syst 17796 project

Deliverable 1

Clash of cards-WAR

Group Name: The Warriors

By: Kushagra Rajdev
Jay Patel
Ashutosh Rana
Mehakdeep Kaur

Table of Contents

Team Contract

SYSTSYST 17796 TEAM PROJECT Team Name: _____The Warriors _____SYST 17796 TEAM PROJECT 17796 TEAM PROJECT

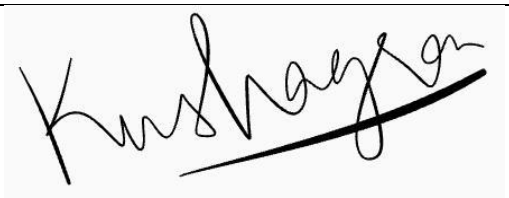
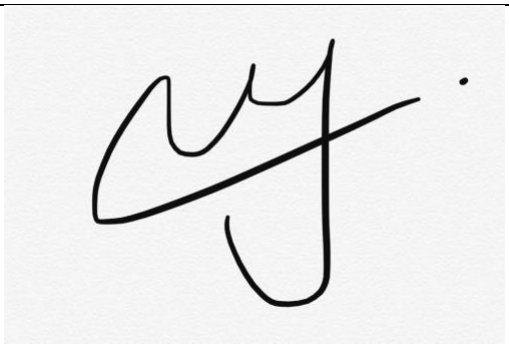
Team Contract

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: Kushagra Rajdev		991592180
Jay Patel		991592044
Ashutosh Rana	Ashutosh Rana	991585298
Mehakdeep Kaur	Mehakdeep Kaur	991591857

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	Initial: **a** Kushagra: **Y** Jay: **Y** Mehakdeep: **Y** Ashutosh: **Y**	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	Initial: **b** Kushagra: **Y** Jay: **Y** Mehakdeep: **Y** Ashutosh: **Y**	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	Initial: **b** Kushagra: **Y** Jay: **Y** Mehakdeep: **Y** Ashutosh: **Y**	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	Initial: **a** Kushagra: **Y** Jay: **Y** Mehakdeep: **Y** Ashutosh: **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 __ occurrences __
A piece of production equipment fails such as a printer, disk drive, or laptop	Initial: **b** Kushagra: **Y** Jay: **Y** Mehakdeep: **Y**	a) Backup copies will be made and kept in the college __ b) A locker or "share" directory will be used for joint access __

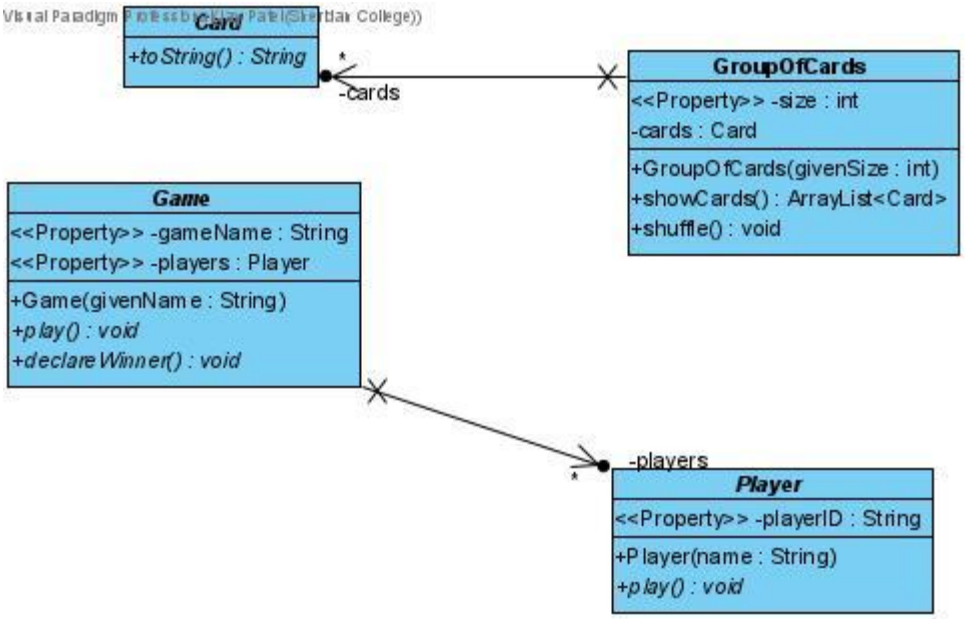
	Ashutosh: **Y**	c) A photocopy and duplicate disk of all deliverables will be made __ d) Other:
An unforeseen constraint occurs after the deliverable has been allocated and scheduled (a surprise test or assignment)	Initial: **b** Kushagra: **Y** Jay: **Y** Mehakdeep: **Y** Ashutosh: **Y**	a) Team meets 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	Initial: **a** Kushagra: **Y** Jay: **Y** Mehakdeep: **Y** Ashutosh: **Y**	a) Team agrees to abide by majority vote __ b) Team flips coin __ c) Other:
Team members do not share expectations for grade desired	Initial: b Kushagra: **Y** Jay: **Y** Mehakdeep: **Y** Ashutosh: **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 __ 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	Initial: **d** Kushagra: **Y** Jay: **Y** Mehakdeep: **Y** Ashutosh: **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 __
Team member assumes or requests that his/her name be signed to a submission but has not participated in production of the deliverable	Initial: **a** Kushagra: **Y** Jay: **Y** Mehakdeep: **Y** Ashutosh: **Y**	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	Initial: **b** Kushagra: **Y** Jay: **Y** Mehakdeep: **Y** Ashutosh: **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	Initial: **b** Kushagra: **Y** Jay: **Y** Mehakdeep: **Y** Ashutosh: **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 __

