



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.

For further information, read Academic Integrity Policy here :

[https://caps.sheridancollege.ca/student-guide/academic-policies-and-](https://caps.sheridancollege.ca/student-guide/academic-policies-and-procedures.aspx)

Team Member Names (Please Print)	Signatures	Student ID
Project Leader: Yoon-Ho Choi		991809741
Stefan Ikonomovski	Stefan I.	991523660
Soham Deepakkumar Parekh		991797785

[procedures.aspx](https://caps.sheridancollege.ca/student-guide/academic-policies-and-procedures.aspx)

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

Responsibilities of the Project Leader include:

Team Contract

- 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.
- Notifying the professor of any issues in a timely manner so that corrective measures can be taken.
- Any other duties as deemed necessary for project completion.

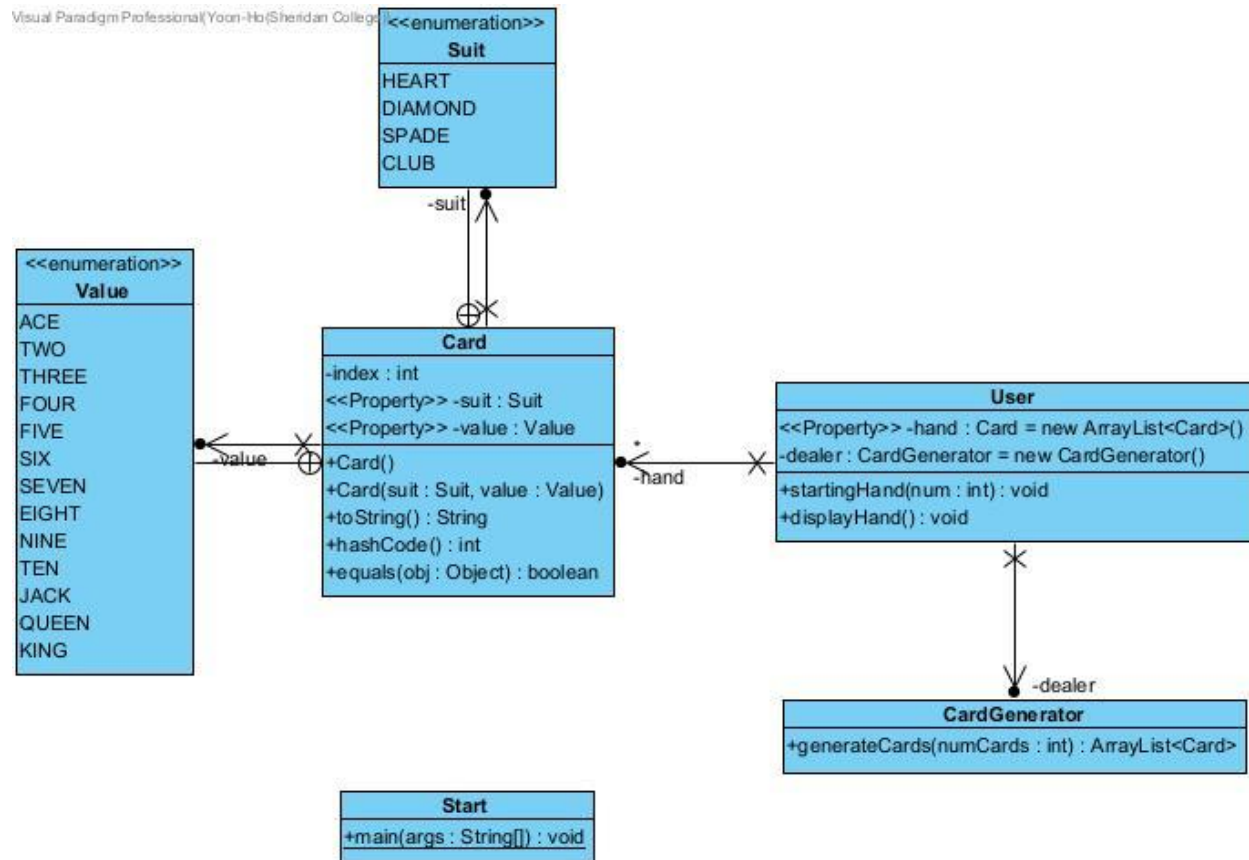
What we will do if . . .

Scenario	Accepted initials	We agree to do the following (Put an X corresponding to your choice in each box)
Team member does not regularly attend team meetings and/or does not respond to communications in a timely manner.		Project leader emails the student citing the concerns and cc's the professor so they are aware of the situation at the very onset <u> X </u> ( <b>Mandatory</b> ).  a)     ___ In addition to above, the leader/team will (add your own content here):
Team member does not deliver component on time due to severe illness or extreme personal problem.		a)     Team absorbs workload temporarily ___  b)     Team seeks advice from professor ___  c)     Team shifts target date if possible ___  d)     ___ Other (specify):
Team member has difficulty delivering component on time due to lack of understanding or ability.		a)     Team reassigns component ___  b)     Team helps member ___  c)     Team member must ask professor for help ___  d)     ___ Other (specify):

