

## Programming Assignment 3

**Submission Date :** 16.04.2021

**Due Date :** 07.05.2021 (23:59)

**Programming Languages:** Java

**Subject :** Polymorphism, Exceptions

### 1 Introduction

In this experiment, you are expected to implement an application which is the logic core simulator of a simple board game. Main focus point of this experiment is to get you familiar with polymorphism. Implementing the program without using the benefits of polymorphism will be penalized.

### 2 Problem

There are two sides which are called Zorde and Calliance. Main theme of the game is the war between two sides. Each side has 3 different types of characters. The game is played on a board where each character occupies a cell on the board (just like chess). During the progress of the game, each side makes moves in turns. The aim of the game is to kill every opponent character.

#### 2.1 Hit Point

Each character has a hit point (HP) value which indicates how healthy they are. This value is lowered equally to the damage taken when a character is being attacked by another character. If a character's HP value is 0 or below, they are considered dead.

#### 2.2 Attack Power

Attack power (AP) of a character indicates how hard it can attack its opponents. This value determines how much damage will be taken by the defending character. By default, damage done by an attacking character is equal to its AP so when it attacks another character, this value will be subtracted from the defender's HP.

The table below shows cell types of each side and their default (initial) HP and AP values.

	Zorde Characters			Calliance Characters		
	Ork	Troll	Goblin	Human	Elf	Dwarf
HP	200	150	80	100	70	120
AP	30	20	10	30	15	20

## 2.3 Board and Initial Armies

The game takes place on a board which is very similar to a chess board. Size of the board and initial armies of both sides (including their initial positions) will be provided via an input file, the initials file.

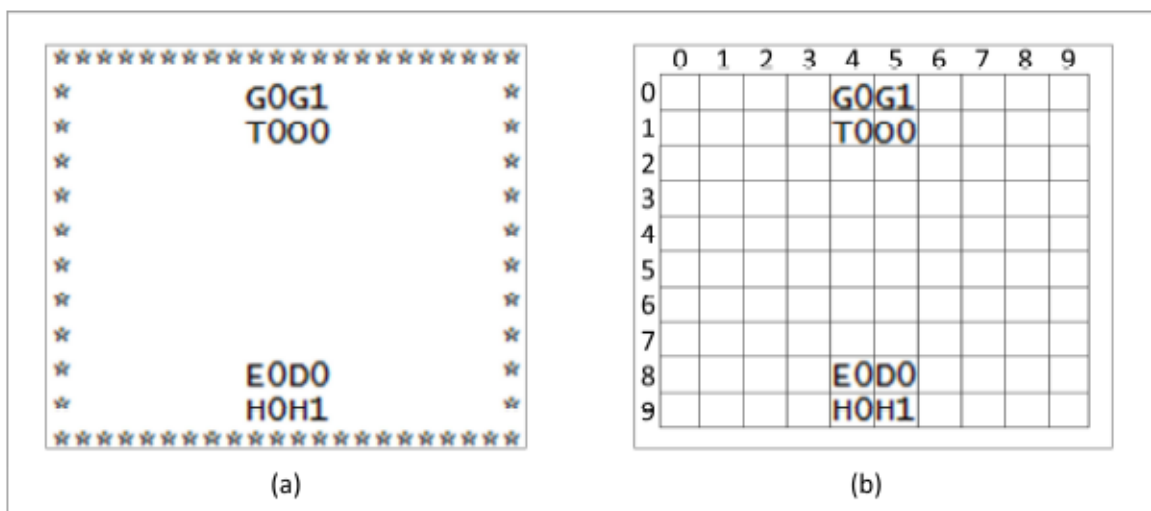


```

1 BOARD
2 10x10
3
4 CALLIANCE
5 ELF · E0 · 4 · 8
6 DWARF · D0 · 5 · 8
7 HUMAN · H0 · 4 · 9
8 HUMAN · H1 · 5 · 9
9
10 ZORDE
11 GOBLIN · G0 · 4 · 0
12 GOBLIN · G1 · 5 · 0
13 TROLL · T0 · 4 · 1
14 ORK · O0 · 5 · 1

