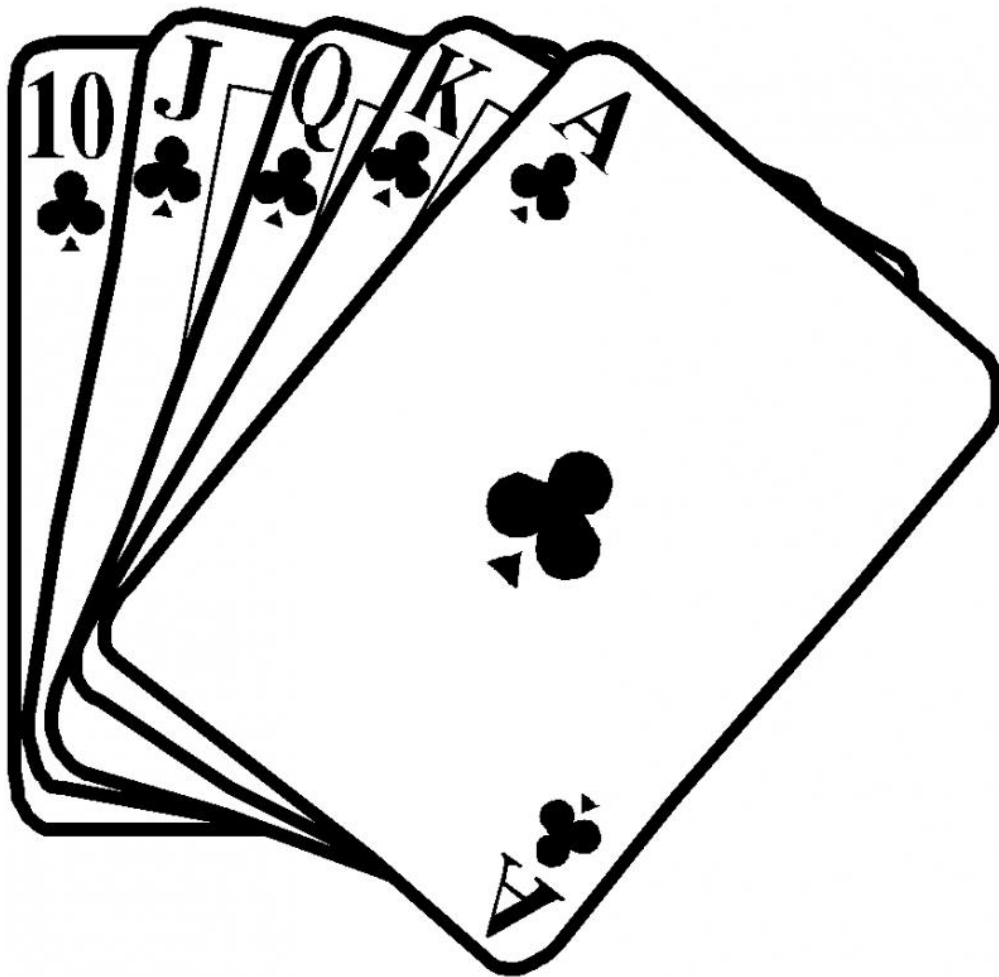


SYST 17796 Project

Deliverable 1

Version 1.0 • 6/13/2019



*Muaz Ahmad
Daniel Minami
Stuart Bollinger*

Table of Contents

Contents

SYST 17796 Project Deliverable 1 Version 1.0 • 6/13/2019	1
1. Team Contract.....	3
2. Project Description.....	8
2.1 Project Background and Description.....	8
2.1.1 Card Game – President - Rules/How to Play:	8
2.1.2 Project Goal:.....	9
2.1.3 Base Code:	9
2.2 Project Scope:	10
2.3 High Level Requirements:.....	10
2.4 Implementation Plan:.....	11
2.4.1 Project Repository:	11
2.4.2 Coding check Points and Planning.....	11
2.4.3 Repository Rules	11
2.4.4 Project Documents	11
2.4.5 Coding Standards.....	12
2.5 Design Considerations.....	12
3. UML Diagram.....	13

1. Team Contract

Team Contract

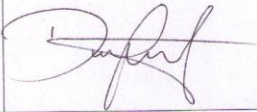

SYST 17796 TEAM PROJECT

Team Name: Quad Aces ♥ ♦ ♠ ♣

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.

