**UNO Game**

Names:

Kaleel Evans

Student ID: 991638770

Edwin Johnson

Student ID: 991667321

Liliana Mantilla Guevara

Student ID: 991633617

College: Sheridan College

Professor name: Sivagama Srinivasan

Course name: SYST 17796 Fundamentals of Software Design and Development

Date: 13/07/2022

Table of Contents

[**UNO Game** 3](#_Toc108648324)

[1. Project Background and Description 3](#_Toc108648325)

[1.1 Project Goals: 3](#_Toc108648326)

[1.2 Vision final: 3](#_Toc108648327)

[1.3 Description and rules of UNO game: 3](#_Toc108648328)

[1.4 Description of the code: 4](#_Toc108648329)

[2. Project Scope 4](#_Toc108648330)

[3. High-Level Requirements 5](#_Toc108648331)

[4. Implementation Plan 6](#_Toc108648332)

[6. Use case diagrams of the UNO game: 8](#_Toc108648333)

[7. Narrativa of the use case dagram of the UNO game: 8](#_Toc108648334)

[8. Design Considerations 9](#_Toc108648335)

[**Appendix** 12](#_Toc108648336)

[1.1 Team Contract 12](#_Toc108648337)

Table of Images

[Figure 1. UML basic game 7](file:///D:\Sprint_Summer2022\Fundamentals%20of%20Software%20Design%20and%20Development\Project\Projectdelivere2.docx#_Toc109078337)

[Figure 2. UML UNO game 8](file:///D:\Sprint_Summer2022\Fundamentals%20of%20Software%20Design%20and%20Development\Project\Projectdelivere2.docx#_Toc109078338)

[Figure 3. Use case UNO game 9](file:///D:\Sprint_Summer2022\Fundamentals%20of%20Software%20Design%20and%20Development\Project\Projectdelivere2.docx#_Toc109078339)

[Figure 4. Enum class 11](file:///D:\Sprint_Summer2022\Fundamentals%20of%20Software%20Design%20and%20Development\Project\Projectdelivere2.docx#_Toc109078340)

[Figure 5. Generate cards 11](file:///D:\Sprint_Summer2022\Fundamentals%20of%20Software%20Design%20and%20Development\Project\Projectdelivere2.docx#_Toc109078341)

# **UNO Game**

# Project Background and Description

* 1. Project Goals:
* To design and develop a code that permits playing a UNO game.
* To create a code that is simple, functional and accomplished with secure programming rules.

# Vision final:

It has the finality to create a functional UNO game, which will develop with the force of the group team.

# Description and rules of UNO game:

UNO is a multi-player card game which has as the goal of the game is to empty a player's deck of cards. The game is formed by regular number cards 1 to 9, which can be the colour red, green, blue or yellow. The player can only play the card when the current card number or colour is the same if the player does not have a card that matches in their hand. The player needs to draw a card and add their hand. The player with only a card in his hand must yell "UNO."

Rules:

* Every player starts their hands with 7 cards.
* The player can only play the card when the current card number, colour, or symbol/action is the same.
* If the player does not have a card to match needs draw a car.
* The player with only a card in his hand must yell "UNO".
* Reverse cards can only be played on other reverse cards or same colour cards, and they change the order of the game.
* Skip cards; when a player plays this card, the next player loses the turn.
* Wild This card represents all colours. It can be placed on any card.
* Draw two; when a player plays this card, the next player must pick up two cards and forfeit the turn.
* Draw four; when a player plays this card, the next player must pick up two cards and forfeit the turn.

# Description of the code:

The current code is developed using Java’s Object-Oriented Programming, which works with abstractions like objects, classes and variables that represent code and processes more complex and deletes the duplicities in the code. Also, the code uses the encapsulation method to preserve its integrity and work with the inheritance; it permits the reusability of the code and easy extension of the code if needed.

# Project Scope

The Team Members:

|  |  |  |
| --- | --- | --- |
| **No.** | **Name** | **Roles** |
| 1 | Liliana Mantilla Guevara | Leader  Programmer |
| 2 | Kaleel Evans | Programmer |
| 3 | Edwin Johnson | Programmer |

Uno game will be developed using the Java language, which works on a local machine with NetBeans Editor and is stored in a GitHub repository. The game interface was though with a simple interface that permits the players to interact simply with the application.

