Contents

[Overview 3](#_Toc36425826)

[1. Project Background and Description 3](#_Toc36425827)

[2. Design Considerations 3](#_Toc36425828)/4/5

syst 17796 Deliverable 2

design document template

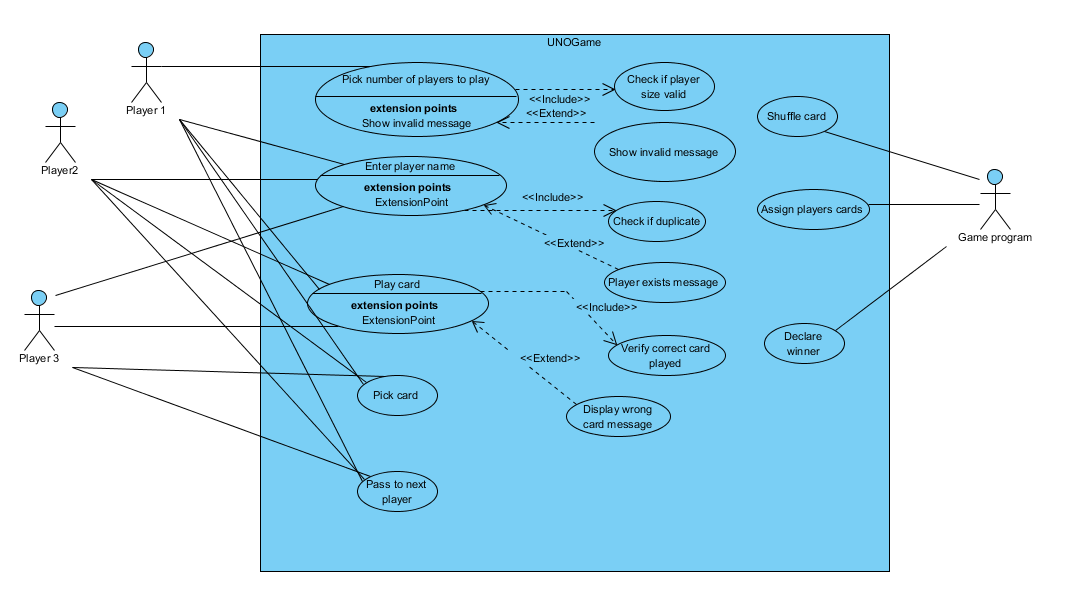
# Overview

## Project Background and Description

In the game, UNO, the maximum number of players will be 4 with 2 being the minimum to play. Each user will be awarded a hand of 7 cards in a normal game but I have decided to make version faster instead to finish faster, they’ll instead be given 4 cards to start with. Players must choose the right card from their deck and play it, if they put a wrong card, it will be denied, and if they don’t have the card, they must pick a card, and decide to play it or pass the round to the next person. A player will not allow to see what cards other players have but they’ll just be informed on how many cards they have. The player who finishes playing all their cards is declared a winner and they can decide to play again.

