Battleships Online

Java Console Game

CONTENTS

1. Architecture

The Battleships online console game is implemented as a client-server application. The business logic of the game is implemented entirely on the server side. The client sends text messages to the server, which it tries to read as executable commands. The client starts a second thread, concurrent to the main one, that listens for messages that the server can send back to the client. In this sense, the client is multi-threaded and represents the implementation of an ordinary chat client.

1.1. Server

The server implementation is based on the Java NIO API, which allows non-blocking (asynchronous) I / O operations. This on the one hand allows efficient operation of the server with a huge number of parallel clients and requests, and on the other hand, there will be no need for synchronization in the server, as the thread that will execute the requests will be only one. The fact that only one thread executes commands on the server is not an issue, because each command is executed either for constant time complexity or linear in terms of the number of players who connected to the server (online) at the time of its invocation.

1.2. Client

The client, as already mentioned above, is a simple chat-client. When starting the client code, the user is expected to enter his username with which to connect to the server. (Note that here the client will only connect to the server with the appropriate username. The server does not yet maintain a status for this client). It is checked by regular expression whether it contains only lowercase or uppercase Latin letters. In addition, the username must be at least 4 characters long and at least 9 characters long. If these username restrictions are not met, the client does not even try to connect to the server. The user is asked to re-enter a username, being informed that the previous username he entered does not meet the requirements.

When connecting to the server, the client executes the set-username command by default with a valid (validated by regular expression) username argument. If this username is already taken and there is another user with the same username, then the command will return a response indicating the problem and the client will be prompted to re-enter the username and to try to connect to the server with another one.

Only after the server returns a message indicating that the client is successfully connected to it, the client runs a second thread that works in parallel with the thread sending the commands. The newly generated thread, instead of sending messages to the server – listens to whether the server wants to send a message to it and the moment it receives one, it prints it on the client console.

2. Комуникация

The communication between the client and the server is carried out through the TCP protocol:
The client sends commands to the server in the form of simple strings. The server takes care to verify that these strings match a valid command that can be executed on his side. In case the command is not valid, the server indicates that by a proper message returned. If the command is valid, the server can send a message to more than one client after the execution of the command. In both cases, the messages are in JSON format. In short, the client sends raw strings to the server, and the server returns JSON strings to the client.

3. Функционалности

3.1. man

User manual command without arguments. Formats and displays online commands that can be executed on the server. Depending on whether the client is in game mode (host / guest) or just surfing (online), a menu is returned that shows him what the usual commands are in the mode in which he is. For example, if he is a guest in a game and executes the man command on the server – the server will return the manual to a user who is in game mode. If the user is not in game mode,

then when executing the man command he will be shown what the commands are in non-game mode.

не игрови режим

```
Please enter your username: pesho
Trying to connect to Battle ships online with username pesho
Welcome to Battleships online!
> Your username is now pesho
battleships menu> man
> User manual in SURFING mode:
> man
> who-am-i
> set-username <username>
> change-username <username>
> list-users
> list-games
> register
> login <api-key>
> create-game <game-name>
> join-game {<game-name>}
> current-game
> save-game
> saved-games
> load-game <game-name>
> disconnect
```

игрови режим

```
battleships menu> create-game my-game
> You must login or register first
battleships menu> register
> You are now logged in with api-key: 821742744946511
battleships menu> create-game my-game
> Created game "my-game", players 1/2
battleships menu> man
> User manual in PLAYING mode:
> display
> place <[A-J]> <[1-10]> <[UP, RIGHT, LEFT, DOWN]> <[2-5]>
> place-all
> attack <[A-J]> <[1-10]>
> hacks
```

A user who is in game mode can also execute commands that are not common in game mode, such as list-users, to check how many users are currently connected to the server, or list-games to check which games are currently created and can be joined or are already played and etc.

3.2. who-am-i

Command without arguments. Displays the effective username with which the user is connected to the server.

```
Please enter your username: 100yo
Accepts usernames only from 4 to 8 lowercase or uppercase latin letters!
Enter a valid username: stoyo
Trying to connect to Battle ships online with username stoyo
Welcome to Battleships online!
> Your username is now stoyo
battleships menu> who-am-i
> You are stoyo
```

3.3. set-username

KCommand with one required argument - <username>. This command is executed by default by each new user who tries to connect to the server. If the given as argument username is already occupied by another user, the server expects to re-enter a new free username.

```
Please enter your username: stoyo
Trying to connect to Battle ships online with username stoyo
Welcome to Battleships online!
> Username stoyo is taken, please select another username
Please enter your username: pesho
Trying to connect to Battle ships online with username pesho
> Your username is now pesho
battleships menu> set-username gosho
> Your username is already set to pesho
```

3.4. change-username

Command with one required argument – <new username>. This command can only be executed by a client who is not in game mode, but has only connected to the server (online). When choosing a new username that is already taken, the client remains with his current username, which he originally connected to the server and is informed that the desired new username is already taken. If the client re-enters his current username – he receives a message that he already has this username.

```
Please enter your username: pesho
Trying to connect to Battle ships online with username pesho
Welcome to Battleships online!
> Your username is now pesho
battleships menu> change-username andy
> Username andy is taken, please select another username
battleships menu> change-username tosho
> Username changed. Your username is now tosho
battleships menu> who-am-i
> You are tosho
```

3.5. list-users

Command without arguments, which shows the number of all users connected to the server in the first line, then on each new line prints the current name of the user and his status in the server (whether just browsing or in game mode)

```
> Total users connected now: 3
> desi is HOST
> pesho is ONLINE
> maria is GUEST
```

3.6. list-games

A non-argument command that displays a table of information about the available games that are currently being played and served by the server.

```
battleships menu> list-games
> | NAME                       | CREATOR     | STATUS      | PLAYERS     |
> |----------------------------+-------------+-------------+-------------|
> | the battle of the vikings  | maria       | in progress | 2/2         |
> | my-game                    | pesho       | pending     | 1/2         |
```

3.7. register

A command without arguments, with which a user registers on the server in order to create a game and join as a second (guest) player in already created games. The registration consists in generating a unique key from the server, which is used as a password by the respective user. We will call this key shortly api-key. It is formed by the difference, measured in milliseconds, between the current time and midnight on January 1, 1970, UTC (Coordinated Universal Time) multiplied by a predefined random prime number (in this case we use the number 499). As an improvement, random characters can be added to the key that is returned at registration at the end. This will enhance the security of the key.

Along with this unique key, which identifies each registered user, a hash table entry is created, which has as key the unique api-key and value – a new hash table with key – game name and value – SavedGame class. This SavedGame class is used only as a data structure that stores everything needed for a game so that it can be restored again in case a user wants to complete it, after being saved.

