Battlesnake

1. Preface

Battlesnake.io was created to have fun while programming, learn new skills and technologies. It is an official competition that takes place once a year in the US, where people get together and write bots to play the game Snake to compete in a tournament environment. Tasks for this assignment will mainly test your algorithmic and problem-solving capabilities using the Battlesnakes.io game engine.

2. Tech guide

2.1 Gradle

Gradle commands can be executed in several different ways:

- When working on Windows command line or PowerShell: *gradlew.bat <command>*
- When working on MacOS, Linux or Windows 10 Shell: *./gradlew <command>*

2.2 Docker

If you do not have Docker installed and you are not familiar with Docker, then head over to https://docs.docker.com/get-started/ and install the latest version of Docker.

2.3 Annotation processor

IntelliJ IDEA:

- Install the Lombok plugin.
- Enable the annotation processor:

File->Other Settings->Default Settings
Expand Build, Execution, Deployment
Expand Compiler
In Annotation Processors check Enable annotation processing

For other IDE's please consult your IDE manual.

2.3 Assignment setup

The gradle tasks you'll need, working from the assignment directory:
gradlew bootRun – runs the application
gradlew test - runs the tests

2.4 The Game Engine

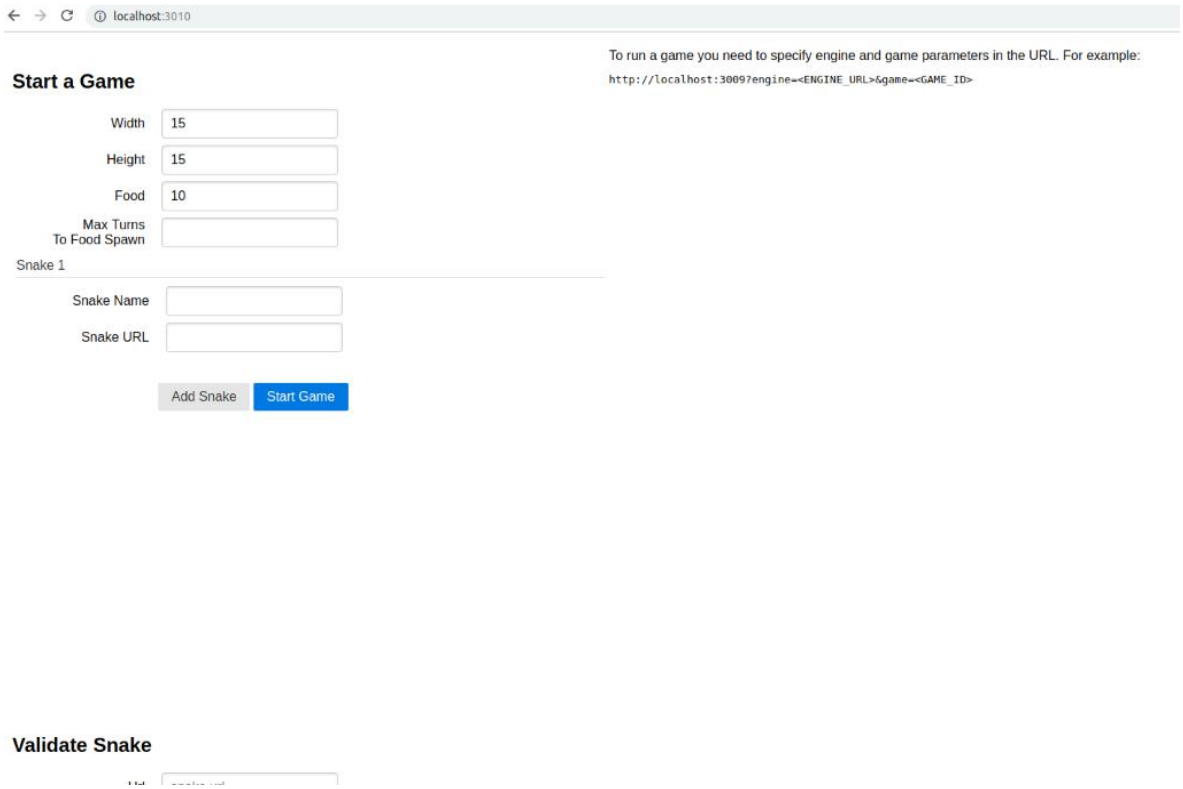
The game engine can be run as a Docker container, through the command line, on Linux with:

