**Software Design  
Document**

for

**CentipedeArmy Chess Game**

Version 1.0 approved

**Prepared by Emmanuel (Tobi) Afolayan, Drew Grubb, Jed Hutto, Jesse Miara, Ryan Weeks**

**CS 3398.264**

February 9, 2018

# Contents

[Revisions ii](#_Toc117484243)

[1 Introduction 1](#_Toc117484244)

[1.1 Purpose 1](#_Toc117484245)

[1.2 System Overview 1](#_Toc117484246)

[1.3 Definitions, Acronyms and Abbreviations 1](#_Toc117484247)

[1.4 Supporting Materials 2](#_Toc117484248)

[1.5 Document Overview 2](#_Toc117484249)

[2 Architecture 2](#_Toc117484250)

[2.1 Overview 3](#_Toc117484251)

[2.2 Component 1..3 3-4](#_Toc117484252)

[3 High-Level Design 4-6](#_Toc117484253)

# Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| 1.0 | Jed Hutto Drew Grubb Jesse Miara Ryan Weeks Tobi Afolayan | Writing of the SDD | 02/25/18 |

# Introduction

# Purpose

## The software described and modeled in this document is meant to simulate a game of chess that can be played between two players, and will have a variety of modes: Player vs. Player, AI vs. Player, and AI vs. AI. Users interacting with the software should be able to select game modes and preferences with relative ease thanks to access to an integrated menu system.

# System Overview

The software will be programmed in Java to run using Java VM. The structure of the program shall use a Game Loop system, where the game is updated first and then rendered separately. This software is not going to interact with any external systems.   
  
  
The Game Loop has a very piece by piece functionality structure. When updating, the first step is to check any updates to the user input. The loop then reacts based on the current state of the Game State and the Input Listener.

The final step of the game loop is rendering. First, the render method will display the window and current Game State to the screen. The Game State render method will display all the buttons and render the board. The game state will then loop through every existing entity on the board and call its render method, rendering it to the screen based on its coordinates in the 2D Plane.

# Definitions, Acronyms and Abbreviations

•AI- Artificial Intelligence: a software developed replication of human intelligence, allowing the software to operate without direct, manual control

•API- Application Programming Interface: an interface that communicates with software components regarding how to build a specific software, usually providing definitions and tools

•GUI- Graphical User Interface: an interface that allows users to interact with software with textless, visual tools, such as windows, scroll bars, and buttons

•IDE- Integrated development environment: a component of software development that allows programmers and users to analyze the software in question to looks for defects and attempt to debug the software

•Java- The programming language that was used to code this chess application, Java relies on an object-oriented design and a general-purpose implementation of various software components

•JRE- Java Runtime Environment: an application that allows users to develop and run coded software that relies on the Java programming language

•OS- Operating System: a part of the computer that provides both the hardware and software with common features

•UML- Unified Modeling Language: A language used to develop diagrams that visualize the design of software.

# Supporting Materials

"Software Requirements Specification for CentipedeArmy Chess Game"

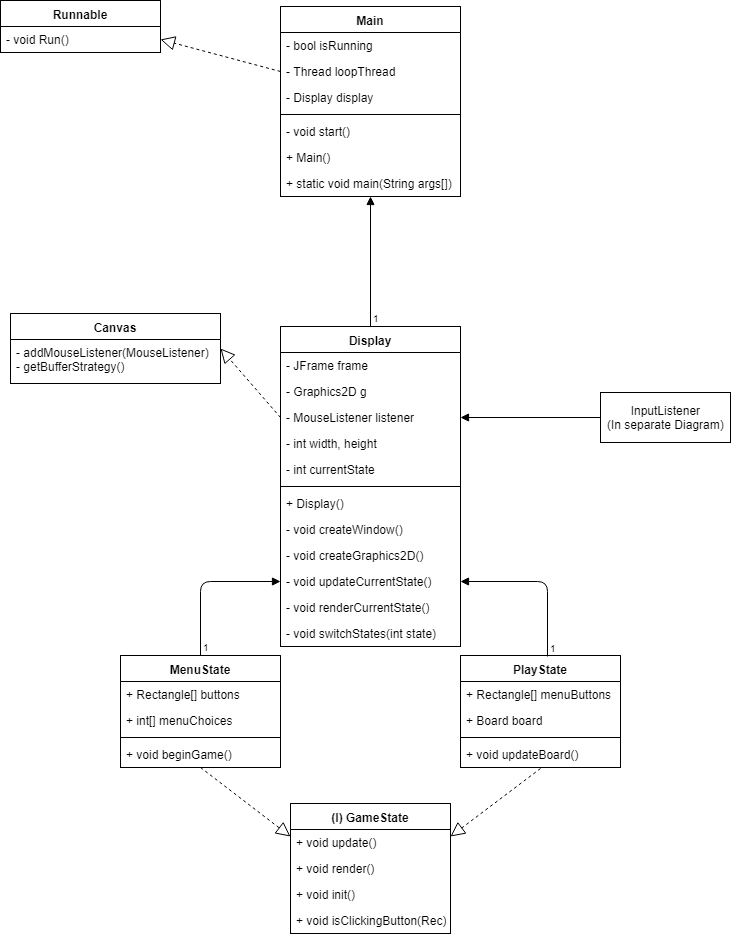
"Test Plan for CentipedeArmy Chess Game"

# Document Overview

This document discusses the architecture and design aspects of our chess game, especially regarding how the game and its respective menus should function upon user input.

