PROGRAMMING ASSIGNMENT 1

TAs : Nebi YILMAZ
Due Date : 31.03.23 (23:59:59)

Ball Collision Game

1 Introduction

In this assignment, you will implement an interactive program to play a board game. In this way, you are expected to gain knowledge on basic JAVA programming. The program you are going to develop will deal with variables, loops, string operations and file read/write operations. Besides the programming task, you will also learn to comply with coding standards.

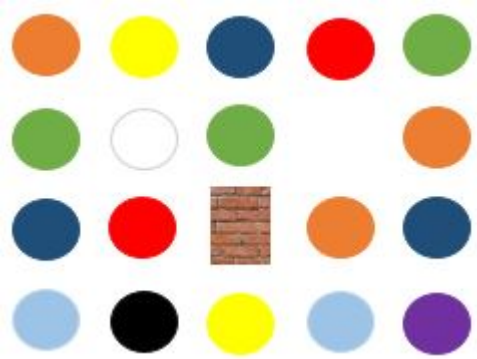

Figure 1: Overview of a board

2 Game

In this game, the white ball which is placed in a board will move according to the commands and collect points according collision with balls of different colors. The overview of the board is given in Figure 1. The user plays the game by moving the white ball denoted by character '*****' to horizontally or vertically adjacent cells. The user earns/lose points if he/she move the white ball to a cell containing balls of Red (R), Yellow (Y), and Black (B). If the user move the white ball to a cell containing a Hole (H), the game is over since the ball will fall in the this hole. The board may contain Walls (W). The white ball cannot break the wall. If the ball is commanded to move towards the wall, the ball hits the wall and moves 2 units to the opposite direction from wall. The user moves the white ball (*) by giving directions to the computer about which direction it is to be moved. There are four allowed moves: Right (R), Left (L), Up (U), and Down (D). The white ball can move one cell at a time.

Color	Abbreviation	Color	Abbreviation
Black	B	Orange	O
Gray	G	Dark blue	D
Yellow	Y	Light blue	L
Red	R	Fuchsia	F
Pink	P	Navy blue	N

Figure 2: List of color and its abbreviation

If the cell that the white ball is trying to move is already occupied, i.e., if there is another color there, the white ball replaces those character with the * character. There are additional rules for the color Red (R), Yellow (Y), Black (B), and Hole (H):

- If the swapped character is R, Y, or B, after swapping, they are replaced with X character.
- If the cell that the white ball is trying to move is Hole (H), then White ball ('*') fall into to Hole (H) and is disappear. That is, the '*' character will not swap with anything and the previous position of the '*' character will be space.
- If the cell that the white ball is trying to move is already occupied, i.e., if there is "X" character there, the white ball replaces those character with the * character
- If the character to be replaced is W, the white ball cannot break the wall. If the ball is commanded to move towards the wall, the ball hits the wall and moves 2 units to the opposite direction from wall, as shown in Figure 3. It should be noted that there will never be a wall 2 units beyond a wall in the input files as the game will go into an infinite loop. Therefore, you will not be given such an input file, as shown in Figure 4.

