**Software Design  
Document**

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

Revisions ii

1. Introduction 1

1.1 Purpose 1

1.2 System Overview 1

1.3 Definitions, Acronyms and Abbreviations 1

1.4 Supporting Materials 1

1.5 Document Overview 2

2. Architecture 2

2.1 Overview 1

2.2 Component 1..5 2

2.2.1 Game Manager 2

2.2.2 Checker Board 3

2.2.3 human 4

2.2.4 ai 5

2.2.5 user interface 5

3. High-Level Design 5

3.1 State Diagram for Starting a New Game 5

3.2 Sequence Diagram 6

4. Use cases 6

4.1 Start Game/Options 6

4.2 Player vs. Ai 7

# Revisions

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Primary Author(s) | Description of Version | Date Completed |
| Draft Type and Number | Full Name | Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded. | 00/00/00 |

# Introduction

## Purpose

The purpose of this Software Design Document (SDD) is to provide an overview of the high-level design aspects for the Rabbit Checkers application. It shows the Model-View-Controller (MVC) architecture pattern used, the classes and components along with several diagrams illustrating the interactions of the application. It follows the requirements that were laid out in the SRS and serves as a navigational tool for the software developers.

## System Overview

Rabbit Checkers will be developed entirely in Java and can be run on any system capable of running Java code. The Observer/Observable pattern will be implemented in the Model-View-Controller architecture system. Rabbit Checkers will consist of four main classes including the Game Manager (GM), Checker Board (CB), Human and AI. These components are explained in more detail later in the document.

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| AI | Artificial Intelligence |
| CB | Checker Board class |
| GM | Game Manager class |
| GUI | Graphic User Interface |
| Kinged | When a piece reaches the opposite side of the board and can move forwards and backwards. |
| MVC | Model-View-Controller |
| Player | Human User |
| RC | Rabbit Checkers |
|  |  |
|  |  |

## Supporting Materials

"Checkers." Wikipedia, The Free Encyclopedia. 25 Aug 2018, 15:17 UTC. 20 Sep 2018, 23:30 <https://simple.wikipedia.org/w/index.php?title=Checkers&oldid=6232760>.

“MVC Architecture.” MDN Web Docs, Mozilla, 8 June 2018, developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern\_web\_app\_architecture/MVC\_architecture.

## Document Overview

The document is separated out into four main sections with subsections.

Section 1 contains an overview of this document including definitions and supporting material.

Section 2 contains the overall architecture of the application, which consists of the specific classes and components listed out in more detail.

Section 3 contains the high-level design overview with specific diagrams of the different components of the system.

Section 4 contains use cases illustrating different interactions between the user and the GUI.

# Architecture

The following diagram describes the overall design of the system design.

# Overview

This system utilizes the common software architectural pattern, Model-View-Controller. The unpredictable rate of exchange to and from a user justified using the pattern. It is described in detail in Figure 1.

