SYST 17796 TEAM PROJECT

Team Name: Sunday Afternoon

Group Members:

Jinyoung Jeon

Juyoung Jung

Tamara Dang

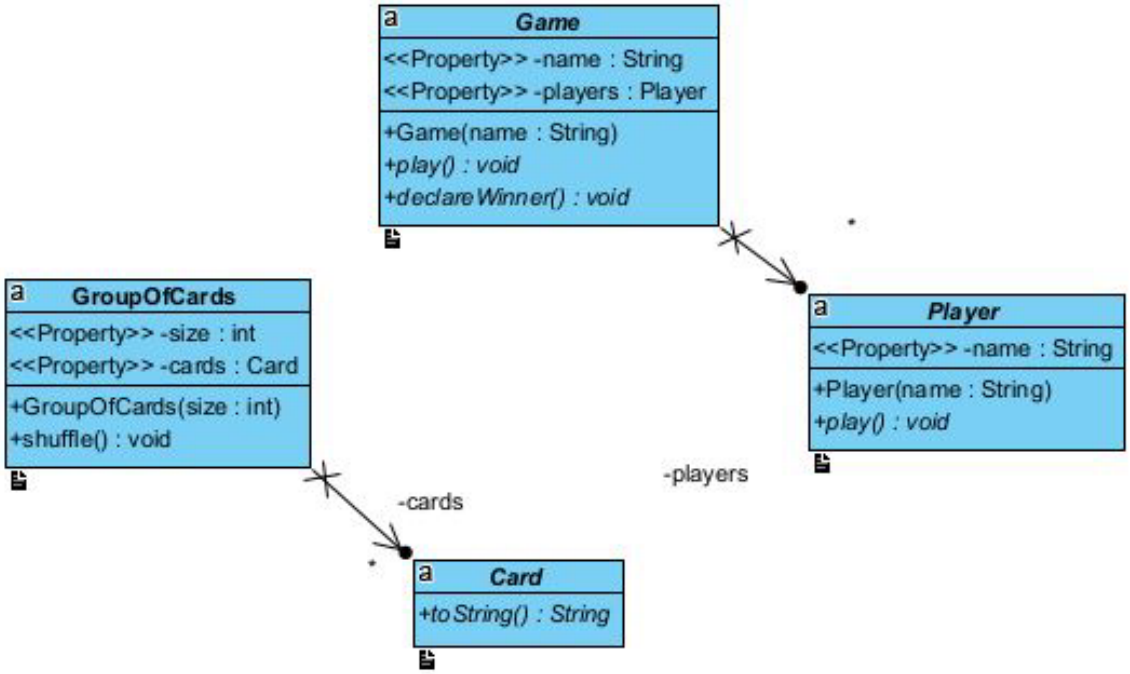
Winston Martinez

Deliverable 1

June 24, 2021

Contents

Domain Class Diagram



Design Document Template

OVERVIEW

1. Project Background and Description

Project goals and final vision

Our project goal is to create an interactive card game program using Java language. The program must function properly and smoothly. On the coding side, each class should be loosely coupled so that classes operate independently, and it will help us maintain the whole project more efficiently. Also, we will achieve high cohesion by describing a class as a single entity and making them logically connected to each other. Our final vision is to accomplish writing clean code through the entire project using debugging tools and code refactoring, applying OO concepts, and cooperation with team members on GitHub.

How to play the "War" card game

1. Each player gets dealt half the deck, 26 cards, and the cards are put face down in a stack in front of the players.
    - Both players turn their top card face up at the same time.
2. The person with the higher card wins that draw and takes both the cards.
    - Card Rank : (Highest) A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2 (Lowest)
3. If both players draw a card of the same rank, then there's a war.
    - The face up cards are left on the table and each player puts three cards face down on the table, and then puts one card face up.
    - The face up card determines who wins the war and gets all 10 cards that are on the table at this point.
    - If the face up card is again the same rank, then the war goes on, three more face down, one face up etc.
4. Once a player has all 52 cards, the person wins the game.
    - Or if a player finishes their cards during a war without having enough cards to finish the war, then the player loses immediately.

References

How To Play War. Gather Together Games. (n.d.). https://gathertogethergames.com/war.

WAR Card Game. Card Game | Play it online. (n.d.). https://cardgames.io/war/.

Wikimedia Foundation. (2021, April 16). *War (card game)*. Wikipedia. https://en.wikipedia.org/wiki/War_(card_game).

