

Deliverable 1

SYST17796 - Software Development

Guided under - Professor Amandeep Sindhu

Date of Submission: Thursday, February 13, 2020

Group Members: Steven ( Group Leader), Jasmine

Daniel and Zawad

Content

Cover page 1

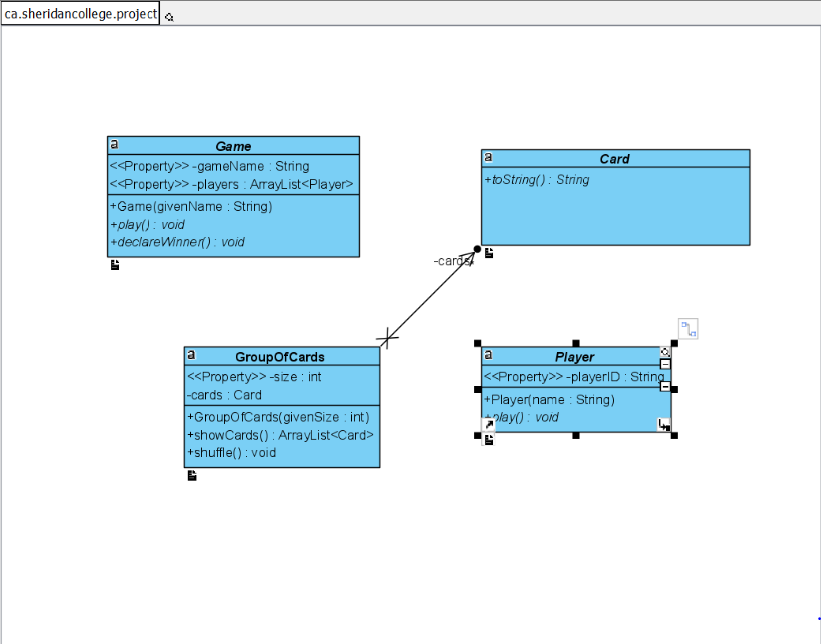
Content page 2

Group Contract 3-8

UML Diagram 9

Design Document 10

UML Class Diagram for the starter code



1. Project Background and Description

1. Project Goals.
   1. The final goal of the project wasn’t introduced to us yet, inasmuch as it’s how our course is organized, but speculating of which, we could say that our game project needs to follow the next restrictions:
      1. It should have all the game rules, and the game mechanics implemented.
      2. It should be intuitive and easy to use.
      3. We should add our new knowledge about the OOP and design patterns we’ve learned in class and from the course textbook so far.
   2. Also as a subgoal of completing this project, we will develop our soft skills which include:
      1. Project management skills.
      2. Team organization skills.
      3. Communication and problem-solving skills.
      4. Project design and structure organization.
      5. Product presentation.
2. Final Vision.
   1. As a final vision of our project, we need somehow to make our game work.
      1. It means we need to decide whether it will be just a console game project (1 screen, 1 program ran, several players are waiting for their turns next to just a 1 computer, plus maybe some GUI interface), nor will we achieve something like online game, using network programming and parts of our game software on each machine our users play. Also, we could’ve used something like javascript for this goal, and I guess it would be easier than using java here, because we can create a site with hosting and so on, but we are still figuring out this question with our team.

The card game we chose for our project is called the UNO. The funny story of UNO started in 1971 Merle Robbins, invented the game while arguing with his son about another card game. Its a shedding type card game meaning whoever plays all the hand cards wins. This game can be played with 2 or more players. The deck has over 100 cards, each player starts with 5 or 7 cards depending on the number of players playing the game. All cards except wild cards have a color assigned to it. It could be blue, yellow, red or green. A player needs to match the color/ number on the card at the top of the pile of cards played. If the player does not have a card to play and not even a wild card, then the player needs to draw the top card from the deck.

There are three types of cards in the deck

1. Number cards - Cards number starts from 0 to 9
2. Power cards - Each color has a set of power cards
   1. Draw Two card - when played the next player draws 2 cards from the deck
   2. Skip card - when played the next player will miss the turn
   3. Reverse card - when played it changes the direction of the gameplay, from clockwise to anticlockwise vice versa.
3. Wild cards - these cards can be played anytime during the gameplay.
   1. Wild Color card - It allows the player playing it to change the current color of the cards being played.
   2. Wild Draw Four card - When played, the next player has to draw 4 cards from the deck.

When a player has played his/her second last card he/she has to say ‘UNO’. And if the player fails to say ‘UNO’ he/she is penalized by drawing 6 cards from the deck.