That is, every time a user registers – we create a place to store his saved games that he can do (in case he leaves the server, if he is in game mode with another user or just wants to save a game he plays to see how it would develop if he made another move after reloading it from where he saved it).

```
battleships menu> register
> You are now logged in with api-key: 821763825190034
```

3.8. login

Command with one required argument <api-key>. With this command, a pre-registered user connects to the server. All saved games are loaded on the server in the RAM reserved for this user. These games are loaded from a file in the file system of the server having the same name as its corresponding api-key.

```
battleships menu> login 1234
> No such api-key
battleships menu> login 821763825190034
> You are now logged in with api-key: 821763825190034
```

3.9. create-game

Command with one required argument – <game name>. This command is available for execution only for users who are registered with api-key on the server and are not in the current game at the time of the command invocation. When you try to create a game with a name that is already taken by another game – a message is returned to indicate this and nothing is created. In all other cases, a new game with the submitted name is created, in which the creator is the host, the status of the game is pending and a second player is waited to join. Boats can be placed while waiting (see place commands below)

```
Please enter your username: pesho
Trying to connect to Battle ships online with username pesho
Welcome to Battleships online!
> Your username is now pesho
battleships menu> create-game
> Requires one argument. Usage: create-game <game-name>
battleships menu> create-game "pesho's game"
> You must login or register first
battleships menu> login 821763825190034
> You are now logged in with api-key: 821763825190034
battleships menu> create-game "taken game-name"
> Game "taken game-name" already exists. Choose another game-name
battleships menu> create-game "pesho's game"
> Created game "pesho's game", players 1/2
```

3.10. join-game

Command with one optional parameter – <game name>. In case this optional parameter is missing, the user joins a random pending game. If the argument is set, the player tries to join the game with the name given as the argument. If such a game exists and has the status of pending - the pairing will be successful. Otherwise, the user will be indicated by a message that he has not been able to join such a game. Similar to the create-game command, this command is only available to users who have registered with the api-key on the server. This is because upon successful connection to a game, these users may be disconnected from the server during a multiplayer (2/2) game or try to save the current multiplayer (2/2) game and must have where to store the game if one of the two described events occur. If the command is called by a user who is already in the game, nothing will be executed except returning a message reminding him that there is already an ongoing game for that user.

```
battleships menu> join-game some-game
> You must login or register first
battleships menu> register
> You are now logged in with api-key: 821764375004701
battleships menu> join-game some-game
> There is no such game
battleships menu> join-game
> Joined game "pesho's game". Place your ships!
```

When a player successfully joins a pending game, a message is sent to the host player indicating that the guest player has joined and the game can be started.

```
battleships menu> > maria joined the game. Place your ships!
```

3.11. current-game

Command without parameters. Returns the current game to a user who is registered and is currently in a game as creator or guest. In all other cases, a message is returned from the server indicating that the user is not in the current game.

```
battleships menu> current-game
> No current game
battleships menu> create-game "pesho's game"
> Created game "pesho's game", players 1/2
battleships menu> current-game
> You are currently HOST in pesho's game
```

3.12. save-game

Command without parameters that saves the current game of the user calling it. The command can only be called if the user's current game is in progress. When the user disconnects from the server (executes the disconnect command), this save-game command is executed covertly (without being explicitly called by the user).

```
battleships menu> > maria joined the game. Place your ships!
battleships menu> save-game
> Game "pesho's game" saved
```

If the user already has a game saved with the name of the current game, the game is not saved and a message is sent informing him that the game has not been saved. When saving a game, the SavedGame data structure (class) stores:

- the name of the game;
- the current state of the two boards;
- a bit indicating whether the host player is OK;

- a bit indicating whether the player who saved the game was the host.

This is the minimal information we need to be able to recover a saved game.

3.13. saved-games

Command without parameters that displays the currently saved games of a registered user.

```
battleships menu> saved-games
> | NAME                        | CREATOR     | STATUS      | PLAYERS     |
> |-----------------------------+-------------+-------------+-------------|
> | pesho's game                | host        | saved       | 0/2         |
```

Note that the saved games have 0 players, as they are just data structures used to make resume play possible.

3.14. delete-game

A command with one required argument – <game name> to be deleted. The command can be executed with several arguments – <game 1> <game 2> ... <game n>. When calling with several arguments, the server will try to delete each game with the given name if it finds such a game in the games saved by the user.

```
battleships menu> saved-games
> | NAME                        | CREATOR     | STATUS      | PLAYERS     |
> |-----------------------------+-------------+-------------+-------------|
> | pesho's game                | host        | saved       | 0/2         |
battleships menu> delete-game "some game" "pesho's game" other-game
> Trying to delete selected game(s)
> Could not delete "some game". No such game-name
> Game "pesho's game" was deleted
> Could not delete "other-game". No such game-name
battleships menu> saved-games
> | NAME                        | CREATOR     | STATUS      | PLAYERS     |
> |-----------------------------+-------------+-------------+-------------|
```

3.15. place

Command with 4 required parameters – <[A-J]> <[1-10]> <[UP, RIGHT, DOWN, LEFT]> <[2-5]>. These parameters are used to position a boat on the board.

- <[A-J]>: A to J letter indicating the row of the board. The letter can be uppercase or lowercase (case-insensitive);
- <[1-10]>: a number from 1 to 10, indicating the column of the board;
- <[UP, RIGHT, DOWN, LEFT]>: direction indicating where the boat will be located. The available direction can be uppercase or lowercase or a combination of both (case-insensitive);
- <[2-5]>: the length of the warship, which determines its type.

  2 – DESTROYER, 3 – CRUISER, 4 – BATTLESHIP, 5 – CARRIER.

Sample calls:
- place B 2 RIGHT 2 – puts a ship of length 2, starting cell B2 and direction – right
- place A 1 down 3 – puts a ship of length 3, starting cell A1 and direction – down
- place J 10 Up 4 – puts a ship with a length of 4, starting cell J10 and direction – up
- place e 6 LeFt 5 – puts a ship of length 5, starting cell E6 and direction – left

The ship placement command line interface (CLI) is designed in such way, just to be as flexible as possible. The same boat can be placed in 8 different ways in the general case (if it is not placed in some corner and not clamped by other boats).

When this command is called, checks are performed for:
- missing argument;
- the status of the calling user (whether he / she is a host / guest in the current game);
- all ships already placed;
- all ships of this type have already been placed;
- successfully convert (parse) all arguments, before normalizing to lowercase, to achieve a case-insensitive friendly interface;
- positional arguments (row and column) that go beyond the board;
- valid boat length;
- leaving the boat outside the board;
- placing a boat on an existing ship's seat.

In case each of the above checks has passed successfully – a ship is placed on the board.

