Project Report [COMP2021]

Object- Oriented Programming

SHOAIB Muhammad [18079999D]

1. Introduction

This document describes the design and implementation of the Jungle game. The project is part of the course COMP2021 Object-Oriented Programming at PolyU. The following sections describe the requirements that were implemented, and the design decisions taken. The last section describes the available commands in the game.

2. The Jungle Game

This game is played on a 9 x 7 board and with two players against each other. There are dens, traps and rivers on the gameboard. Each player has 8 different animals with different ranking. All animals can only capture animals with lower rank or equal rank.

The animal rank in descending order is: Elephant > Lion > Tiger > Leopard > Wolf > Dog > Cat > Rat.
With one exception that Elephants do not beat rat. On the contrary, rat can capture the elephants.

You win the game either by getting your animal to the den or by defeating all your opponent's animals.

2.1 Requirements

Req01: As the program is launched, the user is asked whether he/she wants to load a new game or load an existing (saved) one.

Method startOperation() in class JungleGame takes the user input and validates it in the method startOpCmd(). startOpCmd() returns true if the command that user entered is valid, and false if it is invalid.

Req02: At the beginning of the program, two players, X and Y, are asked to input their names respectively. The initial board is printed, and Player X is asked to enter his/her (first) command.

Method playerInitialize() asks for the names of both players if a new game is started. If a saved game is loaded, the player names are loaded from the data[][] array. Method printBoard() prints the current board. Method gameStart() asks for Player X to make his move first.

Req03: A command can be one of the following:

- save: saves the current game into the [filePath].
- open: loads a saved game from [filePath]. If current game is not saved yet, the user is prompted to save it first.
- move: to move a piece from [fromPosition] to [toPosition].

The program uses the method cmdHandler(String userInput) to know what kind of command the user has entered.
For the save command, the game is saved to filePath using the method saveCmd(String filePath).
For the open command, the opened from filePath using the method openCmd(String filePath). If the current game is not saved yet, user is prompted to save it (prompt given in case statement of "open" in method cmdHandler).
For the move command, the method moveCmd(String pos1, String pos2) is used to move from pos1 to pos2.
If the command is invalid (does not match any of the above), the method cmdHandler uses case statement to identify that, and calls the method errorMessage() to prompt the user accordingly.

Req04: If the move command is valid, it is executed. If the move command is invalid, an error message should be displayed, and this invalid command should not affect the state of the game.

Method cmdHandler(String userInput) that takes userInput (command entered by the user) and the case that matches the move statement, it is validated by the method moveCmd(String pos1, String pos2) where pos1 is the initial position and pos2 is the position where the piece needs to be moved. This method uses the abstract class Chess (and of course its methods) and its sub-classes to validate the move. The method move (String pos) returns true if the move command to position pos is valid and false if it is not. If the move is valid, it is made, and the updated game board is printed. The game state is not affected by a false move and in such a case, the user is prompted that his command is invalid.

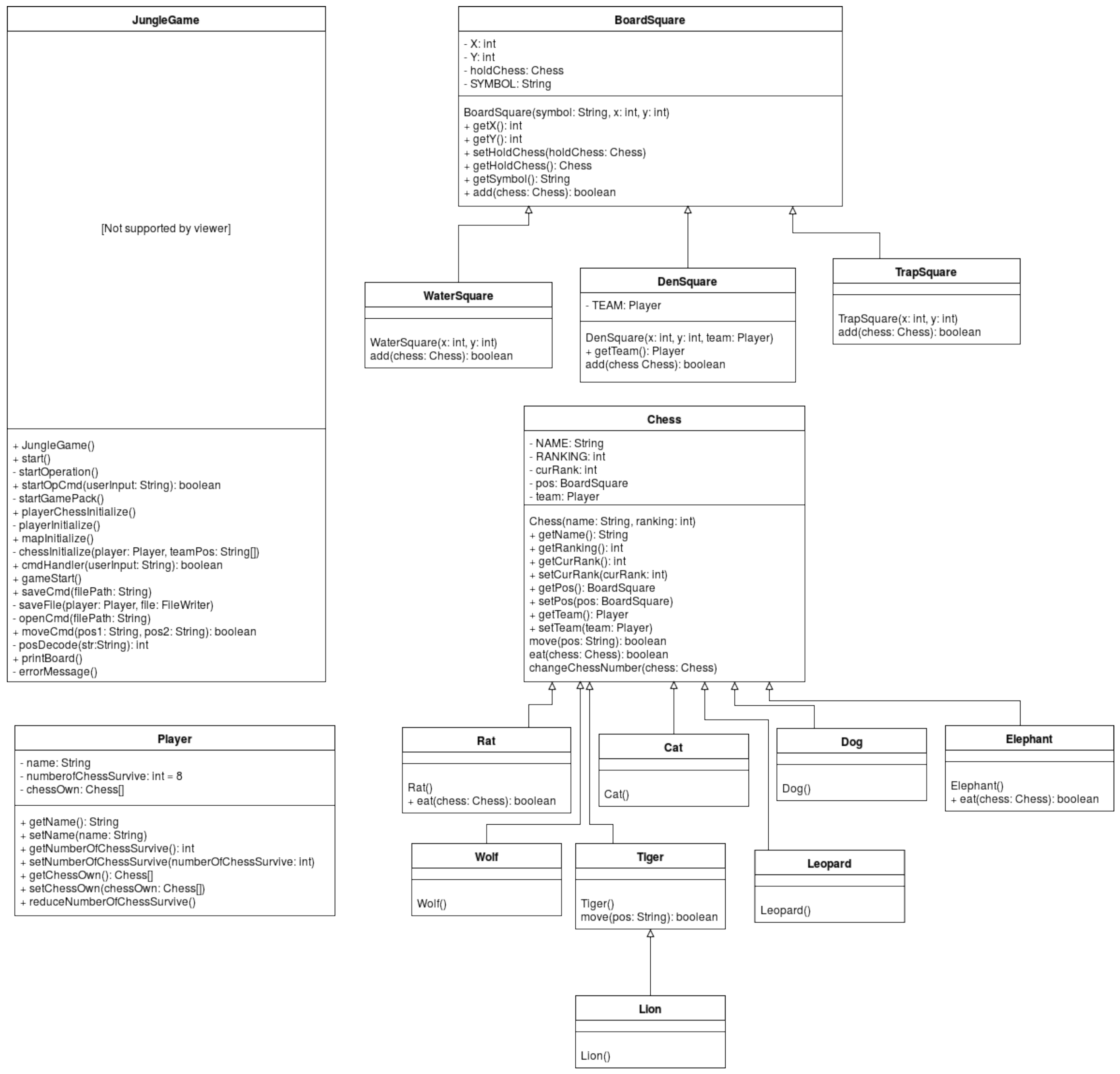
Req05: After each valid move, the updated game board is printed, and the game checks if a player has achieved the goal.