Team Member Names (Please Print)	Signatures	Student ID
Project Leader: Daniel Minami	 DM	991 548 108
Stuart Bollinger	 SB	991 555 191
Muaz Ahmad		991 528 416

Brent Sterling		
---------------------------	--	--

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

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 does not attend team meeting <i>that they agreed to attend.</i>	SB (Y) Dm (Y) MA (Y)	a) Team proceeds without him/her and will assign work to the absent member ✓ b) Team doesn't proceed and records team member's absence c) Team proceeds for that meeting but "fires" member after <u>3</u> occurrences ✓
An unforeseen constraint occurs after the deliverable has been allocated and scheduled (a surprise test or assignment)	SB (Y) Dm (Y) MA (Y)	a) Team meets and reschedules deliverable ✓ b) Team will cope with constraint ✓ <i>Within reason</i> c) Other:
Team cannot achieve consensus leaving one member feeling "railroaded", "ignored", or "frustrated" with a decision which affects all parties	SB (Y) Dm (Y) MA (Y)	a) Team agrees to abide by majority vote ✓ <i>If tied 12-2</i> b) Team flips coin ✓ c) Other:
Team members do not share expectations for grade desired	SB (Y) Dm (Y) MA (Y)	a) Team will elect one person as "standards-bearer" who has the right to ask that work be redone ✓ b) Team votes on each submission's quality ✓

Scenario	Accepted Y/N + initial	We agree to do the following
Team member behaves in an unprofessional manner by being rude or uncooperative	SB (Y) DM (Y) MA (Y)	a) Team attempts to resolve the issue by airing the problem at team meeting ✓ b) Team requests meeting with professor to problem-solve ✓ c) Team ignores behaviour ____ d) Team agrees to avoid use of all vocabulary inappropriate to the business setting ✓
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	SB (Y) DM (Y) MA (Y)	a) Team will actively solicit consensus on all decisions which affect project direction by asking for each member's decision and vote ✓ b) Team will express subordination feelings and attempt to resolve issue ✓ c) Other:
Team has a member who refuses to participate in decision making but complains to others that s/he wasn't consulted	SB (Y) DM (Y) MA (Y)	a) Team forces decision sharing by routinely voting on all issues ✓ b) Team routinely checks with each other about perceived roles ✓ c) Team discusses the matter at team meeting ✓

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	SB (Y) DM (Y) MA (Y)	a) Team absorbs workload temporarily ✓ b) Team shifts target date if possible ✓
Team member cannot deliver component on time due to lack of ability	SB (Y) DM (Y) MA (Y)	a) Team reassigns component ✓ b) Team helps member ✓
Team member does not deliver component on time due to lack of effort	SB (Y) DM (Y) MA (Y)	a) Team absorbs workload ____ b) Team "fires" team member by not permitting his/her name on submission ✓

2. Project Description

2.1 Project Background and Description

2.1.1 Card Game – President - Rules/How to Play:

Object of the game: Play all your cards before the other players to win.

Card rank hierarchy (highest to lowest): 2 A K Q J 10 9 8 7 6 5 4 3

Number of players: 4

How to play: [https://www.wikihow.com/Play-President-\(Card-Game\)](https://www.wikihow.com/Play-President-(Card-Game))

- All 52 cards are dealt evenly (13 each).
- The player with the 3 of clubs plays first (Must play 3 of clubs).
- The game is played in a sequence of tricks.
- The player who starts a trick, can play a single, double, triple or quadruple of the same card.
- The next player must play the same number of cards, of equal or greater rank, or play a 2.
- If a player cannot play, or they don't want to, they may pass.
- A player who passes is not permitted to play for the duration of the trick
- A trick can end 3 ways:
 - All other players pass, the last player to play starts the next trick.
 - Playing a 2, that player starts the next trick
 - Playing equal rank card(s) that the player before played, that player starts the next trick.
- The game ends once 3 players have played all cards.

2.1.2 Project Goal:

The goal of this project is to program the card game President through the programming language Java while using the IDE NetBeans. We plan to individually work on separate classes and reviewing/editing the code written by other group members. Throughout the development of this program, we will update the repository on GITHUB to ensure everyone's code is updated and no data is lost. The game will be played locally via the console, with 4 players passing the device between turns. Ideally, the code will not contain any run-time-errors caused by the user input.