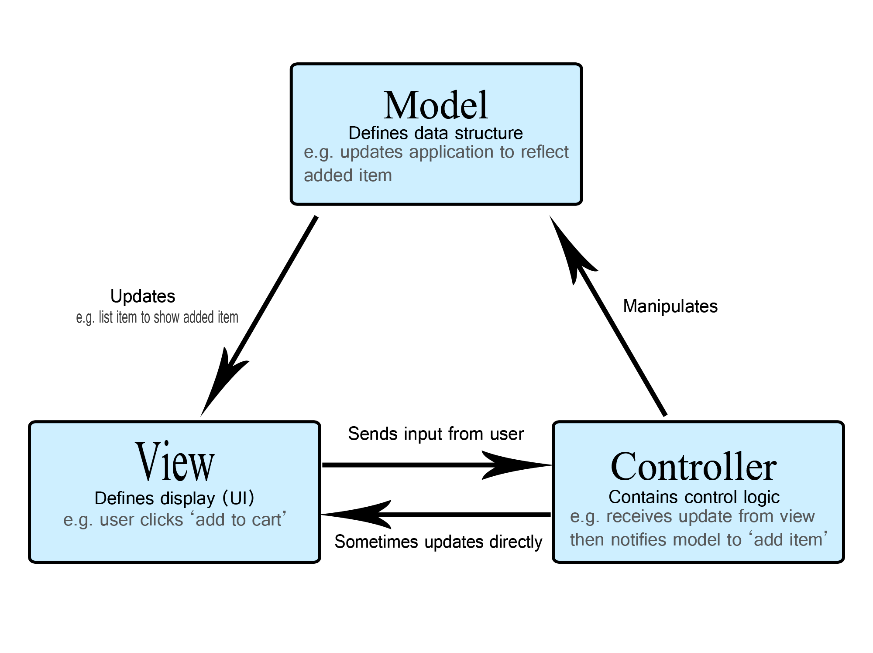


Figure 1: Model-View-Controller Diagram

# Components 1..5

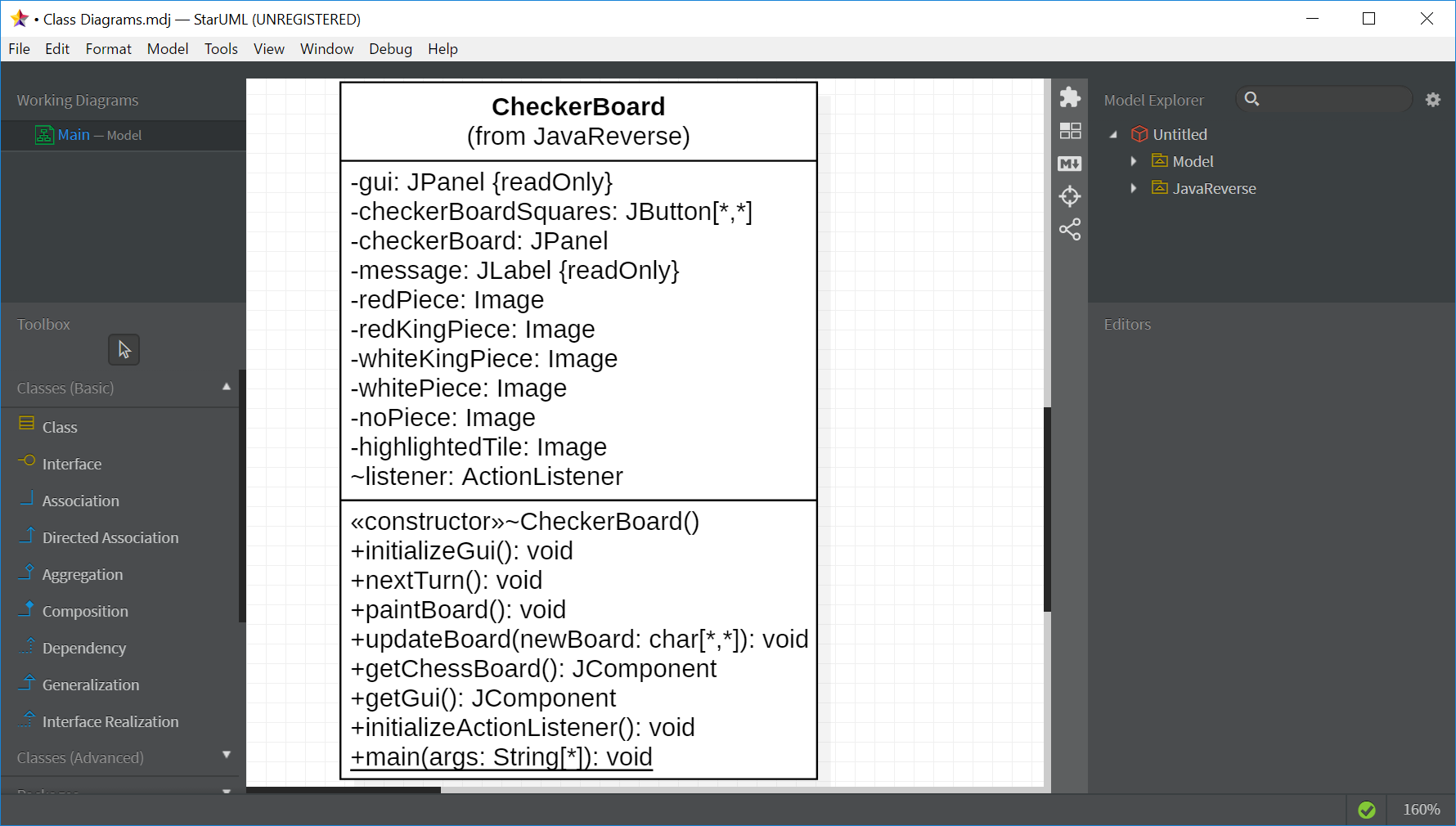
* + 1. **Game Manager Class**

The Game Manager (GM) class acts as the controller component in the Model-View-Controller diagram shown in figure 1.

