

C++ Stock Market

Testarea Sistemelor Software

Alexandru-Christian Codarcea

Florin-Alexandru Gavrilă

Eduard-Valentin Dumitrescul

Prezentare Generală

Acest proiect reprezintă dezvoltarea unei aplicații simple de trading

Funcționalități principale:

- adăugare fonduri
 - plasare de ordine de piață
 - executarea automată a ordinelor
-

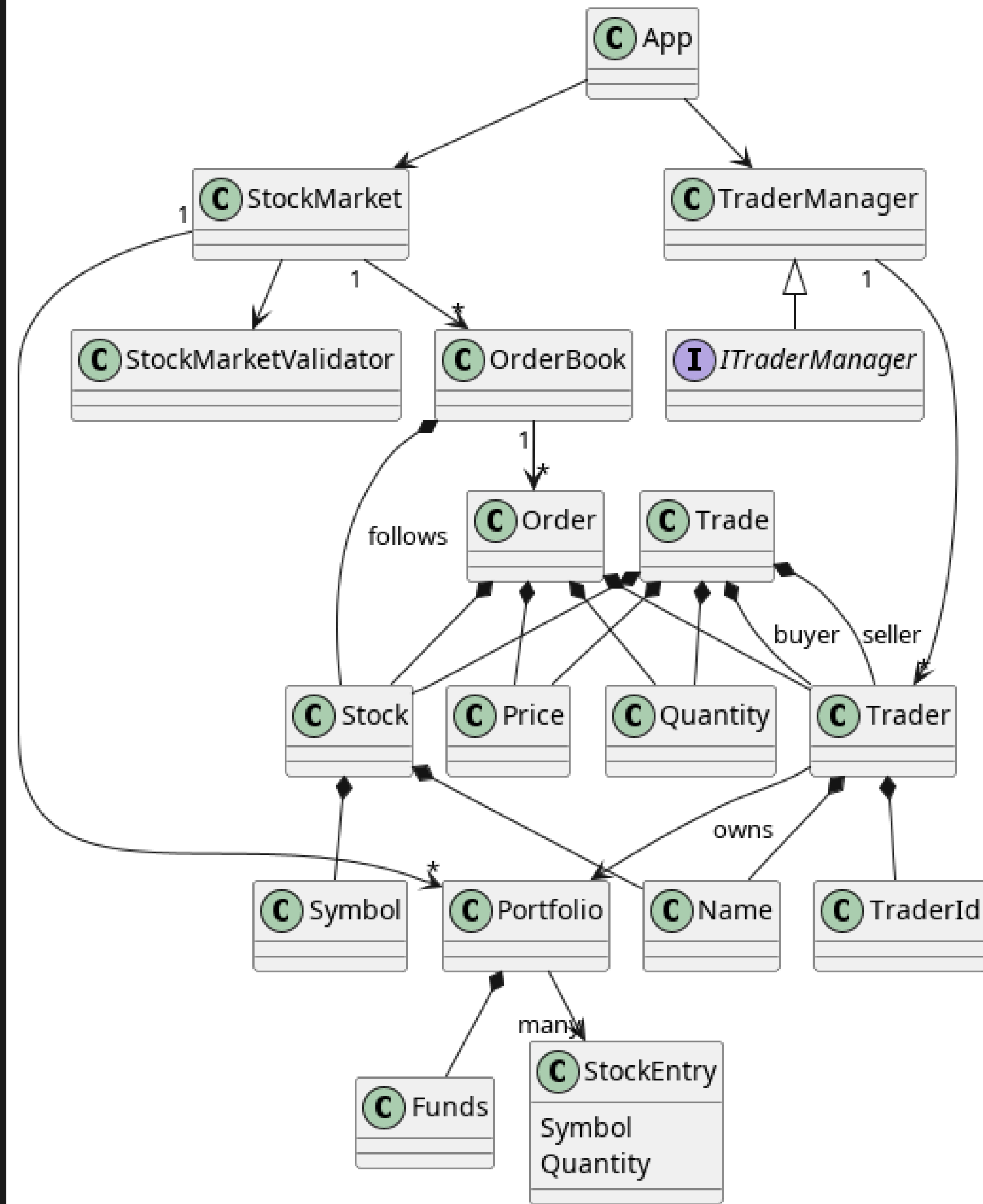
Testare atentă

- testare unitara
- testare de integrare
- testare end-to-end
- mutanți

Arhitectură

Principii POO:

- encapsulare
- abstractizare
- moștenire
- polimorfism



Configurație Software

C++ 20

limbaj de programare

CMake >= 3.30

sistem de build

googletest 1.14.0

framework pentru testare

Windows 11, Ubuntu 24.04, EndeavourOS 6.13

sisteme de operare

Strategii de Testare

01

Testare unitară

Pentru constrângeri stabilite la început și funcționalități simple:

