SYST 17796 Deliverable 1 - Design Document

Project Background and Description

The purpose of this project is to develop a playable version of the **War** card game using the provided base code. The **War** game is a simple two player card game played with a standard deck of 52 cards, where each player draws a card, and the one with the higher-ranked card wins the round. The game continues until one player collects all the cards or the game reaches a predefined condition (e.g., a maximum number of rounds).

Rules of the War Card Game:

Following are the steps and rules of the war card game

1. A standard deck of 52 cards is shuffled and divided evenly between two players.
2. Each player draws the top card from their deck simultaneously.
3. The player with the higher-ranked card wins both cards and adds them to the bottom of their deck.
4. If the drawn cards are of equal rank, it triggers a "War":
   - Each player places three cards face down and a fourth card face up.
   - The player with the higher fourth card wins all the played cards.
   - If the fourth cards are again equal, another War occurs.
5. The game continues until one player has all the cards or a set number of rounds is reached.

Description of the Starting Base Code

So, we were provided with the base code it is written in java and it follows Object Oriented Programming(OOPS) Java.

The base code provided for the project includes:

- **Card.java**: An abstract class that represents a playing card, defining a toString() method that must be implemented by subclasses.
- **Game.java**: An abstract class that models a generic game, with methods to play() and declareWinner().
- **GroupOfCards.java**: A class representing a collection of cards. It manages the deck size and provides a shuffle method.
- **Player.java**: An abstract class representing a player in the game, with a play() method to be implemented by subclasses.
- **Start.java**: It initializes and manages the point of game and initializes the players

The base code is written in **Java** and follows standard Java naming conventions and commenting practices, ensuring maintainability and extensibility.

Project Scope

Team Members and Roles

- **Nabia Mahmood:** Lead Developer, testing and debugging, documentation writing
- **Sepideh Pourshirazi:** Documentation writing, Game flow developement, and code integration
- **Subhkarman Singh Sidhu:** UML diagram

  (All codes and writing were discussed together)

Technical Scope

The **War** game will be implemented using Java, following object-oriented programming (OOP) principles. The interface will be a command-line-based text game where players can see the cards drawn and the winner of each round. The game will be considered complete when:

- Players can draw cards and compare values.
- The program determines round winners.
- The game tracks the number of cards each player has.
- The game declares a final winner when a player collects all the cards.

High-Level Requirements

The system must include the following features:

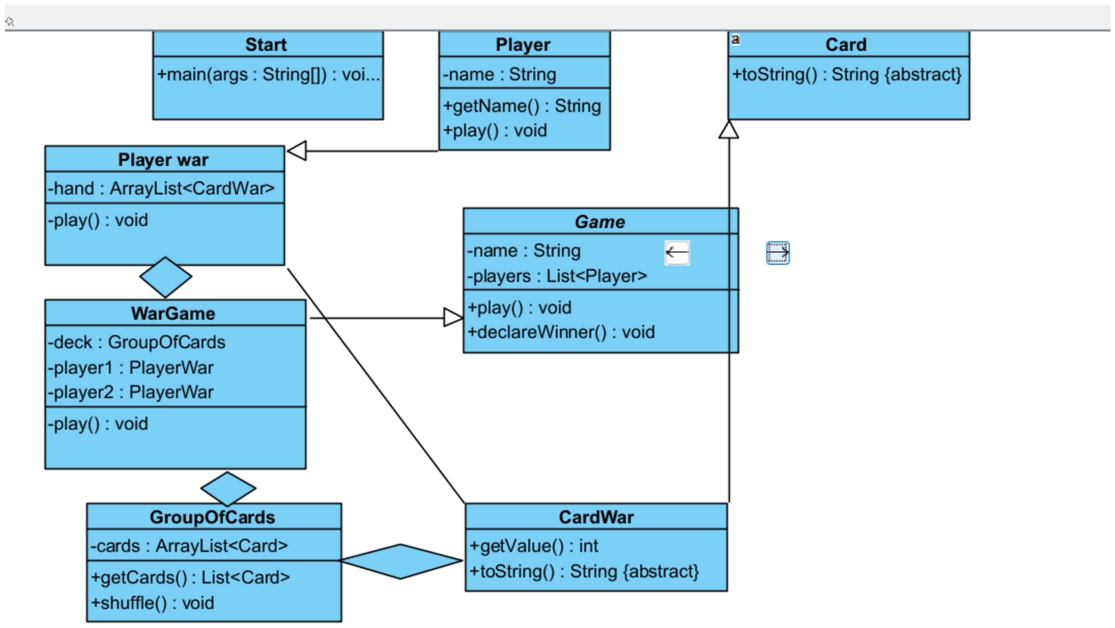
1. Ability for each player to register in the game
2. Game logic to compare card values and determine the round winner
3. Ability to track the score and number of cards for each player
4. Shuffle the deck at the beginning of the game
5. Declare the final winner when a player collects all cards
6. Use of OOP principles to extend and refactor the base code

Implementation Plan

Git Repository

- **GitHub Repository URL:** https://github.com/SepidehPourshirazi/group-7-project
- **Usage Plan:** Each developer checks in code at the end of each working session. Code files are stored in a dedicated folder structure, including source code (src/), UML diagrams (uml/), and documentation (docs/).
  - **Layout:**
    1. **Src: code files**
    2. **uml: UML diagram**
    3. **doc: documents**
    4. **tests: test cases**
- **Tools Used:**
  - **IDE:** NetBeans
  - **Version Control:** GitHub
  - **UML Diagrams**: Visual paradigm
  - **Coding Language:** java

UML DIAGRAM



Design Considerations

Encapsulation

- The Card class has private attributes and provides access via public getter methods.
- The Player class encapsulates a player's name and actions using private attributes and public methods.

