pyMonopoly

Name of project: pyMonopoly

Team members: Tianyi Ma

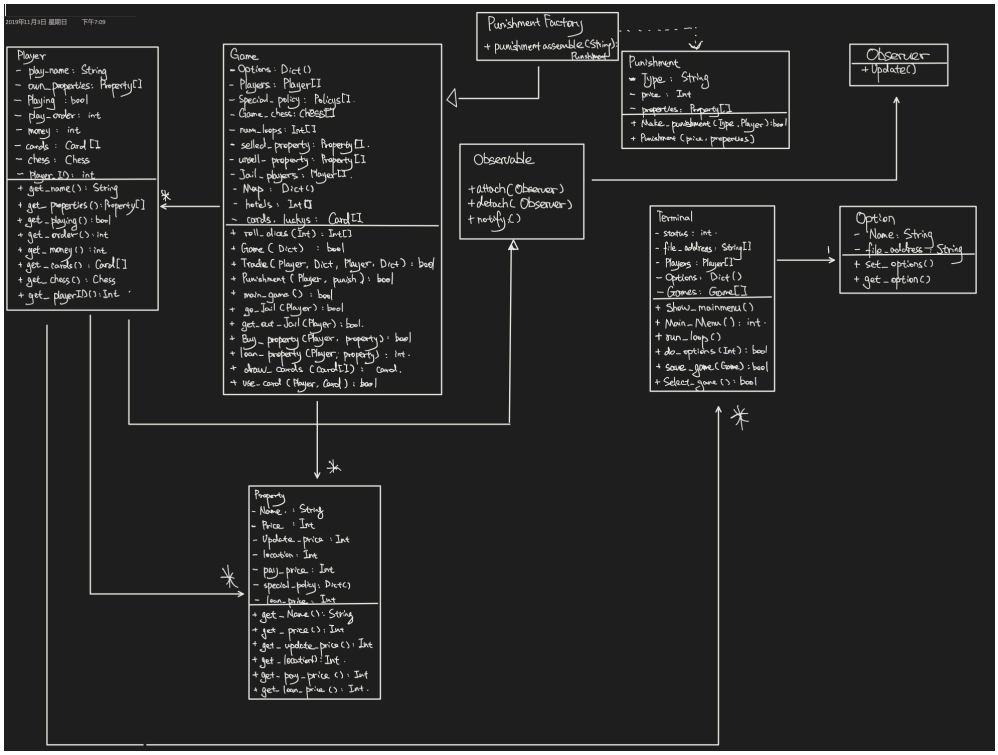
Final State of System Statement :
I finished most functions of the program, and also create some empty functions the allow users to do their own style monopoly games!
Function completed:

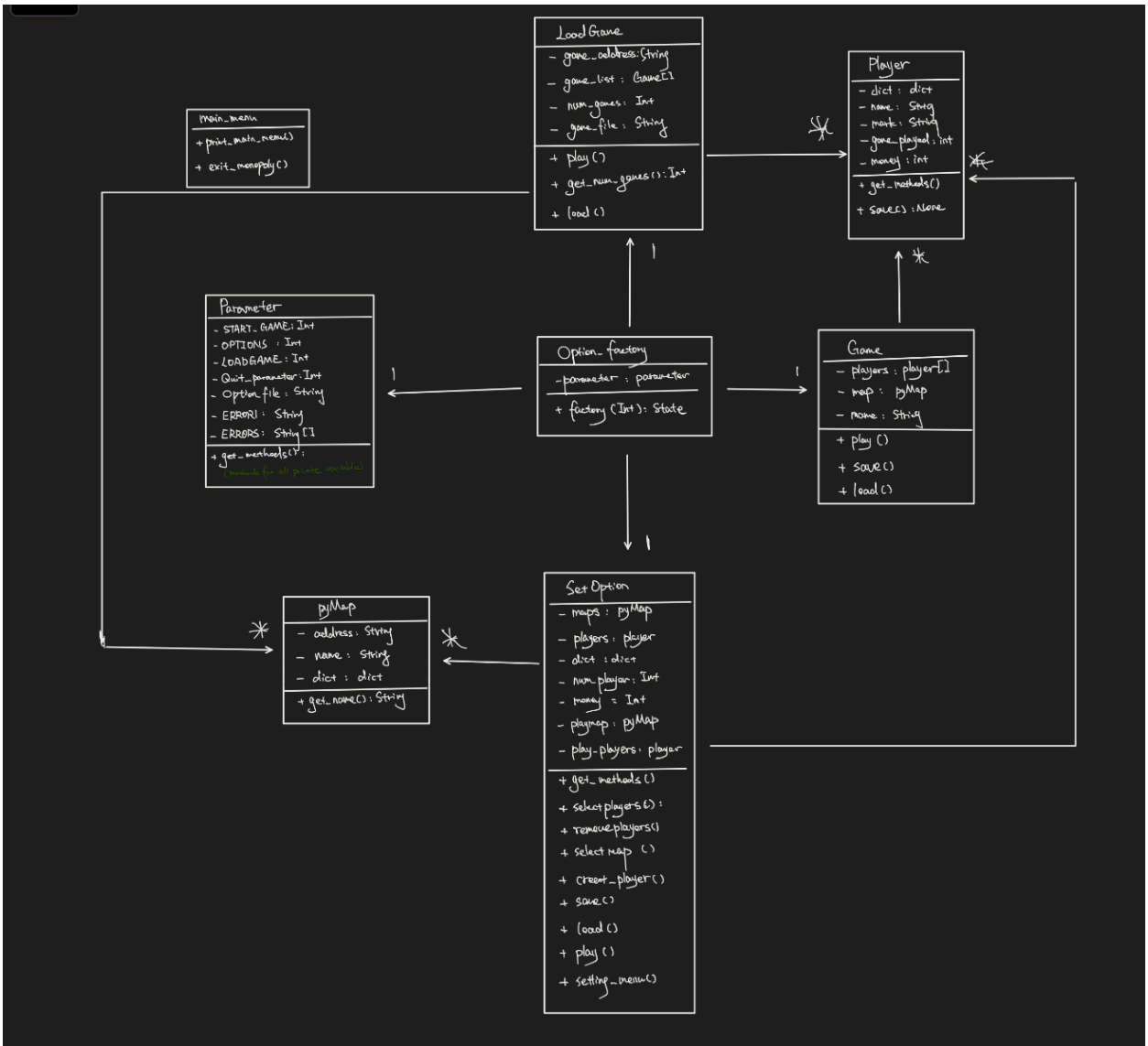
1. Play game:
   - Roll dice
   - Buy properties
   - Switch player
   - Save and quit game
2. Change settings:
   - Create player
   - Load map
   - Set startup money
   - Set number of player
   - Remove selected player
   - Save and go back
3. Load game
4. Exit the menu