2.1.3 Base Code:

The starting base code is composed of 11 Classes and 3 Enumerations:

- **Game:** the main Class, will invoke methods from the other classes to ensure high cohesion is achieved.
- **Rank:** Enumeration containing the 13 values in a standard deck.
- **Suit:** Enumeration containing the 4 suits in a standard deck.
- **Card:** Class that creates objects comprised of Suit and Rank.
- **GroupOfCards:** Class that includes an array of Cards for other classes to access.
- **Hand:** Class that creates an array of 13 Cards using GroupOfCards Class, the array gets smaller as cards are played.
- **Deck:** Class that creates an array of all 52 cards using GroupOfCards Class, shuffles them into a random order, and deals 13 to each player.
- **Board:** Class that displays an array of the Cards played by the previous player using GroupOfCards class and Turn class, composed of 1 to 4 elements.
- **Player:** Class composed of the players Name, and their Hand.
- **Round:** Class that manages the game status, winning order and a series of Tricks.
- **Trick:** Class that manages a series of player Turns,
- **Turn:** Class that manages player actions: the cards they play and turn pass.
- **Error Message Class and Error Enumeration:** will manage exception handling and error message displays.

2.2 Project Scope:

Group members: Daniel Minami, Muaz Ahmad, Stuart Bollinger

Our group has agreed to try and equally contribute to coding the card game.

Daniel Minami (Project Leader): Responsible for ensuring code follows proper standards.

Muaz Ahmad: Responsible for ensuring timelines are in order.

Stuart Bollinger: Responsible for editing and submitting deliverables.

The project will be complete once we have met the requirements outlined in each of the deliverables. Once we have a functioning version of the game coded, we will conduct test games to confirm it is correct and fix any bugs we can find.

2.3 High Level Requirements:

- When a game is started, the console will prompt the user to enter the names of each player.
- At the beginning of each players' turn, a prompt will ask that player if they are ready (device has been passed).
- Once the player has confirmed they are ready, the console will display the cards in their hand and the cards currently on the board.
- The console will prompt the user to type in the cards they wish to play or pass their turn.
- Once three players have played all of the cards in their hand: the game is over, at which point, the console will display the winning order.

2.4 Implementation Plan:

2.4.1 Project Repository:

GIT Repository URL:

https://github.com/danielminami/Sheridan_SYST17796_President

2.4.2 Coding check Points and Planning

- In person project meetings will occur every 15 days.
- Code goals will be defined weekly by the team and divided by the members
- Agile principles of breaking down the game rules into User Stories and tasks will be used to divide the work

2.4.3 Repository Rules

- Each team member will have a copy of the repository into their own terminal
- Check in and merges will happen in the team meetings
- Each developer will work into their own branch

* Repository rules might be changed to accommodate the team needs as the team strives for efficiency

2.4.4 Project Documents

- Project documents will be stored by the member Muaz into the Google Docs account as the group prefers to maintain these files separated from the Source Code
- GitHub will only contain the project Source Code
- All the members will have access to this folder to modify the files both in GitHub and Google Docs

2.4.5 Coding Standards

- This project will be implemented using NetBeans Project Structure
- The code standards will follow the conventions given in the Java 2 course:
<http://www-acad.sheridanc.on.ca/~jollymor/standards.html>
- Project Diagram will be made using Visual Paradigm
- Our group expects to use automated tests as JUnit is part of the second semester class content

2.5 Design Considerations

The structure of our current code revolves around high cohesion and loose coupling as each class is maintained to provide a specific set of rules which will be implemented into the game. The set of OOP principles we've included are:

- **Encapsulation** - The classes in the UML diagram shows that the fields are private. Accessors and mutators are implemented to handle the data in the object. The enums Rank and Suit are also an example of encapsulation, once the values and ranks are accessed through the respective accessor methods.

- **Association** - Shown between many classes where a class has an instance of another related class (Ex. The Turn class is associated with the Player class since the Turn class uses an instance of the Player class called "player")

- **Delegation** - The Classes were designed with loose coupling by delegating responsibilities to different classes. As an example, the Class Turn controls only the current player action. The control of the Cards played in the turn change was delegated to the Class Board. Another example of delegation is the ErrorMessage class that was designed to handle all the error messages, instead of dealing with the messages in the class.

- **Flexibility/Maintainability** - The use of inheritance is a good practice that improves the software flexibility and maintainability by avoiding code repetition and allowing extension. An application of this principle is in the classes Board, Deck and Hand that inherits the members of the parent class GroupOfCards. This allows the child classes to reuse the parent Class. With this class design, another card game, for instance Uno, could be implemented by just creating a new type of Card.

****This code/design is subject to change overtime as well as the OO principles****

3. UML Diagram

Visual Paradigm Professional (Daniel Minami (Sheidan College))

