EDCS Project #7 Battleship game

Authors: Michał Topolski, Yujie Ma

1) Introduction

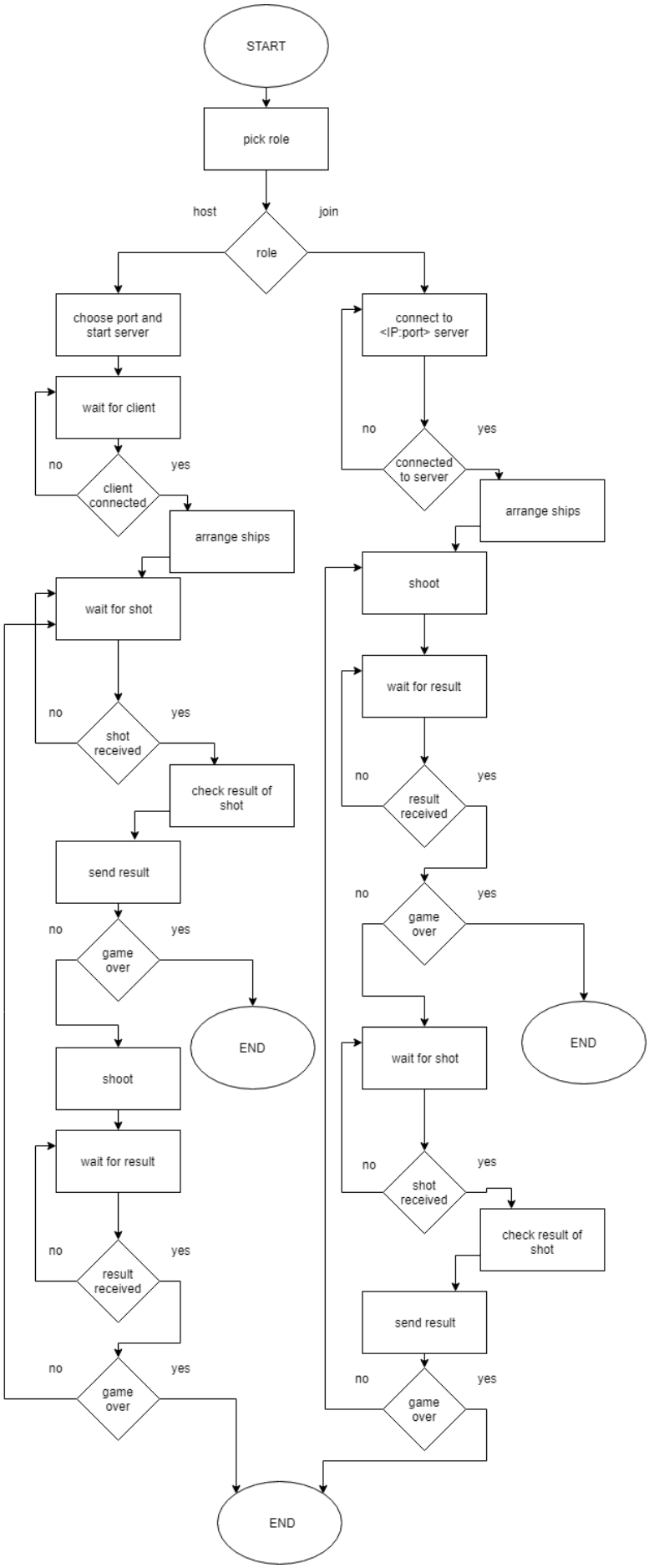
The goal of this project is to create battleships game with online multiplayer for 2 players. At the beginning of the game, both players arrange their ships in secret. After placing ships, the match starts. Players take turns where they choose a target cell coordinates to shoot the enemy. The match is over when one of players destroys all of opponent's ships.

