Software Engineer Assignment

Emanuele Buda

1 - Running Instruction

To run the game it is just necessary to open the terminal and enter in the game path, then launch the command `java -jar Arena.jar`

Once the game is launched just follow the directions of running script to set up the arena of the game. After finishing the setup the game will start automatically.

It is strongly recommended to enlarge the terminal to a size where you can see the field and other panels clearly (e.g. 90x45 if SIZE = 30).

<u>Java must be installed in the machine</u>.

2 - Game Mechanics

Users have the possibility to choose the size of the arena and the duration of the game in turns (iterations).

In the arena there will automatically loaded two standard robots: rand and coward, in order to be sure that the minimum number of robots is reached, in `/Robot` folder there is also vertical.

The maximum number of robots is 5.

Users <u>can import into the game a personal robot</u> following the guidelines of the script, or add an already loaded one.

I decided to implement simple game mechanics, such as:

- Each <u>robot can only perform one single action per turn</u> between shoot or move. So if the implemented algorithm outputs to move the robot will move and it will not be able to shoot until the next turn, and vice versa.
- Every turn the actions of the robots take place in a synchronized manner
- Robots can only move one cell each turn horizontally or vertically, not obliquely
- Robots can only shoot horizontally or vertically, not obliquely
- Missiles can only explode either touching the end of the arena or hitting a robot
- If a robot gets hit by a missile both will be removed from the game with all the missiles of the destroyed robot
- A robot wins when all other robots are destroyed
- If there is no winner, once the turns are finished, the game will end with a draw

2.1 - Arena Layout and Characteristics

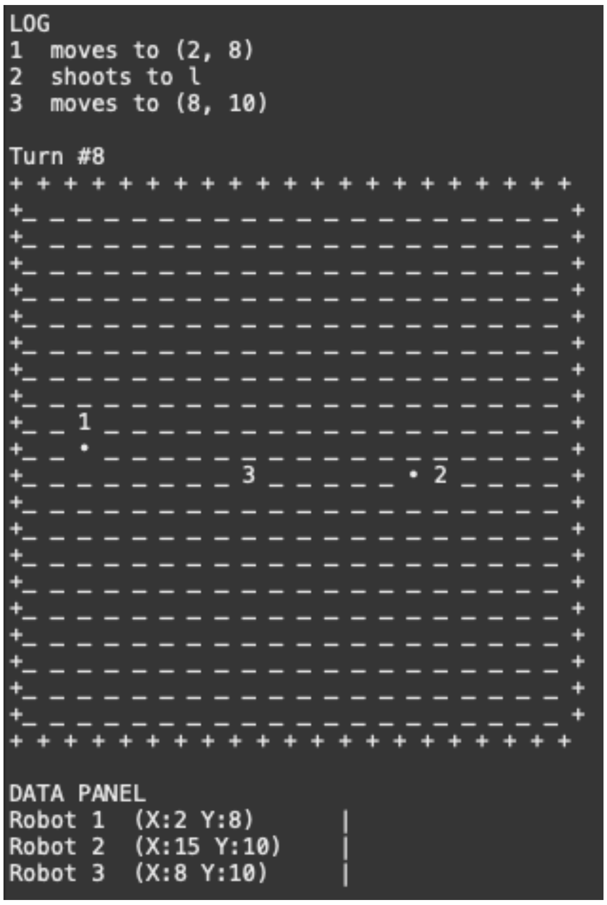
The arena is a bidmensional matrix large SIZE*SIZE where SIZE is a parameter choose by the user. The matrix is filled with “_” that means the cell is empty.

There are also “•” meaning there is a missile, that when explodes is represented with “▦”.

Each number represents a different robot.

There is also a space dedicated to logs, where are reported all the operations done by the robots in that turn, and another one with the current data of each robot position in the matrix (x,y).

Missiles don’t explode touching each other.



```
LOG
1  moves to (2, 8)
2  shoots to l
3  moves to (8, 10)

Turn #8
+ + + + + + + + + + + + + + + + + + + + + +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ 1 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ • _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ 3 _ _ _ _ _ • 2 _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ +
+ + + + + + + + + + + + + + + + + + + + + +

DATA PANEL
Robot 1  (X:2 Y:8)        |
Robot 2  (X:15 Y:10)      |
Robot 3  (X:8 Y:10)       |
```

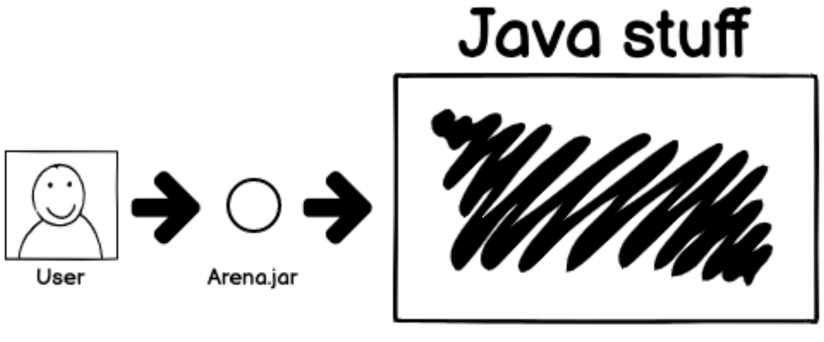
Figure: a little simulation with 3 robots and 2 missiles

3 - Architectural Pattern

The architectural pattern I picked is a very simple decision management in the subdomain of artificial intelligence. Unfortunately the university have not given us the opportunity to go deeper into some topics among which, precisely, the architectural patterns and, thanks to this practical example, I was able to see it and I am going to make up for it.

