**Test Plan**

for

Rabbit Checkers

Version 1.0 approved

Prepared by Theodore Zuniga, Danielle Fernandez, Amy Ripley, and Chris Yanez

Team Rabbit

October 2018

# Table of Contents

Table of Contents i

[1.1 Introduction 1](#_Toc117484244)

[1.1.1 Test Plan Objectives 1](#_Toc117484245)

[2.1 Test Strategy 1](#_Toc117484250)

[2.1.1 System Test 2](#_Toc117484251)

[2.1.2 Stress/Performance Test 2](#_Toc117484252)

[2.1.3 Automated Test 2](#_Toc117484252)

[3.1 Environment requirements 2](#_Toc117484253)

[4.1 Functions to be Tested 3](#_Toc117484254)

# Introduction

The purpose of this document is to provide an overview of the Test Plan for Rabbit Checkers.

### Test Plan Objectives

The purpose of the Test Plan is to verify and validate that the Rabbit Checkers application achieves all of the requirements laid out in the SRS.

*The objective is to verify the following functions:*

1. *Allow the user to play a standard game of checkers against an AI*
2. *Move the checkers pieces according to their respective class using mouse hardware*
3. *Allow the user to play a standard game of checkers against another human player*
4. *The system has a button that will allow the user to restart the game at any time*
5. *The system has checkboxes to allow the user to switch between human vs. AI and human vs. human*

# Test Strategy

<*For each of the following, you should explain how you plan on testing your application, provide a brief description, and possibly scenario/example of how you will accomplish that sort of test. Some of these may not pertain to your application>*

1. *Allow the user to play a standard game of checkers against an AI*
   1. *The tester checks that the game starts in human vs. AI setting*
   2. *The tester checks that the game will switch to human vs. AI if the checkbox is clicked*
2. *Move the checkers pieces according to their respective class using mouse hardware*
   1. *The tester checks that the game starts in human vs. AI setting*
   2. *The tester moves pieces at least three times to ensure the pieces*
   3. *The test captures an AI checkers piece to check if the pieces move to the correct location on the board*
3. *Allow the user to play a standard game of checkers against another human player*
   1. *The tester checks if the game begins in the human vs. AI setting*
4. *The system has a button that will allow the user to restart the game at any time*
   1. *The tester begins a regular game against the AI*
   2. *The tester confirms the game restarts during different states of the game*
   3. *The tester begins a human vs. human game*
   4. *The tester confirms the game restarts during different states of the game*
5. *The system has checkboxes to allow the user to switch between human vs. AI and human vs. human*
   1. *The tester begins a regular game against the AI*
   2. *The tester confirms a new human vs. human game is started when the correct checkbox is selected*
   3. *The tester confirms a new human vs. AI game is started when the correct checkbox is selected*

### System Test

The approach for the system test is to test all components of rabbit checkers are working correctly. The tester will play each game type at least once while making sure that the GUI is updated correctly for each move.

### Stress/Performance Test

The tester will use negative testing to test how Rabbit Checkers handles incorrect inputs from the user.

1. The tester will play a regular game and try moving pieces to positions not allowed by that piece.
2. The tester will attempt to stress the system by attempting to move pieces into invalid positions on the board several times.
3. The tester will stress the system by clicking on different pieces on the board multiple times.

### Automated Test

The automated test will be performed using JUnit testing.

# Environment Requirements

*The environment will take place using a laptop computer. It will be performed using BlueJ IDE to run both the tests and the game.*

# Functions To Be Tested

Class GameManager:

GameManager(), getBoardState(), firstMove(), playerCanMove()

Class CheckerBoard:

CheckerBoard(), updateBoard()

Class Human:

getRandMove(), getMove(), getTurnComplete(), setTurnComplete(), getIsBottom(), getColor(), getKingColor()

Class AI:

getRandMove(), getMove(), setTurnComplete(), getIsBottom(), getColor(), getKingColor()