2) Implementation

We implemented the program in Java for Windows and Python 3 for Linux. The player chooses at the beginning if he wants to host match or join it. Game window presents player's sea grid in size 10x10, 5 arranged ships of lengths 2, 3, 3, 4 and 5, enemy's sea grid in size 10x10 and a message field. Type of the connection suitable for turn-based games is TCP/IP and we decided to use it. There are 4 crucial decision about the program:

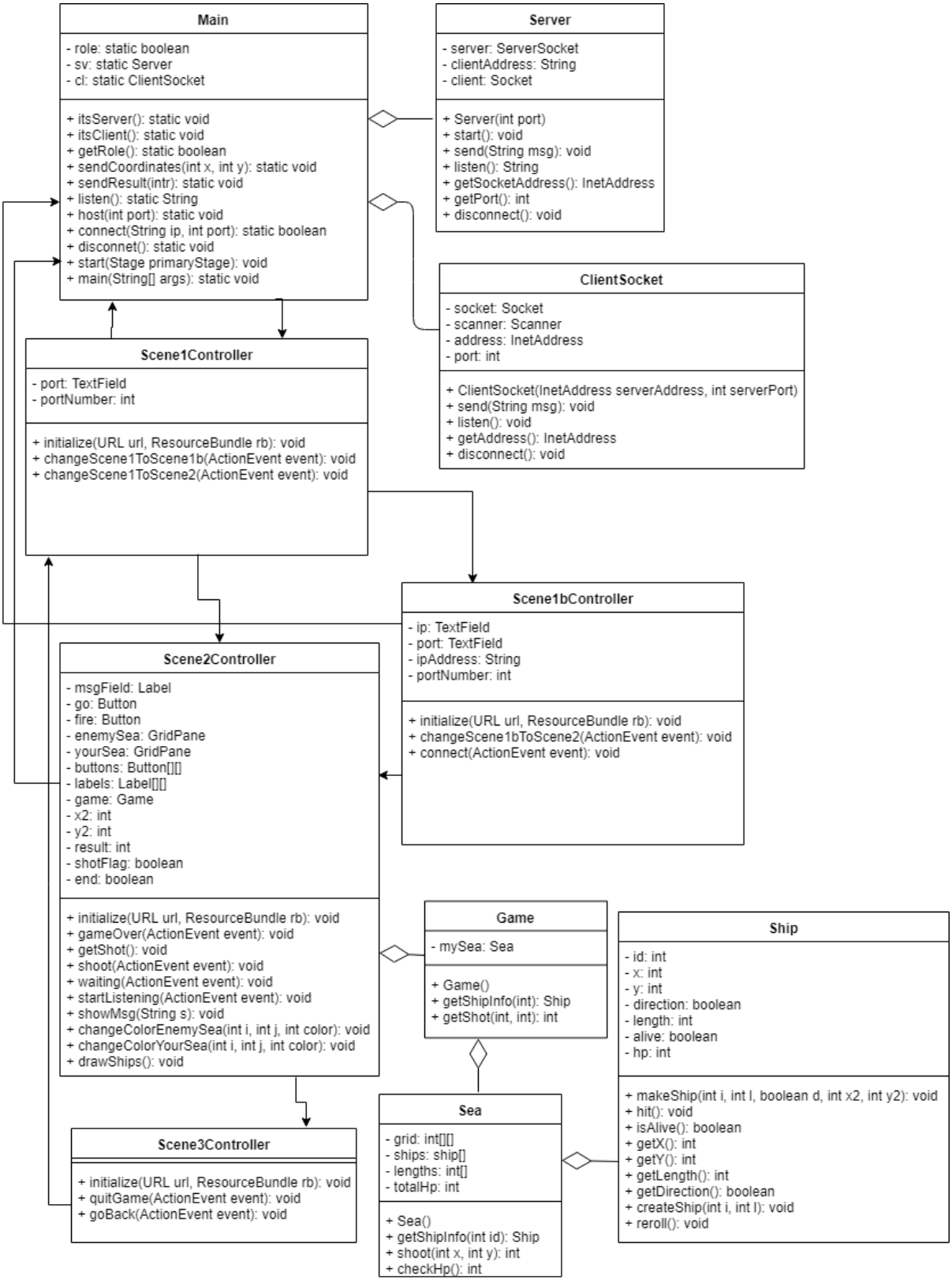
- to place the ship, users chooses direction (1 – horizontal, 0 – vertical), then x and y coordinates of the leftmost/topmost cell of the ship
- there must be chosen the player that will make the first move, it is the player who joins the game
- there are 2 kinds of TCP messages used in this project: <xy> that sends coordinates of the shot i.e. 55 for x=5 and y=5, <result> that sends the result of the shot (0 – miss, 1 – hit, 2 – game over)
- each player stores information only about his ships