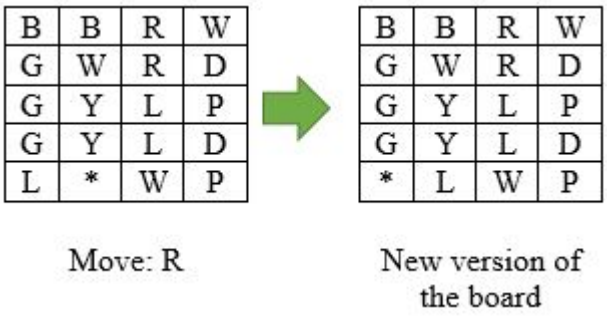

Figure 3: Movement of the white ball moving towards the wall

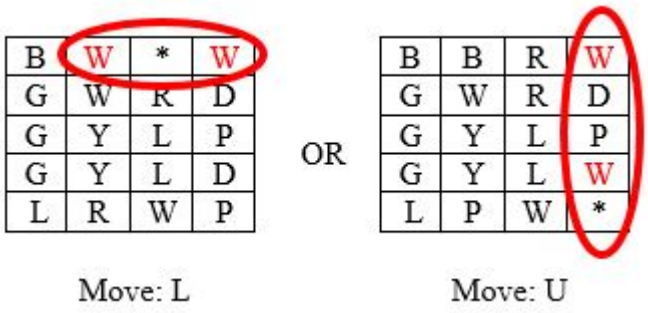

Figure 4: Examples of input you are not responsible for

Moreover, the white ball can get out of the board. For example, if the white ball is on the far left and is asked to move to the left, the ball will move to left and appear from the far right. This situation is the same for all directions.

In this game, the board can have different size such as 4x8, 5x4, 6x6 or so on. The maximum size of board can be 20x20. The board can contain a total of 10 colours, listed in Figure 1. Apart from these, it can contain character '*' character, Wall (W) and Hole (H).

Some rules are listed for scoring as follow:

- The user earns 10 points if the white ball collide with Red (R) ball
- The user earns 5 points if the white ball collide with Yellow (Y) ball
- The user loses 5 points if the white ball collide with Black (B) ball
- The white ball is lost and the game is over, if the ball falls in the Hole (H)

3 Input / Output

The board and directions will be given in input file, named board.txt and move.txt, respectively. You will take an input files (board.txt and move.txt) as an argument from the command line. According to this input files, you will make some operations and print the output.txt at the end of the input file operations. Example of board and move files is given as follows:

Example board file:

```
B W O H D
G W R P Y
G Y Y B P
W L L D D
L * Y W P
```

Example move file:

```
L R R R U D D U
```

The specification for the inputs is given below:

- Elements of board can consists of letters given in Figure 2, `W` and `H` in board.txt. However, it is always guaranteed that the map contains one `*` representing the initial position of the white ball. The number of rows and columns in the board will be between 1 and 10.
- Directions of movement can consist of `L`, `R`, `U` and `D` in move.txt. The input list contains a maximum of 30 movement commands.
- Elements other than these elements will not be used in the board.txt or the move.txt files. In your implementation, you do not need to test these conditions. For instance, you are not given an input files as follow:

Example of wrong board file:

```
T A O H
* W R K
G Y * W
```

Example of wrong move file:

```
K L R U A D R
```

Example input and output files

The example input and output files is given as follows:

```
------------------------ Example input-1-----------------------------
```

Example input 1 (board.txt):

```
B W O H D
G G R P Y
G Y Y L P
W L R D D
L * Y W P
```

Example input 1 (move.txt):

```
L R R R U D D D U L
```

Example output 1 (output.txt):

```
Game board:
B W O H D
G G R P Y
G Y Y L P
W L R D D
L * Y W P
```

```
Your movement is:
L R R R U D D D U L

Your output is:
B W O H D
G G R P Y
G Y Y L P
W X * D D
L L X W P

Score: 15
```

---------- Example input-2----------

Example input 2 (board.txt):

```
B W O H D B
G R R B Y Y
G Y Y W P N
W L L D D W
L R Y R * W
```

Example input 2 (move.txt):

```
L R R U L U U L U
```

Example output 2 (output.txt):

```
Game board:
B W O H D B
G R R B Y Y
G Y Y W P N
W L L D D W
L R Y R * W

Your movement is:
L R R U L U U R U

Your output is:
B W O H D B
G R X   Y Y
G Y X W P N
W L X L D W
L R Y D X W

Game Over!
Score: 20
```

4 Find optimal path (Bonus)

This section is designed to improve your research skills. It encourages you to both research and learn by using multiple resources.

Your effort for this section will be reflected as bonus points. In this part of the game, you are asked to find the shortest path through the cells (i.e., minimum number of movement) to get the highest score. That is, your path MUST contains all the red (R) and yellow (Y) balls in board because only these balls give you points. You are asked to print this path to an output file named bonus.txt. Please note that this output file (i.e. bonus.txt) is a different file than the output file (output.txt) you will obtained from part above. In other words, if you are going to implement the bonus task, you should get two output files in total (output.txt and bonus.txt). You cannot just do the bonus task without doing the main task in the assignment.

