Swiggy Round 1 coding test

Overview:

- You have been given two full days to implement a solution to a design problem using Java, Go, or Node.js.
- This exercise is designed to assess your language and low level design skills, as well as your ability to write readable and concise code.
- **Quality of modeling of entities and behavior, well readable & succinct code, test coverage and a clean & frequent commit history** is key**.**
- Please keep these factors in mind as you solve the problem.

Problem Statement:

Design a multiplayer card game that supports multiple players (up to 4) and different types of cards (e.g. number cards, action cards, etc.). The game should follow the following rules:

- Each player starts with a hand of 5 cards.
- The game starts with a deck of 52 cards (a standard deck of playing cards).
- Players take turns playing cards from their hand, following a set of rules that define what cards can be played when.
- A player can only play a card if it matches either the suit or the rank of the top card on the discard pile.
- If a player cannot play a card, they must draw a card from the draw pile. If the draw pile is empty, the game ends in a draw and no player is declared a winner..
- The game ends when one player runs out of cardswho is declared the winner.
- BONUS: Aces, Kings, Queens and Jack are action cards. When one of these is played the following actions occur:
    - Ace(A): Skip the next player in turn
    - Kings(K): Reverse the sequence of who plays next
    - Queens(Q): +2
    - Jacks(J): +4
    - NOTE: actions are not stackable i.e. if Q is played by player 1 then player two draws two cards and cannot play a Q from his hand on that turn even if available

Rules of the Game:

- You can use Java, Go or Node.js to implement the solution, without using any third-party libraries or frameworks (common and essential helper libs and packages are allowed ex. math.rand() is ok.
- The code should be hosted on GitHub, and you should provide us with a link to the repository.

- You should provide a README file in the repository that explains how to run the code and any other relevant information.
- You should provide unit tests for your solution.
- We expect you to commit frequently with relevant commit messages. Multiple incremental commits are valued over one all-inclusive commit.

Evaluation Criteria:

We will evaluate your solution based on the following factors:

- **Simple design**: Does the code have a clear and simple design? Is it easy to understand and modify?
- **Readability**: Is the code well-organized and easy to read? Are the naming conventions clear and consistent?
- **Modelling**: Are the objects and classes used in the code well-designed and appropriate for the problem at hand?
- **Maintainability**: Is the code easy to maintain and modify? Are there any potential areas of concern or technical debt?
- **Testability**: Are there comprehensive unit tests for the code? Does the code have a high degree of test coverage?

We look forward to seeing your solution and wish you the best of luck!