```

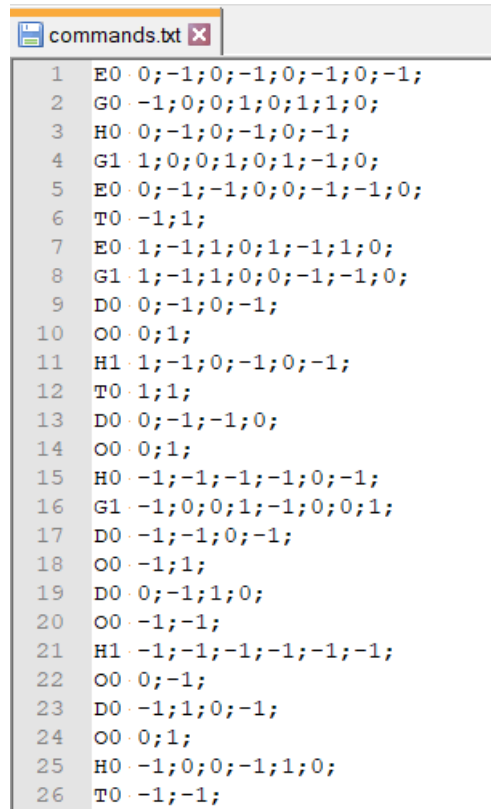
During the game, you will create an output showing the current status of the board. In the output, the board should be formatted exactly as shown in the figure below (a). Note that there is one line for each row (horizontally) and two character spaces for each column (vertically) (b).



## 2.4 Commands

The program you will implement is a turn based strategy game, therefore, the players will make their moves in turns as the game continues. The moves made by each player is supplied

via an input file, the *commands file*. Each line of the commands file indicates a single move made by a player.



```
1 E0 0;-1;0;-1;0;-1;0;-1;
2 G0 -1;0;0;1;0;1;1;0;
3 H0 0;-1;0;-1;0;-1;
4 G1 1;0;0;1;0;1;-1;0;
5 E0 0;-1;-1;0;0;-1;-1;0;
6 T0 -1;1;
7 E0 1;-1;1;0;1;-1;1;0;
8 G1 1;-1;1;0;0;-1;-1;0;
9 D0 0;-1;0;-1;
10 O0 0;1;
11 H1 1;-1;0;-1;0;-1;
12 T0 1;1;
13 D0 0;-1;-1;0;
14 O0 0;1;
15 H0 -1;-1;-1;-1;0;-1;
16 G1 -1;0;0;1;-1;0;0;1;
17 D0 -1;-1;0;-1;
18 O0 -1;1;
19 D0 0;-1;1;0;
20 O0 -1;-1;
21 H1 -1;-1;-1;-1;-1;-1;
22 O0 0;-1;
23 D0 -1;1;0;-1;
24 O0 0;1;
25 H0 -1;0;0;-1;1;0;
26 T0 -1;-1;
```

As you can see in the figure, a move command starts with the ID of the character to be moved, which is followed by a sequence of integers that are inside the interval  $[-1, 1]$  (i.e. the sequence may only contain -1, 0 and 1) and these integer numbers are separated by semicolons. This sequence consists of even number of integer as each couple indicates one step of the move. For example, at the first line of the figure the integer sequence indicates a 4 step move as illustrated in the figure. If we look at the position of E0, its current position is at the position of “4 8”, and if we make a move step with 0;-1, then its position will shift to the “4 7” on the board.

After reading the initials file and locating the characters on the game board, your program should start reading the commands file line by line and execute the moves of the players. The program should exit when it reaches the end of the commands file.

## 2.5 Output

After execution of each line of the commands file, your program should create and output which shows the current status of the board (i.e. positions of the characters) and their current and default HP values as shown in the figure. Please note that the list below the board is ordered by the character IDs. You can use the 'tab' character (i.e. `\t`) between line tokens so that the output will look aligned vertically. Please do not forget to create an initial output right after reading from the initials file and before starting to execute the commands.

```
output.txt
1 *****
2 * .....G0G1 .....*
3 * .....T0O0 .....*
4 * .....*
5 * .....*
6 * .....*
7 * .....*
8 * .....*
9 * .....*
10 * .....E0D0 .....*
11 * .....H0H1 .....*
12 *****
13
14 D0→120→(120)
15 E0→70→(70)
16 G0→80→(80)
17 G1→80→(80)
18 H0→100→(100)
19 H1→100→(100)
20 O0→200→(200)
21 T0→150→(150)
22
23 *****
24 * .....G0G1 .....*
25 * .....T0O0 .....*
26 * .....*
27 * .....*
28 * .....E0 .....*
29 * .....*
30 * .....*
31 * .....*
32 * .....D0 .....*
33 * .....H0H1 .....*
34 *****
35
36 D0→120→(120)
37 E0→70→(70)
38 G0→80→(80)
39 G1→80→(80)
40 H0→100→(100)
41 H1→100→(100)
42 O0→200→(200)
43 T0→150→(150)
```