docker run --net="host" -it --rm siimveskilt/battlesnakes

And on Windows and Mac with:

docker run -it --rm -p 3010:3010 -p 3005:3005 -p 3009:3009 siimveskilt/battlesnakes

Once the Docker container is up and running you should be able to access the engine when visiting: http://localhost:3010 with a modern browser, and you should see a page like this:



When the application is running, three Snake controllers will be available for Linux at the following endpoints:

http://localhost:8090/random

http://localhost:8090/smart

http://localhost:8090/genius

And for Windows and Mac at:

http://host.docker.internal:8090/random

http://host.docker.internal:8090/smart

http://host.docker.internal:8090/genius

2.5 Starting a new game

On the left side of the game engine screen you can see the interface for creating new games. The required fields are the snakes' name and the URL where the snake can be reached (the controller's endpoint). Clicking "Start Game" will run a game. You can also add snakes to see how good they are at battling against each other.

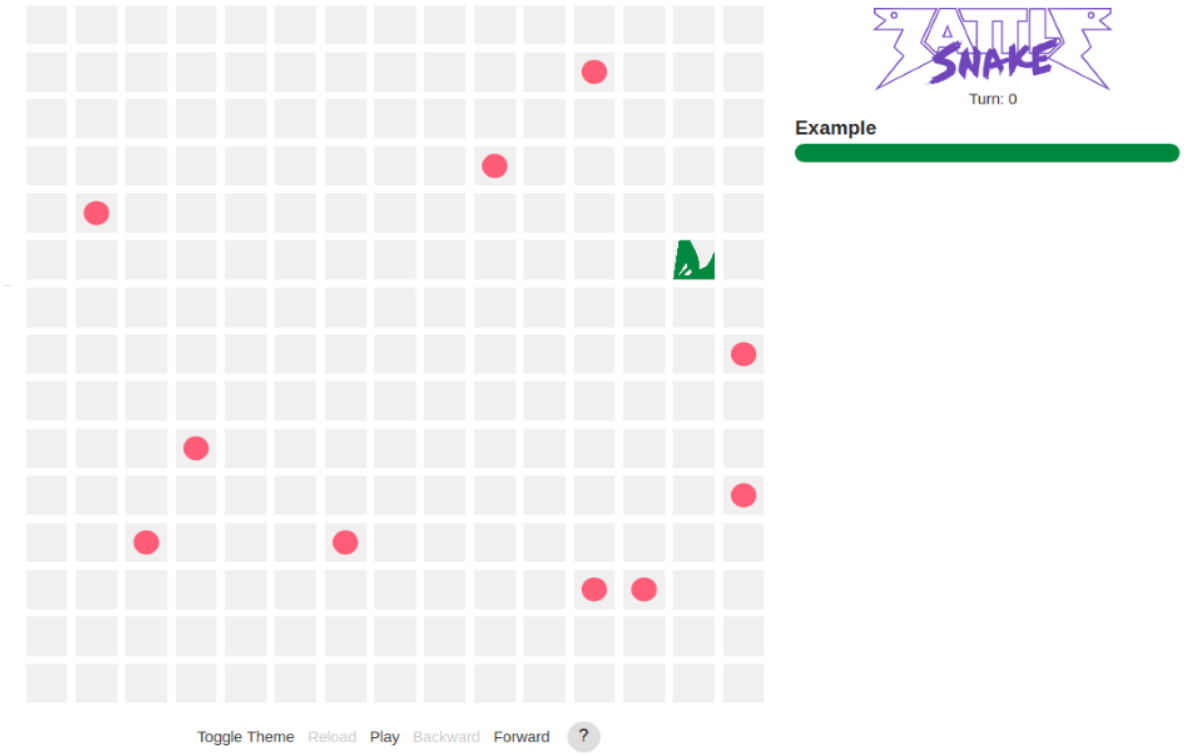
2.6 Validating the controller

Before starting with the assignment, validate the connection between the Application and the Engine by using the snake validator found in the bottom of the game engine screen. All good:


```
{
    "StartStatus": {
        "Message": "Perfect",
        "Errors": [],
        "time": 1,
        "raw": "{\"color\":\"#188936\",\"headType\":\"evil\",\"tailType\":\"bolt\"}",
        "statusCode": 200,
        "score": {
            "checksPassed": 4,
            "checksFailed": 0
        }
    },
    "MoveStatus": {
        "Message": "Perfect",
        "Errors": [],
        "time": 0,
        "raw": "{\"move\":\"left\"}",
        "statusCode": 200,
        "score": {
            "checksPassed": 4,
            "checksFailed": 0
        }
    },
    "EndStatus": {
        "Message": "Perfect",
        "Errors": [],
        "time": 0,
        "raw": "{}",
        "statusCode": 200,
        "score": {
            "checksPassed": 4,
            "checksFailed": 0
        }
    },
    "PingStatus": {
        "Message": "Perfect",
        "Errors": [],
        "time": 0,
        "raw": "{}",
        "statusCode": 200,
        "score": {
            "checksPassed": 4,
            "checksFailed": 0
        }
    }
}
```


If the validation fails, or you cannot get anything to show up at http://localhost:3010, try to answer the following questions:
"Did I run the Application and is the Application running?"
"Did I run the game engine with Docker installed using the correct command and is the Docker container running?"

When everything is working as expected, and a game has been started, a replay option window will appear on the right side:



A replay will start when you click on the "Play" button located under the grid.

3. Assignment

The general rules of the game:

- The world is a grid.
- The snake can only travel orthogonally along this grid.
- This world has a border that kills the snake on contact.
- The snake cannot stop moving.
- If the snake runs into itself or other snakes, it dies.
- Every time the snake eats, it grows longer.
- The goal is to grow as long as possible.

First, look at the example controllers and run a game using the “random” snake controller. Replay the game to see what happened.

3.1 Tasks

Start by reading the *CHANGELOG.md* file located in this assignment folder.

1) Run "*gradlew test*". Tests are failing. Among the fails you will find:

com.battle.snakes.util.SnakeUtilTest > getRandomMove() FAILED

The person writing the tests failed to even write this test. **Write this test and finish the *getRandomMove* method located in the *SnakeUtil* class. Remember, a passing test does not equal to a good test.**

2) Run the *RandomSnakeController*'s snake through the game engine. It does not look very random. **Finish the *getAllowedMoves* method in the *SnakeUtil* class to fix this Snake. Make sure it does not collide with other snakes or itself (unless it is trapped).**

3) Run the snake controlled with the *SmartSnakeController* through the game engine. **Finish the rest of the methods in the *SnakeUtil* class to make the “smart” snake smart.** It is a lot better than the “random” snake. But not really that good for planning more than a step.

4) Run the snake controlled with the *GeniusSnakeController* through the game engine. **Make this snake as good as you can at playing the game. It should be better than the "smart" snake. Bonus points will be awarded for writing at least 3 tests for your "genius" snake, if they are meaningful.** Be warned though, copying other people's solutions will get you no points. Reading about what other people have done and incorporating ideas is encouraged. Remember, the goal here is to create a slightly improved "smart" snake, not solve the many-body problem.