General workflow of the program looks following:


START
pick role
host
join
role
choose port and start server
wait for client
no
yes
client connected
arrange ships
wait for shot
no
yes
shot received
check result of shot
send result
no
yes
game over
END
shoot
wait for result
no
yes
result received
no
yes
game over
connect to <IP:port> server
no
yes
connected to server
arrange ships
shoot
wait for result
no
yes
result received
no
yes
game over
END
wait for shot
no
yes
shot received
check result of shot
send result
no
yes
game over
END

Program 1 (by Michał Topolski)

Language: Java Operating system: Windows 10

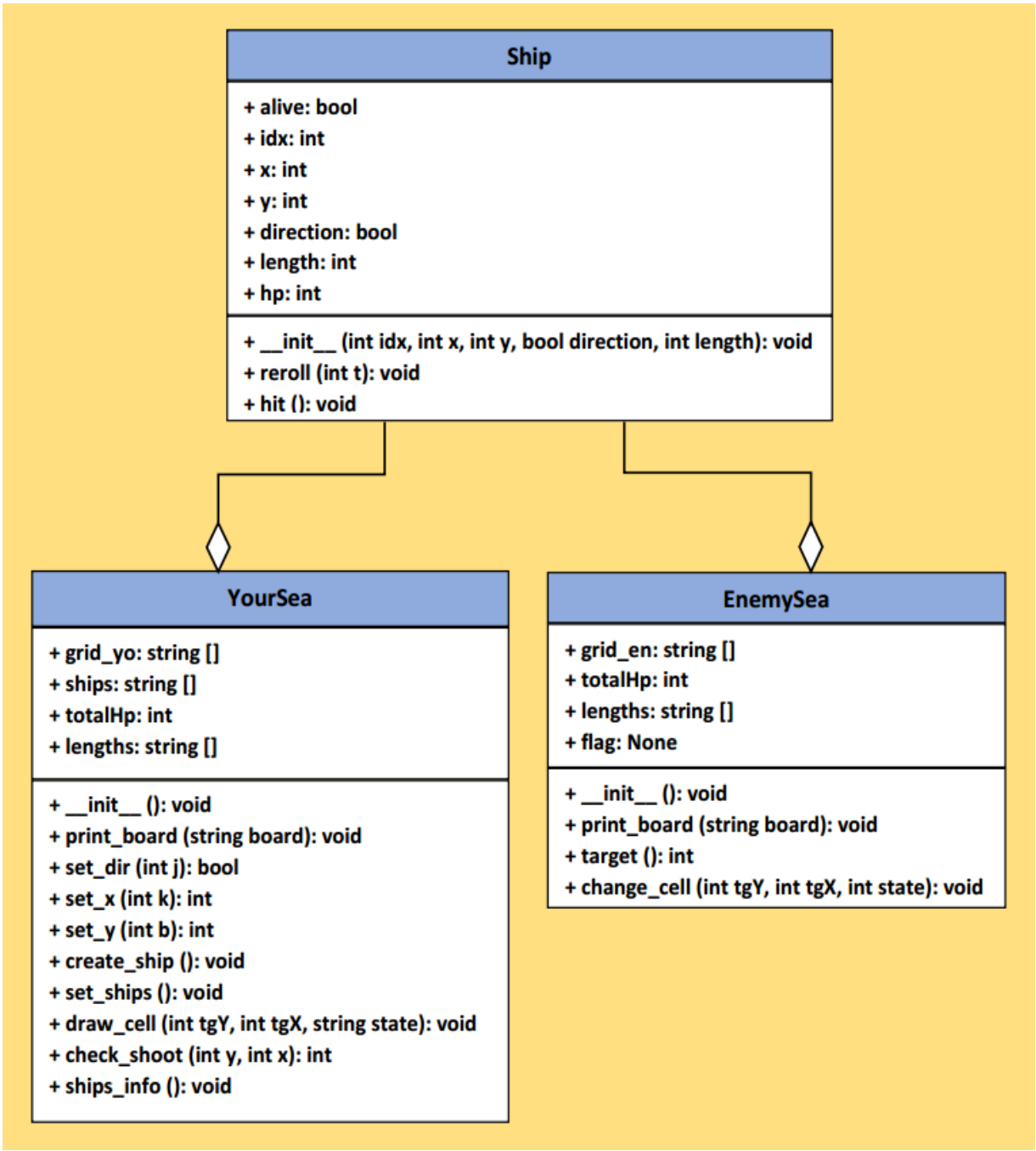


This program works in Model-View-Controller architectural pattern. Controller classes are responsible for connecting UI with the data and implementing game mechanisms. Main is the root class. Depending on the player's role (host or guest), it creates an object Server or ClientSocket that is responsible for establishing connection and communication between players, and initialize scene 1. Scene 1 allows to host the match on desired port or join the match. When choosing option to host, the game waits for incoming connection then asks to arrange ships (using terminal) then goes to scene 2. When choosing option to join, it goes to scene 1b that asks ip address and port of the host to try to connect. If attempt is successful, it asks to arrange ships and proceeds to scene 2. Scene 2 draws two grids that represent player's sea with his ships and enemy's sea which is empty. Host needs to click button START to start listening for incoming message with coordinates. Guest can instantly click on a cell he wants to target then click button WAIT to start listening for incoming shot. Next, host takes the shot. The match continues until one of players lose all of his ships. After the game over, the user should click button FINISH to go scene 3 that allows him to exit the program or go to scene 1 to create/join new game. Class Ship keeps information about a single ship like his length, direction and coordinate of the first cell. Class Sea has information about player's ships as well as sum of ships' health points that is necessary to check if the match is over.

To run the program, user needs to open command line terminal in a directory where battleships.jar is and type "java -jar battleships.jar". Terminal is also used to arrange ships, but the game itself runs in a window.

Program 2 (by Yujie Ma)

Language: Python 3 **Operating system: Kali Linux**



This program works in python 3 in Kali Linux system. The game will have GUI option pane which allows to choose the role, and later game are displayed in terminal. In python the mechanisms work a bit different from Java which not strictly required everything need to be an class. The file main.py is the root widget which creates an option window for choose “Host” or “Join” using built-in python 3 package ‘tkinter’. Click “Host” will call the server.py to run the function click_host() which allows to host the match. After entering the host