*P.S. Representation of sample I/O is given in Appendix (Section 9).*

### 3 Details about Moves of the Characters and Attacking

At the tables below, you can find the move types of each character type. Move types also determine when a character attacks. When a character makes an attack, it attacks to all of the 8 cells around its current position. While some characters make an attack after each step of their move (i.e. if the character makes 4 steps, it attacks 4 times, after each step), some characters can only make an attack after their final step (i.e. if the character makes 4 steps,

<b>ZORDE</b>			
	<b>Ork</b>	<b>Troll</b>	<b>Goblin</b>
<b>Move steps</b>	1	1	4
<b>Attack type</b>	Attacks after the final step.	Attacks after the final step.	Attacks at every step.
<b>Special</b>	Heals friendly neighbors (10) before every move sequence (not move step).	N/A	N/A

<b>CALLIANCE</b>			
	<b>Human</b>	<b>Elf</b>	<b>Dwarf</b>
<b>Move steps</b>	3	4	2
<b>Attack type</b>	Attacks after the final step.	Attacks at every step.	Attacks at every step.
<b>Special</b>	N/A	Makes a ranged attack (instead of the final attack) after every move sequence (not move step). AP:15, range:2	N/A

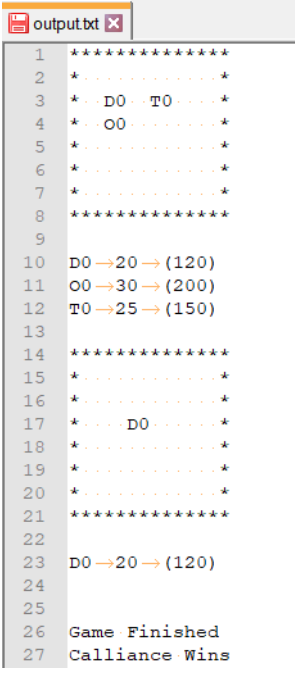
it attacks only once after the final step).

As explained above (see the attack power section), attacking means, decreasing an defender's HP value by the amount of the attacker's AP value.

### 3.1 COMBAT RULES

1. Characters cannot move onto the friendly characters (friendly character means characters that are at the same side). If a character tries to move onto a friendly character, its move is finalized at the current location and the rest of its move sequence will be ignored. For example if a character tries to move onto a friendly character at the second step of a 4-step move, only the first step of the move will be executed.
2. Characters can move onto the enemy characters. If such a situation occurs, two enemy characters that are at the same cell fight to the death for the cell. Before a fight to the death, the attacking character will make a single attack to the defending character with its attack power. A fight to the death can be won by the character that has the highest current HP value but its HP value will decrease by an amount equal to the losing characters current HP value. For example, if a human (current HP : 100) moves onto a cell that is currently occupied by an ork (current HP : 60), the human will make an initial attack to the ork (now, the ork's current HP is 30). Then, the ork will die and human's HP value will decrease to 70.
  - (a) The attack that is made before the fight to the death is not a normal attack. Thus, it will only effect the defending enemy character, not the other characters at the neighboring 8 cells.
  - (b) After a fight to the death, the attacker will stop moving. For example if a character tries to move onto an enemy character at the second step of a 4-step move, even if the attacker is the winner of the fight to the death, only the first and second steps of the move will be executed. The rest of the move sequence will be ignored.
  - (c) If two characters have same HP just before a fight to the death, they both die.

3. Before they start a move sequence, Orks will first heal any friendly character in one of the neighboring cells.
  - Friendly characters will be healed by 10 hit points.
  - A character's hit point cannot exceed its default hit point via healing. If this happens, you should decrease the current hit point to the default hit point of that character.
  - When healing the neighboring characters, Orks will heal themselves too.
4. After they finish a move sequence, Elfs will make a ranged attack (instead of the final step's default attack) to all enemy characters that are in range of 2 cells with an attack power of 15.
  - If an Elf's move sequence is interrupted by moving through an enemy character, then this power cannot be used. This power can only be used if the Elf completes its whole move sequence without being interrupted.
5. Game ends when all characters of one side is dead. When the game ends, you should add a line indicating the winner of the game as shown in the figure.



```
1 *****
2 *.....*
3 *...D0...T0...*
4 *...O0...*
5 *.....*
6 *.....*
7 *.....*
8 *****
9
10 D0→20→(120)
11 O0→30→(200)
12 T0→25→(150)
13
14 *****
15 *.....*
16 *.....*
17 *...D0...*
18 *.....*
19 *.....*
20 *.....*
21 *****
22
23 D0→20→(120)
24
25
26 Game Finished
27 Calliance Wins
```

## 4 Error Handling

You will need to check for only 2 errors by creating custom exceptions. All the input files will be error free apart from these controls :

1. Boundary Check: You need to check if a move in the move sequence exceeds the boundaries of the game board. If it does, you should print an error message such as:

“Error : Game board boundaries are exceeded. Input line ignored.”