# Architecture

CentipedeArmy Chess Game uses a state design pattern. The game has two states that switches between; Menu State and Play State. The Menu State has the graphics for the menu. It will contain multiple buttons, with each performing a specific task. The Play State has the graphics for playing the CentipedeArmy Chess Game. It will be responsible for showing the board, timers, pieces, and options buttons.



# Overview

The CentipedeArmy Chess Game Display class handles the graphics of the game and the inputs to the game. This class also handles the state that the CentipedeArmy Chess Game is in. This is done for simplicity of the code. The PlayState class will handle the board, pieces, movements, and users.

# Component 1: Chess Piece Class

Each piece in the game will be created via the chess piece class, which takes each piece's unique design and move set into account while keeping track of the common characteristics between all pieces. Each piece will have a set of possible moves available to the player when they select that specific piece. Some pieces will also have specific movesets based on whether or not they have been moved yet. The King will constantly need to check to see if he is in check or checkmate.

# Component 2: Board Class

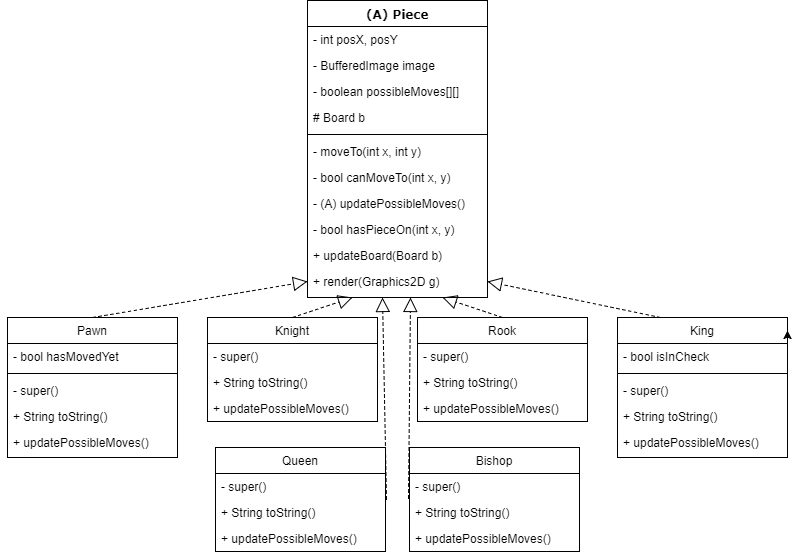
The Board class will contain an eight by eight-piece matrix, called board. The pieces will be arranged in the board in the same order as a typical chess game. Null elements in the board are considered empty spaces for updating move sets. This class is responsible for rendering the whole board, calling each Piece's render method within the 2D matrix.

# Component 3: Menus

There will be a single menu in our game of chess, and it will be run via the Menu State class. The menu will have choices for different game types, timer modes, and AI difficulty. After the user chooses the options they desire, the Menu State will initiate the Play State, starting the game. Although the Play State does not have a specific menu, there is a list of buttons next to the board that can be used for different things. The pause button will allow a user to pause the game if it is timed, turning it into a resume button. The forfeit button will cause the current player to forfeit the game to the opposing player, losing the game.