Delegation

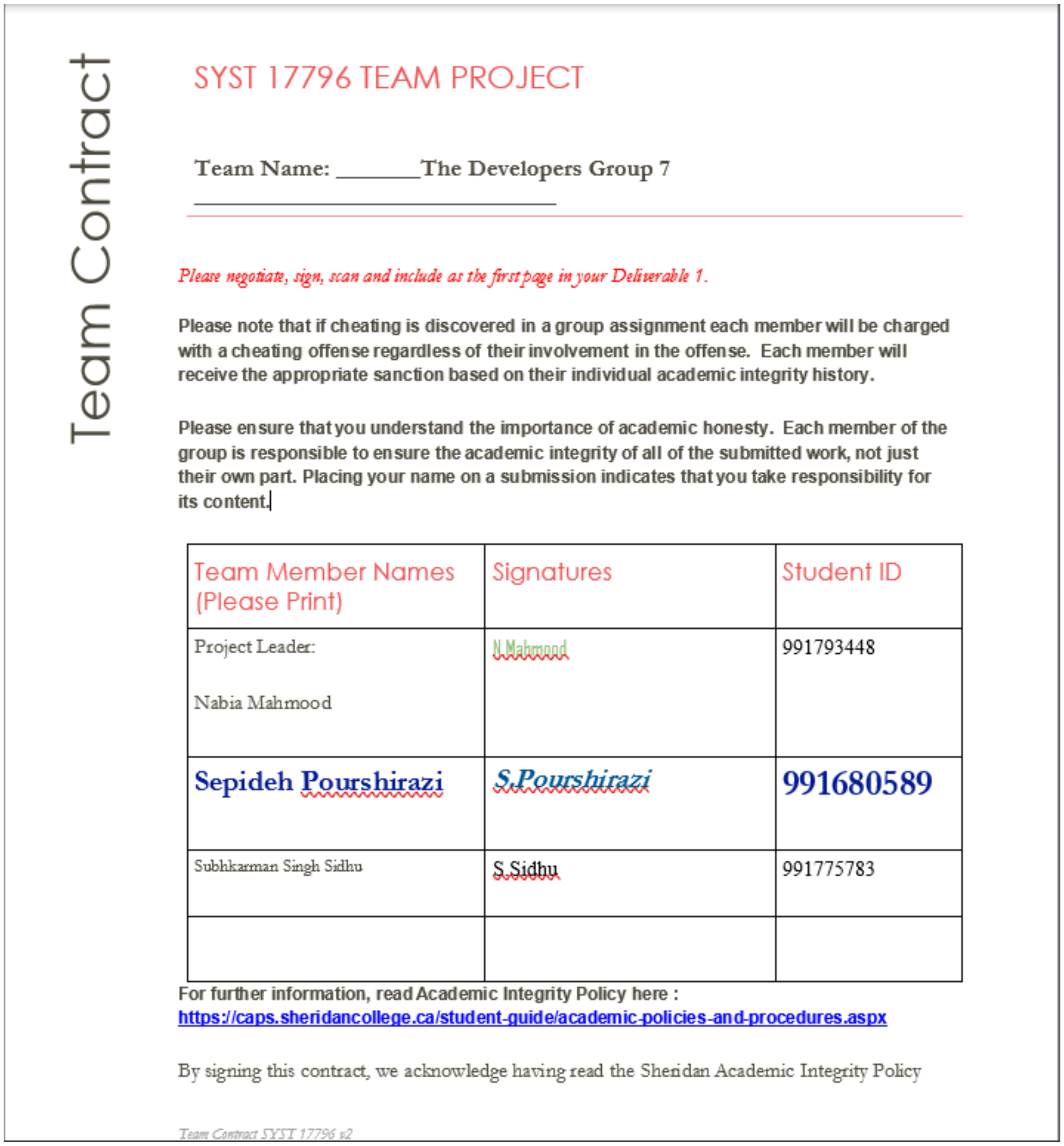
- The Game class delegates game setup and logic to subclasses that define specific rules.

- The GroupOfCards class is responsible for managing and shuffling the deck.

Flexibility and Maintainability

- The Card class is abstract, allowing different types of card games to be implemented.
- The Game class can be extended for future modifications without altering its core structure.

TEAM CONTRACT



Team Contract

SYST 17796 TEAM PROJECT

Team Name: _______The Developers Group 7
____

Please negotiate, sign, scan and include as the first page in your Deliverable 1.

Please note that if cheating is discovered in a group assignment each member will be charged with a cheating offense regardless of their involvement in the offense. Each member will receive the appropriate sanction based on their individual academic integrity history.

Please ensure that you understand the importance of academic honesty. Each member of the group is responsible to ensure the academic integrity of all of the submitted work, not just their own part. Placing your name on a submission indicates that you take responsibility for its content.

Team Member Names (Please Print)	Signatures	Student ID
Project Leader: Nabia Mahmood	N.Mahmood	991793448
Sepideh Pourshirazi	*S.Pourshirazi*	**991680589**
Subhkarman Singh Sidhu	S.Sidhu	991775783

For further information, read Academic Integrity Policy here :
https://caps.sheridancollege.ca/student-guide/academic-policies-and-procedures.aspx

By signing this contract, we acknowledge having read the Sheridan Academic Integrity Policy

Team Contract SYST 17796 v2