Meryem Ülkü Kara
21727355

REPORT

In this assignment we tried to make expected to implement an application which is the logic core simulator of a simple board game. Problem was designing a program to make some operations on Players such as getHP,getAP,getColumn, getRow etc. My approach is; Firstly I created two arraylist. First is for players, second is for output files. Secondly I readed input files and put the informations arraylists. Thirdly I read input files then made operations. Here I created a string in Players class to keep all informations of players line by line. Afterwards, I changed the health points by using the attack points of the characters with the orders we read from commands.txt. Then I checked if they died by looking at their new health points. After each commands.txt line, I checked the exceptions and added the exceptions to the output arraylist, if there are no exceptions, the status and location of the characters, if there are exceptions. Finally, I print the information in the output arraylist to output.txt

Polymorphsim

Actually, all characters have HP, AP, maxMove values, but these values are different for all of them. These are the main data of the game. That's why I created the methods of these values in the players class. Subclass of the Players Class Constants Class. Other classes are subclass of the Players Class.So I override methods based on values in constants class.And I override toString method.

Example: These are overriden methods of elf class.

```
public int getHP() {return Constants.elfHP;}
public int getAP() {return Constants.elfAP;}
public int maxMove() {return Constants.elfMaxMove;}
```

Main Class Functions

`public static String[] readFile(String path):` This reads the file and returns a string array .Elements of the string array is the lines of the file.

`public static void read_player(int board,String line,ArrayList <Players>players):` This function control NotBoardIndexException and calls add_player for adds the players to the players arraylist.

`public static void add_player(int board,String class_name,String id,int column,int row,ArrayList <Players>players ):`

This function check which character the player is and choose accordingly and adds the players to the players arraylist.

`public static void read_command(int board,String line,ArrayList <Players>players, ArrayList <String> output ):` This function control MoveCountCheckExceptionand calls movement for check movement and change players information. Then calls the printOut function, adds the return value of the printOut function to the output arraylist.

`public static int movement(String id, String [] move,ArrayList <Players>players ):`With this function, it controls the steps of the characters and their relations with each other, deletes the character where necessary.

`public static String printOut(int board,ArrayList <Players> players):`
Returns the table of values and healthPoints found in Players arraylist.

