**Design Document Template**

**Project Goals:**

This project is to create a terminal based implementation of the card game "Go Fish" in Java. The goal is to create a fully functional game with shuffling, dealing, player turns, card requests and then determining the winner. The end goal is to have a codebase that can be expanded in the future.

**Game Description**

*Go Fish is a classic card game played by 2-6 players. The goal is to collect as many sets of 4 matching cards (books) as possible.*   
Here (Wikepedia, n.d.):

1. Each player is dealt a number of cards (7 for 2-3 players, 5 for 4-6 players).
2. The remaining cards are the draw pile.
3. Players take turns asking a specific opponent for a specific rank of card.
4. If the opponent has the card(s) they must give them to you.
5. If the opponent doesn’t have the card they say “Go Fish” and you draw a card from the draw pile.
6. Players form and set aside books when they get 4 cards of the same rank.
7. The game ends when all books are collected and the player with the most books wins.

**Current Starting Base Code**

The project is in Java and follows object-oriented programming (OOP) principles. The base code has classes for Card, Deck, Player and Game. Standard Java coding conventions are used, camelCase for variables and methods, PascalCase for class names and appropriate use of access modifiers for encapsulation.

**Project Scope**

1. ***Team Members and Roles***

- ***Harrison Daniel Dsouza*:** *Project Manager and Lead Developer*- ***Misha Punj***: *Developer and Tester*- ***Neil Patrick Saldanha*** : *Developer and Documentation Specialist*

***- Sukhmanpreet Singh Aulakh****:* *Developer and Documentation Specialist (2)*

1. ***Technical Scope:***The project features a text-based interface where players interact with the game via the terminal. The project will be considered complete when the game is fully functional, all defined test cases pass, and documentation is complete.

**High-Level Requirements**

***Player Registration and Login:***

* Players must be able to create accounts with unique usernames and passwords. The system should authenticate players during login and provide appropriate feedback for successful or failed attempts.

***Game Setup and Play:***

* The game must support 2-6 players who can join a game lobby. Once the required number of players join, the game should start with a shuffled 52-card deck, distributing cards among players. Players will take turns asking for specific card ranks from opponents. If the opponent has the requested card, they must hand it over; if not, the player must draw from the deck. Players should form and display sets of four cards of the same rank.

***Status Updates and Communication:***

* The game should provide real-time updates on the current player’s turn, the number of cards each player has, and the sets collected by each player. The game must announce the winner and provide a summary once all sets are formed.

***Scoring and Progress Tracking:***

* The game should display real-time scores and the status of each player, showing the sets collected and the number of cards. Additionally, the system should maintain a history of game outcomes, allowing players to view past game results.

***Code Quality and Project Management:***

* The project should follow standard Java coding conventions, ensuring that the code is modular and well-documented for maintainability. The team must use Git for version control, making regular commits and updates to facilitate collaboration.

**Implementation Plan  
  
*Git Repository URL:***

* <https://github.com/HarrisonDsouza/Project-GoFish.git>

***Repository Structure:***

* **src/:** Contains all Java source files.
* **test/:** Contains all test files.
* **resources/:** Contains text files, images, and other resources.
* **uml/:** Contains UML diagrams and other design documents.

***Expected Use:***

* Each developer is expected to check in code at the end of each day.
* Changes to the code bases should be added with appropriate commit messages.
* Changes should be made to the section assigned unless discussed prior.
* Text files are stored under a separate directory, while code and UML diagrams have their respective folders.

***Coding Standards and Tools:***

* + ***Coding Standards:*** Follow Java best practices and conventions.
  + ***Tools*:** Java, Visual Paradigm
  + ***IDE*:** NetBeans.
  + ***Version Control*:** Git and GitHub for repository management.
  + ***UML Tool****:* Visual Paradigm (VP) for creating UML diagrams.

**Design Considerations**

***Encapsulation:***

***Encapsulation is achieved by keeping class attributes private and providing public getter and setter methods.***

**Example 1**: The Card class has private attributes suit and rank, accessed via public getSuit() and getRank() methods.

**Example 2:** The Deck class maintains a private list of Card objects, with methods to shuffle and deal cards.

***Delegation:***

***Delegation is used to distribute responsibilities among different classes, promoting single responsibility.***

1. **Example 1:** The Game class delegates card dealing to the Deck class.
2. **Example 2:** The Player class delegates drawing cards from the deck to the Deck class's deal() method.

***Flexibility/Maintainability:***

***The code is structured to be flexible and maintainable, allowing for future enhancements.***

1. **Example 1:** The use of interfaces or abstract classes can allow for different types of players (e.g., AI players in the future).
2. **Example 2:** Clear separation of concerns allows for easy modification of specific game rules or adding new features without affecting the entire codebase.

**UML Diagram**

***The final UML diagram, as provided:***

A diagram of a card game

Description automatically generated

A diagram of a diagram

Description automatically generated

**References:**

Wikepedia. (n.d.). *Go Fish*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Go\_Fish