The gameStart() method prints the board after game is started, and after every valid move/update of the board. The variable isCompleted is set to true after checking if the game has been finished in the method add of class DenSquare that extends/is a sub-class of BoardSquare. When the game is
completed, the winner variable is assigned the winner's name,
and the winner's name, along with the victory message, is
printed.

Req06: Upon an invalid command, an error message is displayed, and the same player should is prompted to input another move. If the invalid command is a move command, the current game board should also be printed.

As mentioned previously, the method cmdHandler(String userInput) validates user input and if the input is an incorrect command, the method errorMessage() is called to prompt the user that the command is invalid. If it is a move command, the valiable valid in DenSquare is set to false and the user is not only prompted that the command is invalid but is asked to make the move again and the game board is printed.

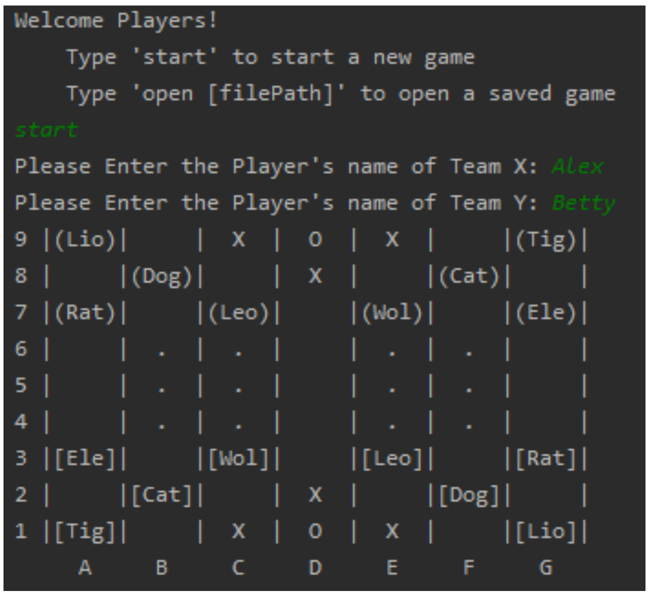
2.2 Design


JungleGame
[Not supported by viewer]
+ JungleGame()
+ start()
- startOperation()
+ startOpCmd(userInput: String): boolean
- startGamePack()
+ playerChessInitialize()
- playerInitialize()
+ mapInitialize()
- chessInitialize(player: Player, teamPos: String[])
+ cmdHandler(userInput: String): boolean
+ gameStart()
+ saveCmd(filePath: String)
- saveFile(player: Player, file: FileWriter)
- openCmd(filePath: String)
+ moveCmd(pos1: String, pos2: String): boolean
- posDecode(str:String): int
+ printBoard()
- errorMessage()
BoardSquare
- X: int
- Y: int
- holdChess: Chess
- SYMBOL: String
BoardSquare(symbol: String, x: int, y: int)
+ getX(): int
+ getY(): int
+ setHoldChess(holdChess: Chess)
+ getHoldChess(): Chess
+ getSymbol(): String
+ add(chess: Chess): boolean
WaterSquare
WaterSquare(x: int, y: int)
add(chess: Chess): boolean
DenSquare
- TEAM: Player
DenSquare(x: int, y: int, team: Player)
+ getTeam(): Player
add(chess Chess): boolean
TrapSquare
TrapSquare(x: int, y: int)
add(chess: Chess): boolean
Chess
- NAME: String
- RANKING: int
- curRank: int
- pos: BoardSquare
- team: Player
Chess(name: String, ranking: int)
+ getName(): String
+ getRanking(): int
+ getCurRank(): int
+ setCurRank(curRank: int)
+ getPos(): BoardSquare
+ setPos(pos: BoardSquare)
+ getTeam(): Player
+ setTeam(team: Player)
move(pos: String): boolean
eat(chess: Chess): boolean
changeChessNumber(chess: Chess)
Player
- name: String
- numberofChessSurvive: int = 8
- chessOwn: Chess[]
+ getName(): String
+ setName(name: String)
+ getNumberOfChessSurvive(): int
+ setNumberOfChessSurvive(numberOfChessSurvive: int)
+ getChessOwn(): Chess[]
+ setChessOwn(chessOwn: Chess[])
+ reduceNumberOfChessSurvive()
Rat
Rat()
+ eat(chess: Chess): boolean
Cat
Cat()
Dog
Dog()
Elephant
Elephant()
+ eat(chess: Chess): boolean
Wolf
Wolf()
Tiger
Tiger()
move(pos: String): boolean
Leopard
Leopard()
Lion
Lion()

