Sheridan College

SYST17796 – Fundamental of Software Design and Development

The Card Game Project

Deliverable Design Document

Due day: June 25th, 2023

*Group number: 4*

*Student* Name: Min Liu, Sichao Quan, Shuya Hu, Wanqin Liang

# 

# 

# 

Table of Contents

[Section 1: Group Members Contract 3](#_Toc138793606)

[Section 2: UML Diagram 4](#_Toc138793607)

[Section 3: Design Document 5](#_Toc138793608)

[1. Project Background and Description 5](#_Toc138793609)

[**Project Goals and Final Vision** 5](#_Toc138793610)

[**Gameplay Description and Rules** 5](#_Toc138793611)

[Starting Base Code 5](#_Toc138793612)

[2. Project Scope 6](#_Toc138793613)

[**The names and roles of each team member** 6](#_Toc138793614)

[**The technical scope of the project** 6](#_Toc138793615)

[3. High-Level Requirements 7](#_Toc138793616)

[4. Implementation Plan 7](#_Toc138793617)

[**Task Assignment and Timeline** 7](#_Toc138793618)

[**Git Repository URL** 8](#_Toc138793619)

[**Coding Standards** 8](#_Toc138793620)

[**Tools and IDEs** 8](#_Toc138793621)

[5. Design Considerations 8](#_Toc138793622)

[1. **Encapsulation:** 8](#_Toc138793623)

[2. **Maintainability:** 9](#_Toc138793624)

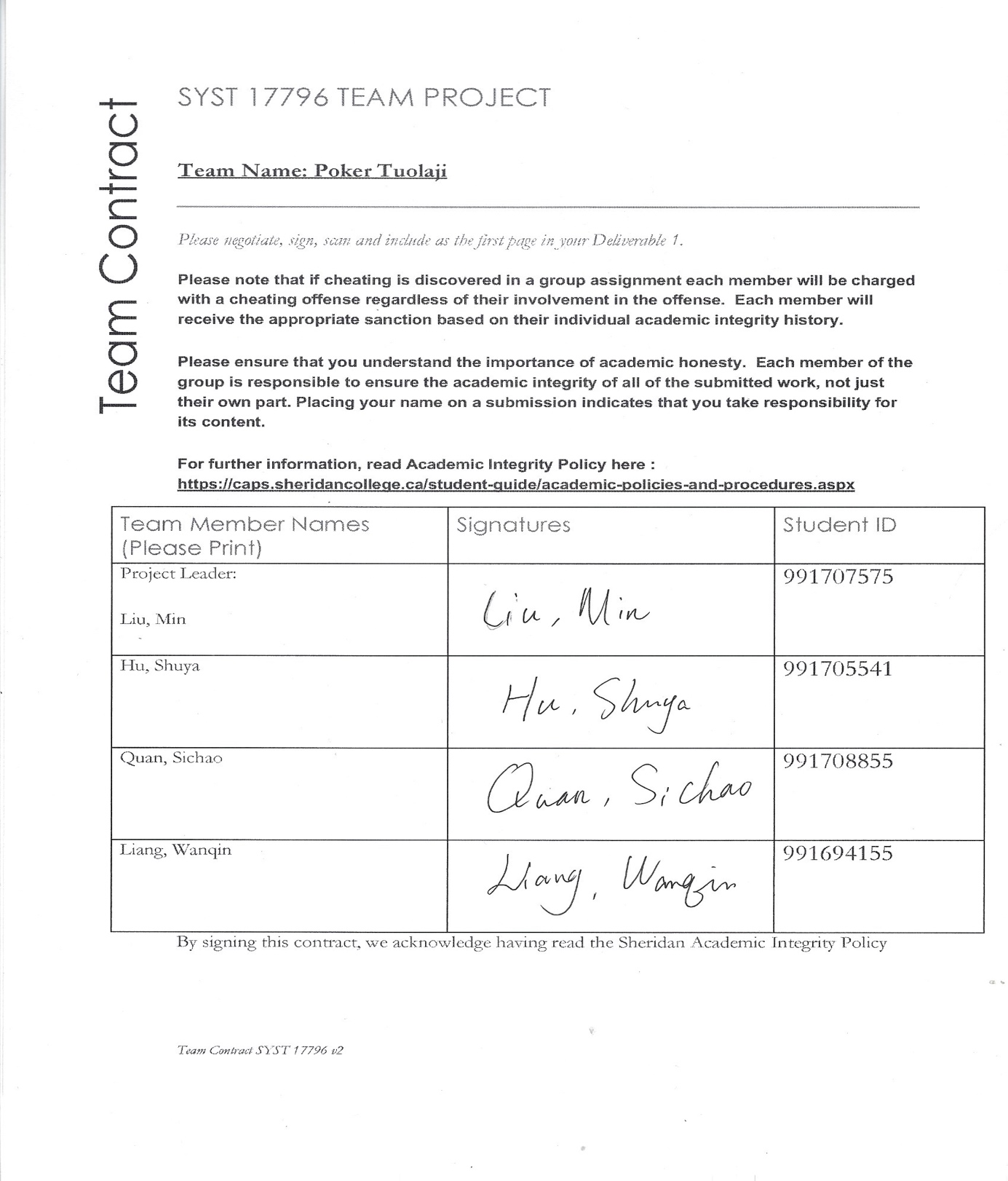
[3. **Flexibility:** 9](#_Toc138793625)