Ship placement commands can be executed asynchronously and do not depend on which of the two players turn is or even whether there is a second player. With attack commands, however, things are different (see attack commands below).

```
battleships menu> place b 2 right 2
> GAME STATE:
       YOUR BOARD
   1 2 3 4 5 6 7 8 9 10
   - - - - - - - - - -
A |_|_|_|_|_|_|_|_|_|_|
B |_|X|X|_|_|_|_|_|_|_|
C |_|_|_|_|_|_|_|_|_|_|
D |_|_|_|_|_|_|_|_|_|_|
E |_|_|_|_|_|_|_|_|_|_|
F |_|_|_|_|_|_|_|_|_|_|
G |_|_|_|_|_|_|_|_|_|_|
H |_|_|_|_|_|_|_|_|_|_|
I |_|_|_|_|_|_|_|_|_|_|
J |_|_|_|_|_|_|_|_|_|_|
DESTROYER added
```

```
battleships menu> place e 4 up 4
> GAME STATE:
       YOUR BOARD
   1 2 3 4 5 6 7 8 9 10
   - - - - - - - - - -
A |_|_|_|_|_|_|_|_|_|_|
B |_|X|X|X|_|_|_|_|_|_|
C |_|_|_|X|_|_|_|_|_|_|
D |_|_|_|X|_|_|_|_|_|_|
E |_|_|_|X|_|_|_|_|_|_|
F |_|_|_|_|_|_|_|_|_|_|
G |_|_|_|_|_|_|_|_|_|_|
H |_|_|_|_|_|_|_|_|_|_|
I |_|_|_|_|_|_|_|_|_|_|
J |_|_|_|_|_|_|_|_|_|_|
BATTLESHIP added
```

```
battleships menu> place e 6 left 3
> GAME STATE:
       YOUR BOARD
   1 2 3 4 5 6 7 8 9 10
   - - - - - - - - - -
A |_|_|_|_|_|_|_|_|_|_|
B |_|X|X|X|_|_|_|_|_|_|
C |_|_|_|X|_|_|_|_|_|_|
D |_|_|_|X|_|_|_|_|_|_|
E |_|_|_|X|_|_|_|_|_|_|
F |_|_|_|_|_|_|_|_|_|_|
G |_|_|_|_|_|_|_|_|_|_|
H |_|_|_|_|_|_|_|_|_|_|
I |_|_|_|_|_|_|_|_|_|_|
J |_|_|_|_|_|_|_|_|_|_|
There already is a ship placed there
```

```
battleships menu> place i 10 left 5
> GAME STATE:
       YOUR BOARD
   1 2 3 4 5 6 7 8 9 10
   - - - - - - - - - -
A |_|_|_|_|_|_|_|_|_|_|
B |_|X|X|X|_|_|_|_|_|_|
C |_|_|_|X|_|_|_|_|_|_|
D |_|_|_|X|_|_|_|_|_|_|
E |_|_|_|X|_|_|_|_|_|_|
F |_|_|_|_|_|_|_|_|_|_|
G |_|_|_|_|_|_|_|_|_|_|
H |_|_|_|_|_|_|_|_|_|_|
I |_|_|_|_|_|X|X|X|X|X|
J |_|_|_|_|_|_|_|_|_|_|
CARRIER added
```

```
battleships menu> place i 3 right 3
> GAME STATE:
       YOUR BOARD
   1 2 3 4 5 6 7 8 9 10
   - - - - - - - - - -
A |_|_|_|_|_|_|_|_|_|_|
B |_|X|X|X|_|_|_|_|_|_|
C |_|_|_|X|_|_|_|_|_|_|
D |_|_|_|X|_|_|_|_|_|_|
E |_|_|_|X|_|_|_|_|_|_|
F |_|_|_|_|_|_|_|_|_|_|
G |_|_|_|_|_|_|_|_|_|_|
H |_|_|_|_|_|_|_|_|_|_|
I |_|_|X|X|X|X|X|X|X|X|
J |_|_|_|_|_|_|_|_|_|_|
CRUISER added
```

```
battleships menu> place j 10 down 2
> GAME STATE:
       YOUR BOARD
   1 2 3 4 5 6 7 8 9 10
   - - - - - - - - - -
A |_|_|_|_|_|_|_|_|_|_|
B |_|X|X|X|_|_|_|_|_|_|
C |_|_|_|X|_|_|_|_|_|_|
D |_|_|_|X|_|_|_|_|_|_|
E |_|_|_|X|_|_|_|_|_|_|
F |_|_|_|_|_|_|_|_|_|_|
G |_|_|_|_|_|_|_|_|_|_|
H |_|_|_|_|_|_|_|_|_|_|
I |_|_|X|X|X|X|X|X|X|X|
J |_|_|_|_|_|_|_|_|_|_|
Out of border [DOWN]!
```

3.16. place-all

Command without arguments, which randomly places all necessary ships on the board of the user who called it. In case the user has already placed all the ships, a message is returned telling him that he can now notify the other player, if any, and start an attack process between the two players. If, before calling this command, the user has already placed several of the available ships on the board, the command will add the remaining boats randomly, keeping the positions of the ships already available on the board.

The placement of the ships is absolutely stochastic in each scenario, which makes it extremely difficult to calculate the time complexity of this function, but with an arsenal of 30 cells occupied by ships of 100 cells water part – we can assume that the command will be executed quickly enough (in constant time) in the average case.

```
battleships menu> place-all
> All ships were added, you are ready to attack!
GAME STATE:
       YOUR BOARD
   1 2 3 4 5 6 7 8 9 10

   _ _ _ _ _ _ _ _ _ _
A |_|_|X|X|X|_|_|_|_|_|
B |_|_|_|_|_|X|X|_|_|_|
C |_|_|_|_|_|_|X|_|_|_|
D |X|X|X|X|_|_|X|_|_|_|
E |_|X|X|X|_|_|_|_|X|_|
F |_|X|X|X|_|_|_|_|X|_|
G |_|X|_|X|_|X|X|_|X|_|
H |_|X|_|X|_|X|_|_|X|_|
I |_|_|_|_|_|X|_|_|_|_|
J |_|_|_|_|_|X|_|_|_|_|
All ships are placed. Type "Start" to start the game
```

3.17. start

Command without arguments that is used to send a message to the other player. The message indicates that the opponent is ready to start attacking (he has already placed all his ships on the board). If there is no other player, the command will not be executed.

```
battleships menu> start
> Cannot start a single player mode
battleships menu> > maria joined the game. Place your ships!
battleships menu> start
> You are ready to start the battle
battleships menu> attack a 1
> One of the players did not press start
battleships menu> > maria is ready to start the battle
battleships menu> attack a 1
```

3.18. attack

Command with two mandatory parameters – <[A-J]> <[1-10]>, which are analogous to the first two parameters of the place command.

- <[A-J]>: A to J letter indicating the row of the board. The letter can be uppercase or lowercase (case-insensitive);
- <[1-10]>: a number from 1 to 10, indicating the column of the board;

Sample calls:
- attack b 2 – attack the cell of row B and column 2 (the cell with coordinates B2)
- attack J 10 – attack the cell of row J and column 10 (bottom and rightmost cell)

