Knight's Party Table Code Design Document

Tim Adams, Chris Lucas, Tyler Sancibrian, Tony Burke

Version 1.2

Table of Contents

1. Introduction

The primary components of the Knight’s Party Table are the launcher and games; games being Battleship, Checkers, Chess, Settlers of Catan, and Tim’s Text-Based RPG. The code design explains how these and other parts of the Knight’s Party Table will be implemented.

1.1 Design Goals and Assumptions

The design goals remain the same as for the system design:

- Easy to implement
- Easy to understand
- Isolation for changes

The system assumes that users are familiar with both classic arcade machines, as well as the rules to the games included in the Knight’s Party Table. It is worth noting that material included in this document are liable to change. Such changes made during development will be added to this design document, and the version number will be incremented.

1.2 Objectives and Benefits

The primary objective of the Knight’s Party Table is to provide entertainment to the users by allowing them to play a variety of board games without the user needing to own or store several different physical board games. The Knight’s Party Table will accomplish this with the utilization of well-known hardware resembling that of a classic arcade and software that is intuitive. The games to be implemented on the Knight’s Party Table are as follows:

- Battleship
- Checkers
- Chess
- Settlers of Catan
- Title Pending Text RPG

2. System Architecture

Architectural Design Patterns

The architectural design pattern that the Knight's Party Table will use is a layered architecture. This design was chosen because while it is not a universal design pattern, it will best suit the Knight's Party Table. By establishing separate layers in the system's architecture, we can effectively ensure simplicity and reliability in the Knight's Party Table. Additionally, establishing separate layers in the system's architecture allows the Knight's Party Table to be more easily modified in the future.

Deployment View

The deployment view of the system shows the physical nodes that will be interacting with each other in the on-location system. As shown in the diagram, the physical deployment of the Knight's Party Table is very simple, and all one unit. The two players control the Knight's Party Table by utilizing the attached Controller Units on either side of the gaming table. The Controller Units are composed of a joystick and buttons attached directly to the surface of the table.

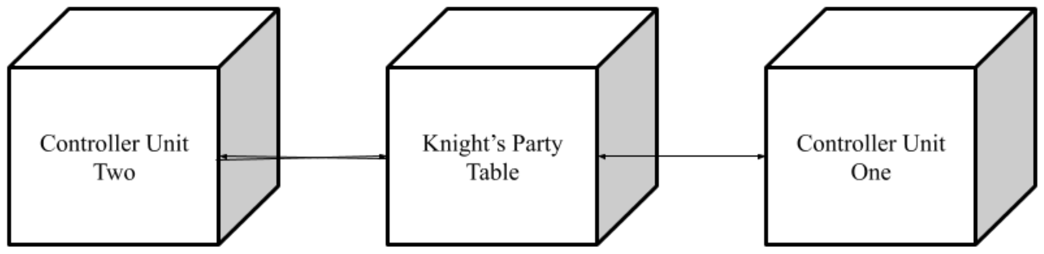

Figure I: The Deployment View

Component View

The component view describes the different systems and subsystems of the Knight's Party Table. The user will access the Knight's Party Table through the Controller Units, using the User Interface. The User Interface will accept input from the Controller Units and display output to the user, as well as the audience. Using the User Interface, the user will select a game from the Games Library. The user will then interact with the game through the User Interface and directly play the game.