A screenshot of a cell phone

Description automatically generated

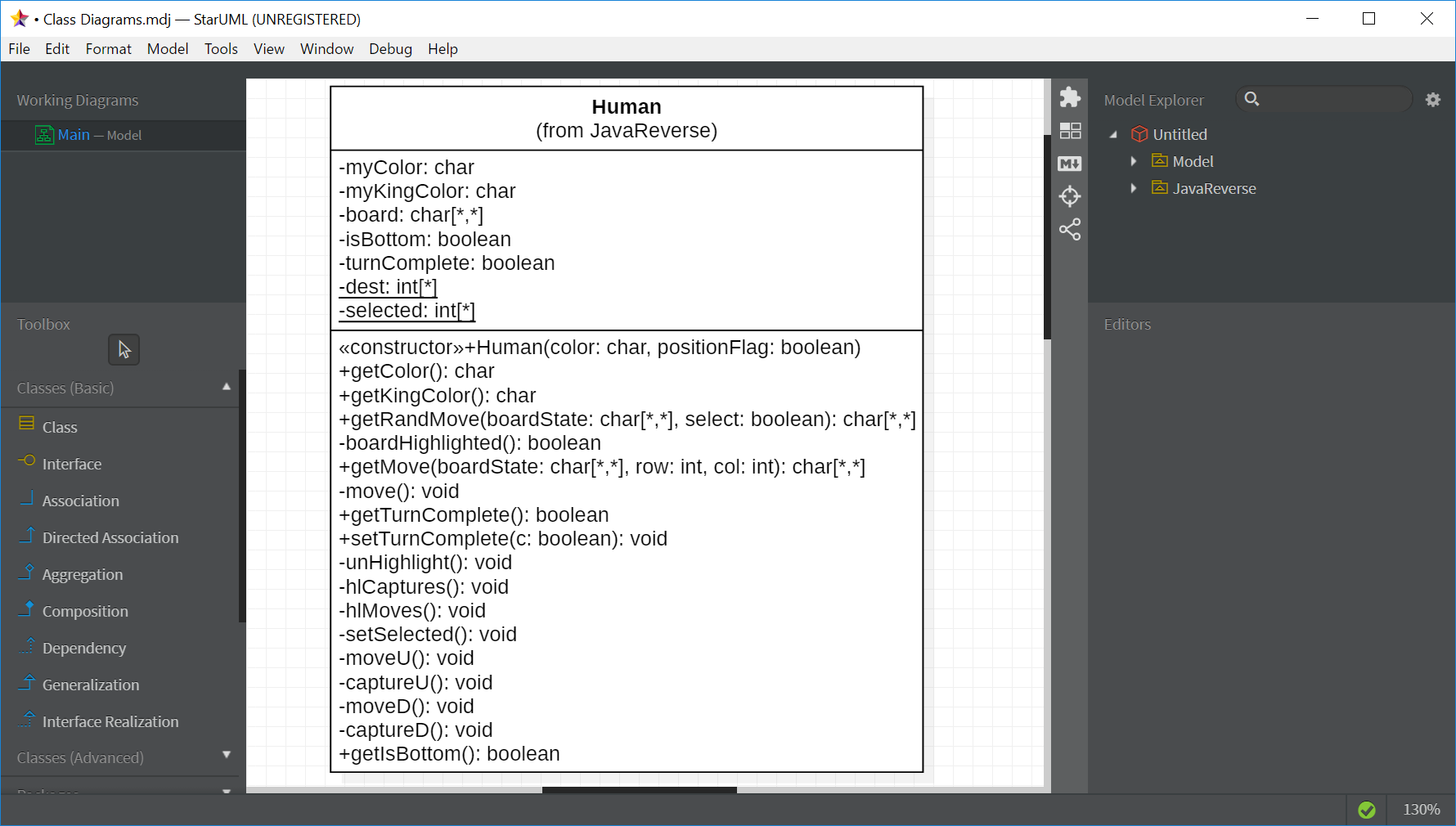
* + 1. **Checker Board Class**

The Checker Board (CB) class acts as the model component of the MVC design.