Description of the base code

There are four base classes. A concrete class GroupOfCards has two attributes; one is "size" whose type is integer and the other is "cards" whose type is ArrayList<Card>. In this class, we can define the size of "group of cards" and mix the "group of cards" using shuffle() method. An abstract class Card represents a card that our team chose (a regular deck of playing cards). We will have to override toString() method in its child classes. Another abstract class Game models our game. It also has two attributes; one is

"name" whose type is String and the other is "players" whose type is ArrayList<Player>. In this class, we will define the title of the game, and get an ArrayList of players from Player class. We will have to override play() method and declareWinner() method in its child classes. The last abstract class Player has an attribute "name" and its type is String. When we extends this class, we will have to override play() method.

2. Project Scope

Member Roles

This project comprises of four main roles, with such responsibilities as follows:

- **Programmer:** Responsible for writing code.
- **Code reviewer:** Responsible for ensuring that code conforms to scope/requirements outlined in design document.
- **QA/Bug Tester:** Responsible for testing code to try and break or exploit it.
- **Project lead:** Responsible for checking in on members and status of their tasks.

Role	Jinyoung	Juyoung	Tamara	Winston
Programmer	Card.java Main.java	Game.java Main.java	GroupOfCards.java Main.java	Player.java Main.java
Code Reviewer			X	X
QA/Bug Tester	X	X		
Project Lead			X	

Technical Scope

The project will be coded in Java and will be played through the console. The project will be considered complete when the requirements outlined High-Level Requirements are fulfilled with no errors during runtime.

3. High-Level Requirements

The new system must include the following:

- Ability for each player to register with the game
- Ability for player to check number of cards in hand
- Ability for player to forfeit the game
- Ability for the game to determine the value of the card
- Ability for the game to determine which card is ranked higher
- Ability for the game to communicate a win or loss
- Ability for players to play another round
- Ability to display final score when players no longer want to keep playing

The new system may include the following:

- Display card face in ASCII art

The new system will not include the following:

- Ability for players to change their names

4. Implementation Plan

GitHub Repository:
https://github.com/WinMillz/SundayAfternoonCodingProject.git

The expected use: each member checks in code at the end of each week as a regular routine as well as whenever we are told to have updates in code by team members. Codes that each member designs will be saved to this repository when it is pushed from a local repository.

Under the "Text files" directory, text files are stored. UML diagrams have their own folder which is "UML". All the codes are under "src/ca/sheridancollege/project".

Team Sunday Afternoon follows Programs industry standard as coding standards:

1. Easier to read: code that is difficult to read will take longer to decipher, even if you wrote it yourself. Proper standards like good indentation, spacing, documentation, self-documenting identifiers are required.
2. Easier to maintain: clients frequently require changes to existing programs. Whether you wrote the program or someone else did, it is difficult to modify and maintain code that violates standard practices: it can be difficult to find the module you need to change or the values that need to be updated.
3. Easier to debug and avoid coding errors.
4. More robust and reliable
5. More user-friendly and consistent
    - Standards for Java
        1. Naming Conventions
            - Primitive and Object Variables
            - Classes
            - Methods
            - Code
        2. Using Proper Indentation
            - Line length
            - Using proper spacing
            - Documentation

We expect ourselves to use NetBeans, Visual Paradigm Professional, and GitHub.

5. Design Considerations

When coding our project, we intend to adhere to the object-oriented principles of encapsulation, delegation, and flexibility/maintainability.

Encapsulation

Delegation

Flexibility/Maintainability

Ensuring loose coupling and high cohesion within our code allows us to be flexible and reduces maintenance.

Separating the Cards and GroupOfCards classes is an example of loose coupling. The Cards class is responsible for defining the total deck that we are working with. GroupOfCards, on the other hand, defines how these cards can be combined. If we want to play war with only half a deck, we just need to modify GroupOfCards and we do not have to touch Cards. If we want to create a different game using a deck of cards, we can reuse the Cards class without needing to remove any code. This reduces the amount of work needed for maintenance and lets us be more flexible with what we can create.

To maintain high cohesion in future classes, we will create classes that are responsible for only one thing. For example, when the cards played are the same value, the players go to “war.” We should separate the action playCard() and the condition required for “war” checkWar(). This allows us to reuse playCard() in another game. It would also allow us to easily find and change the conditions for “war” if we wanted to change the game to compare suits instead of card value.