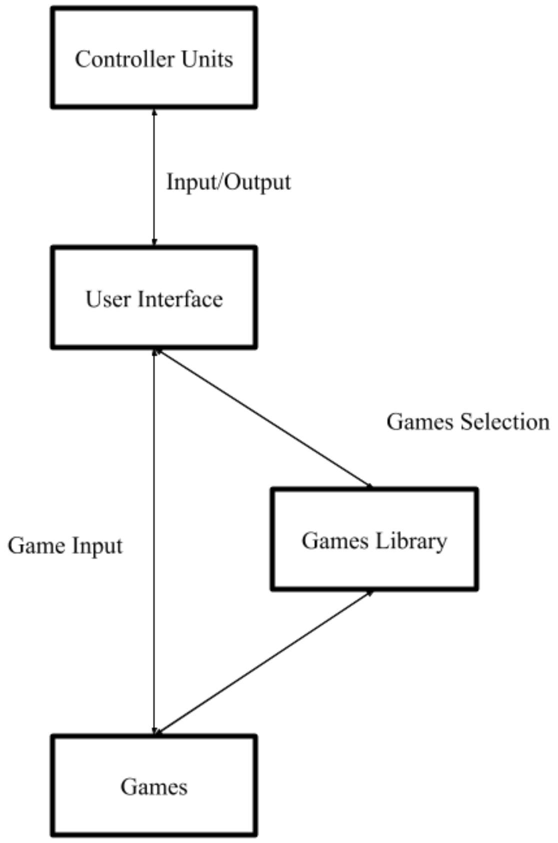

Figure II: The Component View

3. Overall Class Design

The design of the Knight's Party Table is mostly focused around the components of the system. The following section describes the code design in several UML diagrams.

Launcher Design

The Knight's Party Table will allow users to select a game to play on the system through a launcher. The launcher will be comprised of a graphical user interface that allows the user to select a game. Given that Java is the language chosen, the Launcher will utilize Java's Swing GUI framework. The Launcher will utilize the method *launch(Game)* to launch a game. *launch(Game)* accepts the class Game, a superclass that houses the shared methods of each game. The relationship between Launcher and Game is shown below:

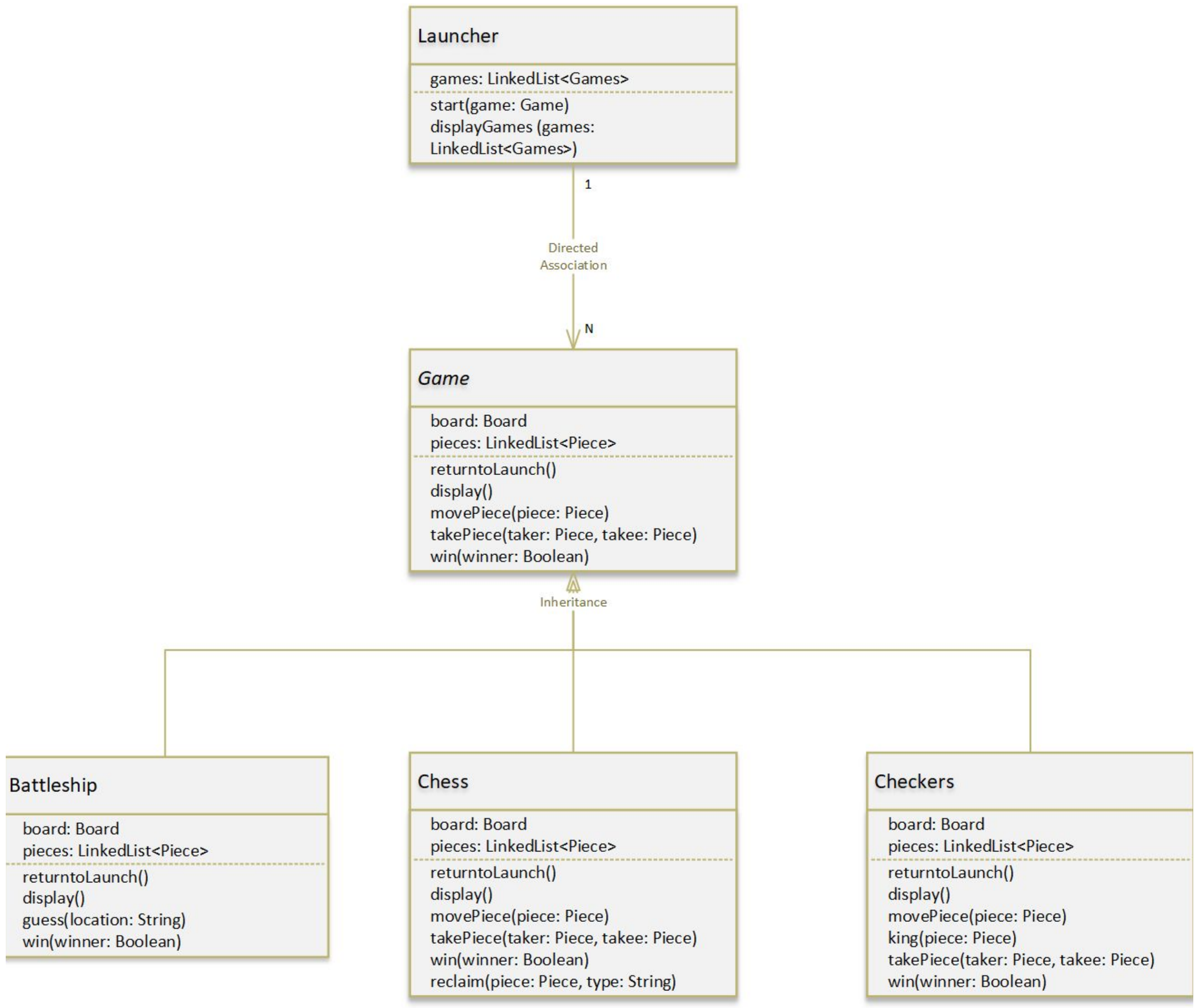

Figure III: Launcher and Games

Game Design

Each game used by the Knight's Party Table will be a subclass inherited from Game. Game will contain methods that will be used by each game, such as *display()* for displaying the game board and *movePiece(Piece)* for moving pieces in the game. Each game will utilize the class Piece, a class representing game pieces utilized in the game. One attribute of Piece will be *type*, a String that represents what type of Piece it is (Checker, CheckerKing, Cruiser, Pawn, Queen, etc.), and the other will be *player*, a boolean representing which user can control the Piece. A checkerboard represented by the class Board will be used for the following games: Battleship, Checkers, and Chess. Each user will have their own instance of Board for Battleship. The class Settlers, used to represent the board game Settlers of Catan, will have specialized classes specific to the game itself, as it is a more complicated board game.

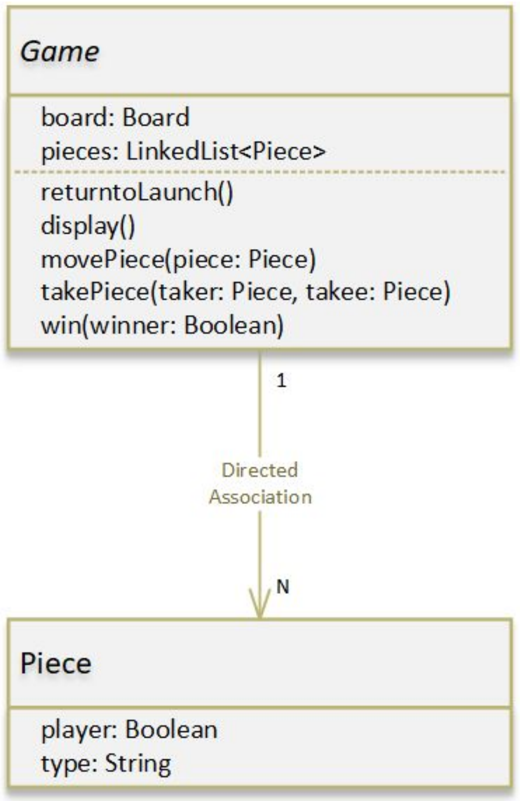

Figure IV: Game and Piece

Starting a Game

Each game will be accessed through the Knight's Party Table launcher, which will present the user with an interactive list. This interactive list will contain each game, and upon selection of the name by the user, will call the method *start(Game)* utilizing the game's class. Each game will be represented as a Java subclass of a superclass, Game, as illustrated in the aforementioned UML diagram.

