Hello World! - Kethan, Samarth, Sameer

Overall Architecture, structure, and Design. (Highlight your Object-Oriented Design)

Architecture

The project is divided into different modules, where each module represents a specific aspect of the game:

Game Design Package: Contains all the classes that are directly related to the game's functionality.

Player: Manages player information such as name and the current amount of money the player.

Game: Controls the flow of the game, including starting rounds, managing bets, and determining winners.

Deck: Handles the deck of cards, ensuring cards are shuffled and dealt correctly.

Card: Defines the properties of a playing card.

Structure

Class-Based Structure: The project uses classes to define the blueprint of each module. Each class has its properties and methods relevant to its function. For example:

Player Class: Contains details about the player and methods to modify their money.

Game Class: Houses the logic to run the game and update the graphical user interface (GUI).

Deck Class: Stores a collection of cards and includes methods to shuffle and deal cards.

Card Class: Represents a single card with a suit and rank.

Object-Oriented Design Principles:

Inheritance: This is used to create specific types of players from a general player class.

Encapsulation: Each class keeps its data private and exposes functionality through public methods. This hides the internal state of the objects and protects them from unwanted external interference.

Abstraction: Some classes are designed to only outline methods without defining them completely, allowing them to be defined more fully in derived classes.

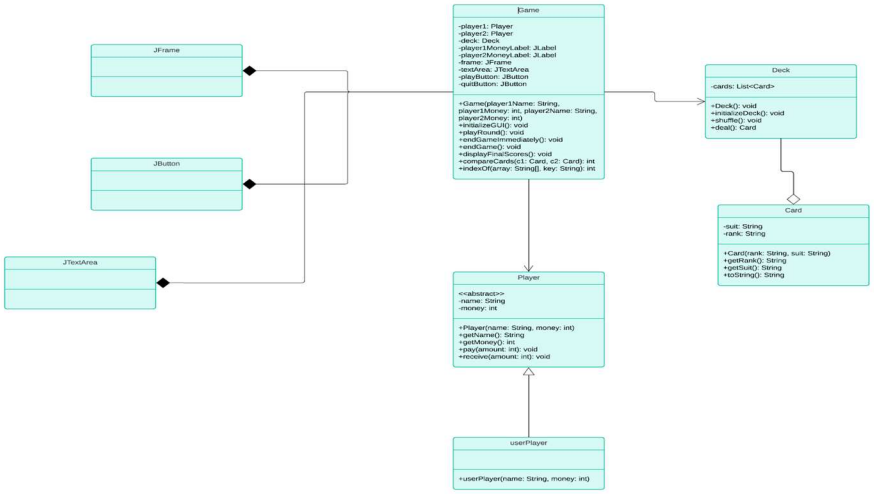
Polymorphism: The game can interact with different types of players through the same interface, treating them as general players, which allows flexibility in adding new player types without changing much of the game logic.

GUI: The game uses Java Swing to create an interactive graphical user interface that allows players to input data through dialogs, click buttons to play rounds, and view game status updates in real time.