About the distribution of the classes I used a separate presentation pattern, where files for the user are located in a specific package (Public).

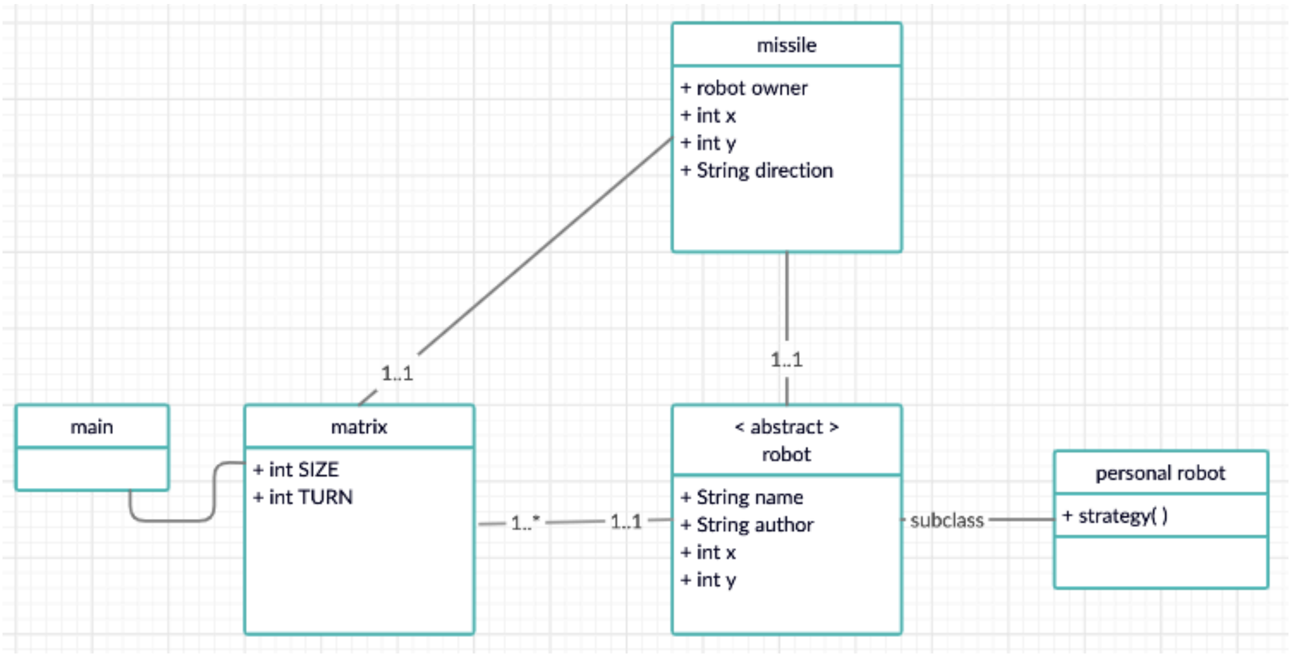
I used the Arena.jar , an executable script of the class compiler in the main folder, to act as an intermediary between the user and the code,