Playing a Game

User interaction with games will be determined individually based on which game is being played. However, each game will have shared methods, determined by the Game superclass. These methods are detailed below:

Move: Moves a piece on the game board. This method will check the type of Piece used, and enforce the rules accordingly

Return: Returns the users to the Launcher.

Display: Displays the game as a GUI.

TakePiece: Takes a piece on the game board.

Win: Called when a user wins.

Lose: Called when a user loses.

Given that there are only two users, the simplest way to identify them is by assigning a Piece a value of "true" or "false" - user one is true, and user two is false.

Settlers of Catan

Given that Settlers of Catan is a much more complicated board game, it will have several classes of its own. These include DevCard, Tiles, and Resource, all of which will be used by Settlers. DevCard is a class used to represent the 25 Development Cards available in the game - 14 Knight cards, 5 Victory Point cards, 2 Road Building cards, 2 Year of Plenty cards, and 2 Monopoly cards. Each of the different Development Cards will be represented as a String, with a method *doCard(type: String)* that will perform the card's action based on the type of card. Tiles will represent the different tiles that make up the game board, and Resource will represent the different types of resources available to users. Settlers will also utilize the Player class, a class representing a player that contains the resources and Development Cards that are usable by the player. Player will utilize methods to add Development Cards and Resources. The relationship between the various classes of Settlers of Catan will be shown in the following UML diagrams:

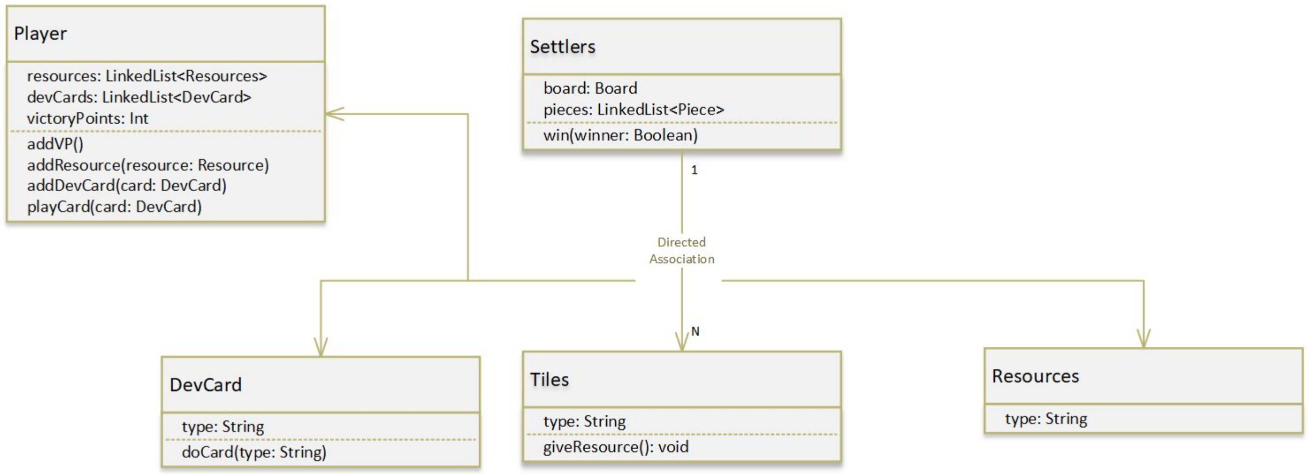

Figure V: Settlers Base Classes

Finishing a Game

Each game will have seperate win conditions, however, one shared feature between all of the games will be the method *returnLaunch()*. This method will be called when the game is finished and the users choose to return to the list of games. Each time a user wins, the Game method *win(user: Boolean)* will be called. This method will depend on which game is currently being played. The win conditions for each game are as follows:

Battleship: All of one user's ships are destroyed.

Checkers: All of one user's checkers are taken.

Chess: One user is placed in checkmate.

RPG: The main quest is completed.

Settlers of Catan: One user has gathered ten Victory Points.