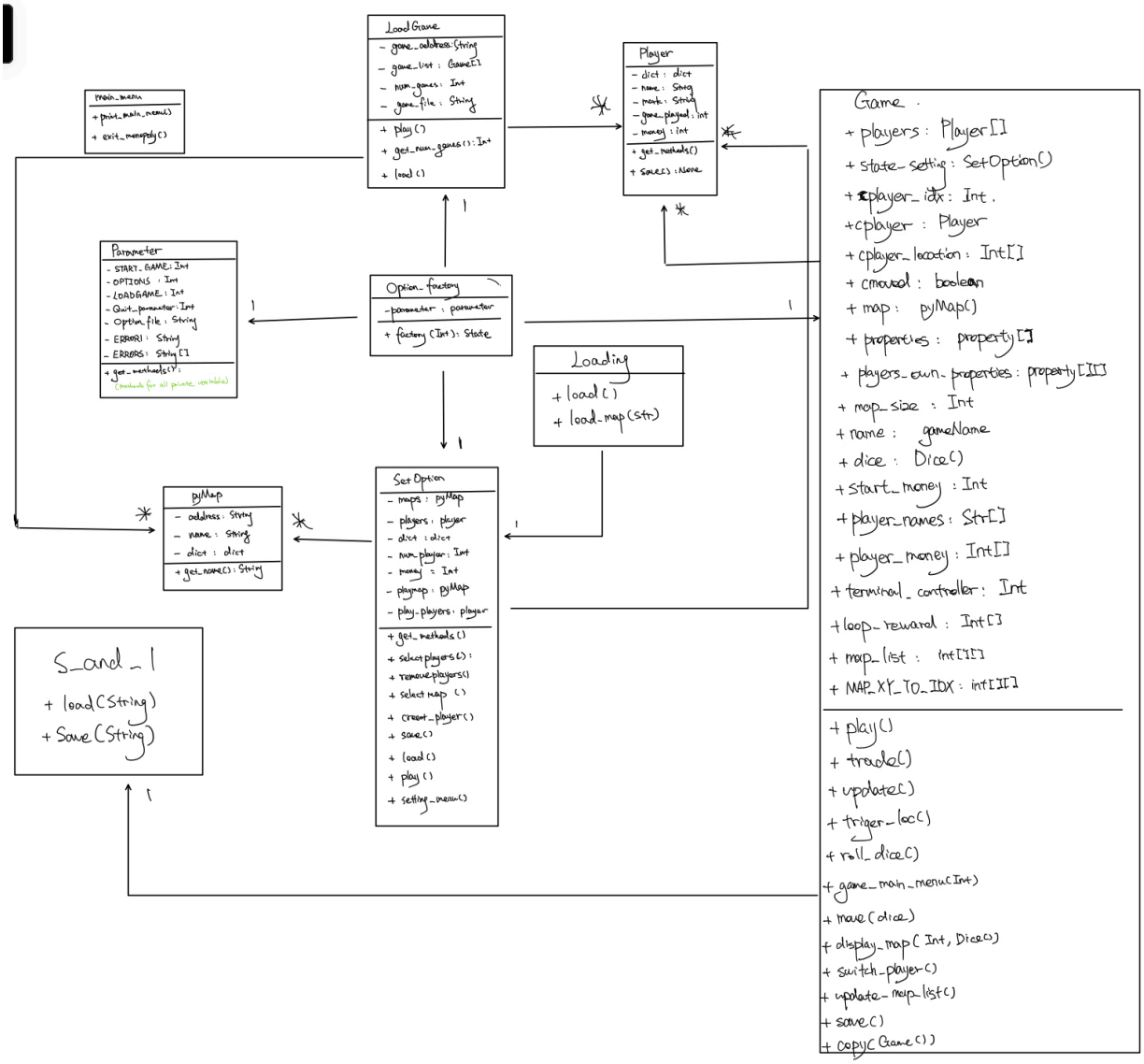
Function not completed, and open for users to develop:

1. Trade
2. Special cards (chance and welfare)
3. Special map

Final Class Diagram and Comparison Statement:
Patterns: Factory pattern, strategy pattern.
Final Class Diagram and Comparison Statement:
The main difference between the UML in project 4 and the final submission is some specific variables' names, and observer class. I did not use observer pattern, instead, I use update() function to keep tracking the players status.
The difference between the UML in project 5 and the final submission is class game. I finished implement class game before the final submission.
Diagram in project 4, and key changes:
Bellow is UML diagrams from project 4, project 5, and final report. There is no significant change between project 4 and final submission, except more variables.


Punishment Factory
+ punishment assemble (String)
Punishment
Punishment
- Type : String
- price : Int
- properties : Property[]
+ Make_punishment (Type, Player) : bool
+ Punishment (price, properties)
Observer
+ Update()
Player
- play_name : String
~ own_properties : Property[]
~ Playing : bool
~ play_order : int
- money : int
- cards : Card[]
~ chess : Chess
~ Player_ID : int
+ get_name() : String
+ get_properties() : Property[]
+ get_playing() : bool
+ get_order() : int
+ get_money() : int
+ get_cards() : Card[]
+ get_chess() : Chess
+ get_playerID() : Int
Game
- Options : Dict()
- Players : Player[]
- Special_policy : Policies[]
- Game_chess : Chess[]
- num_laps : Int[]
- selled_property : Property[].
- unsell_property : Property[]
- Jail_players : Map[].
- Map : Dict()
- hotels : Int[]
~ cards, luckys : Card[]
+ roll_dices (Int) : Int[]
+ Game (Dict) : bool
+ Trade (Player, Dict, Player, Dict) : bool
+ Punishment (Player, punish) : bool
+ main_game () : bool
+ go_Jail (Player) : bool
+ get_out_Jail (Player) : bool
+ Buy_property (Player, property) : bool
+ loan_property (Player, property) : int
+ draw_cards (Card[]) : Card
+ use_card (Player, Card) : bool
Observable
+attach(Observer)
+detach(Observer)
+notify()
Terminal
- status : int
- file_address : String[]
- Players : Player[]
- Options : Dict()
- Games : Game[]
+ Show_mainmenu()
+ Main_Menu() : int
+ run_loop()
+ do_options (Int) : bool
+ save_game (Game) : bool
+ Select_game () : bool
Option
- Name : String
- file_address : String
+ set_options()
+ get_option()
Property
- Name : String
- Price : Int
- Update_price : Int
- location : Int
- pay_price : Int
- special_policy : Dict()
- loan_price : Int
+ get_Name() : String
+ get_price() : Int
+ get_update_price() : Int
+ get_location() Int
+ get_pay_price() : Int
+ get_loan_price() : Int


Load Game
- game_address: String
- game_list : Game[]
- num_games : Int
- game_file : String
+ play()
+ get_num_games() : Int
+ load()
Player
- dict : dict
- name : Strig
- mark : String
- game_played : int
- money : int
+ get_methods()
+ save() : None
main_menu
+ print_main_menu()
+ exit_monopoly()
Parameter
- START_GAME : Int
- OPTIONS : Int
- LOADGAME : Int
- Quit_parameter : Int
- Option_file : String
- ERROR1 : String
- ERRORS : String[]
+ get_methods():
Option_factory
- parameter : parameter
+ factory (Int) : State
Game
- players : player[]
- map : byMap
- name : String
+ play()
+ save()
+ load()
byMap
- address : String
- name : String
- dict : dict
+ get_name() : String
Set Option
- maps : byMap
- players : player
- dict : dict
- num_player : Int
- money = Int
- playmap : byMap
- play_players : player
+ get_methods()
+ select players()
+ remove_players()
+ select map ()
+ creat_player()
+ save()
+ load()
+ play()
+ setting_menu()



Third-Party code vs. Original code Statement:
All the codes in my program are made by myself.

Statement on the OOAD process for your overall Semester Project:
1. My program is a good open source for developers to create their own games
2. My code is consider of “code smell”.
3. I develop some error response for users, if they does not use the program correctly.

Github link:
`https://github.com/MTYBilly03/pyMonopoly.git`