2.3 Quick Start Guide

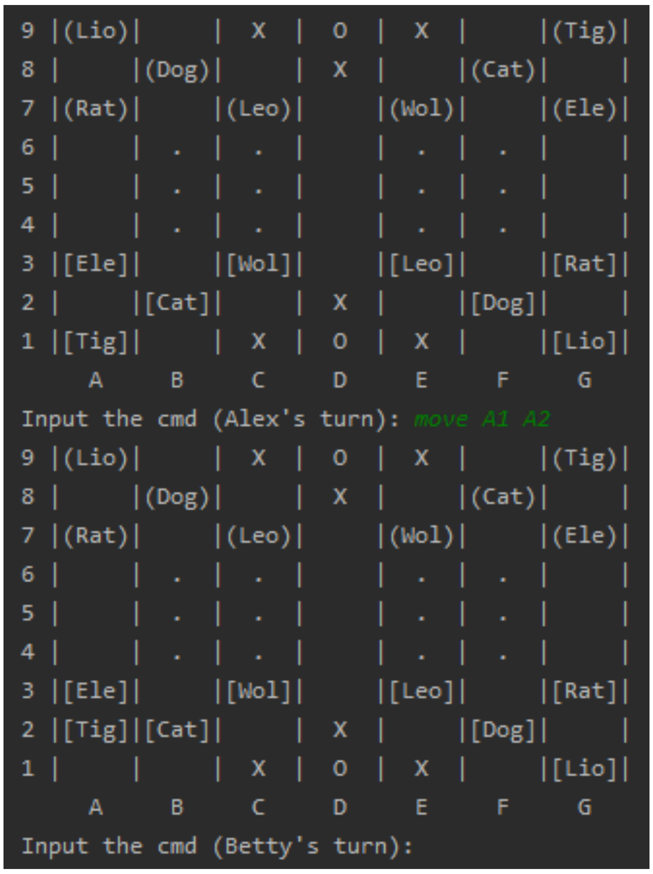
Quick Start Guide shows the scenarios that may occur during the game.

User is asked to start a new game or open an existing one. Player X's pieces are enclosed in [] and Player Y's pieces are enclosed in (). User choses to start a new game as follows:



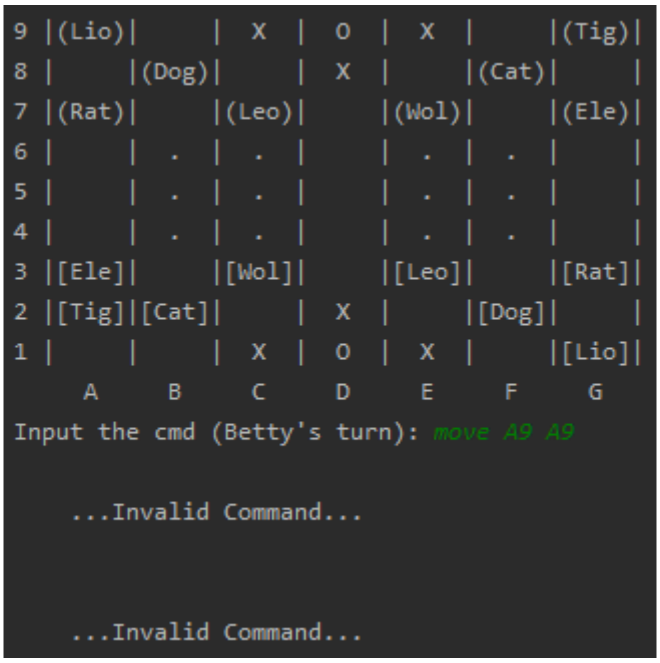
```
Welcome Players!
    Type 'start' to start a new game
    Type 'open [filePath]' to open a saved game
start
Please Enter the Player's name of Team X: Alex
Please Enter the Player's name of Team Y: Betty
9 |(Lio)|     |  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|     |  X  |     |(Cat)|     |
7 |(Rat)|     |(Leo)|     |(Wol)|     |(Ele)|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |     |  .  |  .  |     |  .  |  .  |     |
3 |[Ele]|     |[Wol]|     |[Leo]|     |[Rat]|
2 |     |[Cat]|     |  X  |     |[Dog]|     |
1 |[Tig]|     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
```