When the command is called, checks are performed for:
- missing argument;
- the status of the calling user (whether he is in the current game);
- the status of the current game (whether the current game is in progress);
- readiness of both players (whether both players in the current game have executed the start command on the server successfully);
- the player's order (whether it is the server's command call to attack the server);
- validity of the attack and the result of it, which will be needed to respond to both players.

```
battleships menu> attack a 1
> GAME STATE:
       YOUR BOARD
   1 2 3 4 5 6 7 8 9 10
   _ _ _ _ _ _ _ _ _ _
A |_|_|X|X|X|_|_|_|_|_|
B |_|_|_|_|_|X|X|_|_|_|
C |_|_|_|_|_|_|X|_|_|_|
D |X|X|X|X|_|_|X|_|_|_|
E |_|X|X|X|_|_|_|_|X|_|
F |_|X|X|X|_|_|_|_|X|_|
G |_|X|_|X|_|X|X|_|X|_|
H |_|X|_|X|_|X|_|_|X|_|
I |_|_|_|_|_|X|_|_|_|_|
J |_|_|_|_|_|X|_|_|_|_|
      ENEMY BOARD
   1 2 3 4 5 6 7 8 9 10
   _ _ _ _ _ _ _ _ _ _
A |-|_|_|_|_|_|_|_|_|_|
B |_|_|_|_|_|_|_|_|_|_|
C |_|_|_|_|_|_|_|_|_|_|
D |_|_|_|_|_|_|_|_|_|_|
E |_|_|_|_|_|_|_|_|_|_|
F |_|_|_|_|_|_|_|_|_|_|
G |_|_|_|_|_|_|_|_|_|_|
H |_|_|_|_|_|_|_|_|_|_|
I |_|_|_|_|_|_|_|_|_|_|
J |_|_|_|_|_|_|_|_|_|_|
You missed
```

```
battleships menu> attack d 5
> It's not your turn
battleships menu> > GAME STATE:
       YOUR BOARD
   1 2 3 4 5 6 7 8 9 10
   _ _ _ _ _ _ _ _ _ _
A |_|_|X|*|X|_|_|_|_|_|
B |_|_|_|_|_|X|X|_|_|_|
C |_|_|_|_|_|_|X|_|_|_|
D |X|X|X|X|_|_|X|_|_|_|
E |_|X|X|X|_|_|_|_|X|_|
F |_|X|X|X|_|_|_|_|X|_|
G |_|X|_|X|_|X|X|_|X|_|
H |_|X|_|X|_|X|_|_|X|_|
I |_|_|_|_|_|X|_|_|_|_|
J |_|_|_|_|_|X|_|_|_|_|
      ENEMY BOARD
   1 2 3 4 5 6 7 8 9 10
   _ _ _ _ _ _ _ _ _ _
A |-|_|_|_|_|_|_|_|_|_|
B |_|_|_|_|_|_|_|_|_|_|
C |_|_|_|_|_|_|_|_|_|_|
D |_|_|_|_|_|_|_|_|_|_|
E |_|_|_|_|_|_|_|_|_|_|
F |_|_|_|_|_|_|_|_|_|_|
G |_|_|_|_|_|_|_|_|_|_|
H |_|_|_|_|_|_|_|_|_|_|
I |_|_|_|_|_|_|_|_|_|_|
J |_|_|_|_|_|_|_|_|_|_|
maria's last turn: A4
```

```
battleships menu> attack d 5
> GAME STATE:
       YOUR BOARD
   1 2 3 4 5 6 7 8 9 10
   _ _ _ _ _ _ _ _ _ _
A |_|_|X|*|X|_|_|_|_|_|
B |_|_|_|_|_|X|X|_|_|_|
C |_|_|_|_|_|_|X|_|_|_|
D |X|X|X|X|_|_|X|_|_|_|
E |_|X|X|X|_|_|_|_|X|_|
F |_|X|X|X|_|_|_|_|X|_|
G |_|X|_|X|_|X|X|_|X|_|
H |_|X|_|X|_|X|_|_|X|_|
I |_|_|_|_|_|X|_|_|_|_|
J |_|_|_|_|_|X|_|_|_|_|
      ENEMY BOARD
   1 2 3 4 5 6 7 8 9 10
   _ _ _ _ _ _ _ _ _ _
A |-|_|_|_|_|_|_|_|_|_|
B |_|_|_|_|_|_|_|_|_|_|
C |_|_|_|_|_|_|_|_|_|_|
D |_|_|_|_|X|_|_|_|_|_|
E |_|_|_|_|_|_|_|_|_|_|
F |_|_|_|_|_|_|_|_|_|_|
G |_|_|_|_|_|_|_|_|_|_|
H |_|_|_|_|_|_|_|_|_|_|
I |_|_|_|_|_|_|_|_|_|_|
J |_|_|_|_|_|_|_|_|_|_|
You hit a CRUISER at position D5
```

For a valid attack is considered an attack which is within the limits of the board, whether there was a ship on it and if there was, whether it has already been attacked at this position or even sunk completely. In other words, only an attack that has coordinates beyond the board or that has

invalid arguments (number / type) is considered invalid. In the case of an invalid attack, the player's order does not change and he has the opportunity to attack again until he makes a valid attack.

3.19. hacks

A command without arguments that acts as an intermediate command or a hint command. When called, it shows how to use a cheat command (attack-all <args> usage) or in other words – a command that is not typical for a given game, but is implemented in order to more quick testing during game demonstration. This hacks command will only be valid for invocation by a user who is in game mode.

```
battleships menu> hacks
> attack-all <[A-J]> {<[A-J]>}...
```

3.20. attack-all

Command with at least one required argument – <[A-J]> and many optional. The arguments of this command indicate the letters of the lines to be completely attacked. It is not a legal command, as there should be no such command in the original game, but it exists here only for the purpose of faster presentation of the game. It can be used to finalize a game much faster by simulating multiple single attacks. Behind this command are numerous calls to the attack command with two arguments, which we discussed above.

this player:

```
battleships menu> attack-all a b c d g j
> GAME STATE:
       YOUR BOARD
   1 2 3 4 5 6 7 8 9 10
   _ _ _ _ _ _ _ _ _ _
A |-|_|_|_|_|_|X|X|_|_|
B |X|_|_|_|_|_|_|_|_|_|
C |X|_|X|X|_|_|_|_|X|_|
D |X|_|_|_|*|_|_|_|X|X|
E |X|_|_|_|X|_|_|_|X|X|
F |X|_|_|_|X|_|X|_|X|X|
G |_|_|_|_|_|X|X|_|_|_|
H |_|_|_|_|_|X|X|_|_|_|
I |_|_|_|X|X|X|X|_|_|_|
J |_|_|_|_|_|_|_|X|X|_|
      ENEMY BOARD
   1 2 3 4 5 6 7 8 9 10
   _ _ _ _ _ _ _ _ _ _
A |-|-|X|X|X|-|-|-|-|-|
B |-|-|-|-|-|X|X|-|-|-|
C |-|-|-|-|-|-|X|-|-|-|
D |X|X|X|X|-|-|X|-|-|-|
E |_|_|_|_|_|_|_|_|_|_|
F |_|_|_|_|_|_|_|_|_|_|
G |-|X|-|X|-|X|X|-|X|-|
H |_|_|_|_|_|_|_|_|_|_|
I |_|_|_|_|_|_|_|_|_|_|
J |-|-|-|-|-|X|-|-|-|-|
You hit 16 time(s) and destroyed 5 ship(s)
```

other player:

```
battleships menu> > GAME STATE:
       YOUR BOARD
   1 2 3 4 5 6 7 8 9 10
   _ _ _ _ _ _ _ _ _ _
A |-|-|*|*|*|-|-|-|-|-|
B |-|-|-|-|-|*|*|-|-|-|
C |-|-|-|-|-|-|*|-|-|-|
D |*|*|*|*|-|-|*|-|-|-|
E |_|X|X|X|_|_|_|_|X|_|
F |_|X|X|X|_|_|_|_|X|_|
G |-|*|-|*|-|*|*|-|*|-|
H |_|X|_|X|_|X|_|_|X|_|
I |_|_|_|_|_|X|_|_|_|_|
J |-|-|-|-|-|*|-|-|-|-|
      ENEMY BOARD
   1 2 3 4 5 6 7 8 9 10
   _ _ _ _ _ _ _ _ _ _
A |-|_|_|_|_|_|_|_|_|_|
B |_|_|_|_|_|_|_|_|_|_|
C |_|_|_|_|_|_|_|_|_|_|
D |_|_|_|_|X|_|_|_|_|_|
E |_|_|_|_|_|_|_|_|_|_|
F |_|_|_|_|_|_|_|_|_|_|
G |_|_|_|_|_|_|_|_|_|_|
H |_|_|_|_|_|_|_|_|_|_|
I |_|_|_|_|_|_|_|_|_|_|
J |_|_|_|_|_|_|_|_|_|_|
You were massively attacked and hit 16 time(s) and lost 5 ship(s)!
```