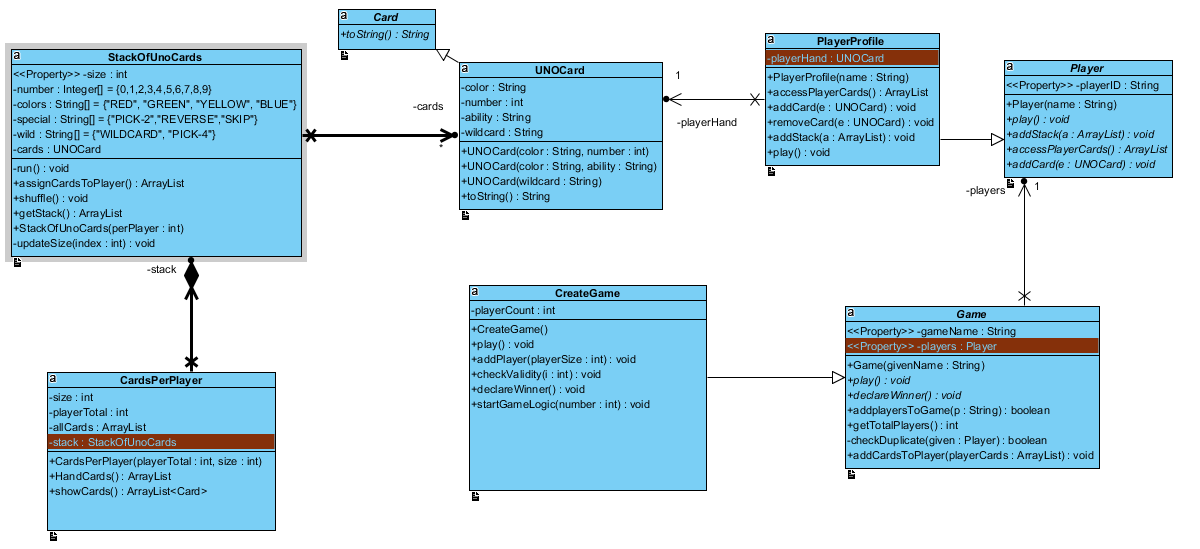
## Design Considerations

**Use case diagram**



The above diagram shows possible use case based on the way the game is meant to run by. Players are allowed to play cards, pick card, enter their name and pass to next player. A user may choose to enter the wrong card or choose same name and the system will force the user to try again since it would be invalid. The game will then decide the game winner based on how the game goes.

**Class diagram**



In the above diagram, there exists association. From Game class and Player class, in order for the game to begin, it stores all the players in an ArrayList which stores the players as a parameter. This arraylist of players is used by the createGame class to update the stack of cards each player has. In the player class, since it’s abstract, it has a playerProfile class where each player has their own Array of cards, where they can add or remove cards. In the Game class each player is unique in the arrayList.

Another association is between the player profile class and UNO card class, the playerHand attribute is an arrayList that stores a player’s cards, and each card is an UNO card, which is unique for each player, thus one single UNO card unique in their array, thus forming multiple unique UNO cards for each player. And in the StackOfUnoCards class, to create a stack of UNO cards, it needs to create a unique UNO card for each card created from the stack.

2) Comment on each of the following as it pertains to the class groupings you have decided upon and if you have included methods, modifiers and return types, comment on those here as well. You may wish to describe any data structures you wish to use (i.e an enumeration) when you are explaining your design choices. Be specific for full credit.

* Encapsulation

In the PlayerProfile class and CardsPerPlayer class, the arrayList of cards is private so as it can’t be manipulated with or accessed, this attribute although has methods that allow only to add or remove a card from a player’s stack of cards.

In the UNO class and StackOfUnoCards class, all attributes are private, because when the game is created, the cards are made and thus making them private would allow no class to tamper with it or change it, since each UNO card is unique when created, 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

The CardsPerPlayer class, the HandCards method generates a stack of cards for each player thus when the addCardsToPlayer method is called from the Game class, it simply calls the addStack method in PlayerProfile that takes in ArrayList from HandCards method and assigns the cards to the players. Hence generating a stack of cards between players is done by two methods(HandCards() and addStack()) to simplify the task.

Another example, is the play () method in CreateGame which starts the game, and the Player class also has play () method which will be used by players in the game to go to the next person after their move.

* Cohesion

The StackOfUnoCards class was created in order to generate UNO cards for the game, and its methods are related to it. And the UNO card which is created from the UNO card class for every card created in the StackOfUnoCards class.

PlayerProfile which extends from the player class, designed for each player to store their game details and also to keep track of their game cards.

* Coupling

An arrayList is used in the stackOfUnoCards class to generate UNO cards, but if other special cards were to be added, then it can simply be added to the array and the stack will have that card, hence other parts of the code won’t be altered with,

* Inheritance

CreateGame class extends the Game class so as when the game is initiated, the play () method is used since it’s abstract, and it is implemented in the createGame class, so as declareWinner () method, and since the Game class is associated with the Player Profile class through extending the player class, it can access all the players deck of cards and update it during the game. The Uno card class extends the Card class so as to print each card.

* Composition

In the CardsPerPlayer class, the stackOfUnoCards class is created in it, where cards are generated and each player is given same size of cards from the CardsPerPlayer parameter, size (int), once the stackOfUno class is created.

* Flexibility/Maintainability

The classes are designed to be separated, most of the gameControls are done from the gameLogic() method in createGame class, and the playerProfile classes are created for each user, this way, a new feature can easily be added for each player to perform, the StackOfUnoCards and UNO card are responsible for generating the cards for the game and since an arrayList was used, a card can easily be added to the deck without changing much. The player class is abstract because it makes it easy to read code and break it down for easy understanding.