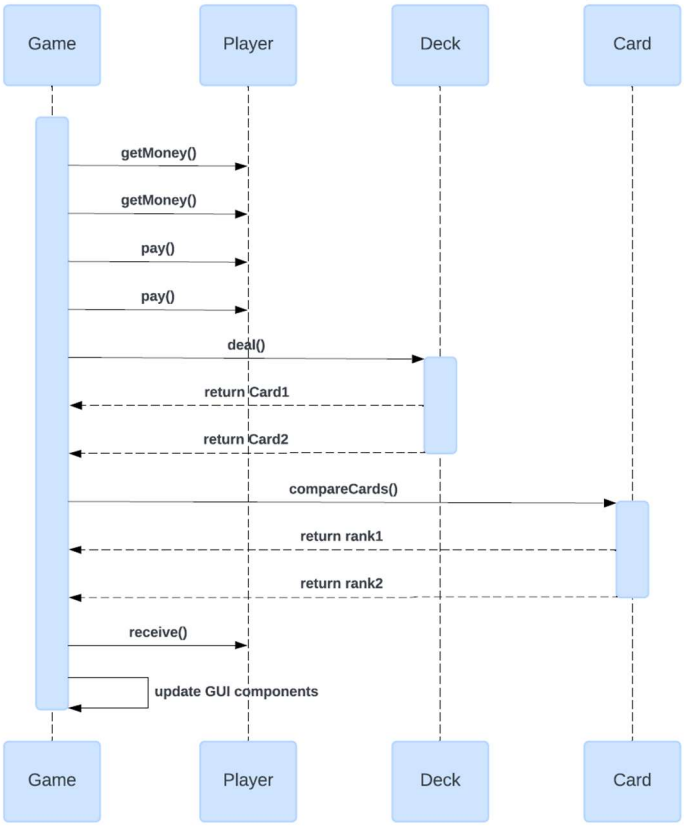
Exception Handling: The code includes basic error management to deal with potential issues during runtime. For example: The deal method in the Deck class throws an “IllegalStateException” if a card is requested when the deck is empty, preventing the game from crashing and providing a clear error message instead.

UML Diagrams (Class, Package, Sequence, Use-Case)

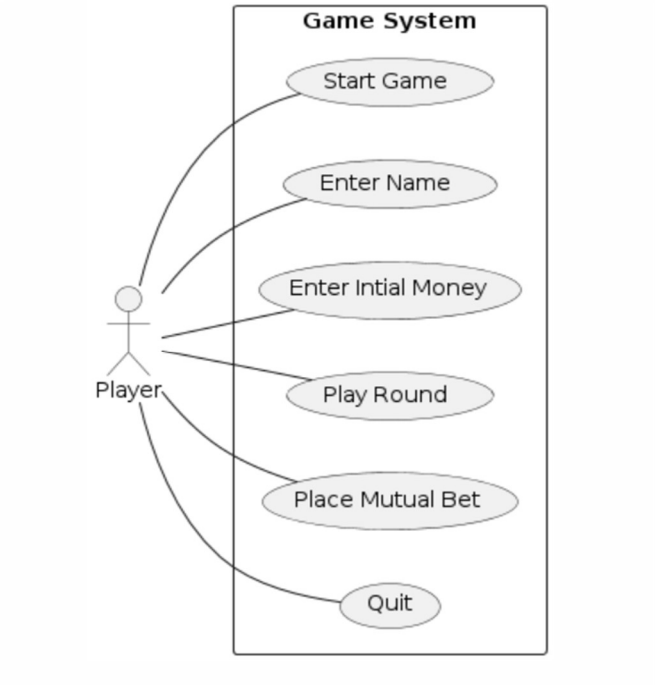
UML Class Diagram:



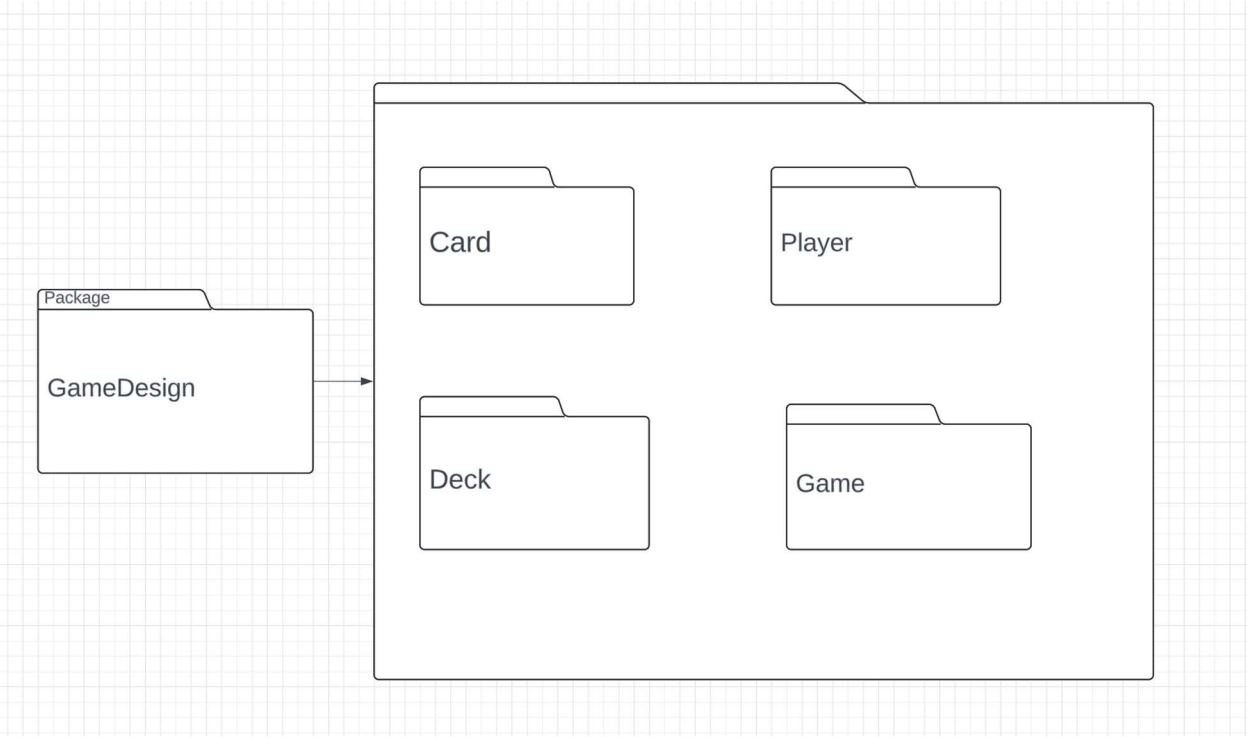
Sequence Diagram:



Use-case Chart:



Package Diagram:



Details of the functionality of the project

The project implements a simple card game where two players bet money on who will draw the higher card.

Initialization: The program starts by prompting the users to enter the names and initial amounts of money for two players.

Game Setup: It then creates a game instance with these two players.

Gameplay Interface: The main game interface is set up using a JFrame, where player information (name and current money) and game controls (Play and Quit buttons) are displayed. A text area is included to show game logs and results.

Playing Rounds:

- Players are prompted to enter a bet amount for each round.
- The game deals a card to each player from a deck.
- The cards are compared, and the player with the higher card wins the round, receiving the total bet amount.

The GUI is updated to reflect changes in player funds and display the outcome of the round.

End Game:

- The game can end if a player runs out of money, chooses to quit or cards in the deck are finished.
- Final scores are displayed, and the game window is closed.

Describe your GUI design and application

The GUI uses Java Swing to create an interactive and visually appealing interface:

Main Window: Displays players' information (name and money), game logs, and control buttons (Play, Quit).

Player Panels: Show each player's current money and name in separate panels, enhancing readability and user experience.

Text Area: Displays game actions, results, and messages, allowing players to follow the game's progress.

Buttons: Control game flow—players can initiate a new round or end the game.

How to use the project source code

You can use this project's source by creating a new Java project, and create Java files for each class (Player, userPlayer, Game, Deck, Card) inside a package named GameDesign and have a main file that creates an object for this game.

Then the game can be played by entering player names and their starting funds via dialog boxes, placing bets, and proceeding through the rounds according to the game's rules, a card is drawn from a shuffled deck and whoever gets the higher card wins the round.

Contribution of each Team member

Sameer 34% - Idea, design, programming, video-making

Samarth 33% - programming, Report, GUI implementation

Kethan 33% - programming, GUI implementation, UML diagrams