- fondurile sunt mereu între 0 și un miliard
- menținerea ordinelor în ordine corespunzător tipului
- precum și alte scenarii

02

Testare de integrare

Pentru verificarea cooperării dintre componente:

- plasarea unui ordin (StockMarket, Portfolio)
- înregistrarea unui utilizator într-o piață de acțiuni (Trader, StockMarket, Portfolio)

03

Testare end-to-end

Pentru testarea unui întreg flux de acțiuni:

- adăugare fonduri
- plasare ordin de tip sell
- plasare ordin de tip buy
- verificare executare

✓ FundsTest.FundsAreNotNegative	10 ms
✓ FundsTest.FundsAreAtMostABillion	10 ms
✓ TraderManagerTest.Insert	10 ms
✓ TraderManagerTest.InsertTradersWithSameNameError	10 ms
✓ OrderBookTest.AddBuyOrder	10 ms
✓ OrderBookTest.AddBuyOrderMantainOrder	10 ms
✓ OrderBookTest.AddSellOrder	10 ms
✓ OrderBookTest.AddSellOrderMantainOrder	10 ms
✓ OrderBookTest.AddBuyOrderDifferentStock	10 ms
✓ OrderBookTest.AddSellOrderDifferentStock	10 ms
✓ OrderBookTest.FullOrderMatching	10 ms
✓ OrderBookTest.PartialOrderMatching	10 ms
✓ OrderBookTest.OrderMatchingReturnsTrades	10 ms
✓ OrderBookTest.FullMatchReturnsCorrectTrade	10 ms
✓ OrderBookTest.MultipleMatchesInOneGo	10 ms
✓ OrderBookTest.EmptyOrderBookThrowsOnBestOrderAccess	10 ms
✓ PriceTest.PricelsNotNegative	10 ms
✓ PriceTest.ReturnDecimalFormat	10 ms
✓ TraderTest.DepositFunds	10 ms
✓ TraderTest.WithdrawFunds	10 ms
✓ TraderTest.WithdrawFunds_ErrorOnOver	10 ms
✓ PortfolioTest.DepositFunds	10 ms
✓ PortfolioTest.WithdrawFunds	10 ms
✓ PortfolioTest.WithdrawFunds_ErrorOnOver	10 ms
✓ PortfolioTest.AddStock	10 ms
✓ PortfolioTest.AddStock_MultipleTimes	10 ms
✓ PortfolioTest.RemoveStock	10 ms
✓ PortfolioTest.RemoveStock_Overdraft	10 ms
✓ OrderTest.OrderSuccessfullyCreated	10 ms
✓ TraderIdTest.Initialization	10 ms
✓ StockMarketInitializationTest.RegisterStock	10 ms
✓ StockMarketInitializationTest.RegisterSameStock_Error	10 ms
✓ StockMarketInitializationTest.GetInexistentOrderBook_Error	10 ms
✓ StockMarketInitializationTest.RegisterTrader	10 ms
✓ StockMarketInitializationTest.RegisterSameTrader_Error	10 ms
✓ StockMarketInitializationTest.GetInexistentPortfolio_Error	10 ms

Testare Riguroasă

Proces în 3 pași:

- adăugare de teste semnificative
- implementarea funcționalității
- rularea testelor și repararea eventualelor erori

O suită de 44 de teste menite să detecteze erorile din implementare

- ✗ TraderTest.WithdrawFunds
- ✓ TraderTest.WithdrawFunds_ErrorOnOver
- ✓ PortfolioTest.DepositFunds
- ✗ PortfolioTest.WithdrawFunds
- ✓ PortfolioTest.WithdrawFunds_ErrorOnOver
- ✓ PortfolioTest.AddStock
- ✓ PortfolioTest.AddStock_MultipleTimes
- ✓ PortfolioTest.RemoveStock
- ✓ PortfolioTest.RemoveStock_Overdraft
- ✓ OrderTest.OrderSuccessfullyCreated
- ✓ TraderTest.Initialization
- ✓ StockMarketInitializationTest.RegisterStock
- ✓ StockMarketInitializationTest.RegisterSameStock
- ✓ StockMarketInitializationTest.GetInexistentOrder
- ✓ StockMarketInitializationTest.RegisterTrader
- ✓ StockMarketInitializationTest.RegisterSameTrader
- ✓ StockMarketInitializationTest.GetInexistentPortfolio
- ✗ StockMarketInitializationTest.PlaceOrders
- ✗ AppTest.End2EndTest

Generare de Mutanți

Script pentru generarea mutanților

main_mutants.cpp

Rularea testelor pe codul modificat

și observarea rezultatului

Îmbunătățirea suitei de teste

În momentul în care mutanții trec toate testele

Analiza Testelor

Efecuată cu ajutorul tool-ului integrat implicit in IDE-ul CLion.