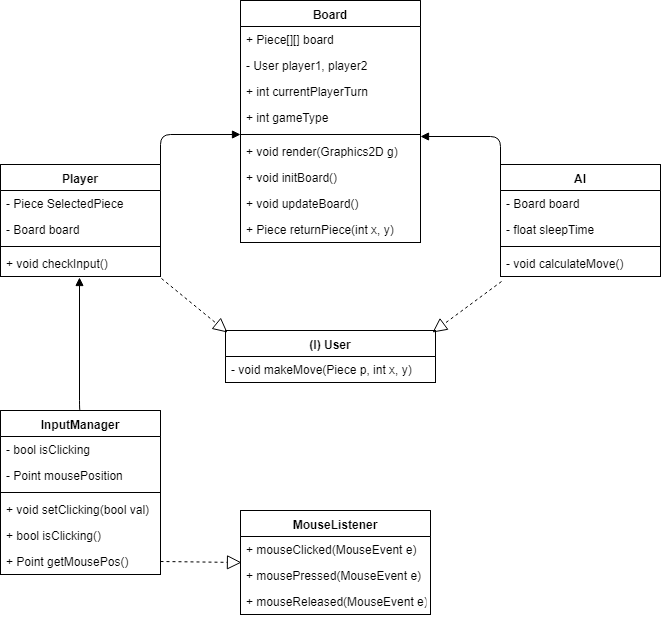
# High-Level Design

# View / Model Component 1: Chess Piece Class



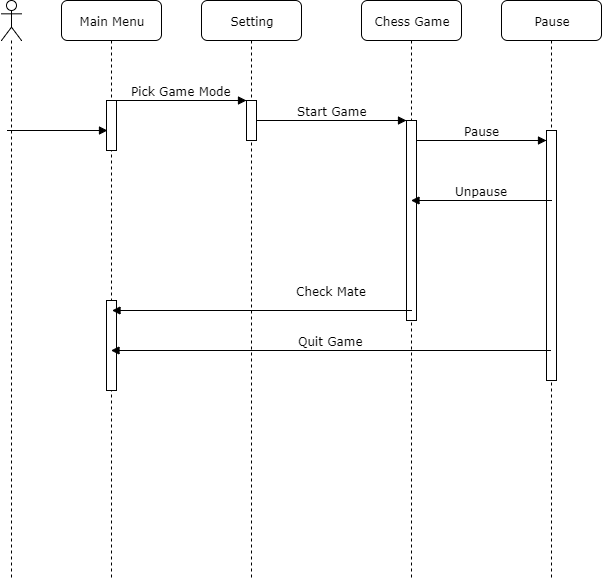
Piece is a component for the Board class. Pawn, Knight, Rook, King, Queen, and Bishop inherit from Piece. Most of the pieces share similar methods and attributes. The method, updatePossibleMoves will be unique to each piece type. The Pawn and King class have an additional attribute for their special purpose moveset. Pawn, for example, can move forward twice but only if it has not moved yet. The King moveset will constantly change depending on whether he is in check.

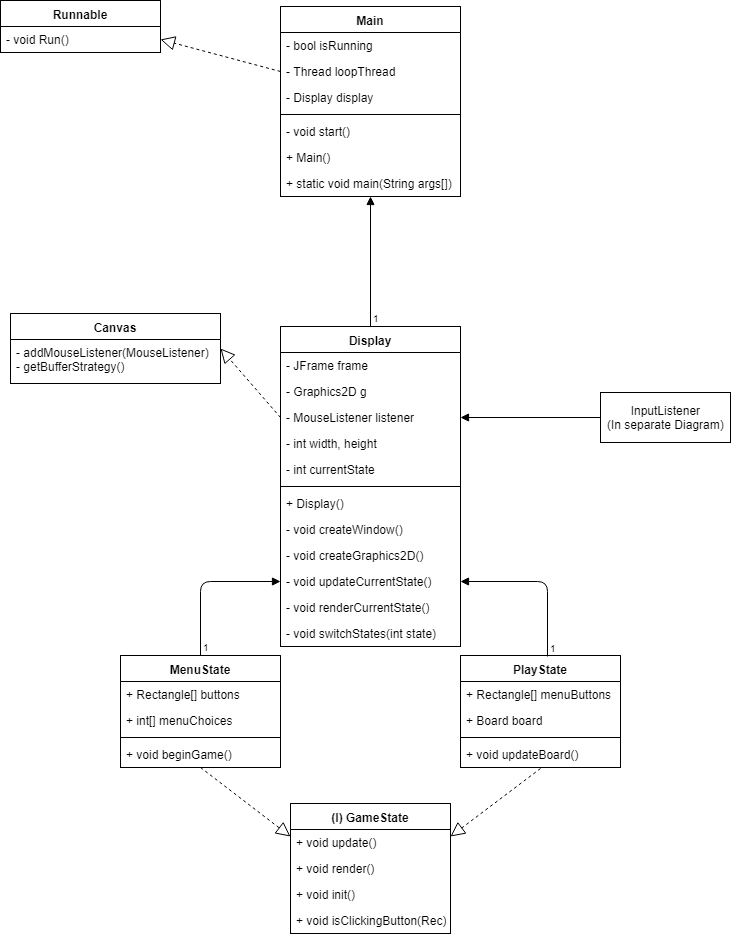
# View/Model Component 2: Board Class



The Board class handle the interaction between the users and the piece board matrix. In order for the user to interact with the board, the Player class uses an input manager to capture the user's selection. The Input Manager class will use java's inherent Mouse Listener interface in cohesion with the "Canvas" class extended by Display to listen for mouse input. The Player class and the AI class both inherit from User, which will have some base methods for those classes.

# View / Model Component 3: Menus





Once the CentipedeArmy Chess Game starts, the Main class will initiate the Display class. The Menu State will be the initial state for the Display class. Once the user has made their selections, the game state will be switched to the Play State. Based on what options the user had selected in the main menu, the Board class will initiate the desired game type. The user will have the option in the Play State to pause the game if it is timed. While paused, the user will be able to forfeit or resume the game.