3.21. display

Command without arguments, which shows the status of the current game. In case the game is pending and there is no opponent yet (guest), only the board of the caller is presented. If the game is in progress and there is a second game, both boards are presented, as the opponent's is coded,

i.e. cells with successful attacks on enemy ships are marked with an "X", and cells with missing ships that are attacked (unsuccessful attacks) are marked with a dash "–".

```
battleships menu> display
> GAME STATE:
      YOUR BOARD
   1 2 3 4 5 6 7 8 9 10
   – – – – – – – – – –
A |-|-|*|*|*|-|-|-|-|-|
B |-|-|-|-|-|*|*|-|-|-|
C |-|-|-|-|-|-|*|-|-|-|
D |*|*|*|*|-|-|*|-|-|-|
E |_|X|X|X|_|_|_|_|X|_|
F |_|X|X|X|_|_|_|_|*|-|
G |-|*|-|*|-|*|*|-|*|-|
H |_|X|_|X|_|X|_|_|X|_|
I |_|_|_|_|_|X|_|_|_|_|
J |-|-|-|-|-|*|-|-|-|-|
      ENEMY BOARD
   1 2 3 4 5 6 7 8 9 10
   – – – – – – – – – –
A |-|-|-|-|-|-|X|X|-|-|
B |_|_|_|_|_|_|_|_|_|_|
C |X|-|X|X|-|-|-|-|X|-|
D |_|_|_|_|X|_|_|_|_|_|
E |_|_|_|_|_|_|_|_|_|_|
F |_|_|_|_|_|_|_|_|_|_|
G |_|_|_|_|_|_|_|_|_|-|
H |-|-|-|-|-|X|X|-|-|-|
I |-|-|-|X|X|X|X|-|-|-|
J |-|-|-|-|-|-|-|X|X|-|
```

3.22. disconnect

Command without arguments. Despite the lack of arguments and the fact that we consider it last, this command is one of the most complex due to business logic that runs on the server. When the client invokes this command, no matter what business logic will be executed on the server and what message will be returned from the server, the client who called the command will print the message on its console and terminate its program (stop executing both customer threads).

The business logic of this server command is more interesting. The first thing to check is whether the user who leaves the server was in a game mode that has a status in progress, i.e. whether the game was multiplayer (2/2) and had an opponent. If this is not true and the game was not multiplayer (2/2), the remote socket address of the socket is removed from the socket channel with which the user connected to the server and the game is not saved.

But if the user has been in a current game that has been in progress and has executed the disconnect command, then the game is transformed from in progress to pending, updating the status of the opposing player (if he was a guest becomes the host), update the socket channels and the players' boards, as well as the correct order, and only then the game is saved in the RAM with the saved games. In this way, after the recording, we will have extracted all the necessary information from the game so that we can recover it correctly when the user logs in again with his api-key.

All saved games from the heap reserved for the client are then saved in a text file on the server, which has the same name as its corresponding api-key.

Heap memory is freed up so that it does not take up space on the server and so that requests that take linear time complexity can be executed faster.

When one of the players leaves, the opposing player is shown a message indicating that he has remained in the single player mode.

this player:

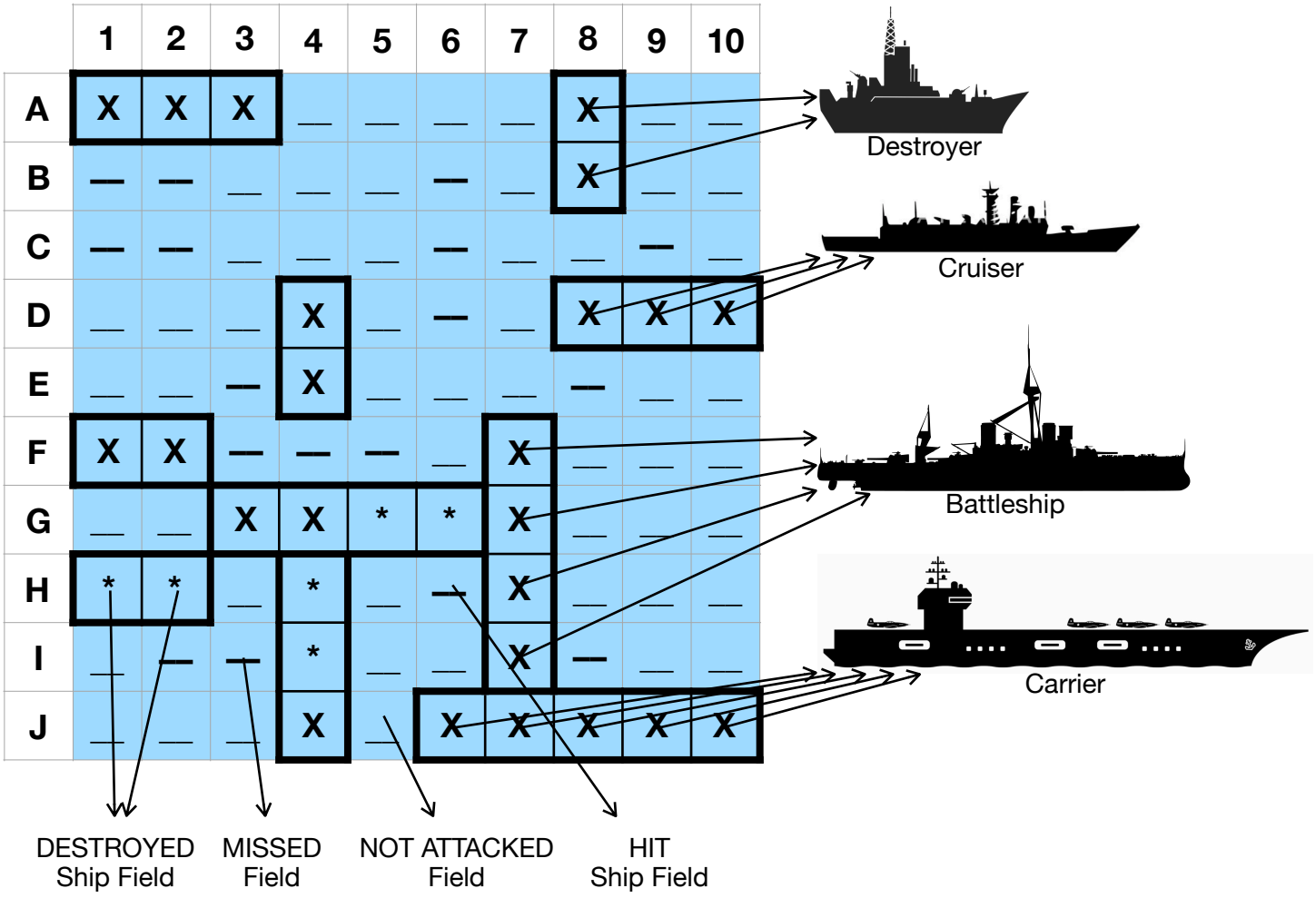
```
battleships menu> disconnect
See you soon on the battlefield pesho
Disconnected from server

Process finished with exit code 0
```

other player:

```
battleships menu> > pesho exit the game. Waiting for another player to join...
battleships menu>
```

4. Data representation

4.1. Board



The Battleships Online board is presented as a Board data structure (class) with many nested dependencies on other classes. The board is a square matrix of Field classes that have a data member enumerating the FieldType (enum) and Ship class. In addition to this matrix, the board also has a data member - a hash table with key number and value list of ships and totalDestroyedShips of integer type. The number from the hash table key represents the length of the ship, and the value is the class that represents the ship.

