Software Construction (L+E) HS 2022

Instructor: Prof. Dr. Alberto Bacchelli — Assignment 1

Tutors: Adam Bauer, Loris De Luca, David Moser, Jonas Zellweger — Week 04

To correctly complete this assignment you **must**:

- Carry out the assignment with your team only (unless otherwise stated). You are allowed to discuss solutions with other teams, but each team should come up its own personal solution. A strict plagiarism policy is going to be applied to all the artifacts submitted for evaluation.
- Prepare the solutions to the exercises by strictly following this structure:
  - A root folder named: `Group[id on OLAT]-a[AssignmentNumber]`[1]
  - One subfolder per each exercise named using the double digit number of the exercise.[2]
  - Inside each subfolder:
    * an `answer.pdf` file that answers the exercise's questions, directly and/or by explaining the decision taken in the source code, depending on the exercise's requirements;
    * only for questions requiring code: a `src` subfolder with the source code of your solution.
- Package your root folder into a single ZIP file named:
  `Group[id on OLAT]-a[AssignmentNumber].zip`[3]
- Upload the solution to the right OLAT task by the deadline (*i.e.*, **Nov 07, 2022 @ 18:00**)

Exercise 1 - A Terminal-based Battleship Game - Design

Referring to the game requirements in Appendix A and to the rules you find in Appendix B, use what you have learned in the lecture about Responsibility Driven Design to do the following tasks:

1. Following the Responsibility Driven Design, start from the game's requirements and rules and derive classes, responsibilities, and collaborations (use CRC cards). Describe each step you make and store the final cards in your answer.
2. Following the Responsibility Driven Design, describe the *main* classes you designed to be your project in terms of responsibilities and collaborations.
3. Why do you consider the other classes as less important? Following the Responsibility Driven Design, reflect if some of those non-main classes have similar/little responsibility and could be changed, merged, or removed.
4. Draw the class diagram of the aforementioned main elements of your game.
5. Draw an object diagram to show the main elements of your game in a step of the game of your choosing.

Exercise 2 - A Terminal-based Battleship Game - Implementation

Implement in Java the game design that you designed in the previous exercise.[4] Make sure to put in practice what you have learned in the first sessions (e.g., encapsulation, input validation, contracts, design patterns) and explain how you did so in the accompanying `answer.pdf` file.

[1] *e.g.*, a correct name would be: `Group1-a1`.
[2] *e.g.*, the subfolder `01` contains the answer to exercise *1*.
[3] *e.g.*, a correct name would be: `Group1-a1.zip`.
[4] For the assignment to be valid, it should also be possible to play the entire game on the terminal.

Appendix A: Requirements for Battleship' Terminal Game

In this game, one user plays Battleship against the computer, inputing her/his moves and seeing the results at each round via the terminal.

When the game starts, the program outputs—on the terminal—the empty ocean grid and target grids. The two grids should be displayed as shown in the following example:

Game start: terminal output

```
===== TARGET GRID =====
  A B C D E F G H I J
 +-+-+-+-+-+-+-+-+-+-+
0| | | | | | | | | | |0
1| | | | | | | | | | |1
2| | | | | | | | | | |2
3| | | | | | | | | | |3
4| | | | | | | | | | |4
5| | | | | | | | | | |5
6| | | | | | | | | | |6
7| | | | | | | | | | |7
8| | | | | | | | | | |8
9| | | | | | | | | | |9
 +-+-+-+-+-+-+-+-+-+-+
  A B C D E F G H I J
=======================

-----------------------

===== OCEAN  GRID =====
  A B C D E F G H I J
 +-+-+-+-+-+-+-+-+-+-+
0| | | | | | | | | | |0
1| | | | | | | | | | |1
2| | | | | | | | | | |2
3| | | | | | | | | | |3
4| | | | | | | | | | |4
5| | | | | | | | | | |5
6| | | | | | | | | | |6
7| | | | | | | | | | |7
8| | | | | | | | | | |8
9| | | | | | | | | | |9
 +-+-+-+-+-+-+-+-+-+-+
  A B C D E F G H I J
=======================
```

The game then asks the human player where they want to place each boat in their fleet, one by one;[5] the user inputs the starting block and the ending block as capitalized characters and numbers, separated by a comma.[6]

During the input of the boats' positions, the game must check that the boat can be placed in the user-specified position. If the position entered by the user is invalid (e.g., because overlapping with another boat), the game informs the user and asks them to enter another position.

Once all the ships have been correctly placed, the updated grids are displayed again, using the first letter

[5] e.g., `Please enter the position of your BattleShip 1`

[6] e.g., `A2,A5`

to represent the type of the boat in the grid (e.g., `P` and `S` are used for blocks occupied by Patrol boats and Submarines, respectively).