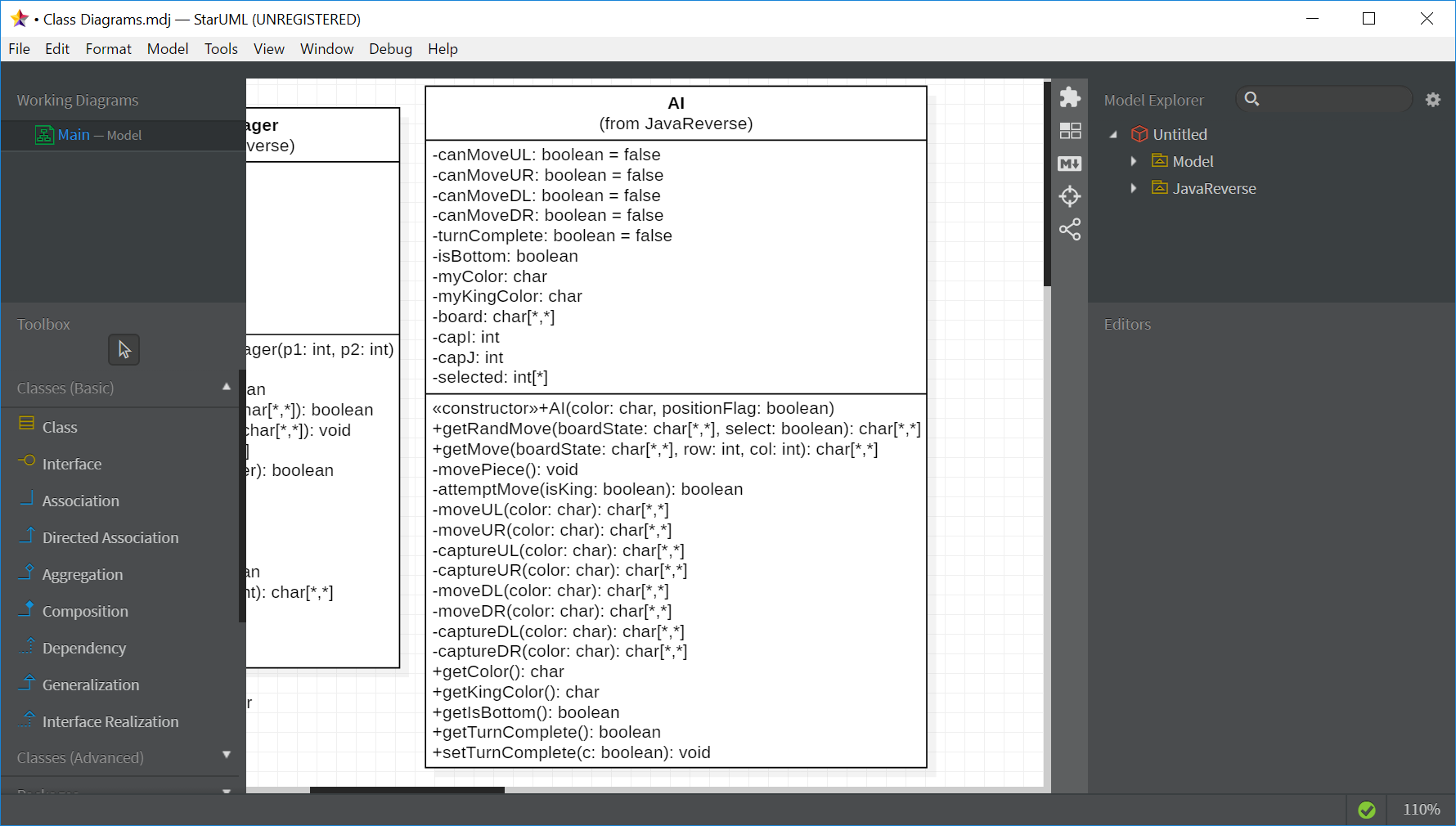
* + 1. **Human Class**

The Human class contains the operations for a human user. There can be up to 2 of these objects in a system.