Player makes a valid move as follows:



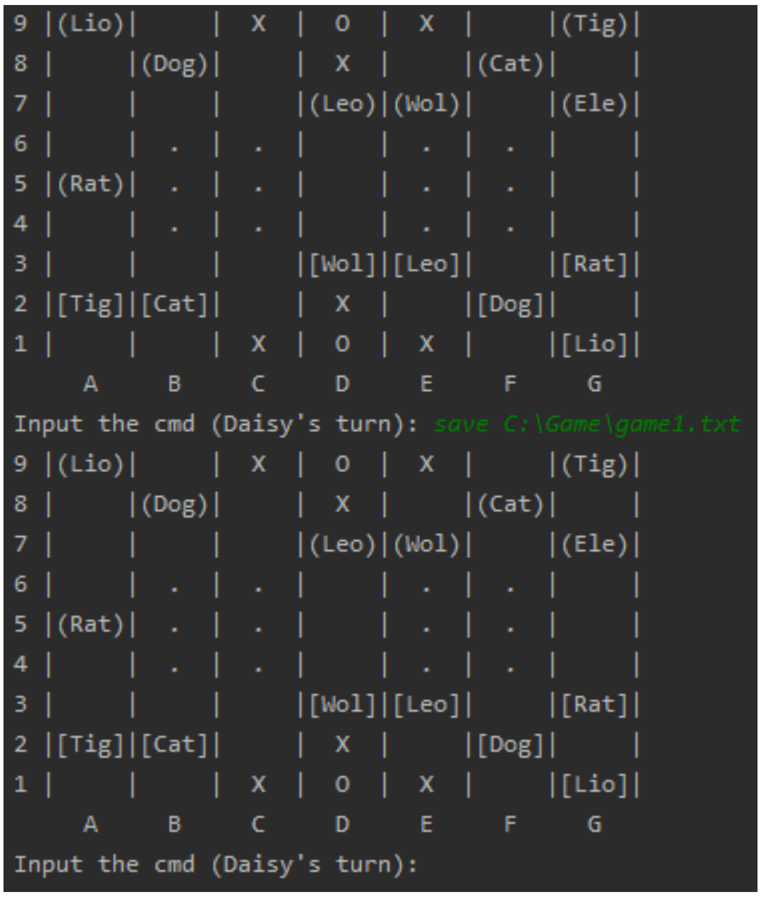
```
9 |(Lio)|     |  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|     |  X  |     |(Cat)|     |
7 |(Rat)|     |(Leo)|     |(Wol)|     |(Ele)|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |     |  .  |  .  |     |  .  |  .  |     |
3 |[Ele]|     |[Wol]|     |[Leo]|     |[Rat]|
2 |     |[Cat]|     |  X  |     |[Dog]|     |
1 |[Tig]|     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Alex's turn): move A1 A2
9 |(Lio)|     |  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|     |  X  |     |(Cat)|     |
7 |(Rat)|     |(Leo)|     |(Wol)|     |(Ele)|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |     |  .  |  .  |     |  .  |  .  |     |
3 |[Ele]|     |[Wol]|     |[Leo]|     |[Rat]|
2 |[Tig]|[Cat]|     |  X  |     |[Dog]|     |
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Betty's turn):
```

Player makes an invalid move (move to the same point):



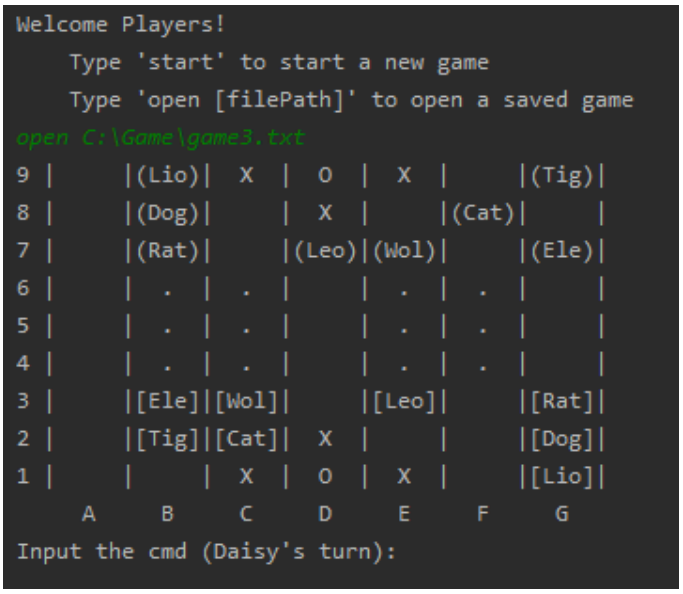
```
9 |(Lio)|     |  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|     |  X  |     |(Cat)|     |
7 |(Rat)|     |(Leo)|     |(Wol)|     |(Ele)|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |     |  .  |  .  |     |  .  |  .  |     |
3 |[Ele]|     |[Wol]|     |[Leo]|     |[Rat]|
2 |[Tig]|[Cat]|     |  X  |     |[Dog]|     |
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Betty's turn): move A9 A9

    ...Invalid Command...


    ...Invalid Command...
```