It will be considered that the project is a finished success when it accomplishes the below parameters:

1. The programmers considered that the code was finished and the application testing super a sample considerable play successfully.
   1. The testing should be considered successful until 5 different people play the game the way success.
2. The overall code is appropriately documented.
3. The project documentation is totally made.
4. All the information is submitted on GitHub.

# High-Level Requirements

The application must permit:

* The user registry their name, and the program shows the user's name any time the program generates a message or task for the user.
* The game permits adding more than one player in a game.
* The application shows a constant count of the player's number of cards.
* The application shows a message when the player has only one card in their hands.
* The program shows a message announcing if the player wins or loses.
* The game assign point to each player.
* The game does not finish until a player has any card in their hands.
* The player decides if they want to play a new game after finishing the current game or not.

# Implementation Plan

Git repository URL: <https://github.com/K4NZEON/GameDEvs-Uno-Deliverable>

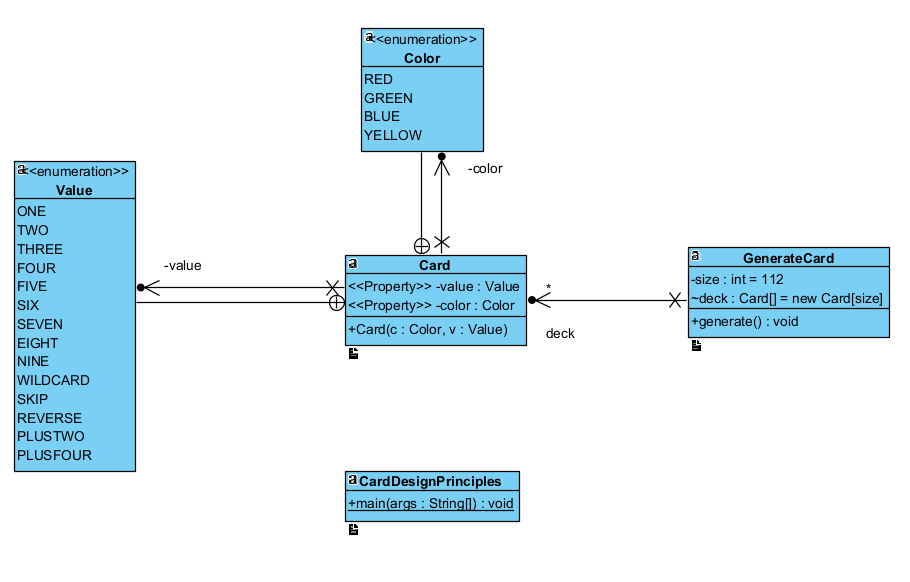
All the team members will use the Git Hub repository to maintain and update the project as required. We are trying to follow the best coding standards by making our code flexible to use and easy to manage and implementing proper methods for getting the best output possible. The tools we are currently using are NetBeans for coding and Visual Paradigm for making a visual representation of the codes in the form of diagrams for a better understanding of the structure of the code.

* 1. UML UNO game
  2. Basic UML UNO game:

Figure 1 UML basic game shows the initial development of the UML diagram for UNO game. It was formed for 5 different principal classes, which include the two enum classes. The Card Class shows a principal class which has an associate 3 class. One of them is GenerateCard class, where the direction of the association is shown from GenerateCard class to the Card class. The relation between both classes is displayed through the Deck instance. It notices the diagram marking a subsetting of the Deck instance in the Card class. Also, it is visible that Card Class has a relation of the type associated with each of the enum classes: Value and Color, and show the direction of the instances value and color also take the partial relationship between the class through the containment association.

1. The diagram shows the principles OOP involves at the moment of the code. The encapsulation method is visible in the Card class and GenerateCard class. The attribute such as value, color and size clary displayed as private. Additionally, the UML diagram shows a class that, at the moment of the project's development, is independent of the other class.