scene, host needs to enter the port which allow client to join the game. If click the “Join” will call the client.py to run the function click_join() which allows client join the match. For after connection successful, both side sneeds to arrange the ships in their own sea. After both side arranging the ships, then host’s side should start waiting and client’s side start enter the x and y coordinate he wants to shoot. Then server side received coordinate and check in his sea by calling function check_shoot() in YourSea class then return state “MISS”, “HIT” or “VICTORY”, then change the corresponding coordinate in his own sea by calling draw_cell() in YourSea class. Then send back the state to client, and client receives the state and change the situation in his sea by calling change_cell in Enemy Class. Then the host and client switch the role, client starts waiting and host start shooting, the same mechanism will be repeated. Both sides will repeat round in while loop until one side all ships are been sunk. Eventually, as winner will see message “You win” in his terminal and if as loser the “You loss” message will display in his terminal. And game will be finished.

To run the program, user needs to open terminal in Linux and goes to the directory of the project folder then enter command “python3 main.py”(if not root user needs to add “sudo” at the beginning of command).

3) Implementation decisions

Common features:

- workflow of the program
- decision about the client having the first turn
- communication by TCP messages
- TCP messages contain 1 or 2 digits (result of shot and coordinates of shot respectively)
- general structure of programs
- distribution of data – each player stores information about his own ships and all taken and received shots
- if one player leaves in the middle of the game, it is considered the end of match by default, if he wants to reconnect, the match must be restarted

Differences:

- Java program has GUI and terminal, Python only terminal (due to using GUI in python is not very convenient)
- Python encodes messages as bytes, Java as strings, so they must be transformed
- for the convenience of testing, Java program has parts of code that allow to restart the game without restarting the program and random generation of ships (first feature is used, second is commented out, but can be added as an option in future)
- Java programs runs fully using classes, while Python combines classes with scripts, which results in simpler class diagram

4) Conclusions

This project allowed us to use in practice some aspects that we have learned during lectures. We managed to make the game working properly in 2 different programming languages and systems. In the future, it is possible to add more functions like giving a choice to arrange ships manually or randomly and GUI for Python program.