And you should also ignore the input line completely.

2. Move Count Check: You need to check if a move sequence in the commands file has exact number of move steps according to the character types. If there are more or less move steps etc., you should print an error message such as:

”Error : Move sequence contains wrong number of move steps. Input line ignored.”

You should not need to check anything other than these two error situations.

## 5 Design Notes

- Class that contains your main method should be named ”Main.java”.
- Some default values are given to you in this document, such as AP values, you will also be supplied a class file named ”Constants.java”. You can find this file via Piazza. This is a JAVA class called Constants and contains variables corresponding to some values given in this document. You have to use the variables from this file while implementing your software. Do not hard-code these values into your code. The variables in the file are defined as public variables so you can easily use them anywhere in your project. Download this file and put it into your project before you start coding. And don’t forget to use it!!!
- The files specified in the pdf are going to be given as program arguments. There will be text files (initials.txt, commands.txt) in your working directory. The input and output files will be given as program arguments from the command line. In order to test your program, you should follow the following steps:
  - Upload your java files to your server account (dev.cs.hacettepe.edu.tr)
  - Compile your code (javac \*.java)
  - Run your program (java Main initials.txt commands.txt output.txt)
  - Control your output data (output.txt).
- Samples of input and output files can be found on Piazza. Examine them carefully during coding.

### 5.1 Report

Your final report should contain these topics:

1. Cover Page
2. Software Design Notes
  - Problem: Re-define the experiment in your own words. Keep the definition short by mentioning only the important parts. DO NOT COPY PASTE FROM THIS DOCUMENT.
  - Solution:

- Explain how you approached the problem, what was the most important part of the problem from your perspective and how it effected your design. And then add anything else that you thing will further explain your design.
- Provide a class diagram at the solution section. Below the diagram, give an explanation for each class’s role in the design. Use only 1 - 2 sentences for each class. If you have more to say, you should say it at the first part of the solution section. If your class diagram is too big to be put in the report file, submit it as a jpg file with the report.

## 6 Submit Format

Do not submit any file via e-mail. You should upload your files via “Online Experiment Submission System” which is at <http://submit.cs.hacettepe.edu.tr>.

- File hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system)

- student id.zip

< src (Main.java, \*.java) >

< report.pdf, \*.jpg(optional)>.

## 7 Grading Policy

Task	Point
Compiled	10
Output	60
Report	30
Total	100

## 8 Notes

- The assignment must be original, individual work. Downloaded or modified source codes will be considered as cheating. Also the students who share their works will be punished in the same way.
- We will be using the Measure of Software Similarity (MOSS) to identify cases of possible plagiarism. Our department takes the act of plagiarism very seriously. Those caught plagiarizing (both originators and copiers) will be sanctioned.
- You can ask your questions through course’s piazza group and you are supposed to be aware of everything discussed in the piazza group. General discussion of the problem is allowed, but DO NOT SHARE answers, algorithms, source codes and reports .
- Ignore the cases which are not stated in this assignment and do not ask questions on Piazza for such extreme cases.



- You are responsible for a correct model design. Your design should be accurate. It is important to **draw a class diagram** to show the whole of the system that you have created.
- Don't forget to write comments of your codes when necessary.
- The names of classes', attributes' and methods' should obey to Java naming convention.
- Do not submit your project without first compiling it on dev machine.
- Save all work until the assignment is graded.
- Do not miss the deadline. Submission will be end at 07/05/2021 23:59, the system will be open until 23:59:59. The problem about submission after 23:59 will not be considered.
- There will be again 3 days extensions (each day degraded by 10 points) in this project.
- Your grades will be announced within 15 days at the latest. Objections about your grades can be made within specified days given by TAs. Objections made outside of the specified days will not be accepted.

## 9 Appendix

Initials.txt	Commands.txt	Output.txt
BOARD 5x5  CALLIANCE HUMAN H0 2 4  ZORDE TROLL T0 2 0	T0 0;1; H0 0;-1;0;-1;0;-1;	***** *      T0      * *              * *              * *              * *      H0      * *****  H0      100      (100) T0      150      (150)  ***** *              * *      T0      * *              * *              * *      H0      * *****  H0      100      (100) T0      150      (150)  ***** *              * *      T0      * *              * *              * *              * *****  T0      20      (150)  Game Finished Zorde Wins

Figure 1: Representation of sample input and output files