Project Goals and Final Vision

Project Goals:

The primary objective of this project is to design and implement a command-line version of the "Go Fish" card game using Java. The goal is to create a functional and playable game that adheres to the traditional rules of "Go Fish" while utilizing object-oriented design (OOD) principles to ensure the code is maintainable, flexible, and reusable. The final vision includes a text-based interface where players can register, play the game, track their scores, and determine the winner.

How to Play "Go Fish":

"Go Fish" is a classic card game typically played by 2-6 players. The objective is to collect the most sets of four cards of the same rank. Here is a brief overview of the rules:

Setup:

A standard deck of 52 cards is used.

The dealer shuffles the deck and deals 5 cards to each player (7 if there are only two players).

The remaining cards are placed face down to form the draw pile.

Gameplay:

Players take turns asking another player for a specific rank (e.g., "Do you have any threes?").

If the asked player has any cards of the requested rank, they must hand them over.

If the asked player does not have any cards of the requested rank, they say "Go Fish," and the asking player must draw a card from the draw pile.

If the drawn card is the rank they asked for, they get another turn.

Players continue to take turns until the draw pile is empty and all possible sets have been made.

Winning the Game:

The game ends when all sets of four cards have been collected.

The player with the most sets wins.

Rules Reference:

For a complete set of rules, refer to Bicycle Cards - Go Fish Rules.

Starting Base Code:

The base code is written in Java and follows standard OOD principles. The primary classes include Card, Deck, Player, and Game. Key design patterns and conventions include:

Encapsulation: Protects the internal state of objects by using private fields and public methods for access.

Delegation: Responsibilities are distributed among classes, ensuring each class handles specific tasks.

Coding Conventions: Follows Java naming conventions (camelCase for variables and methods, PascalCase for classes).

Project Scope

Team Members and Roles:

Harrison: Project Manager and Developer

Member 2: Developer

Member 3: Developer

Member 4: Developer

Technical Scope:

The project will be considered complete when the following features are implemented:

Player Registration: Each player can register with the game, providing their name.

Gameplay Mechanics: Players can draw cards, ask for specific ranks, and go fish.

Score Tracking: Players can view their current score at any time.

Game Communication: The game communicates win or loss conditions.

End-of-Game Summary: A summary of the game is provided at the end, showing the winner and scores.

High-Level Requirements

The new system must include the following features:

Ability for each player to register with the game.

Ability for the game to communicate a win or loss.

Ability for players to know their status (score) at all times.

Implementation Plan

Git Repository URL:

[Git Repository URL]

Coding Standards and Tools:

Coding Standards: Follow Java conventions (camelCase for variables and methods, PascalCase for classes).

Tools:

Version Control: Git

IDE: NetBeans/Eclipse

Testing: JUnit

UML Diagrams: Visual Paradigm

Expected Use:

Each developer is expected to check in code at the end of each day or week. Text files and documents are stored in separate directories for organization.

Design Considerations

Encapsulation:

Example 1: The Card class encapsulates rank and suit, providing getters for these attributes, ensuring the internal state is protected and only accessible through public methods.

Example 2: The Deck class encapsulates the list of cards, exposing methods like shuffle and deal to interact with the deck without exposing the underlying list directly.

Delegation:

Example 1: The Game class delegates the responsibility of drawing a card to the Deck class, ensuring that the game logic does not directly manipulate the deck.

Example 2: The Player class delegates the action of giving a card to the hand list, maintaining a clear separation of concerns and improving code readability and maintainability.

Flexibility/Maintainability:

Example 1: The use of interfaces allows for future extension of different types of players (e.g., AI players), providing flexibility to extend the game without major refactoring.

Example 2: The separation of concerns is maintained by having distinct classes for game logic, player management, and deck handling, making the system more maintainable and easier to debug.

UML Diagram

The final UML diagram with all relations marked, illustrating the structure and interactions between classes:

Conclusion

The project aims to create a fully functional "Go Fish" game using Java, emphasizing good object-oriented design principles. The project structure, coding standards, and tools have been outlined, and the UML diagram illustrates the relationships and design of the system components. The deliverable includes the design document template and detailed use cases, ensuring a comprehensive and well-documented approach to the development process.