The text below shows an example of how the game looks like after the user has positioned their fleet:

Sample placement of fleet: terminal output

```
===== TARGET GRID =====
  A B C D E F G H I J
 +-+-+-+-+-+-+-+-+-+-+
0| | | | | | | | | | |0
1| | | | | | | | | | |1
2| | | | | | | | | | |2
3| | | | | | | | | | |3
4| | | | | | | | | | |4
5| | | | | | | | | | |5
6| | | | | | | | | | |6
7| | | | | | | | | | |7
8| | | | | | | | | | |8
9| | | | | | | | | | |9
 +-+-+-+-+-+-+-+-+-+-+
  A B C D E F G H I J
=======================

-----------------------


===== OCEAN  GRID =====
  A B C D E F G H I J
 +-+-+-+-+-+-+-+-+-+-+
0|P|P| |S| | |B|B|B|B|0
1| | | |S| | | | | | |1
2|B| | |S| |S|S|S| |P|2
3|B| | | | | | | | |P|3
4|B| | | | | | | | | |4
5|B| | | | |P| | | |S|5
6| | | | | |P| | | |S|6
7|P|P| | | | | | | |S|7
8| | | | | | | | | | |8
9| | | | |C|C|C|C|C|C|9
 +-+-+-+-+-+-+-+-+-+-+
  A B C D E F G H I J
=======================
```

The computer will position its boats on the board (without revealing their positions to the user) randomly selecting a starting and final block for each boat. Although the position of each boat is selected randomly, it must follow the rules of the game.

At each turn, the player selects a position within the board to attack. The computer randomly selects positions of the user's board. If the place was already chosen before, the game considers the position invalid and asks for another position.

If a boat is hit in the computer's board, an `X` will appear in the position the bomb was thrown. Otherwise, an `o` should be shown. Once all positions of a ship are hit, the boat is destroyed and the `X`'s are replaced by the boat's initial letter to reveal itself to the opponent.

In the example below, the target grid has been updated to show that: the user hit a computer's ship in `A0` (without destroying it), did not hit anything in `C0`, and destroyed a submarine in `C2,E2`.

Sample result of shots: terminal output

```
 ===== TARGET GRID =====
   A B C D E F G H I J
  +-+-+-+-+-+-+-+-+-+-+
 0|X| |o| | | | | | | |0
 1| | | | | | | | | | |1
 2| | |S|S|S| | | | | |2
 3| | | | | | | | | | |3
 4| | | | | | | | | | |4
 5| | | | | | | | | | |5
 6| | | | | | | | | | |6
 7| | | | | | | | | | |7
 8| | | | | | | | | | |8
 9| | | | | | | | | | |9
  +-+-+-+-+-+-+-+-+-+-+
   A B C D E F G H I J
 =======================

 -----------------------

 ===== OCEAN  GRID =====
... omitted for brevity ...
 =======================
```

At the end of any turn, the game checks whether the entire fleet of a player is destroyed; if so, the game declares the other player as the winner. If the computer won the game, the target grid is updated to show the human user the position of the remaining ships that have not been destroyed.

Appendix B: Battleship' Rules

The following rules have been inspired and adapted from the Hasbro rules:

a. The board is a 10x10 grid of blocks. Each block in the board is associated with a letter (on the horizontal axis) and a number (on the vertical axis). The block in the top-left corner has coordinates `A0` and the block in the bottom-right corner has coordinates `J9`.

b. Each player as a fleet with the following boats: 1x Carrier (6-block long), 2x Battleships (4-block long), 3x Submarines (3-block long), 4x Patrol boats (2-block long).

c. Each player sees only two grids: the *ocean grid* and the *target grid*. The ocean grid is used to record the position of the player's own ships and the shots called by the opponent. The target grid is used to record the results of the player's own shots.

d. Each player starts by placing their ships, according to these rules:

   - Each ship can be placed in horizontal or vertical positions, but not diagonally.
   - No ship can overlap another ship or being placed (partially) outside of the grid.
   - Once placed, ships cannot change their position.

e. After the placement of the ships, the starting player is decided by random choice. Subsequently, the two players alternate turns.

f. At each turn:

   - One player (*i.e.*, the *shooting player* for this turn) calls one shot trying to hit the ships of the other player (*i.e.*, the *receiving player* for this turn). The shot is called by specifying the coordinates of the target block..
   - After a shot is called, the receiving player declares whether the shot is a *miss* or a *hit* (without specifying the type of boat hit).
   - The result of the shot is recorded on the *ocean grid* and the *target grid* of each player.
   - If any of the receiving player's ship has been entirely hit (*e.g.*, two hits for a Patrol Boat), the receiving player must announce that a ship was sunk.
   - If the entire fleet of the receiving player has been sunk, the shooting player is declared as the winner; otherwise the turn finished and the two players exchange their roles.