[4. **Inheritance:** 9](#_Toc138793626)

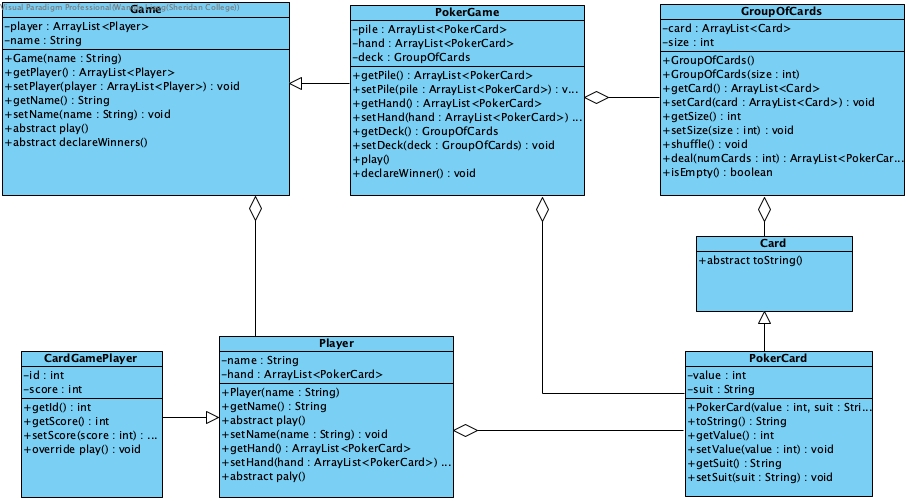
[5. Polymorphism: 9](#_Toc138793627)

# 

# Section 1: Group Members Contract



# Section 2: UML Diagram



# Section 3: Design Document

## 1. Project Background and Description

Our group would like to design a card game named “Tuolaji” in China, which is popular there. This traditional memory and matching game are like the “Match” or “Pairs of Canada. In this game, players need to have a sharp memory, good luck, and an effective strategy to win. This game can be enjoyed by both kids and adults and serves not only as a fun activity but also as a brain-training exercise.

### **Project Goals and Final Vision**

Our project's final goal is to develop a digital version of this card game, ensuring that it's engaging, intuitive, interactive, and accessible game that players can enjoy.

The game should be able to handle player turns, randomly shuffle, and distribute cards, detect matching pairs, count scores correctly, and confirm the winner based on marks. Meanwhile, we also provide a clear user interface that prompts users to implement each step.

### **Gameplay Description and Rules**

1. A deck of playing cards is shuffled and divided equally between two players at the start of the game.

2. Players take turns playing cards, one at a time. The card played is placed on top of the previous cards, forming a stack.

3. When two identical cards (like two 2s) appear consecutively, the player can claim all the cards stacked between and including the identical pair, moving them to their own pile.

4. The game continues until neither player has any more cards to play.

5. At the end of the game, the players count the cards in their piles. The player with the most cards wins.

### Starting Base Code

We designed this game based on the code that our professor has already provided and added some extra functions such as distributing the cards randomly, comparing card points, and counting cards. The current base code is written in Java, a popular language for such projects due to its simplicity and robust libraries for game development. Also, we created a Git repository for our project and stored the base code in our repository. Meanwhile, create the UML class diagram that correctly depicts the current code by Visual Paradigm. The code adheres to the Java style guide, which promotes readability and consistency. Code modularity is maintained by encapsulating specific functionalities in separate functions and classes, making it easier to maintain and scale the code in the future.

## 2. Project Scope

### **The names and roles of each team member**

We four work on this card game. According to each team member’s skills, interests, and the requirement, we divided this project into four parts, each member complete one part. Below is detail:

Liu Min is the leader of our team. She designs and builds the overall structure of the project, organizes weekly member meetings, assigns tasks, tracks the project's progress, and finds problems to improve and optimize the project.

Shuya Hu is responsible for designing the software’s architecture. Based on the code provided by the teacher, she sketched the UML and established the relationship between the different classes and created more specific child classes such as “Main, CardGamePlayer, PokerCard, and PokerGame” to implement this card game.

Sichao Quan is a software developer in our team, her role is to write the actual code based on the design provided by the software architect. She also worked on debugging and optimizing the code to make our project easy to maintain.

Wanqin Liang is a tester in the project. She executed the test plan to find bugs, report them, and ensure the project's quality as she organized relative documents to record the project's process.

