# Table Of Contents

# Team Contract

Team Contract

## SYST 17796 TEAM PROJECT

**Team Name:** ATSDevelopment

*Please negotiate, sign, scan and include as the first section 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 honesty 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.**

**For further information read Academic Honesty Policy on AccessSheridan or visit the faculty office and speak with the Program Support Specialist.**

| Team Member Names (Please Print) | Signatures | Student ID |
|---|---|---|
| Project Leader: Cory Salmon | Cory Salmon | 991526849 |
| Nawaphan Chayopathum | Nawaphan Chayopathum | 991511341 |
| Asher Tompson | agreement via Email (AVE) | |
| Samuel Sousa | (AVE) | |

By signing this contract, we acknowledge having read the Sheridan Academic Honesty Policy as per the link below.

https://policy.sheridanc.on.ca/dotNet/documents/?docid=917&mode=view

## Responsibilities of the Project Leader include:

- Assigning tasks to other team members, including self, in a fair and equitable manner.
- Ensuring work is completed with accuracy, completeness and timeliness.
- Planning for task completion to ensure timelines are met
- Any other duties as deemed necessary for project completion

| Scenario | Accepted Y/N + initial | We agree to do the following |
|---|---|---|
| Team member behaves in an unprofessional manner by being rude or uncooperative | C.S Y<br>NC Y<br>AT, SS(AVE) | a) Team attempts to resolve the issue by airing the problem at team meeting __<br>b) Team requests meeting with professor to problem-solve __<br>c) Team ignores behaviour __<br>d) Team agrees to avoid use of all vocabulary inappropriate to the business setting __ |
| Team member assumes or requests that his/her name be signed to a submission but has not participated in production of the deliverable | C.S Y<br>NC Y<br>AT, SS(AVE) | a) Team agrees that this is cheating and is unethical __<br>b) Friends are friends and should help each other __<br>c) Team will submit with signature but will advise professor who will take action __ |
| There is a dominant team member who is content to make all decisions on the team's behalf leaving some team members feeling like subordinates rather than equal members | C.S Y<br>NC Y<br>AT, SS(AVE) | a) Team will actively solicit consensus on all decisions which affect project direction by asking for each member's decision and vote __<br>b) Team will express subordination feelings and attempt to resolve issue __<br>c) Other: |
| Team has a member who refuses to participate in decision making but complains to others that s/he wasn't consulted | C.S Y<br>NC Y<br>AT, SS(AVE) | a) Team forces decision sharing by routinely voting on all issues __<br>b) Team routinely checks with each other about perceived roles __<br>c) Team discusses the matter at team meeting __ |

| | | |
|---|---|---|
| "ignored", or "frustrated" with a decision which affects all parties | | b) Team flips coin __<br><br>c) Other: |
| Team members do not share expectations for grade desired | C.S Y<br>NC Y<br>AT, SS CAVE | a) Team will elect one person as "standards-bearer" who has the right to ask that work be redone __<br><br>b) Team votes on each submission's quality __<br><br>c) Team will ask for individual marking and will identify sections by author __<br><br>d) Other: |

| Scenario | Accepted Y/N + initial | We agree to do the following |
|---|---|---|
| Team member does not attend team meeting | C.S Y<br>NC Y<br>AT, SS, (AVE) | (a) Team proceeds without him/her and will assign work to the absent member __<br>b) Team doesn't proceed and records team member's absence __<br>(c) Team proceeds for that meeting but "fires" member after ? occurrences __ |
| A piece of production equipment fails such as a printer, disk drive, or laptop | C.S Y<br>NC Y<br>AT, SS, (AVE) | a) Backup copies will be made and kept in the college __<br>(b) A locker or "share" directory will be used for joint access __<br>c) A photocopy and duplicate disk of all deliverables will be made __<br>d) Other: |
| An unforeseen constraint occurs after the deliverable has been allocated and scheduled (a surprise test or assignment) | C.S Y<br>NC Y<br>AT, SS, (AVE) | (a) Team meets and reschedules deliverable __<br>(b) Team will cope with constraint __<br>c) Other: |
| Team cannot achieve consensus leaving one member feeling "railroaded", | C.S Y<br>NC Y<br>AT, SS (AVE) | (a) Team agrees to abide by majority vote __ |

