




SYST 17796 TEAM PROJECT

Team Name: Team UNO

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: Pranav Mistry		991625014
BingChen Sun		991622161
Muhammad Urzam		991614634

Shreyansh Kathrotiya	<i>Shreyansh Kathrotiya</i>	991618807

For further information read Academic Honesty Policy on AccessSheridan.

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 initials	We agree to do the following
Team member does not deliver component on time due to severe illness or extreme personal problem		<p><u>a) Team absorbs workload temporarily</u></p> <p>b) Team seeks advice from professor</p> <p>—</p>

		<p>c) Team shifts target date if possible —</p> <p>d) Other:</p>
Team member cannot deliver component on time due to lack of ability		<p>a) Team reassigns component __</p> <p><u>b) Team helps member__</u></p> <p>c) Team member must ask professor for reference material __</p> <p>d) Other:</p>
Team member does not deliver component on time due to lack of effort		<p>a) Team absorbs workload __</p> <p>b) Team "fires" team member by not permitting his/her name on submission __</p> <p><u>c) Other: Team absorbs workload and informs professor</u></p>

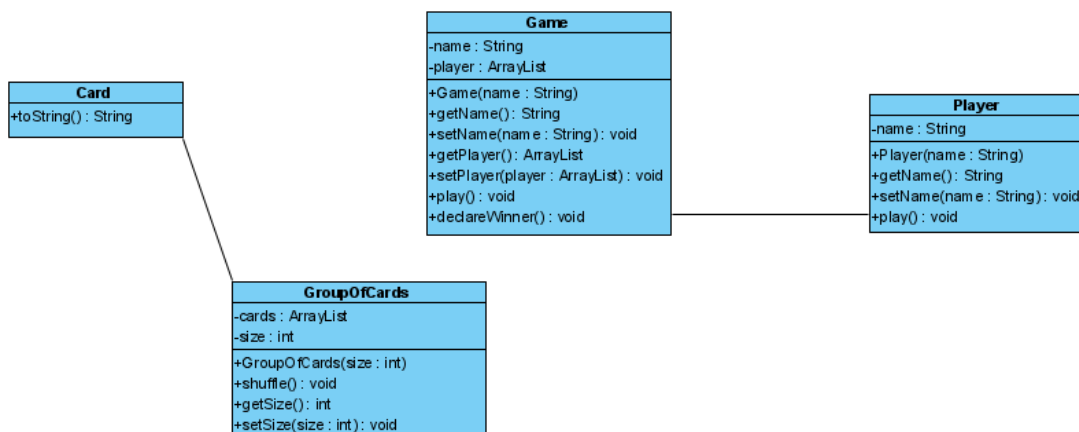
<p>Team member does not attend team meeting</p>		<p><u>a) Team proceeds without him/her and will assign work to the absent member</u></p> <p>b) Team doesn't proceed and records team member's absence __</p> <p>c) Team proceeds for that meeting but "fires" member after __ occurrences __</p>
<p>An unforeseen constraint occurs after the deliverable has been allocated and scheduled (a surprise test or assignment)</p>		<p>a) Team meets and reschedules deliverable __</p> <p><u>b) Team will cope with constraint</u></p> <p>c) Other:</p>
<p>Team cannot achieve consensus leaving one member feeling "railroaded", "ignored", or "frustrated" with a decision which affects all parties</p>		<p><u>a) Team agrees to abide by majority vote</u></p> <p>b) Team flips coin __</p> <p>c) Other:</p>

<p>Team members do not share expectations for grade desired</p>		<p>a) Team will elect one person as "standards-bearer" who has the right to ask that work be redone __</p> <p><u>b) Team votes on each submission's quality</u> __</p> <p>c) Team will ask for individual marking and will identify sections by author __</p> <p>d) Other:</p>
<p>Team member behaves in an unprofessional manner by being rude or uncooperative</p>		<p>a) Team attempts to resolve the issue by airing the problem at team meeting __</p> <p><u>b) Team requests meeting with professor to problem-solve</u> __</p> <p>c) Team ignores behaviour __</p> <p>d) Team agrees to avoid use of all vocabulary inappropriate to the business setting __</p>

<p>Team member assumes or requests that his/her name be signed to a submission but has not participated in production of the deliverable</p>		<p><u>a) Team agrees that this is cheating and is unethical</u></p> <p>b) Friends are friends and should help each other __</p> <p>c) Team will submit with signature but will advise professor who will take action __</p>
<p>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</p>		<p><u>a) Team will actively solicit consensus on all decisions which affect project direction by asking for each member's decision and vote</u></p> <p>b) Team will express subordination feelings and attempt to resolve issue —</p> <p>c) Other:</p>

Team has a member who refuses to participate in decision making but complains to others that s/he wasn't consulted		<p>a) Team forces decision sharing by routinely voting on all issues __</p> <p>b) Team routinely checks with each other about perceived roles __</p> <p><u>c) Team discusses the matter at team meeting__</u></p>
--	--	--

UML Diagram:



Deliverable 1

SYST 17796 Fundamentals of Software Design and Development

Instructor: Elizabeth Dancey

Team Name: Team UNO

Team Members:

BingChen Sun - Project Background and Description & Project Scope

Urzam Muhammad - Design Considerations

Pranav Mistry - High_Level requirements

Kathrotiya, Shreyansh - Implementation Plan

Design Document

1. Project Background and Description

UNO is a board game. The player needs to call out "UNO" when the last card is left, hence the name. It is the English of "1" in Spain/Italy. UNO is popular in vine.

UNO cards are divided into three types of cards: digital cards (76 cards), function cards (24 cards), universal field cards (8 cards), and the cost is 108 cards.