* + 1. **AI Class**

The AI Class contains the operations for a CPU user. There can be up to only 1 of these objects in a system.



* + 1. **User Interface**

When a user opens up the Rabbit Checkers application, a GUI will be displayed. This GUI will use observer and observable design patterns in order to incorporate mouseclicks as a way of communication with a user. A representation of what the display should look like is shown in Figure 2.

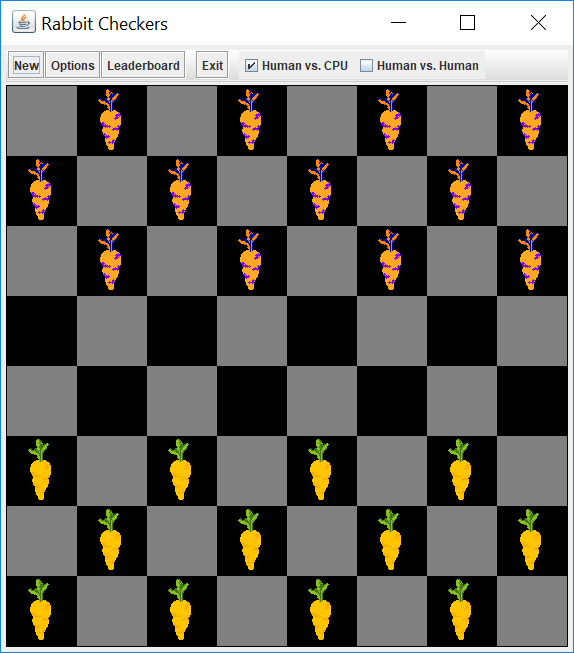
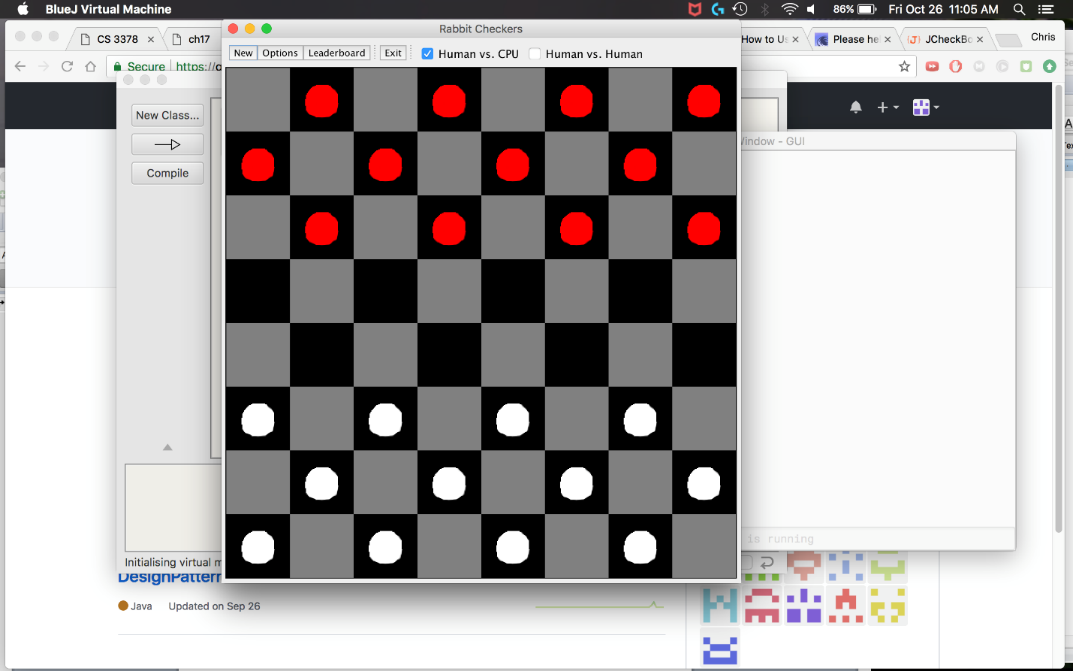


