COS20007-Object Oriented Progamming

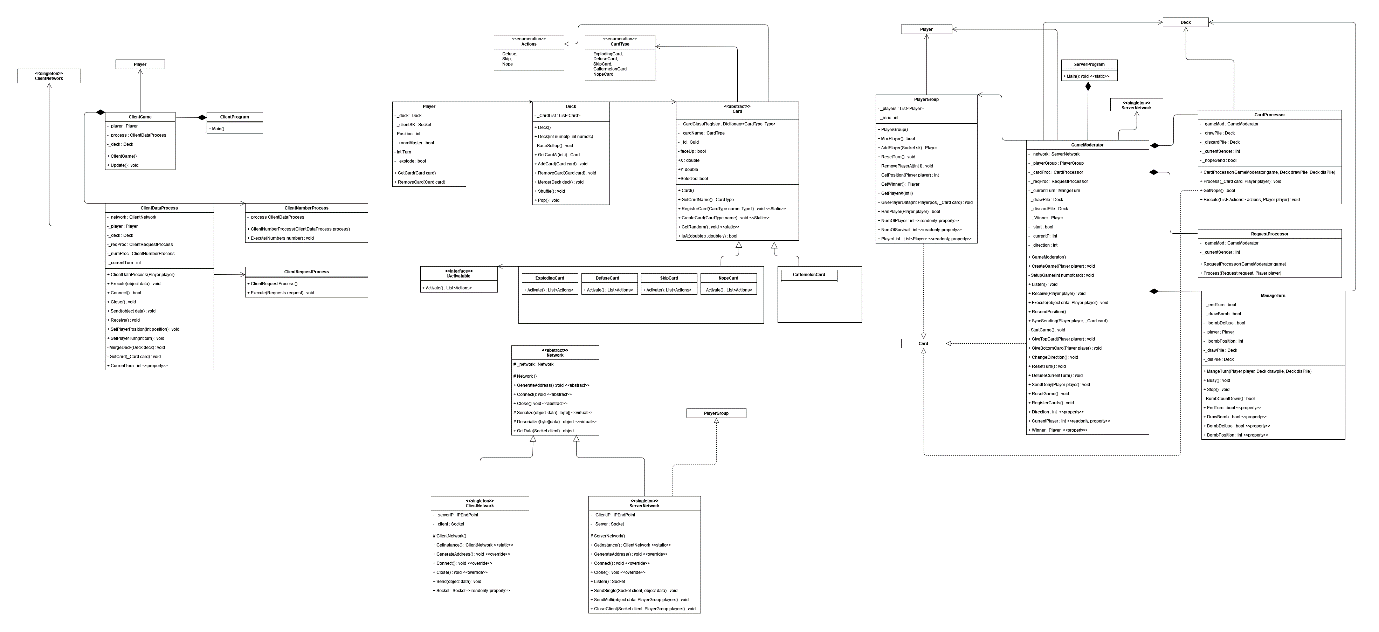
Custom Program Documentation

Exploding Kittens

Last update: October 3, 2021

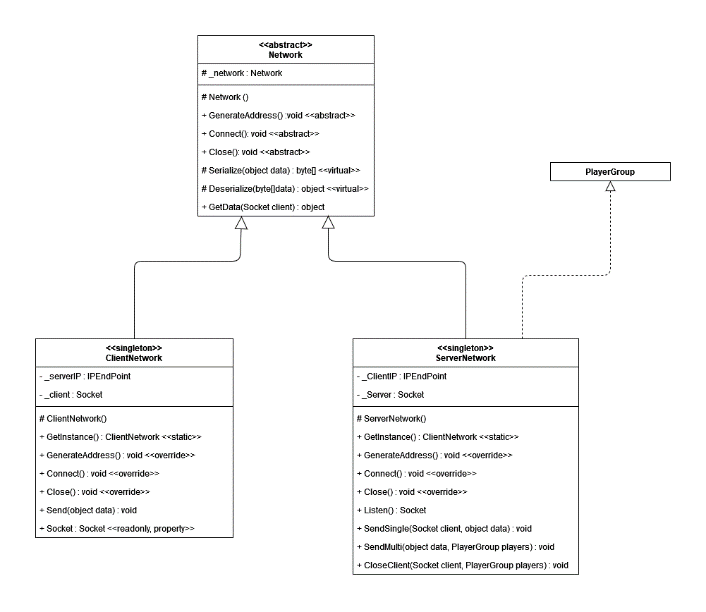
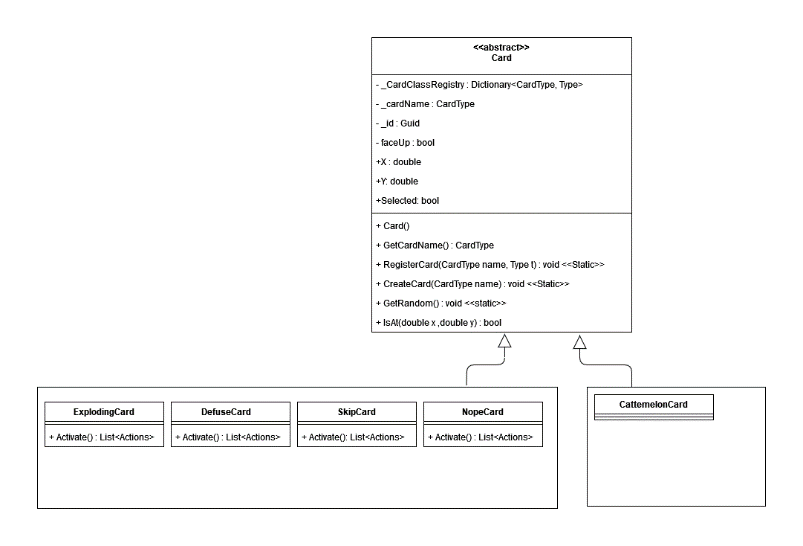
Github Repos: <https://github.com/Jiruu246/Exploding_Kittens.git>

