Project 6

Status Summary :
Project Title : Exploding Kittens Game
Team Members : Abishek Goutham and Arpita Ambavane

Work Done:
Abishek Goutham

- Created a Spring Boot application
- Initialized required Classes and interfaces : Game, Game Operations, Player
- Initialized Controllers and Services for Game, Player and Web Socket
- Created Command Pattern Structure
- Implementation of Strategy Pattern
- Configure the third UI page for main game play, i.e. after the start of the game with Player and Card elements.
- Established a Web Socket connection between React and Spring
- Included Drag and Drop functionality for the Hand cards, where players can pick or drop the card to the Discard Deck.
- Allows the player to draw a card from the deck.
- Display the all cards or current card on play or played by another player

Arpita Ambavane

- Initialized required Classes and interfaces : Hand, Cards
- Implementation of Factory Pattern for cards
- Configured API routes to start and get player name of the game
- Created React Application for User Interface.
- Created two pages which use the above APIs, i.e. the initial screen and the page where the user enters their names.
- Taking the Names of the players (input from user).
- Shuffle the deck of cards at the beginning of the game.
- Deal five cards to each player at the start of the game.
- Allow players to take turns playing a card from their hand.
- Displaying the name of players on the UI

Changes or Issues Encountered:

Issues Encountered :

1. Difficulty with Drag and Drop functionality of cards
2. Web Socket Implementation for the first time

Changes Encountered :

1. Command Pattern functionality changes
2. Added Player and Web Socket Service
3. Added Player and Web Socket Controller
4. Design changes

Patterns:

Factory Pattern :
The factory pattern could be used to create different types of cards. This pattern is applied to Card classes, where each of them has their own properties and behaviors. We have created concrete creators for each type of action cards such as Attack, Defuse, ExplodingKittens, SeeTheFuture, Shuffle, Skip, and StealACard(Favor) in a structure parallel to the main program. There is an abstract CardFactory class which handles all the concrete classes. By using the factory pattern, we can easily add new types of action cards without changing the super class.

Strategy Pattern :
The strategy pattern could be used to implement different playing strategies. The super class could be an PlayingBehaviour interface object and the subclasses could be specific types of playing strategies, such as EndsTurn, DoesNotEndTurn, and MightExplode, each with their own strategies of playing. By using the strategy pattern, we can easily switch between different playing behaviors without changing the PlayingBehaviour super class.

