|  |  |
| --- | --- |
| FRUSTRATION  A SOFTWARE ENGINEERING DESIGN PROJECT | assignment 2017: gROUP pROJECT  GROUP  Rubab Ramzan - S00162293 Brain Mc Gowan - S00165159 Bryan Kerruish - S00173160 |

Table of Contents

[Project Overview 2](#_Toc476822399)

[Part 1: 2](#_Toc476822400)

[Draft Class Diagram 2](#_Toc476822401)

[Draft Class Diagram Notes 2](#_Toc476822402)

[Game Class: 2](#_Toc476822403)

[Board Class: 3](#_Toc476822404)

[Player Class: 3](#_Toc476822405)

[Piece Class: 3](#_Toc476822406)

[Dice Class: 3](#_Toc476822407)

[State Diagrams 3](#_Toc476822408)

[State Diagrams Notes 6](#_Toc476822409)

[Part 2 6](#_Toc476822410)

[Method (Operation) Specification 6](#_Toc476822411)

[Decision Charts 7](#_Toc476822412)

[Draft Expanded Take Turn Decision Chart 7](#_Toc476822413)

[Final Take Turn (Simple) Decision Chart 8](#_Toc476822414)

[Part 3 8](#_Toc476822415)

[Detailed Class Diagram 8](#_Toc476822416)

[Detailed Class Diagram Notes 8](#_Toc476822417)

[Part 4 8](#_Toc476822418)

[Implementation and Testing of Classes in C# 8](#_Toc476822419)

[Part 5 9](#_Toc476822420)

[Code and Walkthrough Notes 9](#_Toc476822421)

[Decoupling the interface from the application logic. 9](#_Toc476822422)

[DRY principles applied. 9](#_Toc476822423)

[Naming conventions. 9](#_Toc476822424)

[SOLID principle applied. 9](#_Toc476822425)

[Code structure. 9](#_Toc476822426)

[Code readability and maintainability. 9](#_Toc476822427)

[Cheats 9](#_Toc476822428)

# Project Overview

For this project, we decided upon developing an application to replicate the board game Frustration. Frustration is a simple board game in which players compete to be the first to send four pieces all the way around a board and is a version of the game Ludo.

This game has a concise set of rules and this documentation allowed for precise and accurate requirements gathering.

From the game’s documentation, we were able to generate a series of user stories and use case templates.

To further ensure we had captured the requirements fully, we developed a program flow chart and a number of process flow charts to cover the main methods.

# Part 1:

### Draft Class Diagram

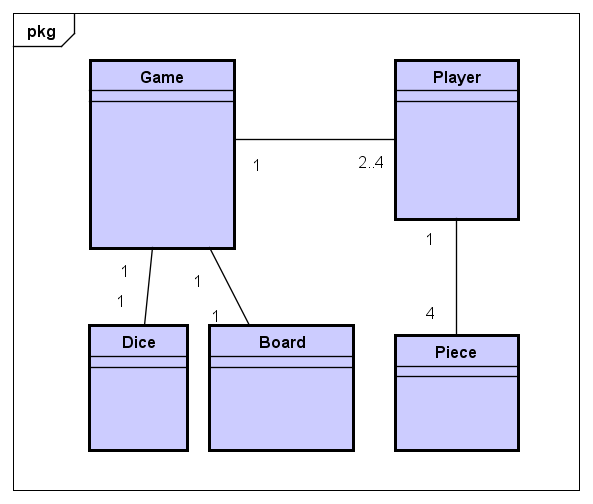


Figure 1 - First Draft Class Diagram

## Draft Class Diagram Notes

### Game Class:

This class will contain the game logic.

It will instantiate two to four players depending on user selection; a board and a dice.

* The game will have one board.
* The game will have two to four players.
* The game will have one dice.

### Board Class:

The board will contain the game logic for the board positions, home position and the pieces’ locations.

* The game will have one board.

### Player Class:

The player class will contain the logic of where the player starts on the board and the pieces they have.

The player will instantiate a list containing four pieces.

* The game will have two to four players.
* The player will have four pieces.

### Piece Class:

The piece will contain the logic for moving around the board and its state within the game.

