Software Engineering GP02 Project

Design Specification

|  |  |
| --- | --- |
| Authors: | Jack Book [jab153] |
| Config Ref: | ? |
| Date: | 27th February 2023 |
| Version: | 0.1 |
| Status: | Draft |

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Copyright © Aberystwyth University 2023

CONTENTS

CONTENTS 2

1. Introduction 3

1.1 Purpose of this Document 3

1.2 Scope 3

1.3 Objectives 3

2. Decompisition description 3

2.1 Programs in System 3

2.2 Significant Classes 3

2.3 Modules shared between programs 3

2.4 Table mapping requirements onto classes 3

3. dependency description 4

4. interface description 4

5. detailed design 4

REFERENCES 4

DOCUMENT HISTORY 5

# Introduction

## Purpose of this Document

The purpose of this document is to give insight into how the game will be designed. It will describe the decomposition of the programs and the modules that fall within them, including any significant classes and how the classes overlap with the functional requirements. It will also describe the relationships and dependencies the modules have between each other. As well as, providing a description of the interfaces used and how any programmers or testers will use the facilities provided by a module. Finally, it will lay out the internal details of any modules that are not obvious to any reader.

## Scope

The document should be read by all members of the project and any readers should have a good understanding of the User Interface Specification document.

## Objectives

The objective of this document is to allow the reader to understand the inner mechanisms of the classes used in the project and go into further detail of how more complicated classes work.

# Decompisition description

## Programs in System

## Significant Classes

The program is broken down into separate packages, each package containing the relevant classes and material needed to run the program. Each of the sections below will describe the packages and briefly explain the classes used.

### Game

The game package contains four classes – Board, Game, Player, and Square, these are used to build the game.

**Board** - creates the chessboard and for testing purposes is used to print the array to the console.

**Game -** collects the relevant information from other classes and holds the settings for the current game in progress.

**Player** - collects the information regarding each player, e.g., name, color, captured pieces.

**Square** - is responsible for looking after each cell on the game board and what those cells are doing.

### Pieces

The pieces package has seven classes – Bishop, King, Knight, Pawn, Piece, Queen, and Rook. These classes contain the material for each specific piece, they all work in similar ways apart from the valid moves that determine what each valid move for the pieces are and allows the movement to take place.

### Moves

The moves package controls the movement of pieces as well as containing how any special moves work e.g., castling and en passant.

**/\*\* These sections will be removed, changed the layout but still using the info in them \*\*/**

### Board

This class implements the game board and the graphical section of the program, it builds the individual squares that the pieces are placed on. It also gives the location and the places that pieces in their starting location.

### Pieces

There are different classes for each individual pieces on the board, these classes lay out the valid moves the pieces can take and determines how these pieces move throughout the board.

### Player

The player class holds the information connected to the two individual players and what colour they are playing with.

### Main

The main class connects all the other classes and launches the GUI and starts the game.

### Square

The square class is responsible for checking what piece or if no pieces are in the square that is called.

## Modules shared between programs

## Table mapping requirements onto classes

|  |  |
| --- | --- |
| Functional Requirements | Classes providing requirement |
| FR1 |  |
| FR2 |  |
| FR3 |  |
| FR4 |  |
| FR5 |  |
| FR6 |  |
| FR7 |  |
| FR8 |  |
| FR9 |  |
| FR10 |  |
| FR11 |  |

# dependency description

# interface description

## Game Package

### Board

* Type: Public
* Public Methods

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
|  | Board() | Constructor. |
| Array | getBoardArray() | Initializes array. |
|  | boardReset() | Creates the board and places the pieces in their cells. |
|  | printBoard() | Used for debugging, prints the array to the console. |

### Game

* Type: Public
* Public Methods

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
|  | Game() | Constructor. |
|  | getBoard() |  |
|  | setWhitePlayer(String name) |  |
|  | setBlackPlayer(String name) |  |
|  |  |  |

## GUI Package

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |

## Moves Package

## Pieces Package

### Bishop

* Type: Public
* Extends: Piece
* Public Methods

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
|  | Bishop(Color color) | Constructor. |
|  |  |  |

## Tests Package

## Util Package

# detailed design

REFERENCES

[1] Software Engineering Group Projects: General Documentation Standards. C.W. Loftus. SE.QA.05. 2.3 For Release

DOCUMENT HISTORY

| *Version* | *Issue No.* | *Date* | *Changes made to document* | *Changed by* |
| --- | --- | --- | --- | --- |
| 0.1 | N/A | 27-02-2023 | N/A - original version | JAB153 |
| 0.2 | N/A | 07-03-2023 | Added interface descriptions | JAB153 |