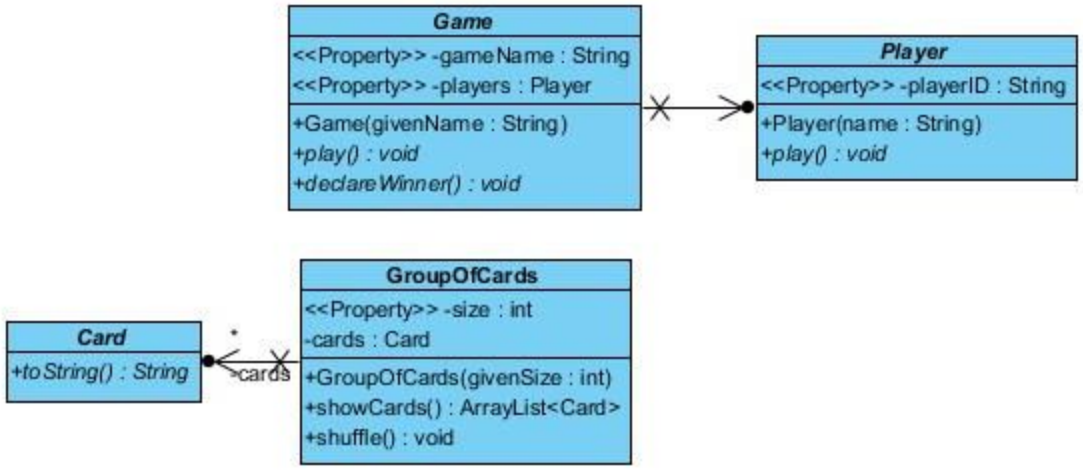
## What we will do if . . .

| Scenario | Accepted Y/N + initial | We agree to do the following |
|---|---|---|
| Team member does not deliver component on time due to severe illness or extreme personal problem | C.S. Y<br>NC Y<br>AT,SS,(AVE) | (a) Team absorbs workload temporarily __<br>(b) Team seeks advice from professor __<br>(c) Team shifts target date if possible __<br>d) Other: |
| Team member cannot deliver component on time due to lack of ability | C.S Y<br>NC Y<br>AT,SS(AVE) | (a) Team reassigns component __<br>(b) Team helps member __<br>(c) Team member must ask professor for reference material __<br>d) Other: |
| Team member does not deliver component on time due to lack of effort | C.S Y<br>NC Y<br>AT,SS (AVE) | (a) Team absorbs workload __<br>(b) Team "fires" team member by not permitting his/her name on submission __ |

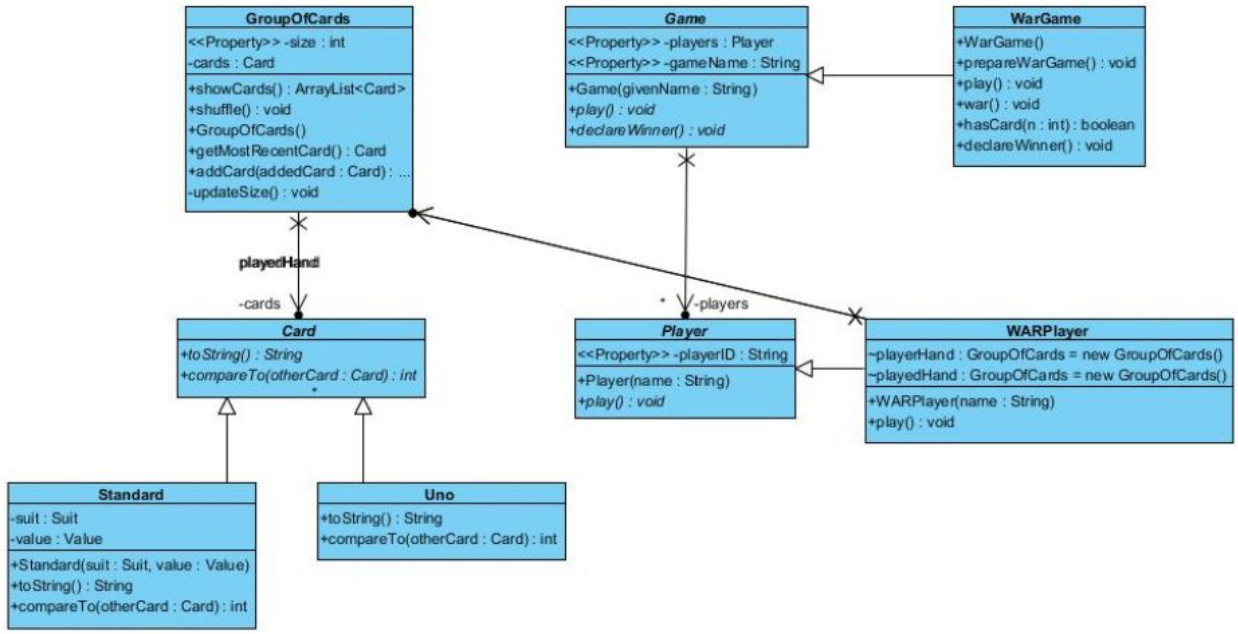
| | | |
|---|---|---|
| | | c) Other: |

# UML Diagrams

## Base UML Class Diagram:




## Advanced UML Class Diagram:



# Design Document

## 1. Project Background and Description

**Project goal**: To create the card game “War” with Object Oriented concepts with the main goal of adhering to main principles like that of strong cohesion and loose coupling.