The Ship class contains a ShipType member of enumerable type, shipDeck of integer type, and a list of Coordinate variables. Coordinate variables have two data members of integer type, indicating the coordinates of which this ship is located. shipDeck is a counter that shows how many of the ship's cells remain healthy (not attacked). Obviously when this counter becomes equal to 0 - the ship will be completely submerged.

* As can be seen from the figure above, when we have a ship located on the board, several adjacent cells point to the same class Ship, which is a ship. In this sense, the list of coordinates on which the ship is located is an essential member of the Ship class, as it will allow us to correctly recover the board from a JSON string. This is exactly what happens in the adjustBoard function of the LoadGame class – for each of the ships on the board, for each of its coordinates, it is redirected to point to the reference of the corresponding Ship class.

The presence of many nested dependencies in the Board also requires the use of mocks, which we deal with dependencies in unit testing.

4.2. Player

The player is represented by the Player class, which has a String username for his username, a String id for the server ID that represents the socketAddress he connected to, and a board (Board class) that we looked at. above in 4.1. and the SocketChannel, which we need to know which server to send a message to if necessary.

4.3. Game

The game is represented by the Game class, which has a data member of two players of the Player type discussed above in 4.2, a name of the game name of the String type, two variables hostReady and a guestReady of the boolean type indicating whether your ships and are ready to attack, game status status of enumerable type GameStatus, boolean variable hostTurn, which indicates whether the player's host is in order and board savedBoardSecondPlayer, which points to the board in some state if the game is loaded from a saved game or points to null if the game is newly created.
When a user tries to join a game, the first thing the Game method joins is to check if savedBoardSecondPlayer does not point to null, if it does not point, it assigns that board to which it points to the joining player, if it points to null. creates a new board for it (ie the game is not loaded by a saved game).

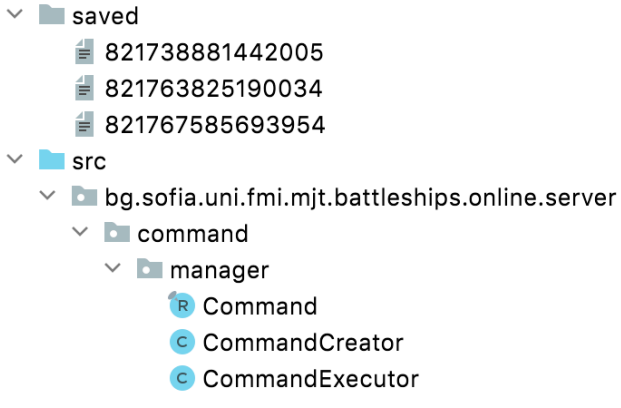
4.4. Storage

The Storage is a class in which all the data needed on the server are combined in order to implement the business logic of the game Battleships Online. These are two hash tables socketAddressUserIn and apiKeySavedGames.

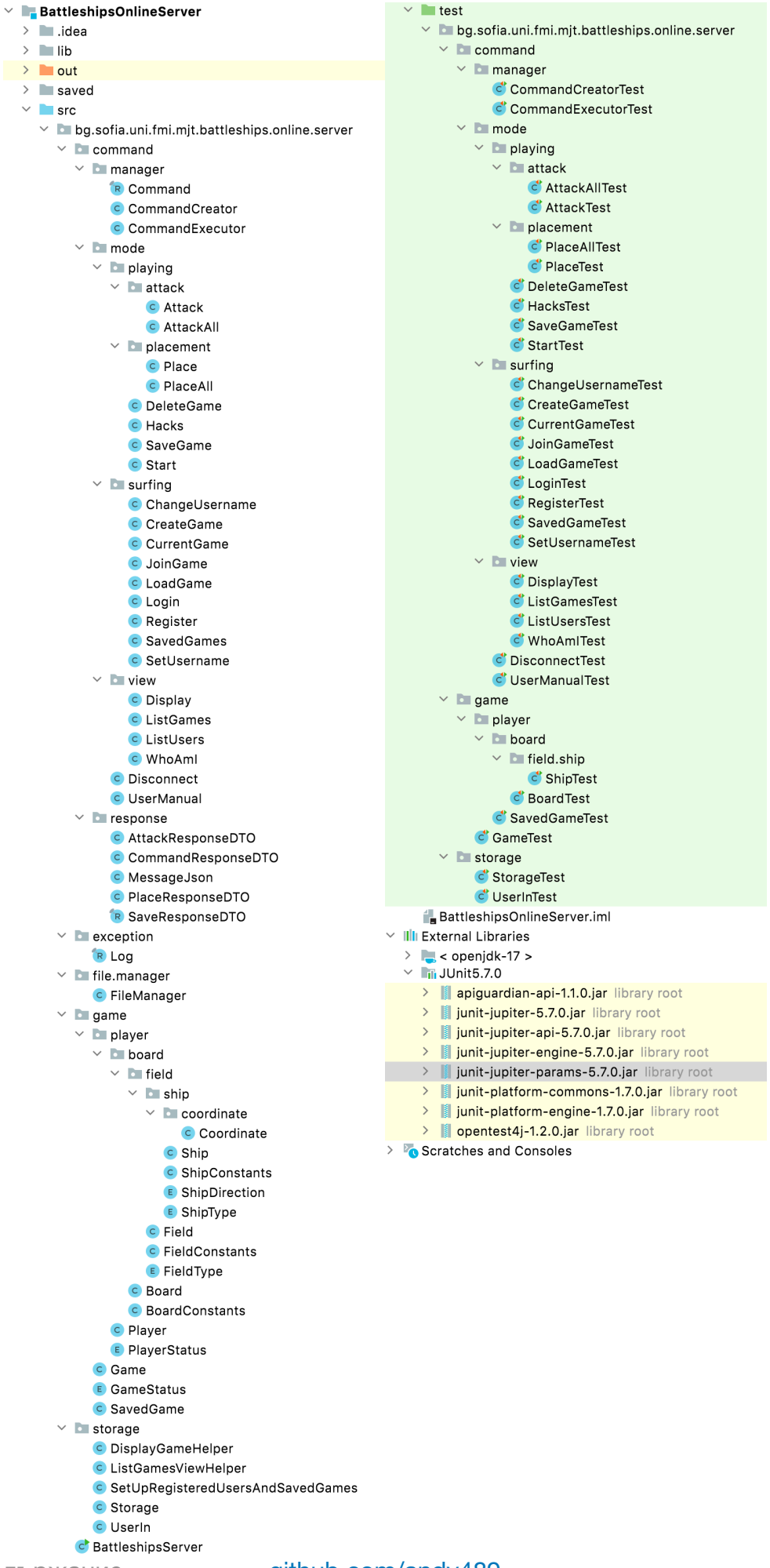
socketAddressUserIn: uses as a key String, which represents the socketAddress of each connection to the server (we use it for id), and for value uses UserIn class, which represents all the necessary information for a user who is currently connected to the server. The UserIn class has data for the member: username username of type String, user status (status) of enumerable type PlayerStatus, apiKey key of type String and class Game, which we discussed above in 4.3

apiKeySavedGames: uses a String key, which is an apiKey key of a user registered on the server, and uses another hash table with a String game name key and SavedGame value structure, which we discussed above.