Note:

- Here, it is assumed that the White ball '*' will behave differently when it hits the Wall (W). The '*' character cannot break the wall, and if the '*' is in the cell next to the wall, it remains in the cell where it is, even if commanded to move towards the wall, as shown in Figure 5.
- The user will not lose/earn any points if the white ball collide with Black (B) ball
- All other rules (except for wall (W)) are as described in Section 2.
- If there is more than one possible shortest path, it is enough to find and print one of them.
- If necessary, the visited cell can be traversed again.

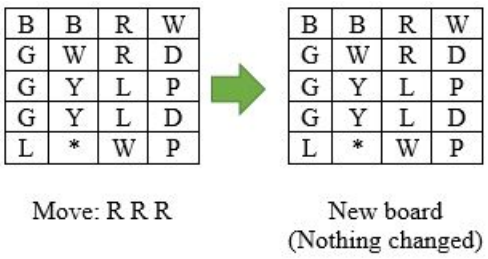

Figure 5: Movement of the white ball moving towards the wall (only for Bonus task)

4.1 Input/output for Bonus

Example outputs (bonus.txt) for input 1 and 2 (given in Section 3) is given below:

---------- Example output-1----------

Example output 1 (bonus.txt):

```
Game board:
B W O H D
G G R P Y
G Y Y L P
W L R D D
L * Y W P

Your optimal path is:
R U U L U R R R

Number of movements: 8

Your output is:
B W O H D
G X X X *
G G X L P
W L X D D
L X X W P

Best score: 40
```

---------- Example output-2----------

Example output 2 (bonus.txt):

```
Game board:
B W O H D B
G R R B Y Y
G Y Y W P N
W L L D D W
L R Y R * W

Your optimal path is:
L L L U U R U L L L L

Number of movements: 11

Your output is:
B W O H D B
X G X B * X
G X X W P N
W X L D D W
L L X X X W

Best score: 70
```

Execution and Test

The input files is going to be given as program arguments. Board file (board.txt) is the first argument and the move file (move.txt) is the second one. 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, or javac Main.java)
- Run your program (java Main board.txt move.txt)
- Control your output file (output.txt)
- If you implemented, control your output file for Bonus (bonus.txt)

Grading Policy

Task	Point
Submit	1
Clean code	10
Coding standard	5
Correct output.txt	79
Correct bonus.txt	10
Total	100 + Bonus

Submit Format

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

```
- <studentid>.zip
    - <src>
       - Main.java, *.java
```

Late Policy

You have three days for late submission. You will lose 10 points from maximum evaluation score for each day (your submitted study will be evaluated over 90 and 80 for each late submission day). You have to submit your solution in deadline date + two days, otherwise it will not be evaluated.

Notes and Restrictions

- Do not miss the submission deadline.
- Save all your work until the assignment is graded.
- Compile your code on DEV server before submitting your work to make sure it compiles without any problems on our server.

- Source code readability is a great of importance for us. Thus, write READABLE SOURCE CODE, comments and clear MAIN function. This expectation will be graded as "clean code".
- Regardless of the length, use UNDERSTANDABLE names to your variables, classes and functions. The names of classes, attributes and methods should obey Java naming convention. This expectation will be graded as "coding standards".
- Name of all files are fixed (input files: board.txt, move.txt) (output file: output.txt)
- 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.
- All assignments must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.

Appendix-A

You can use the following code to read your input files.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class ReadFromFile {

    public static String[] readFile(String path) {
        try {
        int i =0;
        int length= Files.readAllLines(Paths.get(path)).size();
        String[] results = new String[length];
            for (String line : Files.readAllLines(Paths.get(path))) {
                results[i++]= line;
            }
        return results;
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }

    public static void main(String[] args) {
        String[] lines= readFile("testfile.txt");
        for (String line : lines) {
            System.out.println(line);
            }
    }
}
```

Figure 6: Source code to read input files