SYST 17796 Project:

Deliverable 1

For Faculty

Prepared by PO1_1205_11044-7

Sheridan College

Submitted:

19 June 2020

Table of Contents

Contents Page

Team Contract

SYST 17796 TEAM PROJECT

Team Name: PO1_1205_11044-7

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: Harleen Kaur Grewal	H.K.G.	991590305
Puru Dhingra	P.D.	991592033
Ann Rachel Panicker	A.R.P.	991585951
Vikas Rai	V.R.	991590584

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

What we will do if . . .

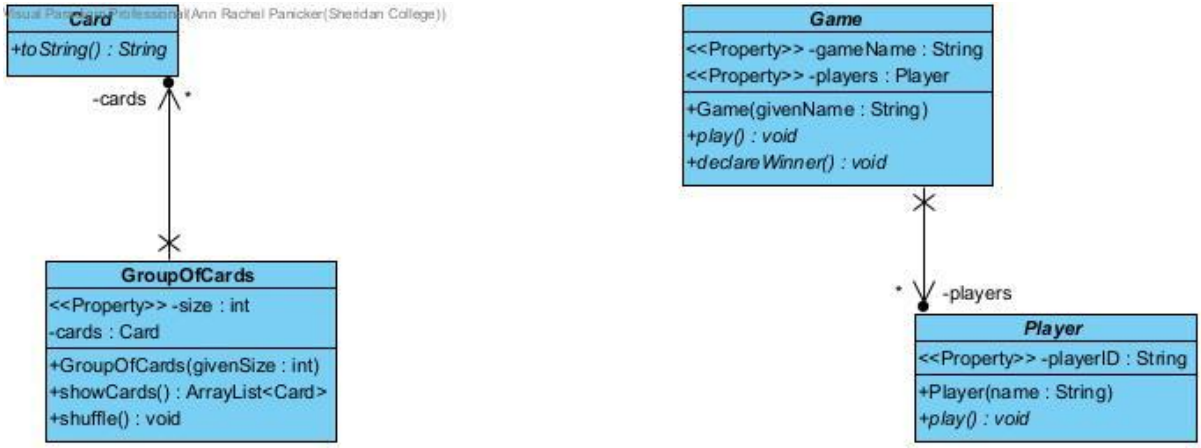
Scenario	**Accepted Y/N + initial**	**We agree to do the following**
Team member does not deliver component on time due to severe illness or extreme personal problem	Accepted **a)** Y- Harleen Y- Puru Y-Ann Y-Vikas	**a) Team absorbs workload temporarily __** b) Team seeks advice from professor __ c) Team shifts target date if possible __ d) Other:
Team member cannot deliver component on time due to lack of ability	Accepted **b)** Y- Harleen Y- Puru Y-Ann Y-Vikas	a) Team reassigns component __ **b) Team helps member __** c) Team member must ask professor for reference material __ d) Other:
Team member does not deliver component on time due to lack of effort	Accepted **a)** Y- Harleen Y- Puru Y-Ann Y-Vikas	**a) Team absorbs workload __** b) Team "fires" team member by not permitting his/her name on submission __ c) Other:

Scenario	**Accepted Y/N + initial**	**We agree to do the following**
Team member does not attend team meeting	Accepted **a)** Y- Harleen Y- Puru Y-Ann Y-Vikas	**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 __ occurrences __
A piece of production equipment fails such as a printer, disk drive, or laptop	Accepted **c)** Y- Harleen Y- Puru Y-Ann Y-Vikas	a) Backup copies will be made and kept in the college __ b) A locker or "share" directory will be used for joint access __ **c) A photocopy and duplicate disk of all deliverables will be made __** d) Other:
An unforeseen constraint occurs after the deliverable has been allocated and scheduled (a surprise test or assignment)	Accepted **b)** Y- Harleen Y- Puru Y-Ann Y-Vikas	a) Team meets and reschedules deliverable __ **b) Team will cope with constraint __** c) Other:
Team cannot achieve consensus leaving one member feeling "railroaded", "ignored", or "frustrated" with a decision which affects all parties	Accepted **a)** Y- Harleen Y- Puru Y-Ann Y-Vikas	**a) Team agrees to abide by majority vote __** b) Team flips coin __ c) Other:

Team members do not share expectations for grade desired	Accepted **b)** Y- Harleen Y- Puru Y-Ann Y-Vikas	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 __** c) Team will ask for individual marking and will identify sections by author __ d) Other:

Scenario	**Accepted Y/N + initial**	**We agree to do the following**
Team member behaves in an unprofessional manner by being rude or uncooperative	Accepted **a)** Y- Harleen Y- Puru Y-Ann Y-Vikas	**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 __
Team member assumes or requests that his/her name be signed to a submission but has not participated in production of the deliverable	Accepted **a)** Y- Harleen Y- Puru Y-Ann Y-Vikas	**a) Team agrees that this is cheating and is unethical __** b) Friends are friends and should help each other __ 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	Accepted **a)** Y- Harleen Y- Puru Y-Ann Y-Vikas	**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	Accepted **a)** Y- Harleen Y- Puru Y-Ann Y-Vikas	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 __**

UML Diagram


Card
+toString() : String
-cards
*
GroupOfCards
<<Property>> -size : int
-cards : Card
+GroupOfCards(givenSize : int)
+showCards() : ArrayList<Card>
+shuffle() : void
Game
<<Property>> -gameName : String
<<Property>> -players : Player
+Game(givenName : String)
+play() : void
+declareWinner() : void
*
-players
Player
<<Property>> -playerID : String
+Player(name : String)
+play() : void

Design Document Template

Project Background and Description

Project goal- To design a card game named as “War”.

Final Vision- To make something, which anyone, who has limited knowledge about card games, can play.

War is a simple card game which is played by two players using a standard playing card deck. The goal is to be the first player to win all 52 cards. “The deck is divided evenly, with each player receiving 26 cards, dealt one at a time, face down.

Each player turns up a card at the same time and the player with the higher card takes both cards and puts them, face down, on the bottom of his stack. If the cards are the same rank, it is War. Each player turns up one card face down and one card face up. The player with the higher cards takes both piles (six cards). If the turned-up cards are again the same rank, each player places another card face down and turns another card face up. The player with the higher card takes all 10 cards, and so on.” (Bicycle Playing Cards, “n.d.”)

The Base or the Project Starter Code given is written in Java. There are 4 classes which are Card, GroupOfCards, Player and Game. Currently, every class except the GroupOfCards is an abstract class. The GroupOfCards is a concrete class which has a parametrized constructor and methods which are showCards(), shuffle(), getSize() and setSize(). The Player class is the one which models a player. It has a parametrized constructor, getPlayerID() and setPlayerID methods including an abstract play() method. The Game class is the one which models our game. It has a constructor, getGameName(), getPlayers(), setPlayers(), play() and declareWinner() methods.

Project Scope

There are 4 members in our team.

Harleen Grewal- Responsible for managing Git repository and organizing it.

Puru Dhingra- Responsible for documents that are going to be submitted.

Ann Rachel Panicker- Responsible for class diagram and testing.

Vikas Rai- Responsible for checking code after each week.

This project will include 3 deliverables.

Due Dates	Deliverables
Week 6	Deliverable 1
Week 10	Deliverable 2
Week 13	Deliverable 3

Deliverable 1 includes team contract, UML diagram, Git repository and design document template.

Deliverable 2 includes requirements, UML diagram showing the extension planned, Use Cases, written description explaining OOD choices.

Deliverable 3 includes completed code and tests.

High-Level Requirements

1. The game itself will decide whose turn it is to begin.
2. Our game would display the rank of the card as well as who gets to take the cards every time the players turn up cards.
3. Players would see their win/loss/draw message at the end of game
4. The names of players and their win/lose/draw status would be recorded each time they play.
5. Players can communicate with each other through chat in between the game.

Implementation Plan

Our Git Repository URL is: https://github.com/Harleen1881/Project_Card_Game

Each team member keeps check in the code after every 2-3 days. Our text files are stored under a separate directory, code, UML diagrams have their own folders etc. For coding standards, we'll be using different tools such as NetBeans, Intellij Idea Software, Computer Assisted Software Engineering (CASE) tool such as Visual Paradigm, etc.

Design Considerations

- **Encapsulation:**

  Encapsulation is the process of binding data fields and methods in a single class so that it is protected from external interference. Encapsulation even helps in implementing loose coupling.

  For example: Our *GroupOfCards* class has methods like shuffle() and setSize() whose definitions vary according to the card game we play. Grouping is different for different card games. Since, all grouping methods are in a single class, we only need to make changes in *GroupOfCards* class, not in any of the remaining three.

- **Delegation**

  We have used 4 classes to write the code for War game where each class performs a specific task. *Card* class is the base class for our project. It has a toString() method which returns a String representation of our card. *Game* class returns the game name and list of players. It also contains methods for playing the game and displaying the winner. *GroupOfCards* is a class used for grouping cards which can then be assigned to each player. *Player* class assigns a unique ID for each player.

- **Flexibility/Maintainability**

  Each class that we have contains the base code which can be used for any card game. We can modify the code according to the game that we are going to design.

  For example: The size variable in *GroupOfCards* is flexible because each card game has a different grouping of cards. The play() method implements different logic for each game.

References

- Bicycle Playing Cards (n.d.) War – Card Game Rules. Retrieved from https://bicyclecards.com/how-to-play/war/