### **The technical scope of the project**

This card game project developed in Java, utilizing the language's Object-Oriented (OO) principles and concepts. The application should be compatible with multiple platforms, given the cross-platform nature of Java.

Interface Design

We designed a friendly- user interface for the players to prompt the user to play, and display ‘the current card pile’ and plays’ scores.

1. Playing Field: This is the main area of the interface where cards are played and stacked. It should clearly display the current cards in the stack.

2. Players' Cards: Each player has an area on the screen where they can see their own cards but not the other player's cards.

3. Score Display: The current score, i.e., the count of cards each player has won, is displayed, and updated in real-time.

Project Completion

The project will be deemed complete when:

1. The game logic functions correctly, allowing two players to play the game with the defined rules.

2. Multiplayer functionality is implemented, allowing two users to be on the same machine.

3. The game has been thoroughly tested to ensure there are no bugs or logical errors.

4. The code follows Java's object-orientation principles, is efficient, readable, and well-documented.

Developing this card game following the Java principles will not only help in managing the complexity of the project but also make the game more flexible, and maintainable in the future.

## 3. High-Level Requirements

Game Mechanics

The system we have completed implements the defined game rules: the ability to shuffle a deck of cards, equally distribute them between two players, allow players to play cards in turns, detect matching cards, and get players' scores. The system can accurately track and manage the turn-based nature of the game.

Win/Loss Communication

This game can determine a win or loss based on the rules, i.e., the player who collects the most cards when no cards are left to play is the winner. The system can provide the game result (win, lose, or draw) to the players promptly and clearly at the end of the game.

Player Status

The program we designed can continuously track and update each player's score (the number of cards they have collected) and display each player's score in real time during the game.

User Interface

The system provides an intuitive and user-friendly interface, allowing players to easily understand and interact with the game. The interface displays the current game state clearly, including the current top card, each player's score, whose turn it is, and who win finally.

## 4. Implementation Plan

### **Task Assignment and Timeline**

Our team divided the task into manageable parts. Each team member worked on their assigned tasks and follow a strict timeline to ensure efficient progress. We also have weekly team meetings to check the status and find any issues.

### **Git Repository URL**

The URL for our Git repository is **https://github.com/Shuya-Hu/poker-game**. Each team member is expected to check in code at the end of each day to keep the repository updated and allow for collaborative work.

The repository will be structured as follows:

- `src`: This directory contains all source code for the project.

- `doc`: This directory contains all project documentation.

- `UML`: This directory stores all UML diagrams that visualize the system's structure and behavior.

- `[nbproject](https://github.com/Shuya-Hu/poker-game/tree/main/nbproject)`: This directory holds additional resources required by the project.

### **Coding Standards**

Our team follows Java's standard coding conventions. This includes naming conventions, indentation and spacing, bracket placement, and comment standards. We also follow principles of clean code, including self-descriptive naming and small functions, and make code more cohesive and loose coupling.

### **Tools and IDEs**

We use the following tools:

We use `NetBeans` for writing, testing, and debugging our code. `Git` is our version control system to manage code changes and collaboration. `Visual Paradigm (VP)` is used for creating UML diagrams to visualize our system design.

## 5. Design Considerations

All the classes in this project follow Java's OO principles:

1. **Encapsulation:** We define private fields (data) and provide public methods to access and manipulate that data. For instance, the `Player` class encapsulates player-related data like ‘name’ and `hand`, and provides methods to operate on that data (`getName()`, `getHand()`, `setHand()`). Similarly, in pokerGame class, both `value` and `suit` datas are defined private.
2. **Maintainability:** the ease with which a software system can be understood, corrected, adapted, and enhanced. The use of object-oriented programming principles, like encapsulation, inheritance, and polymorphism, improves the maintainability of the software.
3. **Flexibility:** It means how easily a system can be modified or extended. The Poker game project uses abstract classes and methods, which allows for more flexible code. For example, the `Game` class and `Player` class are both abstract, meaning they can be extended by these subclass `PokeGame` or `PokerPlayer` classes that implement their own versions of the abstract methods. This makes it easy to add new types of games or new player behaviors without having to change existing code.
4. **Inheritance:** The `Card `class is the superclass and `PokerCard` is a subclass. This is a clear example of inheritance, as `PokerCard` inherits the properties and methods of `Card` and also defines additional properties and behaviors specific to poker cards.
5. **Polymorphism:** The `Game` and `Player` classes demonstrate polymorphism. In the `Main` class, we define a `PokerGame` object (`game`) that is treated as its superclass type `Game`. This allows us to call the `play()` and `declareWinner()` methods, which will dynamically bind at runtime to the `PokerGame` class's implementation of these methods. Similarly, when we call the `play()` method on a `Player` object, it will bind at runtime to the `CardGamePlayer`'s `play()` method, because the objects stored in the `Player` list are actually instances of the `CardGamePlayer` class.