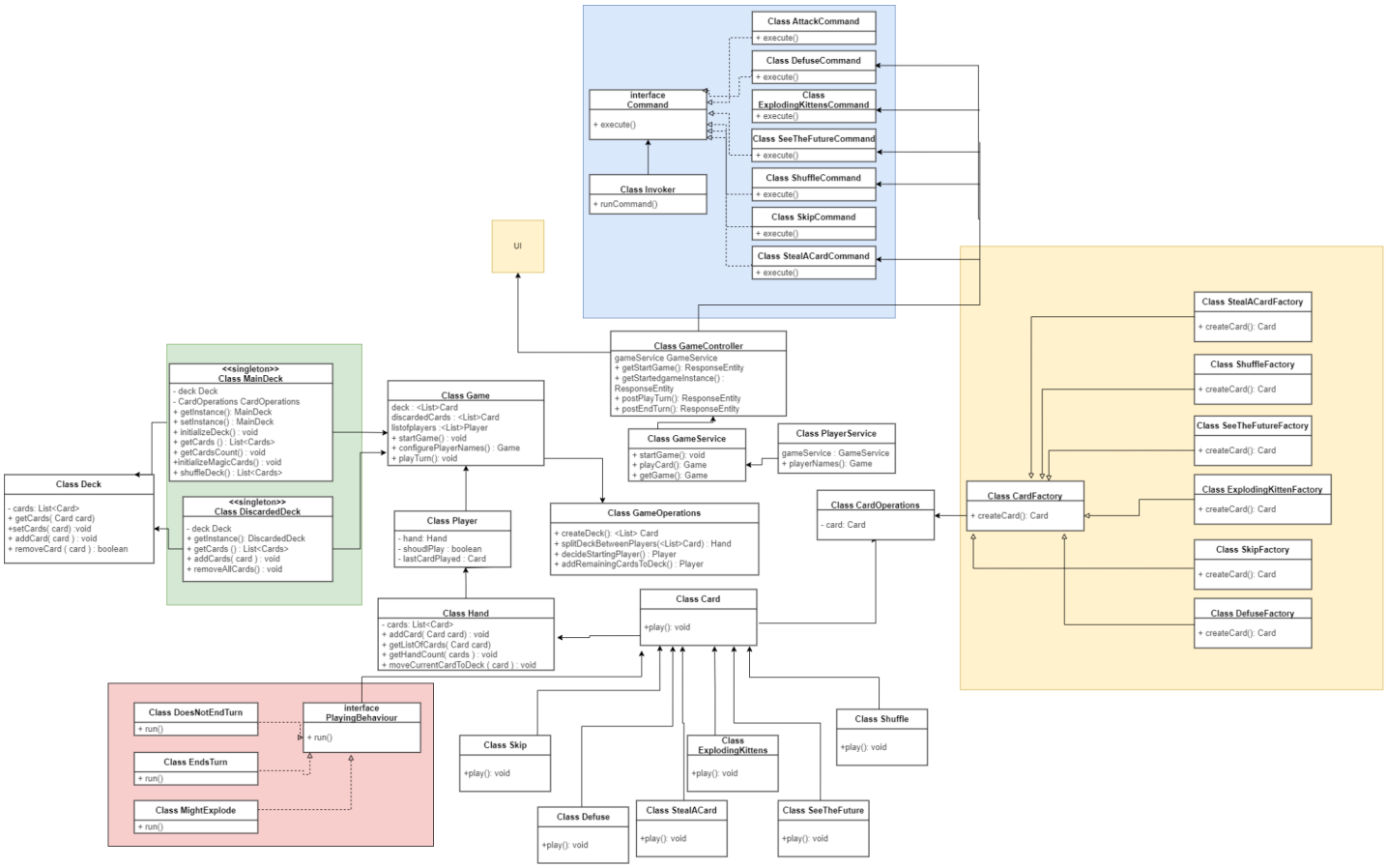
These patterns can make the code more modular and easier to maintain. Additionally, both patterns can help to reduce coupling between classes, making the code more flexible and easier to modify.

Command Pattern :
The command pattern could be used to implement actions of cards such as skipping the turn, shuffling the cards, etc. Each action would be encapsulated as a command object with an execute method that carries out the action. The Command Interface is connected to Game Controller via concrete commands. Invoker Class invokes the commands based on the input of the card and calls the respective card command class to execute() method. The command pattern can also help to decouple the code, since the object invoking the command doesn't need to know anything about the object that receives it, only that it can execute the command.

The command pattern can be a powerful tool for implementing actions in game development and can help to improve the flexibility and maintainability of the code.

Class Diagram :

We have added only those elements that we have worked on the past weeks. Design patterns are highlighted.



Plan for Next Iteration :

- Implementation of Observer and State Pattern
- Deck, Discarded Deck and Withdrawn Deck convert to Singleton classes.
- Display the Number of cards in Hand left after playing the turn.
- Implement the various card actions, such as exploding kittens, defusing, skipping a turn, and so on.
- Notifying the turn of the player - For Example: ‘Abishek’s Turn’
- Alert the user if the option picked is invalid.
- To place the Exploding Kitten in the deck after playing defuse card - Give options to the player to choose where to place the Exploding Kitten card in the pile.
- Update the number of cards from players when a favor (StealACard) card is played
- End the game when only one player is left or when all exploding kittens have been defused.
- Work on UI functionalities to make game more user-friendly.
- Complete the game implementation.