If all the cards from the deck are exhausted and none of the players have played all their cards then the game ends as a tie.

2. Project scope

Our Group members are :

1. Steven, the group leader, who will be delegating the task of the project and owner of the Git Hub repository for the project.
2. Jasmine will be designing the layout and color scheme of the game environment.
3. Daniel is in charge of analyzing the starter code, how to implement it in the project.
4. Zawad will be recording the group meeting outcomes and documentation.

All of us will be working together to come up with algorithms that will form the structure of the backbone of our code and finally implement it. We will try to ensure our coding is efficiently written using the proper software in our case it will be the NetBeans 8.2. And for our team collaboration, we have created a Project repository.

A project is thought to be completed when it does whatever it is needed to achieve, so for us, it will be to run properly without any hiccups.

We are hoping that our final game application will be able to be launched smoothly on devices whichever supports Java Framework.

For a desktop version, the user( game player) will use the mouse to click on the desired card to make his/her move. For other devices (mobile /tablet) it will be touching the screen on appropriate locations.

3. High-Level requirements

1. The ability for each player to register with the game
   1. The registration process will be as follows;
      1. Each player will sign up with a random name of their choice.
      2. Cards will be shuffled and each player will receive their number of cards.
2. The ability for the game to communicate a win or loss
   1. The first player to finish their cards wins the game but before they can be declared the winner, he/she has to click the finish button. If this player fails to do so, then the game won’t stop and another person might win the game.
   2. When someone wins, the game will display “[Name of the winner] has won the game”.
   3. For the losers, the game will display “The losers are [Names of the losers]”.
3. The ability for players to know their status at all times
   1. During the gameplay, when each player plays a card, their card hand will show the current number of cards they have.
   2. Players will also be able to see the number of cards their adversaries have.
4. The ability for a player to draw a card randomly from the deck
   1. During the gameplay, players might run out of cards to play. They would be able to draw cards randomly from the top of the deck.
5. The ability for each player to have their turn to play
   1. A player won’t be able to play when it is not their turn. Our code will display a message to show who’s turn it is to play. For example: “Steven’s turn to play”.
   2. A player won’t be able to play if he/she doesn’t have the required card to play. He/she must pick a card from the pile of cards.
6. Tie Game!
   1. If all the cards from the pile of cards are exhausted and no one has finished playing, the game will display “Oops! It’s a tie game. Would you like to restart the game?”.

4. Implementation plan

1. Each member of the team has their own responsibilities for this project.
   1. Each member of the team has their own responsibilities for this project. As was described above we have set our roles at the moment, and by completing them, and moving forward we will face new challenges, and with the time we will rearrange our roles respectively to the problems we face.
2. Project communication.
   1. In order to share our work, we will be using the GitHub repository, so every member of our team can observe all the changes in the project remotely. (Github repository link <https://github.com/skebila/Deliverable-1.git>). Where we will be storing files for the project like code, text and other files, documentation, UML diagrams, e.t.c, separated by their own directories.
3. Personal communication.
   1. In order to communicate we usually leave sometime after the class to discuss the project, or we can freely use WhatsUp or Discord where we have our team group chat in order to navigate through the project or asking each other questions related to the project or to get help from our team members.
4. Coding standards
   1. For our project standardization, we were recommended to use the same IDE for the project, NetBeans of v.8.0.2 or higher and Visual Paradigm software for creating UML graphical representation of the project code structure.

5. Design Considerations

Currently, our code is a structure of abstract classes and methods, which means we don't have the implementation yet. But all these methods already represent the structure of our game where there are represented following dependant classes:

* (Card) class: class of cards which represents a card, and has Enums for its suit and value.
* (GroupOfCard) class: a group of (Cards) shown above ^, that has a |LinkedList| field for storing these cards, method to create all the cards for a deck, and shuffle method (which I guess should be pseudo-randomized), and overall this class is used to create a deck and pass it to the game.
* (Player) class: a player with a unique ID in the game, his own hand, and methods representing basic game mechanics like drawing cards and playing them, and of course he needs to know where are his turns, and when aren’t and he should wait for everyone else to finish their turns.
* (Game) class: has a |ArrayList| of players who are taking part in the current game, and in the future should implement, or delegate all the remaining game mechanics.

Overall our game uses inheritance for its classes which should be implemented in real game classes later, will be using polymorphism if we will leave this structure of abstracts when we’ve done to override some methods, or overload them for some cases if we’ll need to do so. All it’s classes delegate their responsibilities, which means one class represents and encapsulates one functionality and represents one entity. We don't have our code done so that's it for now.