**Exercise 2:**

**Details:**

- Nizan Naor 313546822 nizan.naor11@gmail.com

- Raz Olewsky 315341396 razol@mta.ac.il

The system is combined of 3 jar files:

1) **DTO** jar:

details from the engine system which are transferred to the graphic user interface as Strings and Objects.

Thanks to that, we can show these details to the user on the screen, without having the ability to update them.

2) **SystemGUI** jar:

Where all the graphic user interface is managed.

Hierarchy:

App controller includes Header controller and Body controller.

Body controller includes Details controller, NewExecution controller and Results controller.

Details controller includes all the data that is shown to the user about the World that is loaded from the current xml file (actions, environments, entities, properties, etc.).

NewExecution controller manages user inputs (entities population and environments values) that will be the initialized values for the new execution.

Results controller includes executions list (all the simulations that have been executed by the user), simulation details (status, current ticks, and current seconds.

In there the user can choose to stop, resume, or pause the simulation, and the user can see the current population for each entity in runtime.

In addition, there's also the simulation results which would be shown after the chosen execution is finished.

There the user can see for each property the histogram, population graph, consistency and average (only if the property is numeric).

Also, the user can choose to rerun a finished execution.

In addition to that, there's a thread which sleeps every 300 milliseconds and withdraws DTO from the chosen simulation, so the user would be able to see it's details.

3) **SystemInterface** jar: No significant changes from previous task except for working with threads and runnable methods.

This time the running simulation for each execution happens in a runnable method which is run by threads from the threadpool.

\*\*\* Where we show the graph of population per ticks, we chose to show only every 1000 ticks and the x axis is up to 100.

We chose that, because there's a lot of data loaded to the graph (many ticks pass), which makes the system work slow.

**New main classes:**

SecondaryEntity - In this task the xml file includes an option for a secondary entity to run actions on, or to use its properties and values.

Two new actions: Proximity action and Replace action.

This time we divided the simulation engine into 3 main classes:

SimulationRunner - Holds a runnable method which executed by threads from the threadpool. This runnable method is the simulation running for each execution.

SimulationDetails - Each simulation that is executed gets its own SimulationDetails object. It holds all the entity instances, active environments, etc.

SimulationManager - Holds all the simulations that have been executed, the world, manages the threadpool and of course transfers DTO details to the GUI.

\* The GUI jar acknowledge the SimulationManager.