Player enters a valid save command:



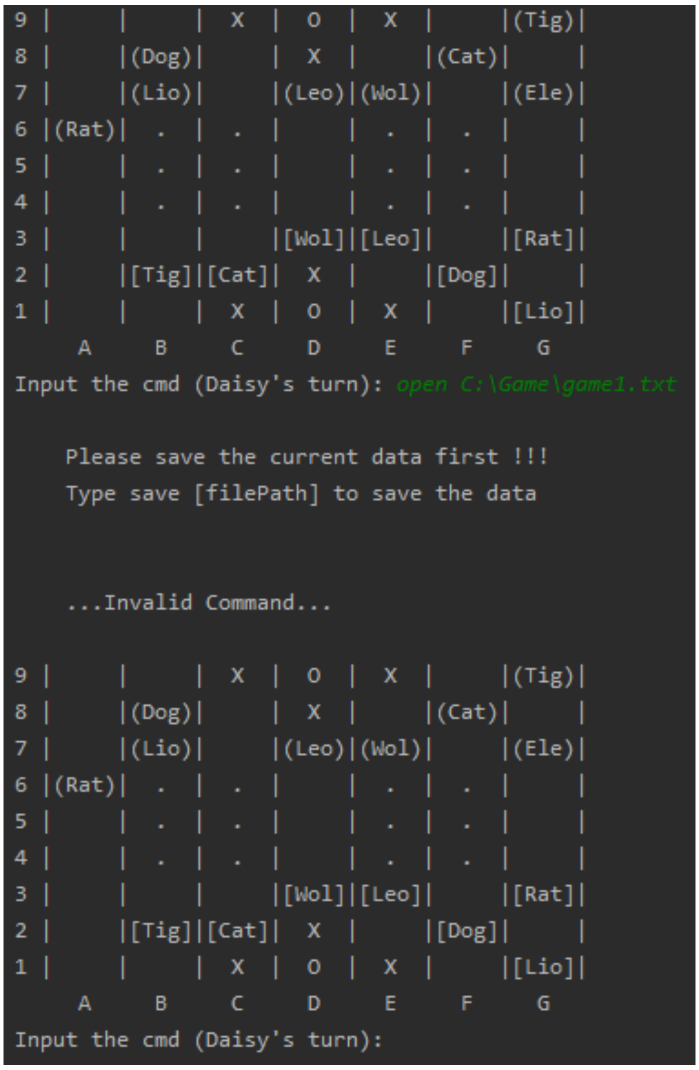
```
9 |(Lio)|     |  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|     |  X  |     |(Cat)|     |
7 |     |     |     |(Leo)|(Wol)|     |(Ele)|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |(Rat)|  .  |  .  |     |  .  |  .  |     |
4 |     |  .  |  .  |     |  .  |  .  |     |
3 |     |     |     |[Wol]|[Leo]|     |[Rat]|
2 |[Tig]|[Cat]|     |  X  |     |[Dog]|     |
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Daisy's turn): save C:\Game\game1.txt
9 |(Lio)|     |  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|     |  X  |     |(Cat)|     |
7 |     |     |     |(Leo)|(Wol)|     |(Ele)|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |(Rat)|  .  |  .  |     |  .  |  .  |     |
4 |     |  .  |  .  |     |  .  |  .  |     |
3 |     |     |     |[Wol]|[Leo]|     |[Rat]|
2 |[Tig]|[Cat]|     |  X  |     |[Dog]|     |
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Daisy's turn):
```

Player enters a valid open command:



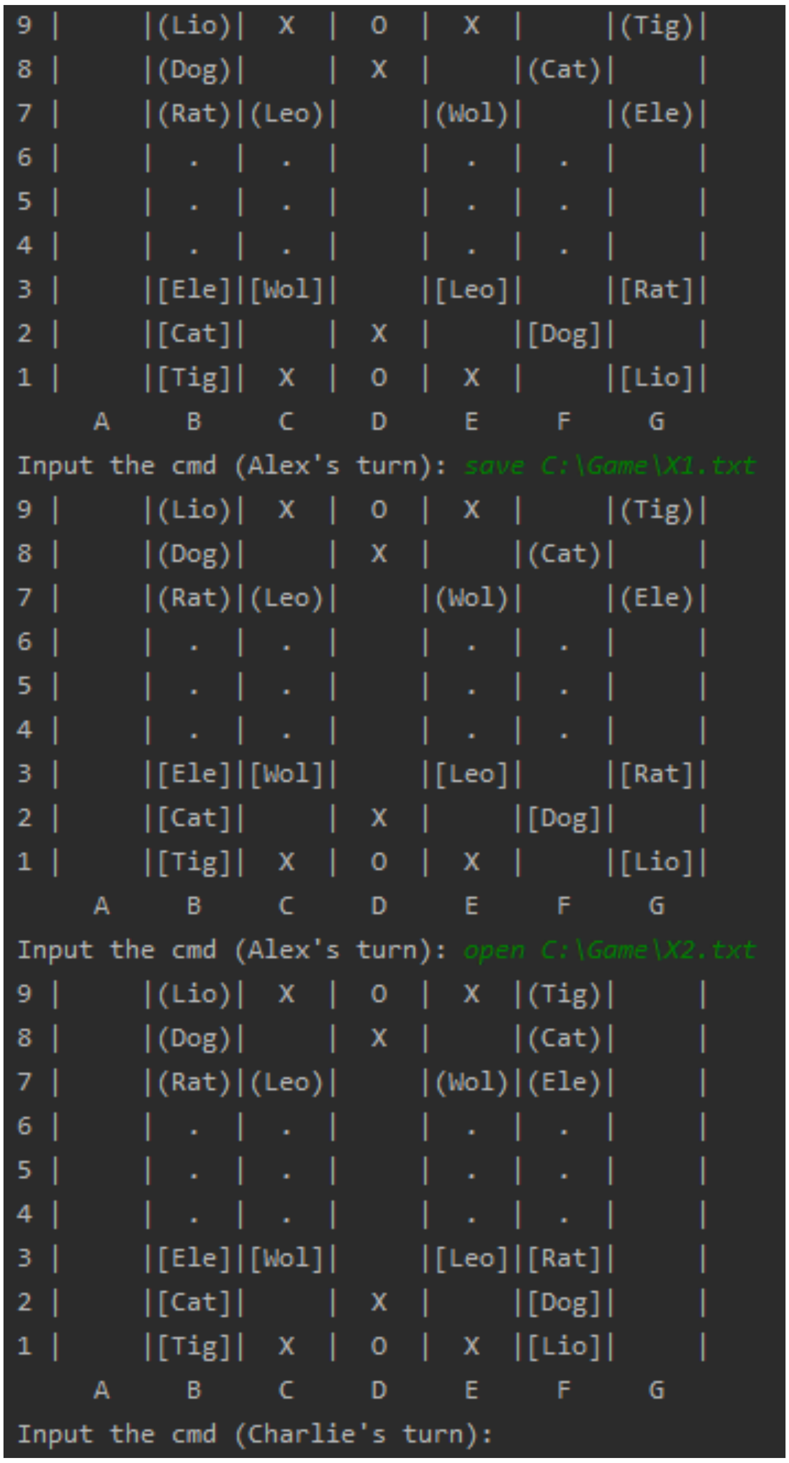
```
Welcome Players!
    Type 'start' to start a new game
    Type 'open [filePath]' to open a saved game
open C:\Game\game3.txt
9 |     |(Lio)|  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|     |  X  |     |(Cat)|     |
7 |     |(Rat)|     |(Leo)|(Wol)|     |(Ele)|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |     |  .  |  .  |     |  .  |  .  |     |
3 |     |[Ele]|[Wol]|     |[Leo]|     |[Rat]|
2 |     |[Tig]|[Cat]|  X  |     |     |[Dog]|
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Daisy's turn):
```

Player asks to open a saved game but is prompted to save the existing one first.



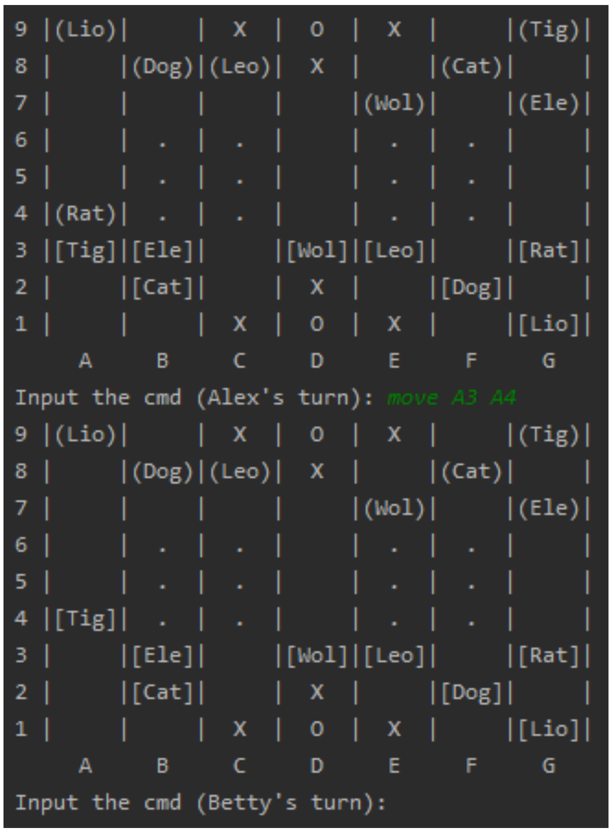
```
9 |     |     |  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|     |  X  |     |(Cat)|     |
7 |     |(Lio)|     |(Leo)|(Wol)|     |(Ele)|
6 |(Rat)|  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |     |  .  |  .  |     |  .  |  .  |     |
3 |     |     |     |[Wol]|[Leo]|     |[Rat]|
2 |     |[Tig]|[Cat]|  X  |     |[Dog]|     |
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Daisy's turn): open C:\Game\game1.txt

    Please save the current data first !!!
    Type save [filePath] to save the data


    ...Invalid Command...

9 |     |     |  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|     |  X  |     |(Cat)|     |
7 |     |(Lio)|     |(Leo)|(Wol)|     |(Ele)|
6 |(Rat)|  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |     |  .  |  .  |     |  .  |  .  |     |
3 |     |     |     |[Wol]|[Leo]|     |[Rat]|
2 |     |[Tig]|[Cat]|  X  |     |[Dog]|     |
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Daisy's turn):
```

Play successfully saves the current game and then loads another existing game.