Acoperirea codului 97%

Acoperirea liniilor 97%

Acoperirea ramurilor 49%

Coverage All CTest x	
Element ^	
✓ tss-cpp-stock-market	97% files, 97% lines c
src	96% files, 97% lines c
app	100% files, 100% lines
order	100% files, 85% lines
orderBook	100% files, 99% lines
portfolio	66% files, 97% lines c
stock	100% files, 91% lines c
stockMarket	100% files, 97% lines c
services	100% files, 95% lines
validators	100% files, 100% lines
stock_market_test.cc	100% lines covered
StockMarket.cpp	93% lines covered
trade	100% files, 100% lines
trader	100% files, 94% lines
manager	100% files, 97% lines c
traderId	100% files, 100% lines
Trader.cpp	81% lines covered
trader_test.cc	100% lines covered
types	100% files, 98% lines
hello_test.cc	100% lines covered

Demo

The screenshot displays the Visual Studio Code interface for a C++ project named "demo tss". The left sidebar shows the project structure with folders like "stockMarket", "services", "validators", "trade", "trader", "manager", "traderId", and "types". The main editor shows the "StockMarket.h" file with C++ code. The right sidebar displays the "Coverage" view, showing line and branch coverage for various files. The bottom panel shows the "Test Results" view, indicating that 44 tests passed out of 44 tests, with a total execution time of 690ms. A red YouTube play button is overlaid on the code editor.

demo tss

- stockMarket: 100% files, 97% lines covered
- services: 100% files, 95% lines covered
 - TradeSettlementService.cpp: 95% lines covered
 - TradeSettlementService.h
- validators: 100% files, 100% lines covered
 - StockMarketValidator.cpp: 100% lines covered
 - StockMarketValidator.h
- stock_market_test.cc: 100% lines covered
- StockMarket.cpp: 93% lines covered
- StockMarket.h
- trade: 100% files, 100% lines covered
 - Trade.cpp: 100% lines covered
 - Trade.h
- trader: 100% files, 94% lines covered
 - manager: 100% files, 97% lines covered
 - TraderManager.h: 100% lines covered
 - trader_manager_test.cc: 100% lines covered
 - TraderManager.cpp: 95% lines covered
 - TraderManager.h
 - traderId: 100% files, 100% lines covered
 - Trader.cpp: 81% lines covered
 - Trader.h
 - trader_test.cc: 100% lines covered
- types: 100% files, 98% lines covered

CMakeLists.txt

```
#ifndef STOCKMARKET_H
#define STOCKMARKET_H

#include "../orderBook/OrderBook.h"
#include "../portfolio/Portfolio.h"

class StockMarket {
    std::unordered_map<Symbol, std::shared_ptr<OrderBook>> orderBooks;
    std::unordered_map<TraderId, std::shared_ptr<Portfolio>> portfolios;

    friend class StockMarketValidator;

public:
    std::vector<Trade> placeBuyOrder(const Order &order) const;
    std::vector<Trade> placeSellOrder(const Order &order) const;

    void registerStock(const Symbol &symbol, const std::shared_ptr<OrderBook> &orderBook);
    void registerTrader(const TraderId &traderId, const std::shared_ptr<Trader> &trader);

    std::shared_ptr<const OrderBook> getOrderBook(const Stock &stock) const;
    std::shared_ptr<const Portfolio> getPortfolio(TraderId traderId) const;
};
```

Coverage

Element	Line Coverage, %	Branch Coverage, %
tss-cpp-stock-market	97% files, 97% lines covered	49% branches covered
src	96% files, 97% lines covered	48% branches covered
app	100% files, 100% lines covered	50% branches covered
order	100% files, 88% lines covered	46% branches covered
orderBook	100% files, 98% lines covered	51% branches covered
portfolio	68% files, 97% lines covered	51% branches covered
stock	100% files, 91% lines covered	62% branches covered
stockMarket	100% files, 97% lines covered	51% branches covered
trade	100% files, 100% lines covered	50% branches covered
trader	100% files, 94% lines covered	51% branches covered
types	100% files, 98% lines covered	44% branches covered
hello_test.cc	100% lines covered	50% branches covered

Test Results

✓ 44 tests passed: 44 tests total, 690ms

Test project /home/eduard/work/tss-cpp-stock-market/cmake-build-debug-coverage

Constructing a list of tests

Done constructing a list of tests

Updating test list for fixtures

Added 0 tests to meet fixture requirements

Parallel test dependencies resolved

Watch on YouTube