test flow
testInsertTradeCommon	Inserts a Trade associated to a Common Stock and checks whether it exists
testInsertTradePreferred	Inserts a Trade associated to a Common Stock and checks whether it exists
testAllTrades	Inserts 3 different Stocks and checks the size the List returned by getAllTrades
testGetTradesByStock	Inserts 4 trades 2 of them associated to the

	"POP" stock and checks the size of the List returned by getTradesByStock
testDeleteTrades	Inserts 3 Trades deletes 1 and checks the size of the List returned by getAllTrades

TestStockService

The Setup function inserts the Stocks given as examples

Test Case	Test flow
testDividendYeldCommon	Tests the dividendYield method for "POP" and price=4 expected output=2
testDividendYeldCommonErr	Tests the dividendYield method for "POP" and price=0 expected output=StockTradeService.ERR
testDividendYeldPreferred	Tests the dividendYield method for "GIN" and price=2 expected output=1
testDividendYeldPreferredErr	Tests the dividendYield method for "GIN" and price=0 expected output=StockTradeService.ERR
testPERCommonErr	Tests the pEratio method for "TEA" and price=2 expected output=StockTradeService.ERR
testPERCommon	Tests the pEratio method for "POP" and price=2 expected output=1/4
testPERPreferred	Tests the pEratio method for "GIN" and price=2 expected output=1/4
testVWSP	Checks the Volume Weighted Stock. Inserts 2 Trades For "TEA" with qtty=20 and price=10 expected output=10
testVWSPNoTrade	Checks the Volume Weighted Stock. No trades expected output=1
testVWSP5Min	Checks the Volume Weighted Stock last 5 minutes. Inserts 2 Trades For "ALE" with

	qtty=20 and price=10 expected output=10
testAllShareIndex	Checks the GBCE All Share Index. Inserts 2 Trades For each Stock with qtty=20 and price=10 expected output=1000000^1/5