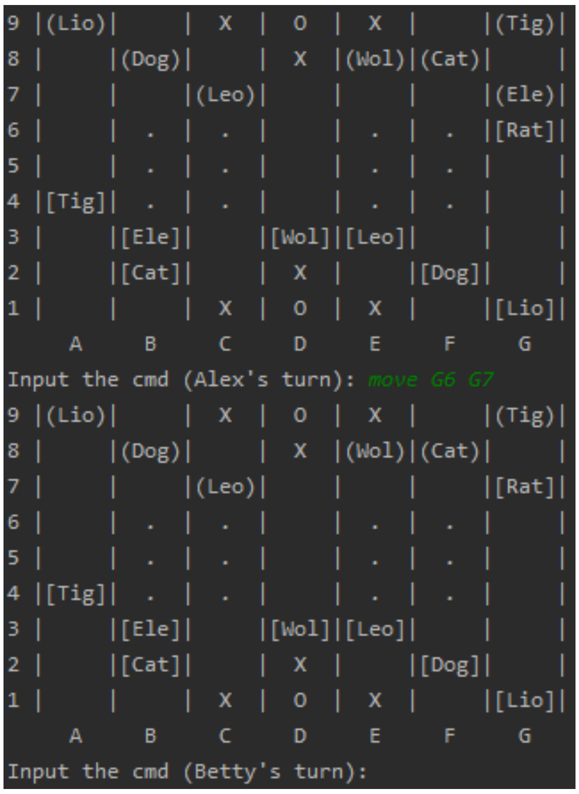
```
9 |     |(Lio)|  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|     |  X  |     |(Cat)|     |
7 |     |(Rat)|(Leo)|     |(Wol)|     |(Ele)|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |     |  .  |  .  |     |  .  |  .  |     |
3 |     |[Ele]|[Wol]|     |[Leo]|     |[Rat]|
2 |     |[Cat]|     |  X  |     |[Dog]|     |
1 |     |[Tig]|  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Alex's turn): save C:\Game\X1.txt
9 |     |(Lio)|  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|     |  X  |     |(Cat)|     |
7 |     |(Rat)|(Leo)|     |(Wol)|     |(Ele)|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |     |  .  |  .  |     |  .  |  .  |     |
3 |     |[Ele]|[Wol]|     |[Leo]|     |[Rat]|
2 |     |[Cat]|     |  X  |     |[Dog]|     |
1 |     |[Tig]|  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Alex's turn): open C:\Game\X2.txt
9 |     |(Lio)|  X  |  O  |  X  |(Tig)|     |
8 |     |(Dog)|     |  X  |     |(Cat)|     |
7 |     |(Rat)|(Leo)|     |(Wol)|(Ele)|     |
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |     |  .  |  .  |     |  .  |  .  |     |
3 |     |[Ele]|[Wol]|     |[Leo]|[Rat]|     |
2 |     |[Cat]|     |  X  |     |[Dog]|     |
1 |     |[Tig]|  X  |  O  |  X  |[Lio]|     |
     A     B     C     D     E     F     G
Input the cmd (Charlie's turn):
```

Player enters a valid command that results in eating (Tiger eats Rat):



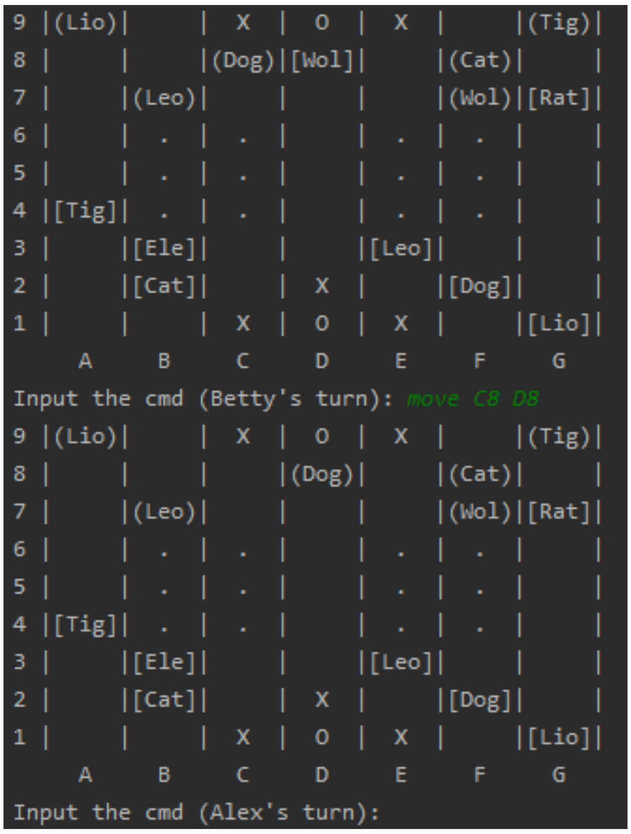
```
9 |(Lio)|     |  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|(Leo)|  X  |     |(Cat)|     |
7 |     |     |     |     |(Wol)|     |(Ele)|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |(Rat)|  .  |  .  |     |  .  |  .  |     |
3 |[Tig]|[Ele]|     |[Wol]|[Leo]|     |[Rat]|
2 |     |[Cat]|     |  X  |     |[Dog]|     |
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Alex's turn): move A3 A4
9 |(Lio)|     |  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|(Leo)|  X  |     |(Cat)|     |
7 |     |     |     |     |(Wol)|     |(Ele)|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |[Tig]|  .  |  .  |     |  .  |  .  |     |
3 |     |[Ele]|     |[Wol]|[Leo]|     |[Rat]|
2 |     |[Cat]|     |  X  |     |[Dog]|     |
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Betty's turn):
```