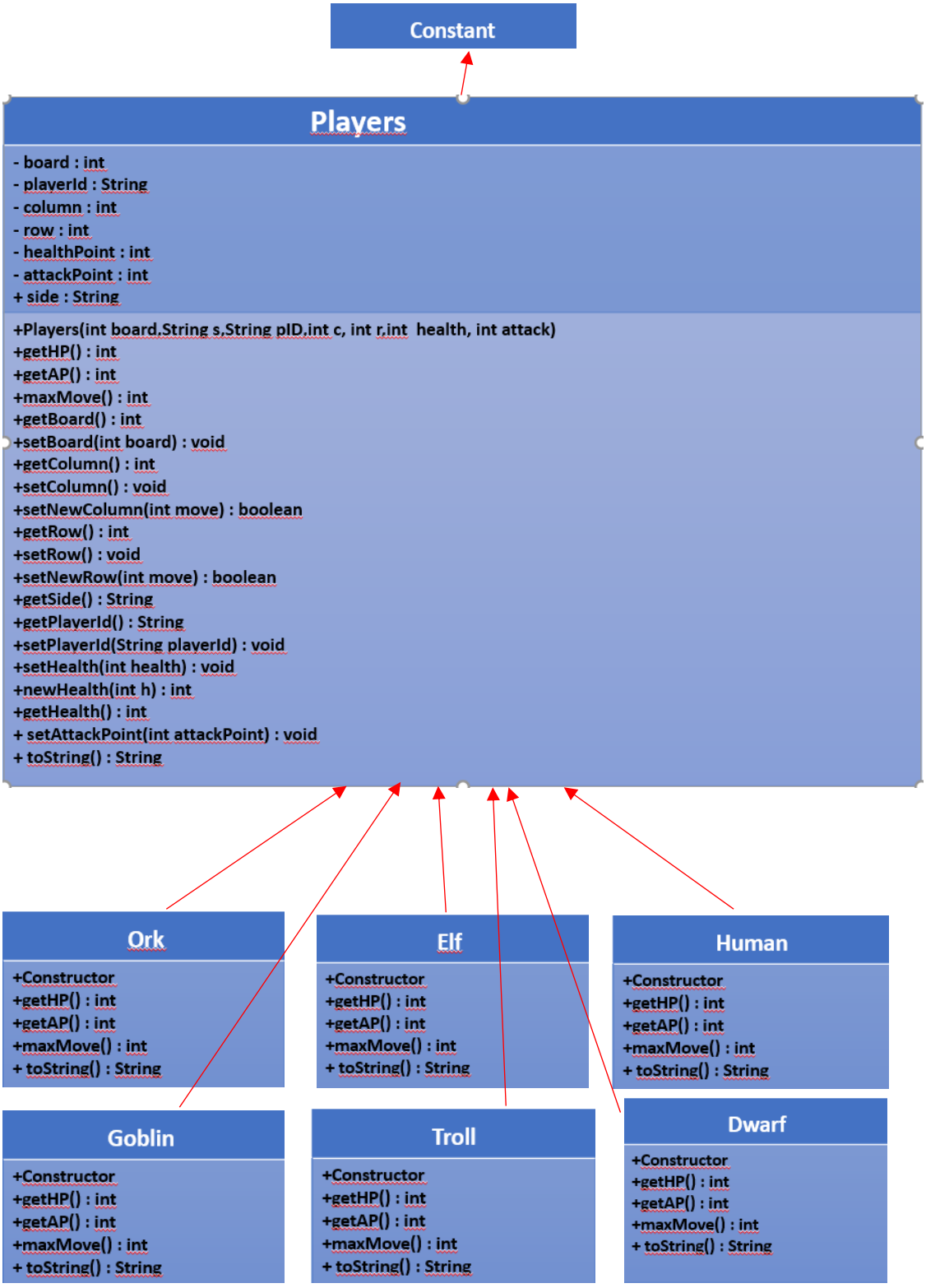
`public static boolean game_finishes_control(ArrayList<Players> players):`
This function checks all players if they are on the same side

`public static void output_file(String file_name,String winner_team, ArrayList <String> output):` This function is called when all characters are on the same side and prints the output to output.txt

Apart from these, I have three more classes:

`NotBoardIndexException` and `MoveCountCheckException` are error Classes.

`PlayerIDSorter implements Comparator<Players> Class:` Comparator to sort the players list by playersID alphabetically.


Constant
Players
- board : int
- playerId : String
- column : int
- row : int
- healthPoint : int
- attackPoint : int
+ side : String
+Players(int board,String s,String pID,int c, int r,int health, int attack)
+getHP() : int
+getAP() : int
+maxMove() : int
+getBoard() : int
+setBoard(int board) : void
+getColumn() : int
+setColumn() : void
+setNewColumn(int move) : boolean
+getRow() : int
+setRow() : void
+setNewRow(int move) : boolean
+getSide() : String
+getPlayerId() : String
+setPlayerId(String playerId) : void
+setHealth(int health) : void
+newHealth(int h) : int
+getHealth() : int
+ setAttackPoint(int attackPoint) : void
+ toString() : String
Ork
+Constructor
+getHP() : int
+getAP() : int
+maxMove() : int
+ toString() : String
Elf
+Constructor
+getHP() : int
+getAP() : int
+maxMove() : int
+ toString() : String
Human
+Constructor
+getHP() : int
+getAP() : int
+maxMove() : int
+ toString() : String
Goblin
+Constructor
+getHP() : int
+getAP() : int
+maxMove() : int
+ toString() : String
Troll
+Constructor
+getHP() : int
+getAP() : int
+maxMove() : int
+ toString() : String
Dwarf
+Constructor
+getHP() : int
+getAP() : int
+maxMove() : int
+ toString() : String

UML DIAGRAM INFORMATION

1- Player is superclass.
2- Ork, Elf, Human, Goblin, Troll and Dwarf are subclasses.
3- Side: zorde or calliance
4- board: the number of rows of the board
5- setNewRow and setNewColumn control MoveCountCheckException.