The number plate is composed of 4 colors of red, yellow, blue, and green, and the actual number plate of each color is 0-9. There are 19 digital cards for each color, 2 for each color from 1 to 9, and 1 for each color. Basic card. The function card is composed of 4 colors: red, yellow, blue, and green. Their names are "skip", "reverse" and "+2/+4" respectively. There are 8 function cards, 2 of each color, each with its own function.

Skip: After skipping the play, your next house cannot play cards, it is your turn to play the next house.

Start: After the start needle is played, the current card sequence will be synthesized, and the next card will be played.

+2: After playing +2, the next player will be fined 2 cards and cannot play cards. It's the next player's turn to play cards.

Wildcards have a very important position in the UNO and have some functions. Their cards are black, and generic cards have no color. The names are "color-changing card" and "+4", each with 4 cards. There are many other names for omnipotent cards, such as black cards, trump cards, and so on.

After changing the color to play a faded card, you can arbitrarily specify the color of the next card (choose one of the four colors), and then the family will play this card.

+4: After playing +4, you can arbitrarily specify the color of the next card. At the same time, the next player needs to draw 4 cards from the pile, and the next player cannot play cards. It's the next house's turn to play the next house

UNO's victory condition is: finish all cards in hands.

Final Goal

We want to let more people come into contact with this game through our UNO Game, simple game rules, and then let players experience the fun through the game we made.

Our final philosophy in this game design is

1. The difficulty of the game should be low, not too complicated, and the display content should be concise
2. Pay attention to the interaction of everyone: must pay attention to the interaction of everyone, you can design more interactive activities, enrich body language, and enrich expressions
3. Players who have played for a short period of time must ensure their retention, (1) establish a strong goal; (2) guide them to integrate into player circles and establish friendships through interest relationships.
4. The combination of simple rules is changeable, and interaction is enhanced. This will not let the player feel bored and opt out of the game

Base Code Description:

The current base code is written in Java and consists of 4 classes consisting of 3 abstract classes called Card, Game, and Player and one concrete class GroupOfCards. Currently, the classes Card and GroupOfCards have an association relationship,

similarly, the classes Player and Game are also associated together. Looking at it we can sense that the classes are set up in a way that allows inheritance and data sharing between classes, there is also room for potential types such as Enums. We are also able to see a potential relationship that can be formed between the class Game and the class GroupOfCards.

Currently, there are in total, five private instance variables in their respective classes with corresponding getters and setters ensuring encapsulation. Apart from getters and setters, the method shuffle() in the class GroupOfCards is the only concrete method in the entire program, all the rest are abstract methods.

2. Project Scope

BingChen Sun - Project Background and Description & Project Scope (Planning & Testing)

Urzam Muhammad - Design Considerations (Developer)

Pranav Mistry - High_Level requirements (Development & Planning)

Kathrotiya, Shreyansh - Implementation Plan (Developer)

We used Whatsapp and Team Viewer to have group exchanges and discussions, and then the group members selected the parts that everyone wanted to make. We have created a shared folder in Google Drive, we will upload all the project parts we have done to the shared folder and discuss and integrate through voice communication.

We will know the project is complete when the basic gameplay is functional without any bugs.

3. High_Level requirements

High-level requirements include:

- The ability for each player to register with the game.
- The ability to randomly assign 7 cards to the player at the beginning of the game.
- The ability for each player to view their current set of cards.
- The ability for the game to communicate win or loss.
- The ability for the players to know their score at all times.
- The ability to include 4 different colors of cards such as Yellow, Red, Blue, & Green.
- The ability to include special cards such as Draw 4 cards, Draw 2 cards, Wild cards, etc.
- The ability for the dropped cards to stack up.

4. Implementation Plan

Link to the GitHub repository: <https://github.com/kathrots/UNO-TEAM>

Each developer checks in with the code every four days.

Tools used in the project:

Netbeans, GitHub, Visual Paradigm, Teamviewer.

5. Design Considerations

Encapsulation: In Java, encapsulation is a mechanism of wrapping variables and methods together in a class, in order to hide its implementation from other classes.

In the given code, Encapsulation is achieved by using the **private** keyword on instance variables.

The **Player class** contains a private instance variable called **name** which is of type **String**, which is hidden from the **Game class** which also contains two **private** instance variables called **name** of type **String** and **player** of type **ArrayList** which are hidden from the Player class.

Similarly, the **GroupOfCards class** contains two **private** variables called **cards** of type **ArrayList**, and **size** of type **int** which are hidden from the **Card class**.

The only way for the other classes to access these private variables is to use either the **getter** or **setter methods**.

Delegation: Delegation is the process of handing over the responsibility of a task to another class or method. Delegation is used to achieve high cohesion in our programs.

Looking at the current code, the class Game has two abstract methods called play(), and declareWinner() which are to be overridden when the concrete class GroupOfCards extends Game in the future. GroupOfCards has its own method called shuffle(). The Card class also has an abstract method toString() which is also to be inherited and overridden by GroupOfCards. However as a result GroupOfCards will lose its cohesion.

Each class should have its own function instead of just one class doing every task.

Flexibility/Maintainability refers to how simple it is to update, change, or reuse code over time.

As the current base code does not have many implementations of methods apart from getter and setter methods, thus it has high flexibility/maintainability. Looking at the current code, methods like play() can be expanded to implement the main logic of the gameplay, declareWinner() expanded to declare the winner of the game, shuffle() can be expanded to shuffle the cards, and so forth. Overall the code is already flexible enough that we can implement any type of card game from it, not just UNO. All we would need to do is change the main logic, and maybe the type of cards(which can be better improved using Enums instead of Strings).

How to access the Git repository:

To access the Git repository click on the link mentioned in the implementation plan, you would need to make a GitHub account to access the code.