Player enters a valid command that results in Rat eating the Elephant:



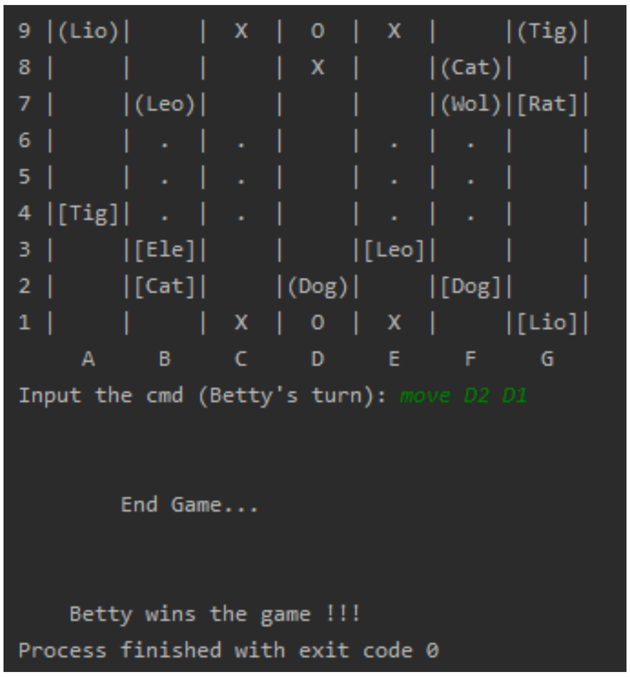
```
9 |(Lio)|     |  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|     |  X  |(Wol)|(Cat)|     |
7 |     |     |(Leo)|     |     |     |(Ele)|
6 |     |  .  |  .  |     |  .  |  .  |[Rat]|
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |[Tig]|  .  |  .  |     |  .  |  .  |     |
3 |     |[Ele]|     |[Wol]|[Leo]|     |     |
2 |     |[Cat]|     |  X  |     |[Dog]|     |
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Alex's turn): move G6 G7
9 |(Lio)|     |  X  |  O  |  X  |     |(Tig)|
8 |     |(Dog)|     |  X  |(Wol)|(Cat)|     |
7 |     |     |(Leo)|     |     |     |[Rat]|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |[Tig]|  .  |  .  |     |  .  |  .  |     |
3 |     |[Ele]|     |[Wol]|[Leo]|     |     |
2 |     |[Cat]|     |  X  |     |[Dog]|     |
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Betty's turn):
```

Player enters a valid command that results in Dog eating the Wolf despite Wolf’s higher ranking since it is in opponent’s trap:



```
9 |(Lio)|     |  X  |  O  |  X  |     |(Tig)|
8 |     |     |(Dog)|[Wol]|     |(Cat)|     |
7 |     |(Leo)|     |     |     |(Wol)|[Rat]|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |[Tig]|  .  |  .  |     |  .  |  .  |     |
3 |     |[Ele]|     |     |[Leo]|     |     |
2 |     |[Cat]|     |  X  |     |[Dog]|     |
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Betty's turn): move C8 D8
9 |(Lio)|     |  X  |  O  |  X  |     |(Tig)|
8 |     |     |     |(Dog)|     |(Cat)|     |
7 |     |(Leo)|     |     |     |(Wol)|[Rat]|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |[Tig]|  .  |  .  |     |  .  |  .  |     |
3 |     |[Ele]|     |     |[Leo]|     |     |
2 |     |[Cat]|     |  X  |     |[Dog]|     |
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Alex's turn):
```

Player wins by moving a piece onto the den on the opponent’s side:



```
9 |(Lio)|     |  X  |  O  |  X  |     |(Tig)|
8 |     |     |     |  X  |     |(Cat)|     |
7 |     |(Leo)|     |     |     |(Wol)|[Rat]|
6 |     |  .  |  .  |     |  .  |  .  |     |
5 |     |  .  |  .  |     |  .  |  .  |     |
4 |[Tig]|  .  |  .  |     |  .  |  .  |     |
3 |     |[Ele]|     |     |[Leo]|     |     |
2 |     |[Cat]|     |(Dog)|     |[Dog]|     |
1 |     |     |  X  |  O  |  X  |     |[Lio]|
     A     B     C     D     E     F     G
Input the cmd (Betty's turn): move D2 D1


        End Game...


    Betty wins the game !!!
Process finished with exit code 0
```

The game ends here and the winner’s name is also printed with a victory statement.