CSE 221
Object Oriented Programming
Chess Project

Design

Berk Kaya

Introduction

This document serves as a documentation regarding the design of my term project relating to a chess game which contains both traditional and fantasy elements. The scope of the project is rather large due to the requirements given in the project definition which includes unit testing, configuration loading, edge cases of chess logic - especially when combined with fantasy elements - and extensibility.

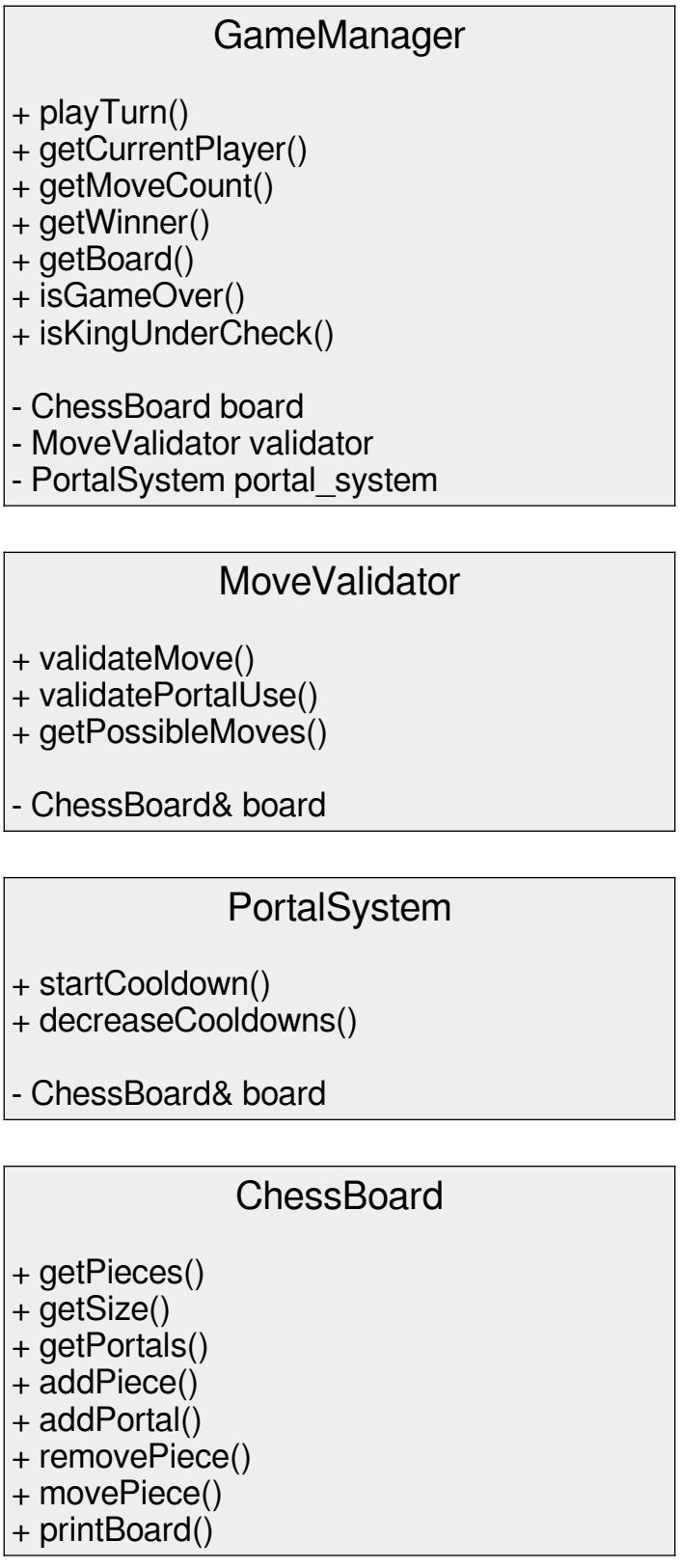
Project Structure

The project is structured in a quite mainstream way. Implementation files are under `src/`, public header files are under `include/`, test files are under `test/`, and configuration files are under `data/`. Lastly, terminal user interface based implementation is under `tui/`, to separate chess logic and user interface concerns. This has the additional benefit of making it easier to write a `Makefile` which produces both a shared library for chess, and an executable which can use it to play chess.

```
.
├── data
│   ├── chess_nopawn.json
│   ├── chess_pieces.json
│   └── fantasy_chess.json
├── include
│   ├── ChessBoard.hpp
│   ├── ChessPiece.hpp
│   ├── ConfigReader.hpp
│   ├── GameManager.hpp
│   ├── MoveValidator.hpp
│   ├── Portal.hpp
│   └── PortalSystem.hpp
├── instructions.pdf
├── Makefile
├── README.md
├── src
│   ├── ChessBoard.cpp
│   ├── ConfigReader.cpp
│   ├── GameManager.cpp
│   ├── MoveValidator.cpp
│   └── PortalSystem.cpp
├── test
│   ├── ChessBoardTest.cpp
│   ├── GameManagerTest.cpp
│   ├── MoveValidatorTest.cpp
│   ├── PortalSystemTest.cpp
│   └── TestRunner.cpp
├── third_party
│   ├── nlohmann
│   └── Unity
└── tui
    └── main.cpp
```

Class Hierarchy

The classes do not extend from each other, and composition over inheritance principles are applied. The main handler of the chess logic is `GameManager`, which communicates with classes `MoveValidator`, `PortalSystem`, and `ChessBoard`.