4. User Interfaces

Launcher User Interface

This portion of the application will be first presented to the user upon starting the Knight's Party Table or by returning to it after a game is completed. In order to promote simplicity, the board games will be organized in a list alphabetically. The board game names will be buttons that the users will be able to select. The following figures are rough illustrations as to how the Game Selection User Interface will look. It is worth noting that designs may change during development.

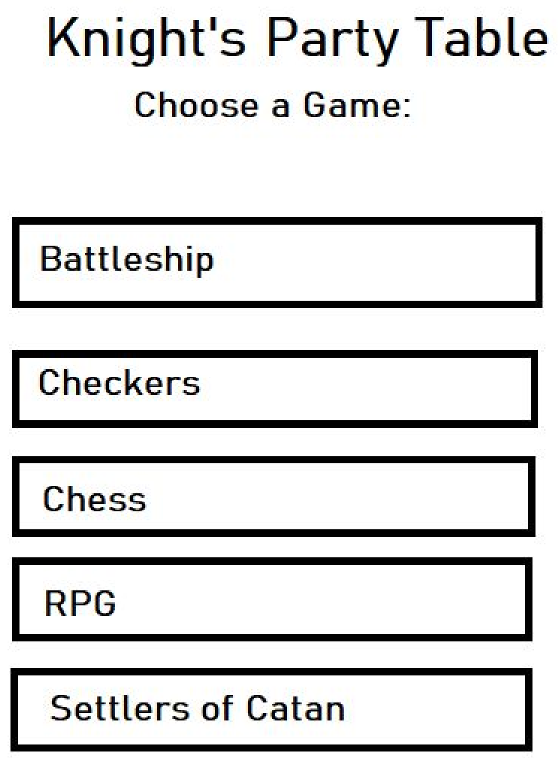

Figure VI: Launcher UI Mockup

Game User Interfaces

The code design structure of each game will be made similarly, using subclasses inherited from the superclasses Pieces. Checkers and Chess will all be able to utilize the same Board, which will make implementation easier. Many image resources, such as game boards and pieces, will be recycled from the previous implementation of the Knight's Party Table for the game's user interfaces. Resources that cannot be recycled will be created during development. For brevity's sake, a figure showcasing the GUI for each board game will not be represented in this document, as most utilize a simple checkerboard.

5. Class Definitions

This section summarizes the classes that make up the Knight's Party Table.

Battleship

Battleship is a subclass of *Game* that represents the board game Battleship. It is used by *Launcher* to play the game.

Board

Board is a class utilized by *Game* that represents a traditional game board.

Checkers

Checkers is a subclass of *Game* that represents the board game Checkers. It is used by *Launcher* to play the game.

Chess

Chess is a subclass of *Game* that represents the board game Chess. It is used by *Launcher* to play the game.

DevCard

DevCard is a class utilized by *Settlers* that represents a Development Card in the game Settlers of Catan.

Game

Game is the superclass that contains the core methods utilized by the games on the Knight's Party Table.

Launcher

Launcher is the main user interface Knight's Party Table. It utilizes the *Game* subclasses *Battleship*, *Chess*, *Checkers*, *Settlers*, and *RPG* to allow users to play said games.

Piece

Piece is a class utilized by *Game* that represents a piece on a game board. *Piece* is utilized in different ways by the different games.

Player

Player is a class utilized by *Settlers* that represents one of the players of the game. *Player* contains the amount of Resources and Development Cards usable by the player.

Resource

Resource is a class utilized by *Settlers* that represents a Resource in the game Settlers of Catan.

RPG

RPG is a subclass of *Game* that represents a traditional text-based role-playing game. It is used by *Launcher* to play the game.

Settlers

Settlers is a subclass of *Game* that represents the board game Settlers of Catan. It is used by *Launcher* to play the game. *Settlers* utilizes several classes, including *DevCard*, *Player*, *Resource*, and *Tile*.

Tile

Tile is a class utilized by *Settlers* that represents a map tile in the game Settlers of Catan.