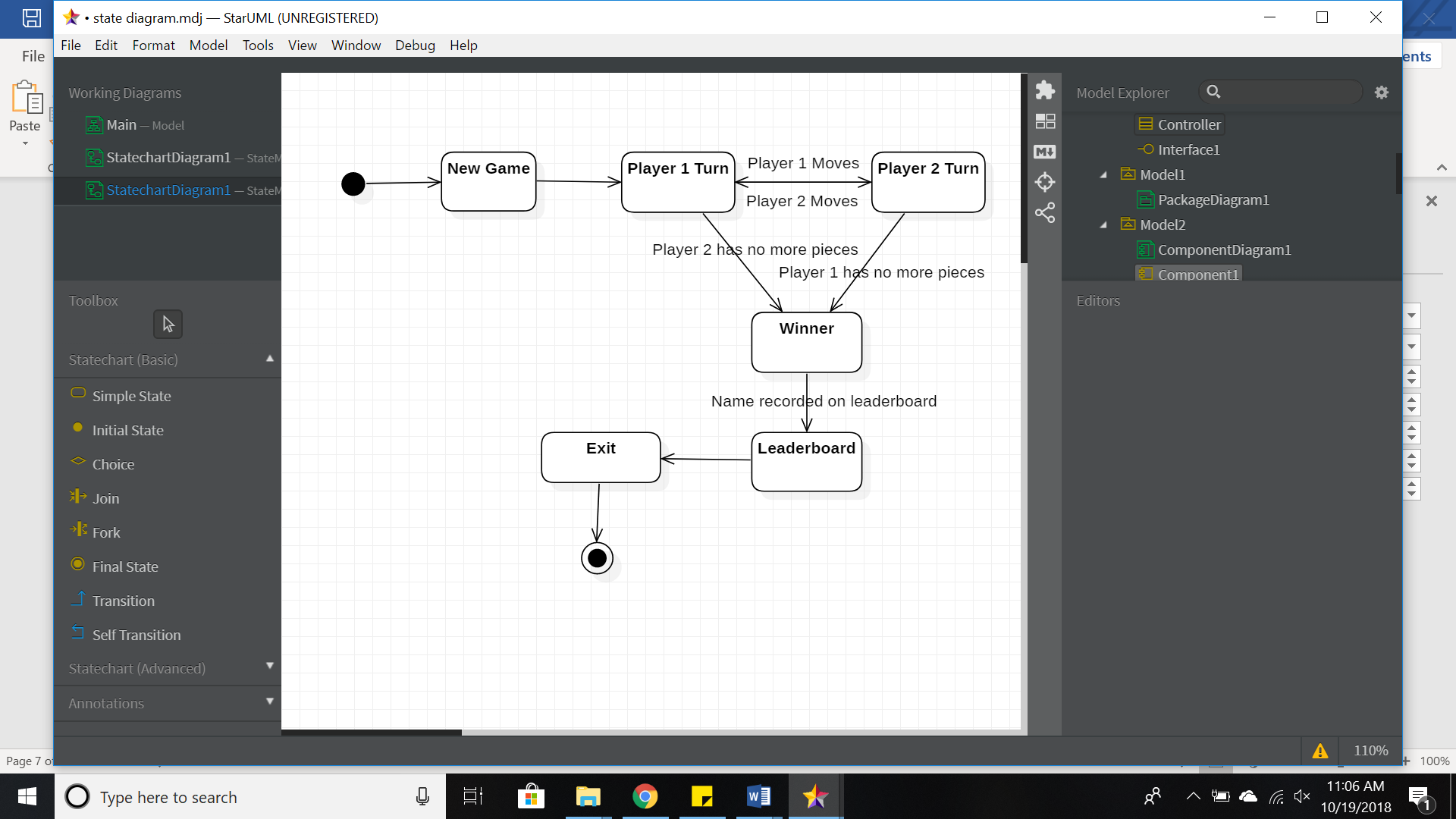
Figure 3: GUI with Carrot Pieces

Figure 2: GUI with Default Pieces

# High-Level Design

The following diagrams embody the design of the flow of the system.