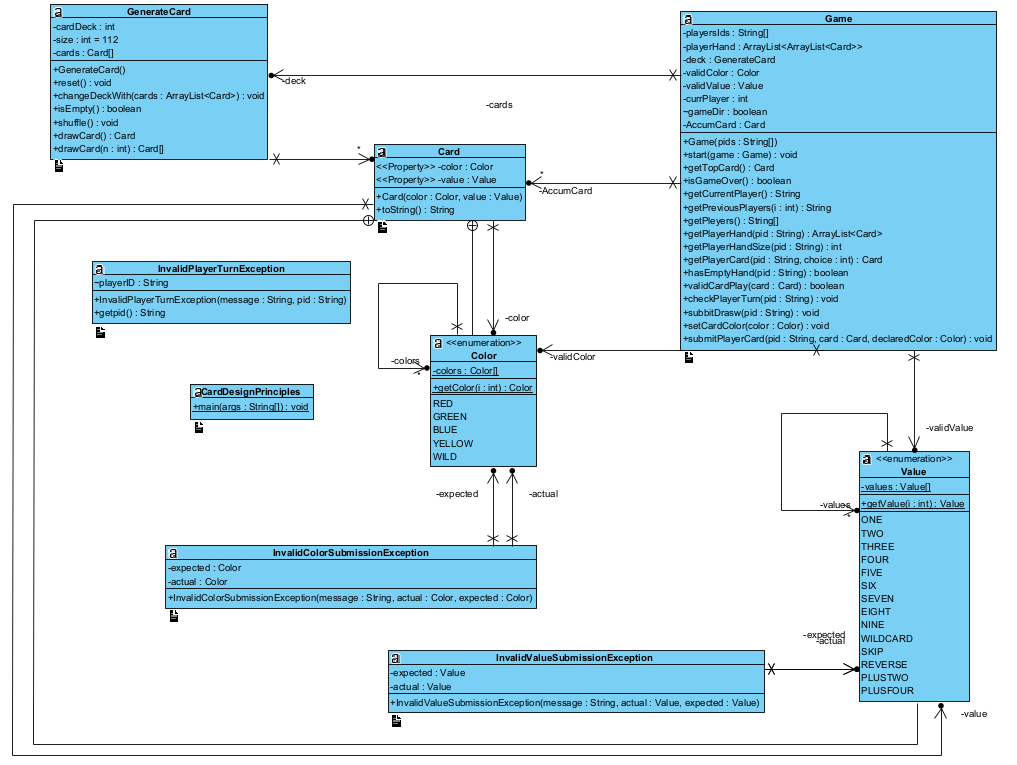
Figure 1. UML basic game code.



# Basic UML UNO game:

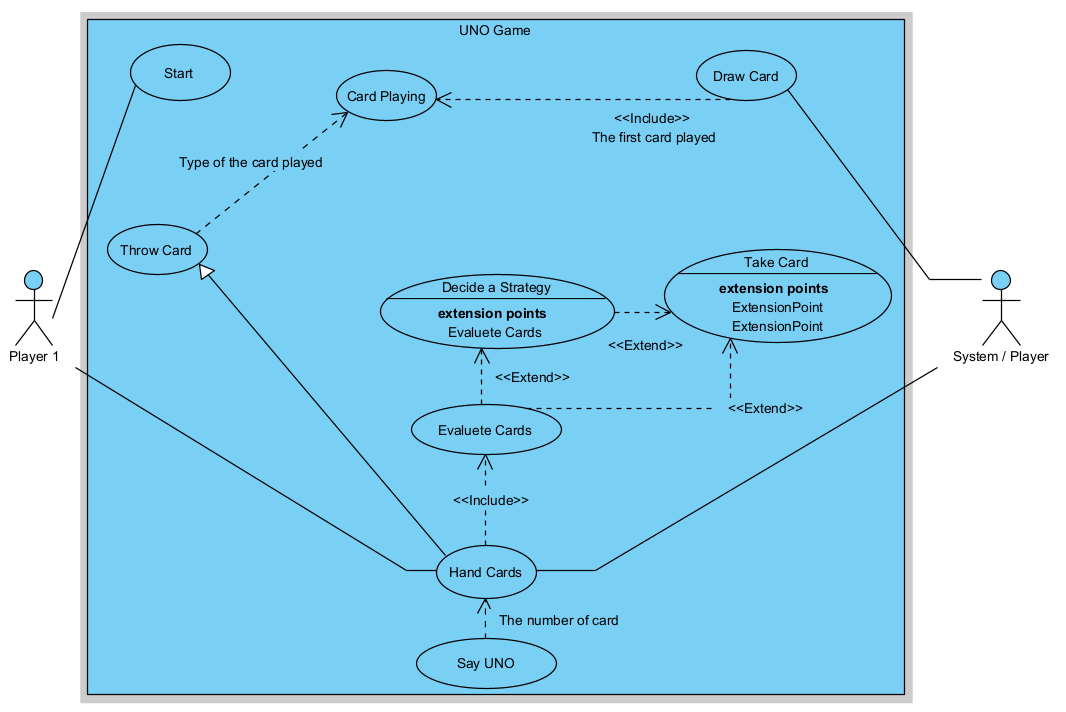
Figure 2 UML UNO game shows a complex UML diagram for UNO game, with an additional class called Game which as well other classes in the last UML diagram, have different private attributes that follow the concept of encapsulation, and it has a number of methods that permit the play the different path of the game. The Game class was designed with a relationship the type association with the Card class. The diagram marks a subsetting of the AcummCard instance in the Card class from the Game Class. Additionally, in the new diagram, there are 3 classes that use the hineritance the concept of hineritance OPP to manage the exceptions about player, colour and values, which exit a relationship with the type association. Also, GenerateCard class added a new method and array list that helped to draw the cards or now it is empty.