Scenario	Accepted initials	We agree to do the following (Put an X corresponding to your choice in each box)
Team member does not deliver component on time due to lack of effort.		a) Team absorbs workload ___ b) Team member(s) ask professor to request a Participation Form from <u>all</u> team members. This <i>may</i> result in individualized grades being awarded for a deliverable ___ c) Both a. and b. above ___ d) ___ Other (specify):
Team cannot achieve consensus leaving one or more member(s) feeling that their voice(s) is/are not being heard in a decision which affects everyone.		a) Team agrees to abide by majority vote ___ b) Team seeks advice from the professor ___ c) ___ Other (specify):
Team members do not share expectations for the quality of work on a particular deliverable.		a) Team members will draw on each other's strengths to help bring the quality of the deliverable to a minimal acceptable level ___ b) Team votes on each submission's quality ___ c) Team member(s) ask professor to request a Participation Form from all team members, which may result in individualized grades being awarded for a deliverable ___ d) ___ Other (specify):
Team member behaves in an unprofessional manner, e.g. being rude, uncooperative and/or making one or more		a) Team agrees to avoid use of all vocabulary inappropriate to a business/college setting ___

Scenario	Accepted initials	We agree to do the following (Put an X corresponding to your choice in each box)
member(s) feel uncomfortable.		b) Team attempts to resolve the issue by airing the problem at a team meeting __  c) Team requests a meeting with the professor to discuss further __  d) __ Other (specify):
There is a dominant team member who insists on making all decisions on the team's behalf leaving some team members feeling like subordinates rather than equal members		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) Team seeks advice from the professor __  d) __ Other (specify):
Team has a member who refuses to participate in decision making but complains to others that s/he wasn't consulted		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



The source code for this UML diagram can be found at:

[https://github.com/Francis-Choi300/SYST\\_Group](https://github.com/Francis-Choi300/SYST_Group)

## Design Document Template

This Card class represents a single playing card in a standard deck.

First the Card class was imported with the objects method. Public enum was declared with the Suit variable with the values HEART, DIAMOND, SPADE, CLUB and public enum was also declared with the Value variable with the values ACE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK, QUEEN, KING, then the Value value and int index variables were declared as private.

The public Card(){} was first set with the no args constructor, then public card has the variables Suit suit, Value value with this.suit = suit; and this.value=value; The getter and setter methods were set with the variables getSuit, setSuit,

getValue, and setValue. For starters, the override method was set to public String toString(), then with the return method saying what suit and value are displayed. Secondly the override method was set to public int hashCode() with hash set to 3, hash = 53 \* hash + Objects.hashCode(this.suit);

hash = 53 \* hash + Objects.hashCode(this.value); return hash; Finally the override method was written with the boolean equals function.

The CardGenerator class is designed to generate a list of playing cards randomly. First the class is part of the cards package. The public class CardGenerator is responsible for generating all cards.

The method public ArrayList<Card> generateCards (int numCards) generates a number of card objects and returns them in the ArrayList. The method ArrayList<Card> hand = new ArrayList<Card>(); creates a new list to hold the cards like a player's hand.

The Random rnd = new Random(); creates a random object to generate random numbers. Card.Suit.values returns an array of all suits values (eg. Hearts, Diamonds, etc). Card.Value.values returns an array of all value enums (eg. Ace, 2, 3, King, etc) .length gets the number of options available for suit and value.

The for loop starts a loop that runs numCards times and then creating one new card each time. The int rndSuit and int rndValue generate random integers. c.setValue and c.setSuit randomly assign a value and a suit to a card.

The hand.add(c) adds the generated card to the hand. Finally after the loop the return hand; is returned.

The Start class represents the beginning of a simple card game simulation. The public class Start is commonly used as the driver to start running the program. After the public class the main method begins the execution. User player1 = new User(); creates an object called player1 of type User.

The player1.startingHand(7); function calls the method startingHand() on player1, asking for 7 cards to be dealt. The player1.displayHand(); function calls a method that

prints or shows all cards the player has in hand. Finally the `System.out.println("Game Complete")` message is printed.

The `User` class is part of a card game simulation in Java. It models a player who has a hand of cards and can receive cards from a dealer. The `cards` class starts from a package, then `java.util.ArrayList` is imported.

The public class `User` represents a player in the game, `private ArrayList<Card> hand = new ArrayList<Card>();` this hand stores the cards the player holds, `private CardGenerator dealer = new CardGenerator();` this dealer is used to generate the cards for the player.

Next, the getter and setter methods are declared for `getHand` and `setHand`. They set the player's to a given list of cards, the public void `startingHand` method deals num random cards to the player using the `CardGenerator`, public void `displayHand` prints each card in the player's hand to the console.

The UML diagram has the variables `Start`, `User`, `CardGenerator`, `Card`, `Suit`, and `Value`. The `Suit` and `Value` variables are both enum. The `Card` class has the index: `int` variable and `Card` with the values `suit: Suit` and `value: Values`, the `toString` method, `hashCode` method, and `equals` object with a boolean.

The `Start` class starts with a main method. The `CardGenerator` generates cards with the variable `numCards: int`. Finally, the `User` class has the `startingHand` and `displayHand` methods.