IMPORTANT! In the structure of the BattleshipsServer class responsible for the server part, the CommandExecutor class with an empty Storage class is initialized. This CommandExecutor takes care of the execution of the server's commands and in its initialization a static method of the SetUpRegisteredUsersAndSavedGames class is called, which reads from the server's file system all files from the saved folder, named with apiKey keys of the users registered on the server and there are stored hash tables of saved games in JSON format. With this information, the server updates its datastore at startup so that there is no data loss in case we need to restart it.



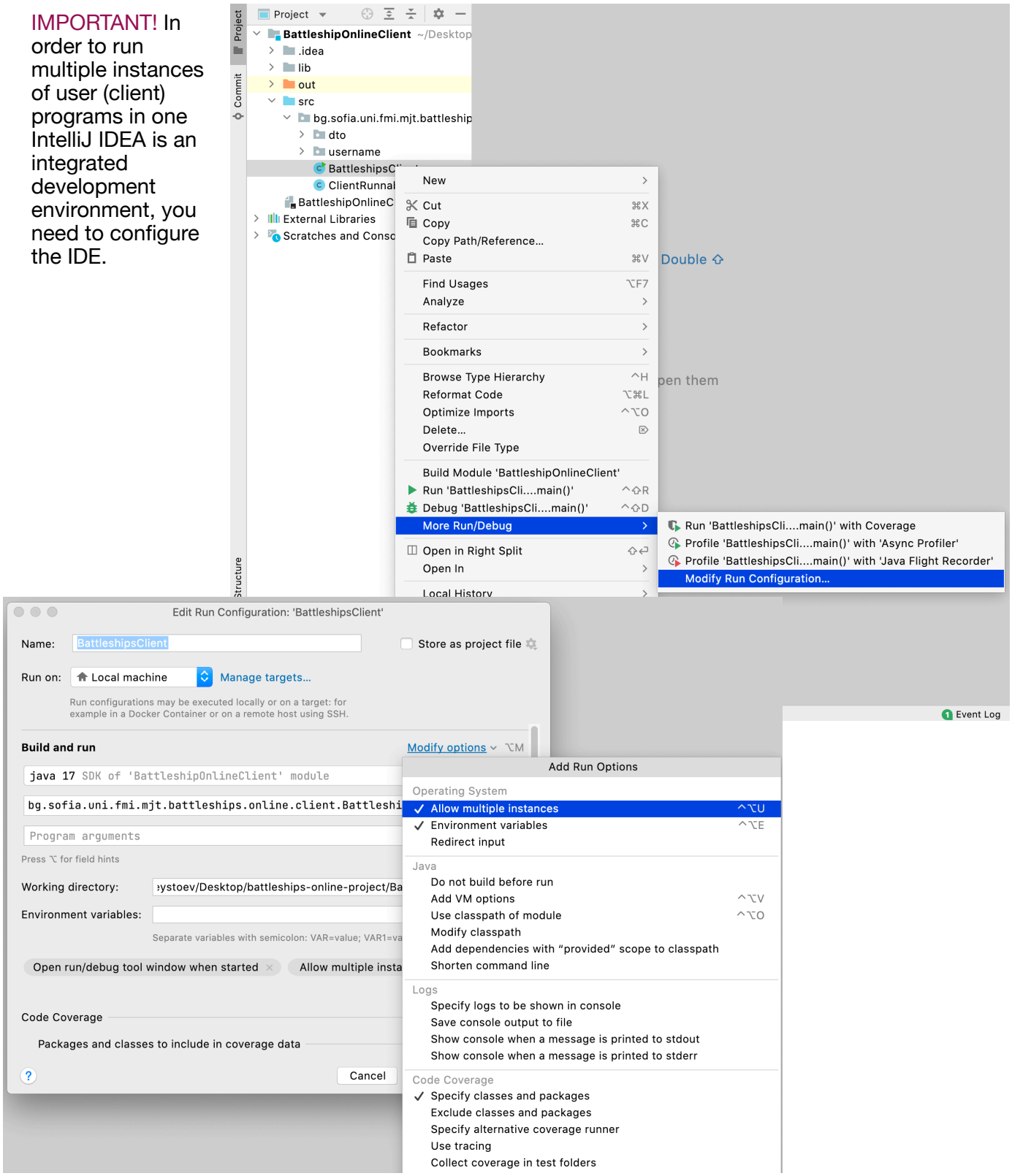
5. Package and class structure

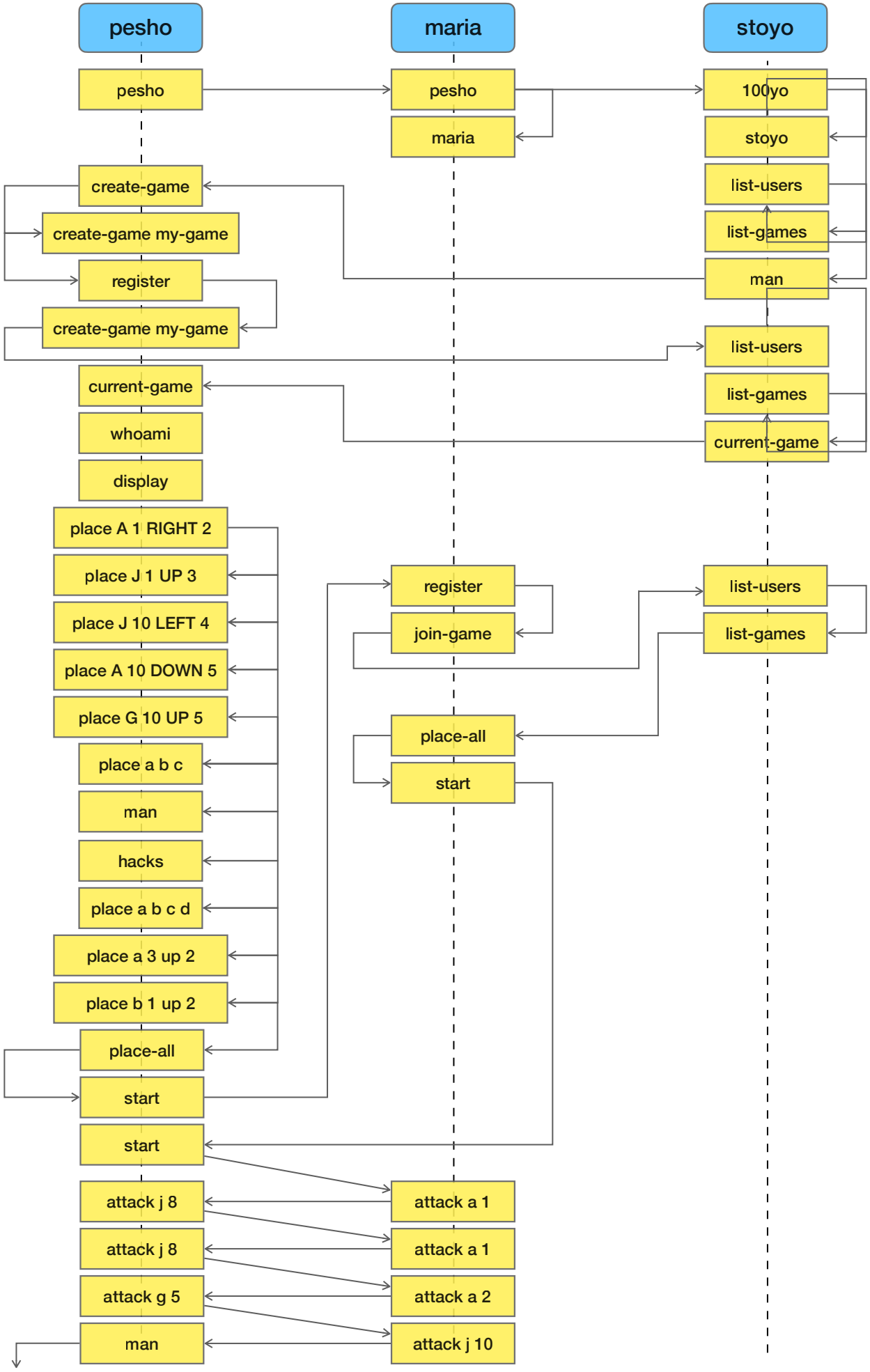


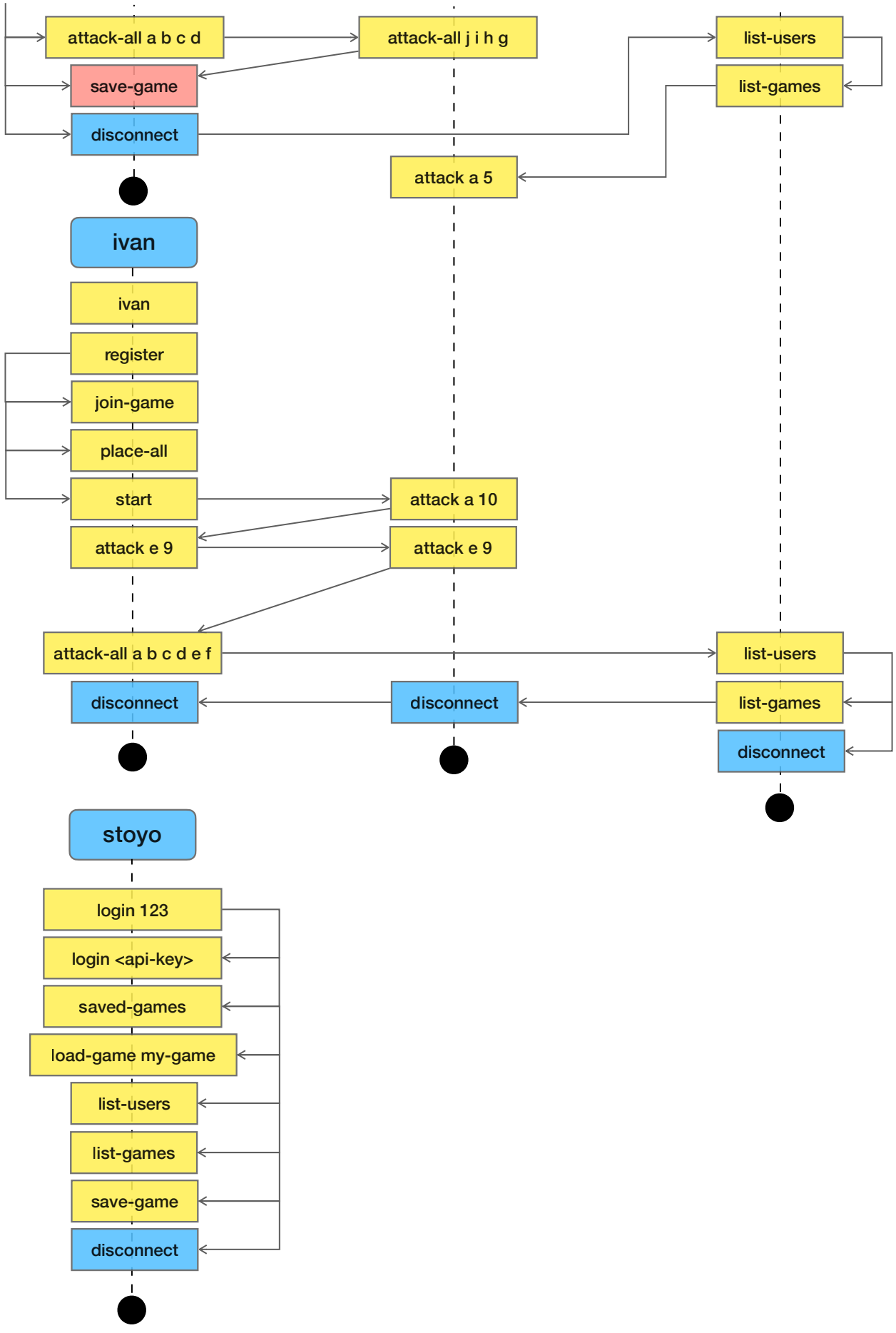
6. Примерен тестови сценарии

The test scenario is a pseudo-sequence diagram that only aims to guide the presentation of the project. The chart borrows some notations from sequence charts, but in no case can it be considered as such. This is because in our case the different users are parallel programs, but for convenience they are shown as a von Neumann sequence. This sequence is a representation of the