UML (for the latest update please do check out the files in the Documentation folder)

**How it works:** There are two main components: Server and Client, they communicate by sending object data throughout the network. The Server will responsible for creating a player object for each client who connects to it, managing the game based on the game rules, listening for player requests, and tell the player what to do. The Client will be responsible for helping the actual player interact with the game using CLI or GUI. It tells the player what they got or what they need to do and sends the player actions/data to the Server.

This Program is based on a board game called “Exploding Kittens”. The game start with two or more players each will draw a card from the draw pile, if someone draw a “exploding card” they will be eliminated, the winner is the last one who survive. Other cards that are not “exploding card” will help the player reduce the chance they got “Explode”.

**Inheritance and Polymorphism**

****

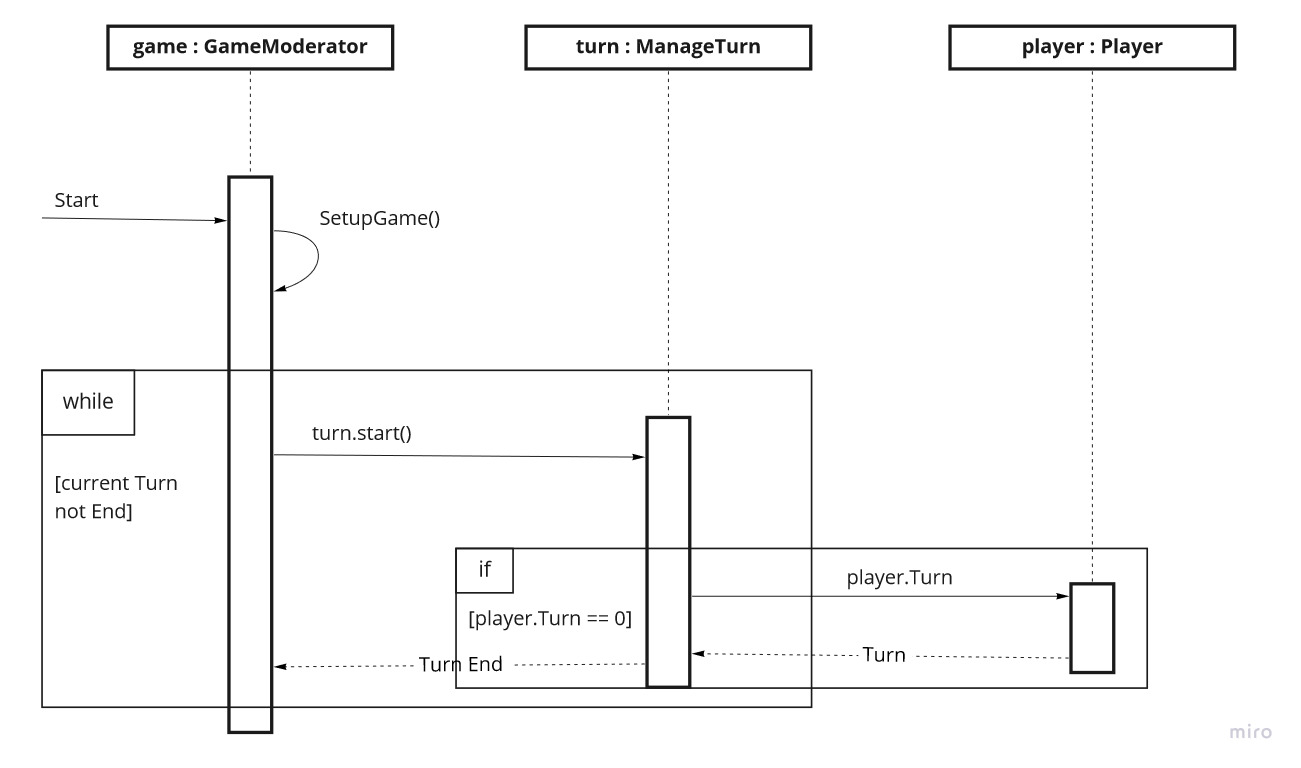
Two significant inheritance is the Network class and Card Class:

+ The Network Class is responsible for transferring data between clients and server. It has 2 children: ClientNetwork and ServerNetwork

+ The Card Class represents a single card in the game. It has many child classes, each is a unique type of card with different actions.

**Main procedure**

The main class that control the game logic and rules is the GameModerator Class

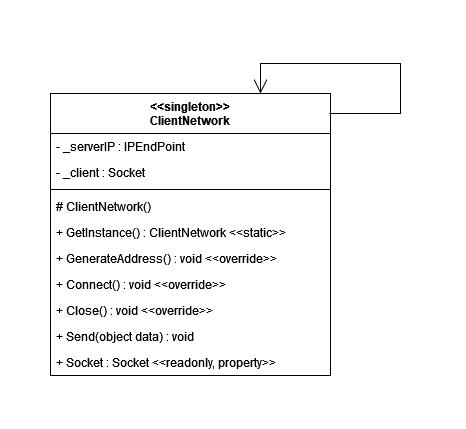
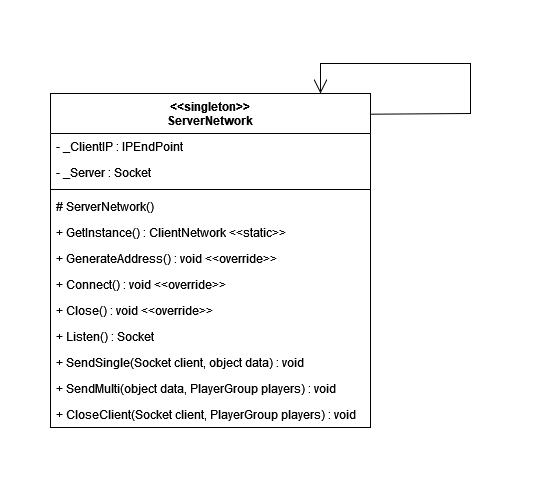
****

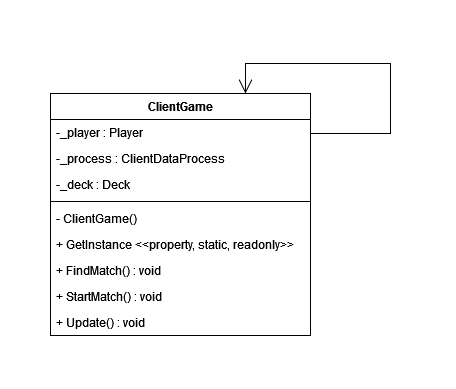
When the player sends a signal to start the game it starts the loop. First, it loops through each player and checks for their turn, while the player’s turn does not end it will keep waiting for the player to send the “draw” request. During this time, player can play any card that they have. After the player send the draw request, their turn will end and that player will receive a card from the draw pile. If they draw an exploding card, they will be exploded and eliminate from the game.