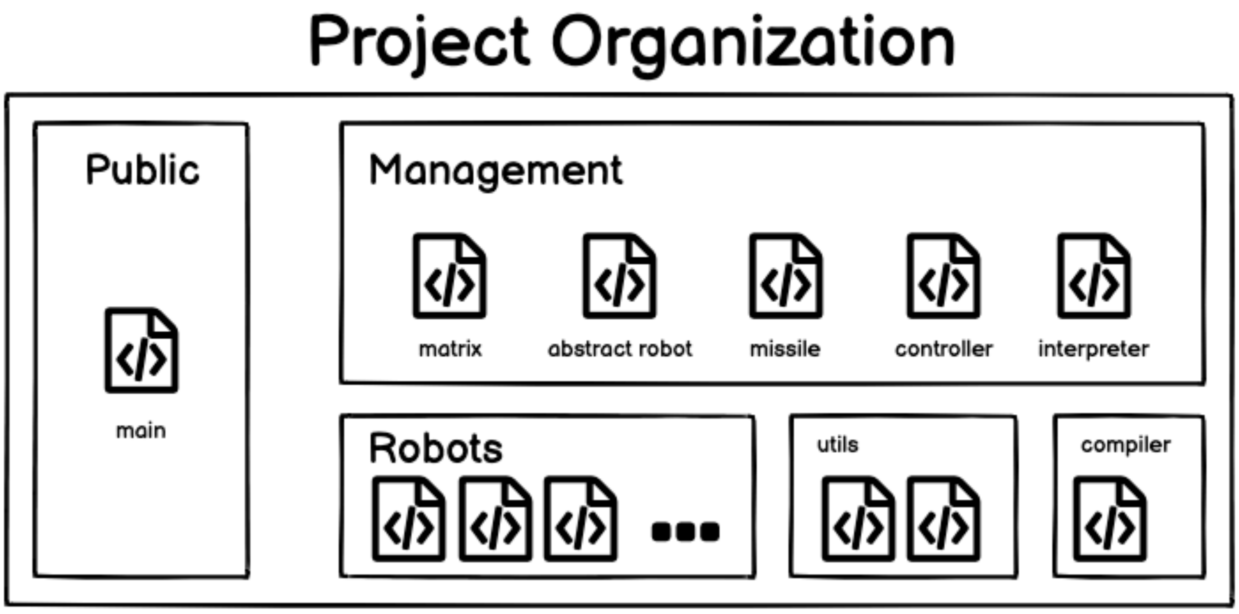


to setup the number of turns, the field size and to the robot import operation, which would otherwise not be easy to do for non-expert users.

I designed the UML of the project then I started to write down the code.

I designed the classes and the methods so that personal robots can not cheat getting sensitive information from other classes and tried to respect the good java rules.





3.1 - Robot AI Language

I've decided to leave as robot language the same as software, Java. Personal robots just have to implement the strategy() method.

3.2 - How to Write Robot AI

As said in the previous chapter the personal robot just have to implement the strategy() method.

The method involves returning as output just a string formatted in this way:

- if robot wants to move “m, x, y”, where “m” stands for move and x,y are the coordinates, they will be double checked from a controller
- If robot wants to shoot “s, x, y, direction” where “s” stands for shoot and direction can be “l” = left, “r” = right and so on …

The output string will be interpreted and checked if it is permitted to make.

Before making a move there is data that can be used to determine which is going to be for your Robot AI the best move like scanning the position of the bot.

With `this.scan` it is possible to scan the arena, it is an ArrayList of strings, where each string is composed in the following way “robotName, robotX, robotY”. Analyzing those information is the key to build a strong robot AI. A few easy examples of how to use `this.scan` can be found in `/Robots`.

A brief description of the already imported robots:

- **Vertical**, moves only downwards, shoots if there are enemies with a common coordinate with him
- **Horizontal**, moves only to the right, practically useless.
- **Rand**, moves randomly, shoots if there are enemies with a common coordinate with him
- **Coward**, moves randomly, but when the situation becomes hot and finds an enemy with a common coordinate with him, he run away

A base file on how to build your own robot can be found in `/utils` .

3.3 - Tradeoffs

Of course I tried to do my best to make sure that the code was at the same time scalable and easy to understand. My focus was being fast and precise, this has inevitably made the software in general, a simple software, although I'm a very creative person. While writing the code, I came up with a lot of ideas about how to make the game interesting.

In the application I have exploited some of the major advantages of the OOP, such as abstraction and encapsulation to my advantage.

4 - Future steps

The game uses really simple mechanics so there's certainly room for improvement. I have many ideas on how to make it more interactive, dynamic and competitive. In order, I would continue the project following these steps:

1. Being just a draft version I haven't optimized memory allocation neither code efficiency, so the very first future step I'd procede doing is the optimization of the code, meaning also the addition of controllers, that are not always present at the moment
2. I would add many more features, which would make the game much more fun and competitive, some of these:
   1. Robots and Missiles may move also obliquely
   2. Different kind of missiles: big ones that explodes in a bigger area, faster ones and so on …
   3. Create rules for which users can make compromises between increasing armor, size and power of shots, while still allowing the game to remain balanced. this would allow greater customization of robots.
   4. Give the possibility to import bots written in other languages
   5. Add randomly generated obstacles in the map
   6. Provide users with the ability to create maps
3. Since the game runs on a terminal it is "laggy", it is not fluid, so I'd use some library to make it more dynamic and maybe animate it
4. Make the simulator accessible also from the web
5. Organize competitive tournaments

5 - Difficulties and Conclusions

Once I processed the information and organized the work, I started doing some visual tests with python, to get an idea of the result I'd get with the prints from the terminal without animations. I liked it, but I also tried some animation library, it was going to be a whole new thing for me and I decided to travel the roads I already knew to present a good work.

It was immediately more or less all clear to me, I was sure from the start that OOP was the best solution, I already had in mind which classes to create and how to use them, the only thing that scared me was the import of an external robot and which architectural pattern and how to use it.

I'd never had to deal with anything like this before, so after chatting with some university colleagues and searching on google I came up with the idea of managing everything using an external script.

Whatever the outcome of the interview will be, this assignment was a great exercise and allowed me to learn new things and discover shortcomings that I didn't know I had and which I will work on to improve myself, for this reason I want to thank you.

6 - Bibliography

- Stackoverflow **-** https://stackoverflow.com/
- Wikipedia page - https://programminggames.org/CROBOTS
- Alessandro Pira's project - https://www.alessandropira.org/pybots/pybots.html
- CROBOTS GitHub - https://github.com/tpoindex/crobots