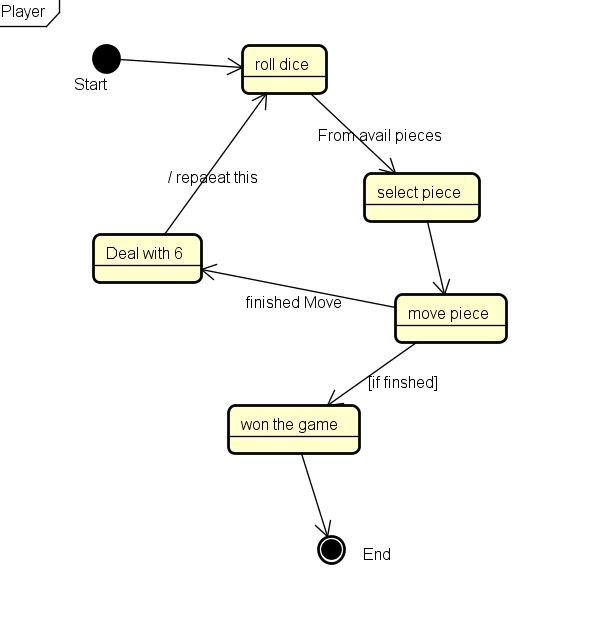
* Each player will have four pieces.

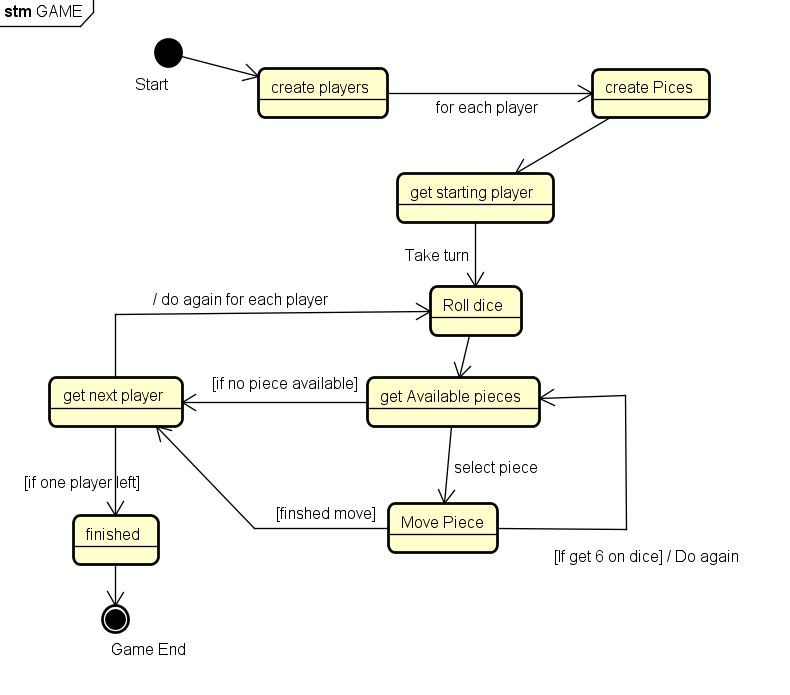
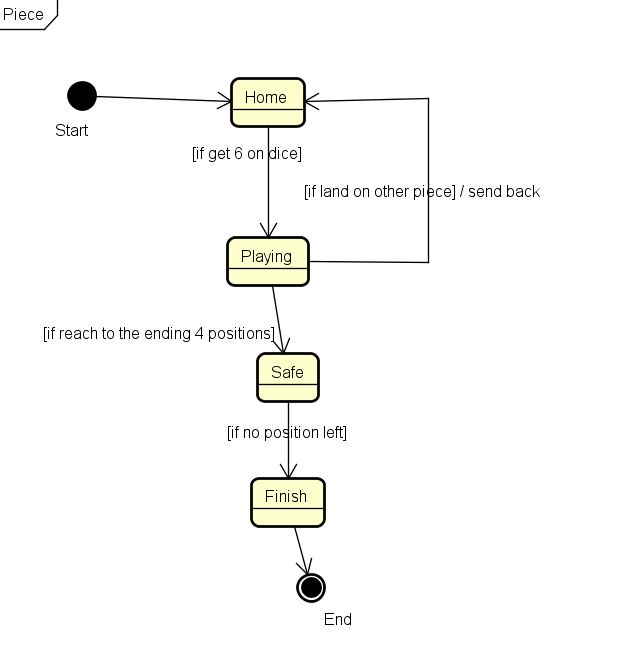
### Dice Class:

The dice will contain the logic to generate a random number in the range of one to six.

* The game will have one dice.

## State Diagrams

****

****

## State Diagrams Notes

# Part 2

### Method (Operation) Specification

## Decision Charts

### Draft Expanded Take Turn Decision Chart



### Final Take Turn (Simple) Decision Chart



# Part 3

## Detailed Class Diagram

## Detailed Class Diagram Notes

# Part 4

## Implementation and Testing of Classes in C#

# Part 5

## Code and Walkthrough Notes

### Decoupling the interface from the application logic.

### DRY principles applied.

### Naming conventions.

### SOLID principle applied.

### Code structure.

### Code readability and maintainability.

### Cheats