**Design Patterns**

1. Singleton pattern

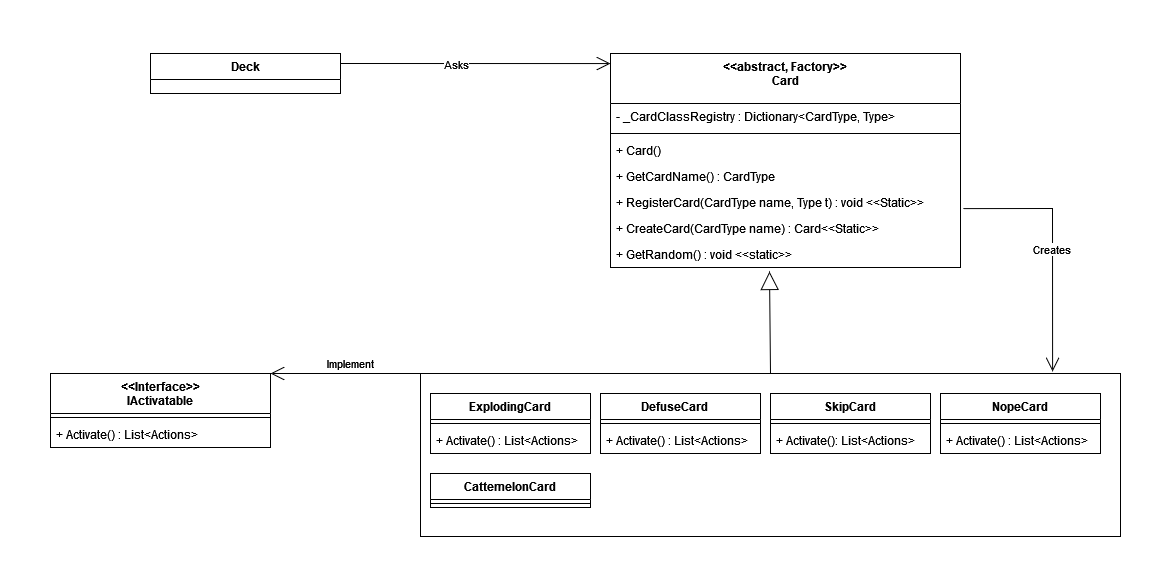
Networking Class use Singleton pattern, as there should only be one network object to manage connection (each belong to the corresponding project Client and Server).



ClientGame use Singleton pattern because there is only one game instance when the player runs the Client project.

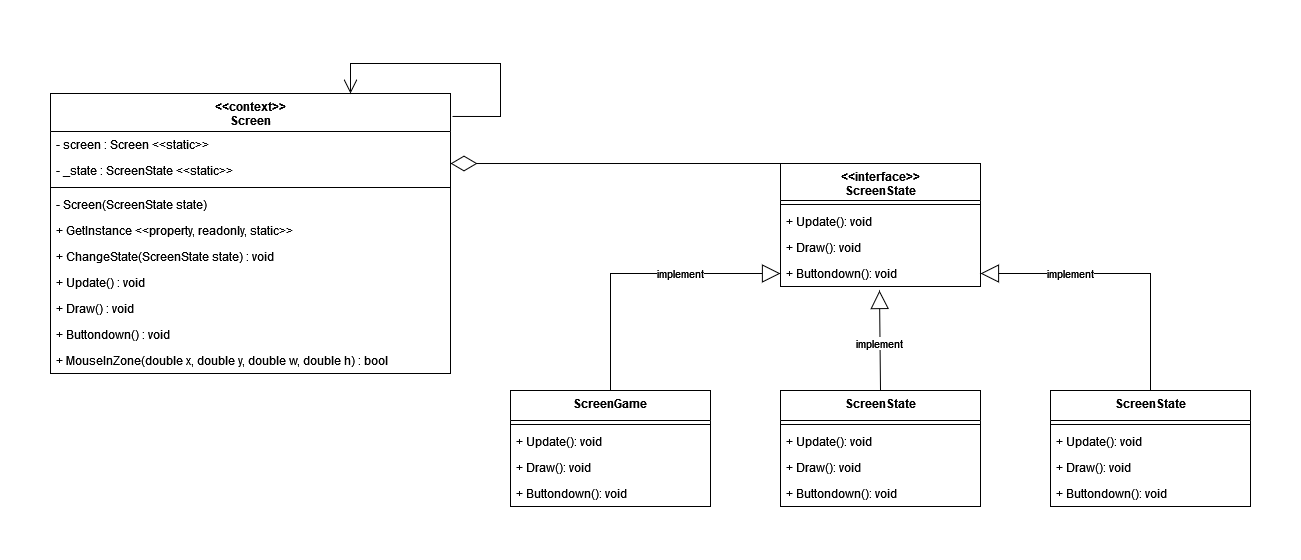
1. Factory pattern

The Card class is the factory that responsible for creating different type of card. Whenever a Deck need to add any kind of card it will ask the Card class to create



1. State Pattern

The Screen will have different state and when it needs to change its state the ChageState Method will be called.



1. Adapter pattern

Client project use a GUI adapter to access to Splashkit library

