CALLIANCE VS ZORDE

THE CONSOLE BOARD GAME

YASİN ŞİMŞEK | 21827847

INTRODUCTION

What is the Calliance - Zorde?

This is a console game. There are two teams that are called Calliance and Zorde. Each teams can include three different characters:

- Calliance Team: Dwarf, Elf, Human
- Zorde Team: Goblin, Ork, Troll

Each characters in game have a default hit point value, attack power value and max move value (For example, Elfs can pass 4 cell in one move , but Orks can move only one cell). Some of them can attack after every step (Dwarf, Elf, Goblin), others attack only after final step. In addition, Ork character has a healing talent, that means Orks can heal the neighboring friends after final step.

The main goal of game is to apply an effective strategy and kill all enemies. Each characters must move in limited place and must move the default max move value times.

SOLUTION

The main goal of the project is to develop this game in Java programming language using object oriented programming techniques especially polymorphism and exceptions. I think, the most important points in the project are to use polymorphism with overriding methods and to develop and design new classes that are designed for this game benefit from collections and generics. Instead of using lists, maps etc in all classes, I designed new classes called Board and Data and used only these in each operations. Previous assignment, I didn't do like that and it forced me too. It provided to write more effective, readable and changeable codes.

I planned to connect all characters to one superclass and add some methods to this superclass. Under this superclass, I added two classes which take their name from team name. And finally, I created all characters by connecting these characters to superclasses.

The program includes 17 classes (one of them is Main class and two of them are Exception class).

1. **Main:** Takes program files as an argument and sends all files – data's to **Method_Execute - Method_Initialize** classes, and creates new Board and Data class.

2. **Constants:** This class includes all default values of characters like hit point, attack power etc**.** The character classes use these default values.

3. **Method_Initialize:** Takes **Board** and **Data** class and reads **initials** file. After the reading, records all characters into system. Includes only a static method.

4. **Method_Execute:** Takes **output** file, **Board** and **Data** class and reads **commands** file. After the reading, directs program to **Characters** class with needed data's to start attack system. Includes only a static method.

5. **BoundaryException** : This error is thrown if any character tries to move out of the board.

6. **MoveCountException :** This error is thrown if any character tries to the wrong number of times.

7. **Board:** The class represents the board of game. Works with an 2D array. Includes methods to record characters, creating board, print current situation etc.

8. **Data:** This class is used to keep character data with location and name. Uses **HashMap** and **ArrayList** in the backend. Includes put, delete and get methods like a map.

9. **Characters:** This class is one of the most important classes together **Board** and **Data.** Takes all details of characters with constructor, and all moving is managed in the **Characters** class. Includes many method to apply right moving like attack, final attack, damage, checking methods etc.

10. **Calliance_Characters:** The subclass of **Characters.** This class does not includes any method or data. Used to only indicate teams.

11. **Dwarf:** Created for Dwarfs. Does not include any method or data.

12. **Elf:** Created for Elfs. Specially only keep final range attack data and includes final attack method. The final attack method is overriden from superclass (**Characters**). Because Elfs has a range attack talent different from others.

13. **Human:** Created for Humans. Does not include any method or data.

14. **Zorde_Characters: Like Calliance_Characters,** used to only indicate teams. Does not include any special things.

15. **Goblin:** Created for Goblins. Does not include any method or data.

16. **Ork:** Created for Orks. Specially only keep heal point value and includes final attack and heal methods. The final attack method is overriden from superclass (**Character**), because Orks has heal talent, so the final attack must be managed in **Ork** class.

17. **Troll:** Created for Trolls. Does not include any method or data.

All methods, values, classes are given in the **"all_classes.jpg"** file. You can see more deeply how to program works.

COMMENTS

During the project, I got a lot of errors. Because I must check many things to run the program rightly. So, I needed to do new things to solve these problems. As I said before, Board and Data classes make easier everything. I gained an experience developing special classes/methods for different problems, I can say it is my biggest earning although the main goal of this assignment is using polymorphism.

REFERENCES

I used these sources to develope this game:

- Intruduction to Java Programming and Data Structures, 11th Edition, Daniel Liang
- Stack Overflow - https://stackoverflow.com/
- Geeks for Geeks - https://www.geeksforgeeks.org/
- Java Tutorial - https://docs.oracle.com/javase/tutorial/
- Intellij Idea (for creating UML diagram)
- BBM102 Lecture Notes