DELIVERABLE 1

UNO GAME

Contents

[1. Team Contract 2](#_Toc32341361)

[2. UML diagram of existing code. 7](#_Toc32341362)

[7](#_Toc32341363)

[3. Project Background and Description 7](#_Toc32341364)

[4. Project Scope 7](#_Toc32341365)

[5. High-Level Requirements 8](#_Toc32341366)

[6. Implementation Plan 8](#_Toc32341367)

[7. Design Considerations 8](#_Toc32341368)

[8. References. 9](#_Toc32341369)

|  |  |  |
| --- | --- | --- |
| Team Member Names (Please Print) | Signatures | Student ID |
| Project Leader: Mohamed | Mohamed | 991590042 |
|  |  |  |
|  |  |  |
|  |  |  |

## Team Contract

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 | Y  MA | a) Team absorbs workload temporarily \_\_  b) Team seeks advice from professor yes  c) Team shifts target date if possible \_\_  d) Other: |
| Team member cannot deliver component on time due to lack of ability | Y  MA | a) Team reassigns component \_\_  b) Team helps member \_\_  c) Team member must ask professor for reference material y  d) Other: |
| Team member does not deliver component on time due to lack of effort | Y  MA | a) Team absorbs workload y  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 | Y  MA | 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 | Y  MA | 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 y  d) Other: |
| An unforeseen constraint occurs after the deliverable has been allocated and scheduled (a surprise test or assignment) | Y  MA | a) Team meets and reschedules deliverable y  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 | Y  MA | a) Team agrees to abide by majority vote \_\_  b) Team flips coin \_\_  c) Other: |
| Team members do not share expectations for grade desired | Y  MA | 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 y  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 | Y  MA | a) Team attempts to resolve the issue by airing the problem at team meeting \_\_  b) Team requests meeting with professor to problem-solve y  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 | Y  MA | a) Team agrees that this is cheating and is unethical y  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 | Y  MA | a) Team will actively solicit consensus on all decisions which affect project direction by asking for each member's decision and vote y  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 | Y  MA | a) Team forces decision sharing by routinely voting on all issues y  b) Team routinely checks with each other about perceived roles \_\_  c) Team discusses the matter at team meeting \_\_ |

## UML diagram of existing code.

## 

## Project Background and Description

The goal of this project is to make a working class UNO card game that allows up to a maximum of 4 players to play. How to play: Uno is a card game that involves 2-10 players (My version will include 4 max). Each player is given 4 cards and they play. Players have to match the color that is played or the same number can be played. Special cards in the game include: Wild-card (Allow users to pick any color), Skip card (Skips a player), Pick 2 and Pick 4 (where the next player has to pick the number of cards and can’t play) and finally the reverse card. The player who finishes playing all their cards is declared a winner and they can decide to play again.

Video link reference: <https://www.youtube.com/watch?v=fOL2sGbHeUo>

Starting code: Starting base code, in Java, includes 4 classes, with most of them being abstract classes since they will depend on what code I’ll have to write and I’ll have to inherit the classes and use the methods in order to have a sense of direction which classes I need. For example extending the Card class will allow me to use the card method and use the toString method (which is abstract) and override it in my class, in order to print the UNO card if a user wants to see their cards.

The classes have comments in their methods which allows me to easily understand what each method does and Player class includes initializer which allows me to create a player and Game class has final attribute that stores title of game hence can’t be changed and it stores the players in an array-list. Attributes are private thus can’t be accessed from outside the class.

## Project Scope

One member in group thus responsible to make sure all the requirements are fulfilled and the code is always committed. For testing features, a branch will be created to make sure the working class code isn’t affected and if satisfied with it, the code will be merged. Every week the code will at-least be pushed to the remote from the local remote and any added things will be well committed with an appropriate message. Project will be complete once tested fully and requirements completed.

## High-Level Requirements

The new system must include the following:

* Ability for each player to register with the game
* Ability for the game to determine the winner of the game.
* Ability for players to forfeit the game
* Players cannot see the other players’ deck of cards and must be hidden.
* Ability for players to decide who wants to play first
* Game can be played with minimum of 2 to a max of 4 players.

## Implementation Plan

Public Git repo: <https://github.com/mohd009/simpleUNO.git>

Code will always be tested before pushed to the remote and any new feature or change must be pushed from the local remote to the remote before the end of the week. UML diagram will must always be updated and stored in a different file. And any text files required will be stored in a different directory.

Coding standards: Any new method or class added must at least have a description of what it does and use of Camel Case for variables and making sure methods have an appropriate name of what they should do.

## Design Considerations

The current code includes Encapsulation, Game class includes two private attributes gameName (final, can’t be changed) and Players in an array List, so that these attributes can’t be called anywhere from outside this class or manipulated with. Another example is the Player class, which has Player ID attribute that stores each player which is also private hence user can’t call it from outside. Thus by having encapsulation makes it easier to test code by just using the helper methods to use the private attributes for classes that inherit it.

Delegation: Classes have different responsibilities for example in the Game class the abstract method play() will be used to start the game and implemented in another class through inheritance and in the Player class, there is also a play() method, but I plan to use this method to start the beginning of the game and will simple just ask user the number of users playing the game and create players based on number meanwhile the play() in Game class will officially start the game after players are created.

I plan to just use an interface and put the play() method in it. Then call the method when I need it for creating players and starting the game. Thus it will be used for two purposes. It will be used to create players and the other purpose for starting the game and handing out cards to players.

Flexibility/Maintainability: The Group of Cards class allows us to store the cards in an array list that accepts Card class as parameter, hence it makes it easy to add cards and since the class also has a size attribute, it would be easy for me to just give it a specific size for my game when inheriting from this class thus less code to be changed (loose coupling) and the use of cohesion in the card class which has a toString method which performs one purpose and that is to just print the value of the string. I plan to extend this class to add extra functionalities for special cards like Pick two, Wild card. This way when a user plays one of these special cards, the class will print when a user uses one of these cards. Player class also has strong cohesion and has methods that involve based on the player, like getting player ID, and thus this class can be reused to create players.

**.**

## References.

Kumar, S., & Kumar, S. (2019, July 2). Coding Standards and Guidelines. Retrieved from <https://www.geeksforgeeks.org/coding-standards-and-guidelines/>

Video link for UNO: <https://www.youtube.com/watch?v=fOL2sGbHeUo>