Figure 2. UML UNO game.



# Use case diagrams of the UNO game:

Figure 3. Use case UNO game.



# Narrativa of the use case dagram of the UNO game:

1. The player1 starts a UNO game.
2. The System draws 7 cards to each player.
   1. Include the first card on the deck for starting the game.
3. The player has cards in their hands (the player has 7 cards at the moment to start the game).

3.1 The player evaluates the card in their hands.

* + 1. The player takes a card if they do not have a card in their hands to play.
  1. The players decide on a strategy to play their cards.

3.2.1 The player takes a card. If the player does not want to play a card that is in their hands at the moment.

* 1. The player takes a card if the card Playing if the card is playing on the deck, obligate take a card.
  2. The player says UNO when the player only has in their hands one card.

1. The player throws a card depending on the type of card played on the game's deck.
2. The player is a winner when the player has any card in their hands.

# Design Considerations

The current code is structured in a way that allows it to be easily updated based on the requirements. The code has implemented OO principles like Encapsulation in order to avoid loose coupling by using enum function instead of String, as shown in the following example:

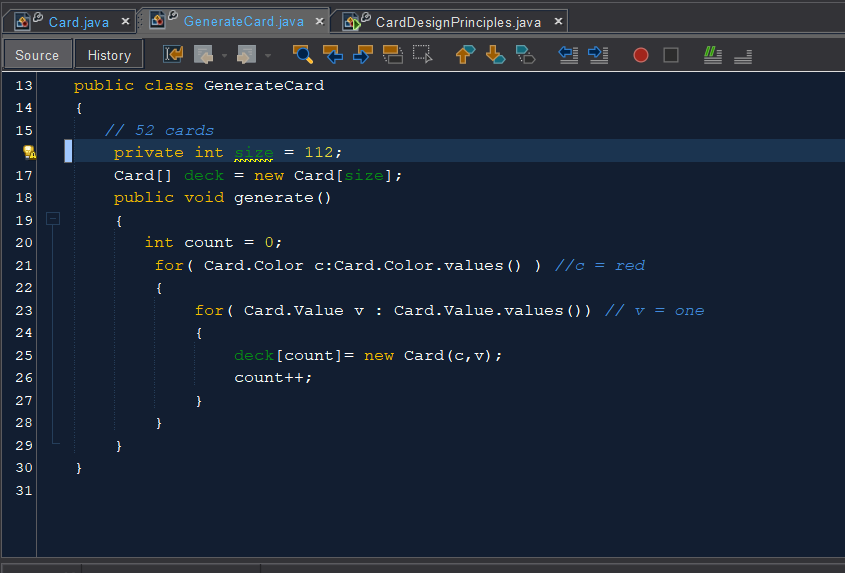
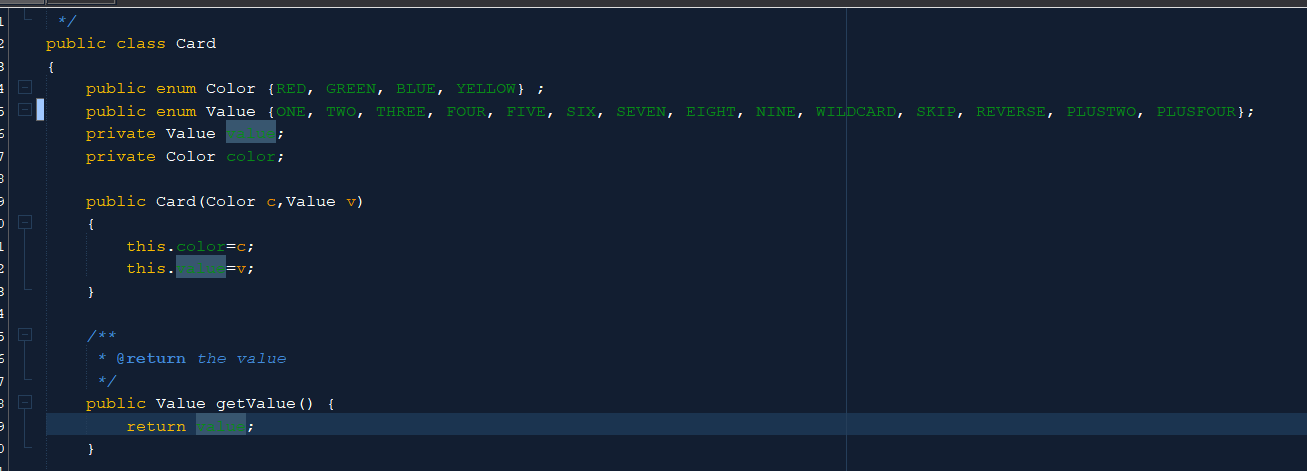


Figure 4. Enum class.

The code is also made flexible by using different methods, and it can easily change or fix any issues when needed. Additionally, the rules of cohesion have also been followed. The code has been split into the classes in which they are doing what they are supposed to do.

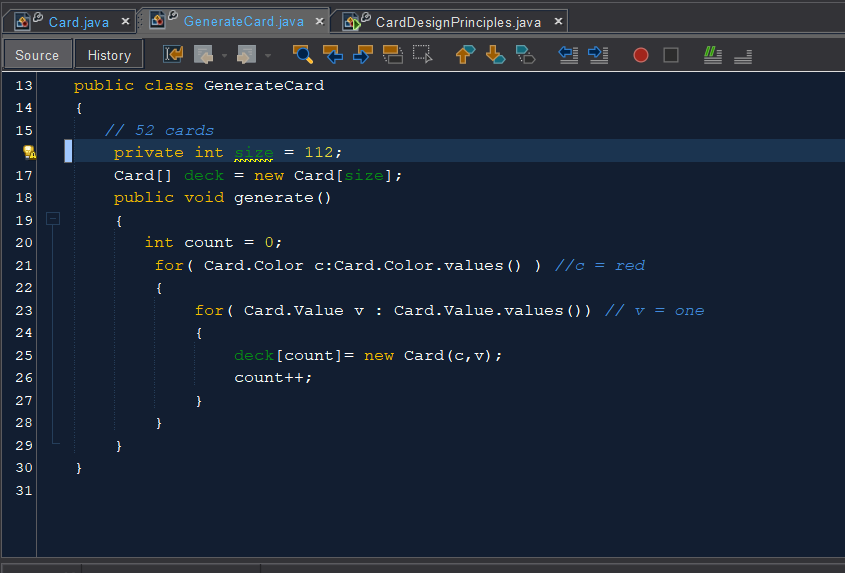


Figure 5. Generate cards.

Below is a possible appreciated example of the cohesion principle that before has been mentioned. As you can see, the primary objective of this class is just to generate the cards.

# **Appendix**

# Table Description automatically generated1.1 Team Contract