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

Goal: To create the card game – WAR as an extension of the code provided by the professor.

Final Vision: To program a game using the principles of software design and development enabling the user(s) to play it on computer without facing any complications such as program crash, limitation on number of players etc.

About the card game- WAR

The deck is divided evenly between the players, dealt one at a time, face down. Anyone may deal first. Each player places their stack 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 deck.
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.

The game ends when one player has won all the cards.(*War- Card Game Rules*, n.d., page 1)

Starter Code

The starter code provided by the professor is a flexible code written in Java programming language. It contains 1 normal class and 3 abstract classes (Game, Card, GroupOfCards and Player respectively) each having their starting variables along with their accessors and modifiers encapsulated appropriately. Comments about the role of every class and the functions inside them have also been provided. Some of the important functions such as play(), declareWinner(), shuffle() etc have also been declared in the code.

Project Scope

Group Members and Their Respective Roles: -

1. **Jay Patel –** Repository Creation and Management
2. **Kushagra Rajdev –** UML Diagram Creation
3. **Ashutosh Rana –** Design Template Creation.
4. **Mehakdeep Kaur –** Final Document Compilation and coordination with the other members

Scope of The Program

The interface of the program will provide the choice of the number of players in the game and it will register them by taking in their game IDs. The player having their turn will be shown their IDs, score (i.e. the number of cards in their deck) and controls over their screen along with the cards flipped by the other players. At the end of the game a scoreboard will be displayed containing the results.

High Level Requirements

The game must have the following:

- Ability to accept any reasonable number of players
- Ability to register the players with their unique IDs
- Ability to communicate a player's win or loss at the end of the game
- Ability to display a player's scores upon their turn
- Ability to provide a scoreboard at the end of the game with the results
- Ability to store the scores as history of the game

Implementation Plan

Git Repository Link: https://github.com/Patel1198/The-Clash-Of-Cards-WAR-.git

Programs used:

- NetBeans 8.2 with JDK 1.8 / NetBeans 11.3 with JDK 11+
- Junit 5, 4
- Visual Paradigm

The repository currently contains the starter code i.e. the four classes. Separate folders have been created for Texts, Domain Class Diagrams and Coding.

The members of the group will be assigned to code different parts (functions or classes) which will be done in their respective branch of the class and after proper inspection, it will be merged with the main remote repository.

Design Considerations

Structure of The Current Code

The present code or the current code contains one normal class Game and three abstract classes Card, GroupOfCards and Player. Implemented Principles of the started code:

- Cohesion and Delegation: The current code is divided into 4 classes each having their own respective roles. For example:
  1. Card abstract class indicates the inherited class about the properties that a card should have in the game such as its suit and number.
  2. The GroupOfCards abstract class indicates the inherited classes about how the cards will be distributed and shuffled.

  This shows that the code is has high cohesion and is flexible enough to be developed into any card game.

- Encapsulation: All the variables in all the class are declared as private to make them inaccessible out of the class. They can only be accessed by their respective accessors and modifiers (getters and setters) which are declared public.

For e.g.: playerID variable in player class is private and has getPlayerID() and setPlayerID() as its getter and setter respectively.

Another example is the size variable in GroupOfCards class which indicates the size of a deck in the game and is also private as its access modifier.Its getter and setter are getSize() and Setsize() respectively.

- The deck of cards in the code is represented using the concept of Array Of Objects by using ArrayList to store the cards which are Objects.

For Future Coding

For future coding we will use the concept of stacks to represent the deck of cards each player uses to play the game. Since the player will only flip the top card in their deck, the Last In First Out property of Stacks will facilitate the gameplay.

Reference List

- *War – Card Game Rules | Bicycle Playing Cards*.(n.d.). Retrieved from URL https://bicyclecards.com/how-to-play/war/#:~:text=Each%20player%20turns%20up%20a,both%20piles%20(six%20cards).