## State Diagram for Starting a New Game



## Sequence Diagram

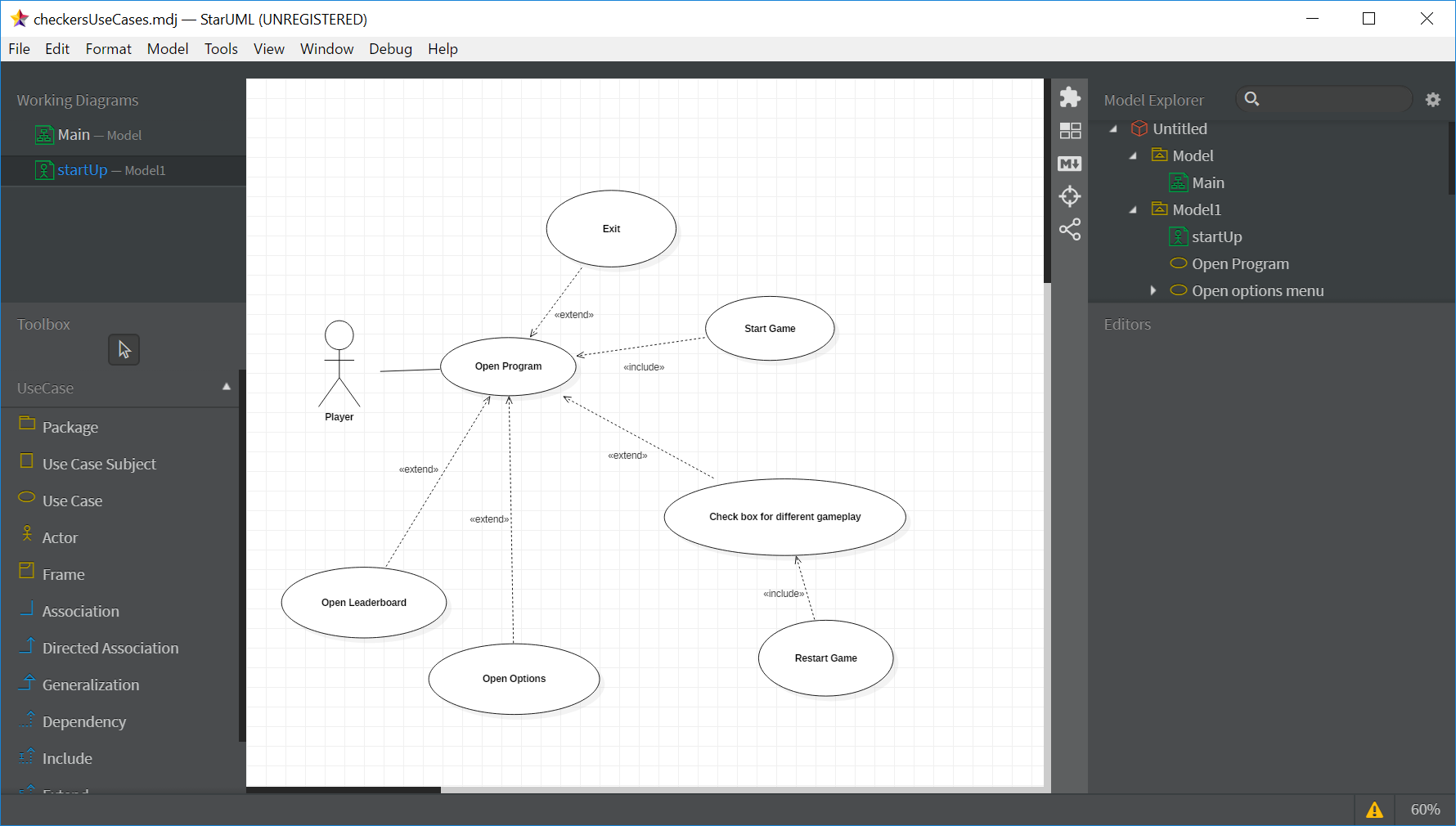
A close up of a map

Description automatically generatedThe diagram to the left shows the transition of messages throughout the system when a different game mode is selected via checkbox in the start-up visual screen.

# Use Cases

The following use case diagrams illustrate the possible interactions the user will have with the GUI.

## Start Game/Options

When the user opens the RC application a new default game will be displayed. The user will have a selection of options that will be displayed on the top of the window.



## Player vs. AI

Once a new RC game has started the user has the option to move a piece if it is their turn or they can terminate the application by selecting the exit button.