User Interface

Here is an example of a chess game played in a terminal user interface environment.

```
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 |.r.|.k.|.b.|.q.|.K.|.b.|.k.|.r.|
  +---+---+---+---+---+---+---+---+
7 |.p.|.p.|.p.|.p.|.p.|.p.|.p.|.p.|
  +---+---+---+---+---+---+---+---+
6 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+
5 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+
4 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+
3 |   |   |   |   |   |   |   |   |
  +===+===+===+===+===+===+===+===+
2 I^p^I^p^I^p^I^p^I^p^I^p^I^p^I^p^I
  +===+===+===+===+===+===+===+===+
1 I^r^I^k^I^b^I^q^I^K^I^b^I^k^I^r^I
  +===+===+===+===+===+===+===+===+

=== Move 1 ===
Turn: White
Select piece (cN): f2

    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 |.r.|.k.|.b.|.q.|.K.|.b.|.k.|.r.|
  +---+---+---+---+---+---+---+---+
7 |.p.|.p.|.p.|.p.|.p.|.p.|.p.|.p.|
  +---+---+---+---+---+---+---+---+
6 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+
5 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+===+---+---+
4 |   |   |   |   |   I   I   |   |
  +---+---+---+---+---+===+---+---+
3 |   |   |   |   |   I   I   |   |
  +---+---+---+---+---+===+---+---+
2 |^p^|^p^|^p^|^p^|^p^I^p^I^p^|^p^|
  +---+---+---+---+---+===+---+---+
1 |^r^|^k^|^b^|^q^|^K^|^b^|^k^|^r^|
  +---+---+---+---+---+---+---+---+

=== Selected pawn ===
Possible Moves: f3 f4
Select destination (cN): f3

    a   b   c   d   e   f   g   h
  +===+===+===+===+===+===+===+===+
8 I.r.I.k.I.b.I.q.I.K.I.b.I.k.I.r.I
  +===+===+===+===+===+===+===+===+
7 I.p.I.p.I.p.I.p.I.p.I.p.I.p.I.p.I
  +===+===+===+===+===+===+===+===+
6 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+
5 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+
4 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+
3 |   |   |   |   |   |^p^|   |   |
  +---+---+---+---+---+---+---+---+
2 |^p^|^p^|^p^|^p^|^p^|   |^p^|^p^|
  +---+---+---+---+---+---+---+---+
1 |^r^|^k^|^b^|^q^|^K^|^b^|^k^|^r^|
  +---+---+---+---+---+---+---+---+

=== Move 2 ===
...
```

Test Suite

Using Unity Test as a framework, a test suite was set up in line with project requirements to ensure stability and conformance to chess rules. The test suite is composed of 17 unit tests and 3 integration tests.

```
Unity test run 1 of 1
TEST(ChessBoard, BoardInitialization) PASS
TEST(ChessBoard, MovePiece) PASS
TEST(ChessBoard, RemovePiece) PASS
TEST(ChessBoard, AddPiece) PASS
TEST(ChessBoard, AddPortal) PASS
TEST(ChessBoard, GetPieceAtPosition) PASS
TEST(ChessBoard, GetKingOfTeam) PASS
TEST(ChessBoard, ExchangePiecePositions) PASS
TEST(MoveValidator, PossibleMoves) PASS
TEST(MoveValidator, ValidateMove) PASS
TEST(MoveValidator, ValidatePortalUse) PASS
TEST(PortalSystem, InitializePortals) PASS
TEST(PortalSystem, StartCooldown) PASS
TEST(PortalSystem, DecreaseCooldowns) PASS
TEST(GameManager, PlayTurn) PASS
TEST(GameManager, IsKingUnderCheck) PASS
TEST(GameManager, PlayTurnReverseOnCheck) PASS
TEST(GameManager, FoolsMate) PASS
TEST(GameManager, ScholarsMate) PASS
TEST(GameManager, Stalemate) PASS

-----------------------
20 Tests 0 Failures 0 Ignored
OK
```

The integration tests play three distinct chess games, Fool's Mate, Scholar's Mate, and a short stalemate, exercising various aspects of chess and making sure every rule is observed. The other tests make sure functions work as designed.

Conclusion

The project was designed by paying attention to modern C++ principles, unit testing, and separation of concerns. The result of this is a product which is easy to maintain, study, and develop.