sequence we want to simulate. Its purpose is to cover some key cases and problems that are solved through architectural, structural and design solutions before and during the implementation of the game.

IMPORTANT! In order to run multiple instances of user (client) programs in one IntelliJ IDEA is an integrated development environment, you need to configure the IDE.




pesho
maria
stoyo
pesho
pesho
maria
100yo
stoyo
list-users
list-games
man
list-users
list-games
current-game
create-game
create-game my-game
register
create-game my-game
current-game
whoami
display
place A 1 RIGHT 2
place J 1 UP 3
place J 10 LEFT 4
place A 10 DOWN 5
place G 10 UP 5
place a b c
man
hacks
place a b c d
place a 3 up 2
place b 1 up 2
place-all
start
register
join-game
list-users
list-games
place-all
start
start
attack j 8
attack j 8
attack g 5
man
attack a 1
attack a 1
attack a 2
attack j 10


attack-all a b c d
attack-all j i h g
list-users
save-game
list-games
disconnect
attack a 5
ivan
ivan
register
join-game
place-all
start
attack a 10
attack e 9
attack e 9
attack-all a b c d e f
list-users
disconnect
disconnect
list-games
disconnect
stoyo
login 123
login <api-key>
saved-games
load-game my-game
list-users
list-games
save-game
disconnect