**Base Code:** The base code includes multiple abstract classes aimed towards being able to reuse the code mainly without modification. Classes like Card for instance are abstract so that each type of card will be a child class with its own specific characteristics. The Java code is split up into specific defined roles to follow the convention of strong cohesion and loose coupling leaving 4 major parts of the code from which we will be working.

It seems that the code as it stands now is merely a template for the logic of a game and will be run from another class. This allows for easy transitioning to a GUI.

**Rules of War**:
Value of cards: Ace, King, Queen, Jack, 10, 9, 8, 7, 6, 5, 4, 3, 2 (Highest to Lowest)
Number of players: 2
How to play war:

1. The deck is shuffled
2. The deck is split evenly between two players
3. In each turn, both players present their top card and place them in on the table
4. The player with the highest card value wins that turn and takes both cards to add to the bottom of their deck
    a. If both cards shown are the same value, it is War. In War,
        i. Each player places a card turned up and one card turned down into the pile
        ii. Whichever player has the visible card with the most value, wins all cards played.
        iii. If there is another draw, a new card facing up and new card facing down are added with all old cards remaining but facing down and the process repeats.
4. The first player to win all 52 cards is the winner.

Reference for the rules:
https://www.bicyclecards.com/how-to-play/war/

# 2. Project Scope

**Chayopathum, Nawaphan -** Main responsibility on WarGame Class and necessary child classes, Design Document preparation
**Salmon, Cory -** Main responsibility on Card Class and necessary child classes, Design Document preparation, Repository Management
**Sousa, Samuel -** Main responsibility on GroupOfCards Class and necessary child classes, Design Document preparation
**Thompson, Asher -** Main responsibility on WarPlayer Class

## Technical Scope

This game will be feature complete when players can decide how many players each game will have, can play the game to completion and will run into no errors along the way. Players should be able to understand what is going on just as they would in real life.

To go above and beyond, a gui may be added after this has been completed

# 3. High-Level Requirements

## Critical Game Features

- War mode, where when 2 players have the same values of playing cards. Here, players will enter a different mode of play
- Winning conditions. Here, whichever player gets all 52 cards first wins
- Ability for players to register with their own names in game
- Ability for players to check their amount of cards that they have in their deck at anytime
- A display of the number of cards in each players deck and hand alongside the visible cards and visible war cards.

# 4. Implementation Plan

- Git repository URL: https://github.com/CS-Username/SYST-SoftwareDesign-GroupProject-1

## Expected Use

- The repository manager will be Cory. Each team member will do their part on their own branch and the code will be merged to the master branch by Cory
- Each type of data will have its own folder

        - The documentation will be saved in the SYST-SoftwareDesign-GroupProject-1/Documents directory
        - The UML diagram will be saved in the SYST-SoftwareDesign-GroupProject-1/vpproject/ directory
        - The Code will be saved in the SYST-SoftwareDesign-GroupProject-1/src/ca/sheridancollege/project/ directory
    - Coding Standards
        - Strong cohesion and loose coupling is a standard key in coding. Only elements that are strongly related will be part of their own classes and classes will coded in a modular factor being as independent as possible (for instance, cards will use enums to avoid relying on other classes)
        - Code will be commented on reasonably to ensure future revisions are done easily
        - Developers will communicate about large general changes through a central chat system as well as labelling all pull requests
    - Tools
        - Github will be used to manage our central git repository
        - Netbeans will be used as the ide to create and modify code
        - Visual Paradigm will be used to create class diagrams and generally outline and plan.

# 5. Design Considerations

- **Encapsulation**: Encapsulation ensures that variables are private and therefore are protected from outsiders
  Examples:
    - Standard Class: The Suit and Value is set to private.
    - GroupOfCards Class: The size and ArrayList is set to private.
    - Game: players and gameName are set to private
- **Delegation**: Delegation ensures strong cohesion and loose coupling by delegating roles to classes.  Each class has a responsibility that mimics the real world counterpart.
    - Standard Class: mimics a standard deck of cards
    - WarGame Class: Mimics the rules and structure for play of the game war
    - GroupOfCards mimics a stack, or deck of cards including the ability to shuffle said deck.
- **Flexibility/Maintainability**:
    - Standard Class: This class is a child class and allows this type of card deck to be changed seperate to any other card type using the Card class. Using Enums prevents the Suit and Value from being invalid leaving any issue of parsing to another class  and also makes the class less dependant on the methods of other classes.

- WarGame Class: This class is a child class and allows this game to be changed seperate to any other game type using the Game class
- All code will be reasonably commented on so that its always clear how code is created.
- Class diagrams that are up to date alert users of any interactions to be aware of.