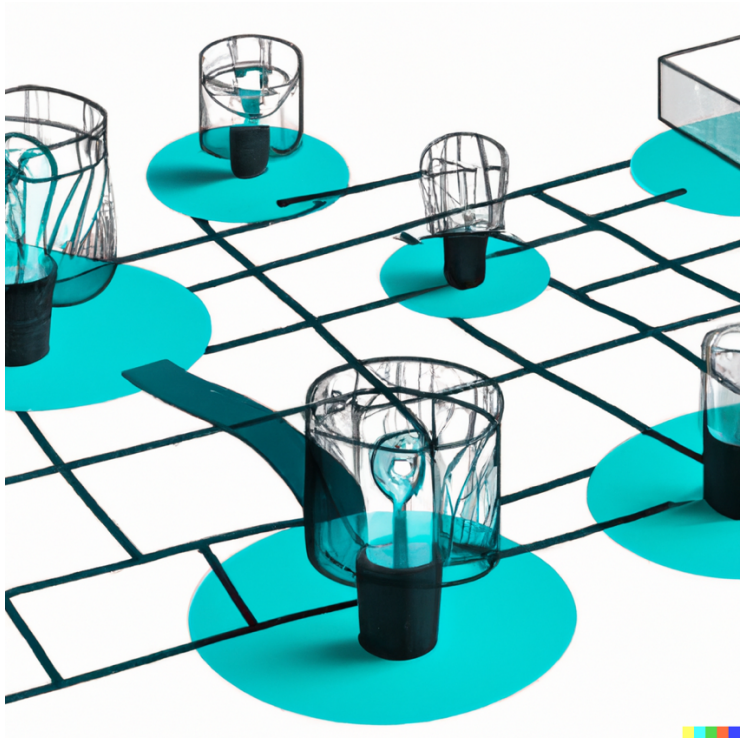
Predictions – By Nitzan

Intro

Welcome to Predictions, an advanced simulation engine designed for researchers who want to model and simulate complex scenarios. This platform provides a powerful and customizable framework for creating diverse simulations by simply uploading an XML file that describes the world. Whether you're simulating ecological systems, social dynamics, or other phenomena, Predictions is the perfect tool to turn your research concepts into dynamic simulations.



About The Project:

Student Name: Nitzan Ainemer (ניצן איינמר)
Email: Nitzan63@Gmail.com
Bonuses Implemented: none.

Design Overview:

There are three modules implemented in this project:

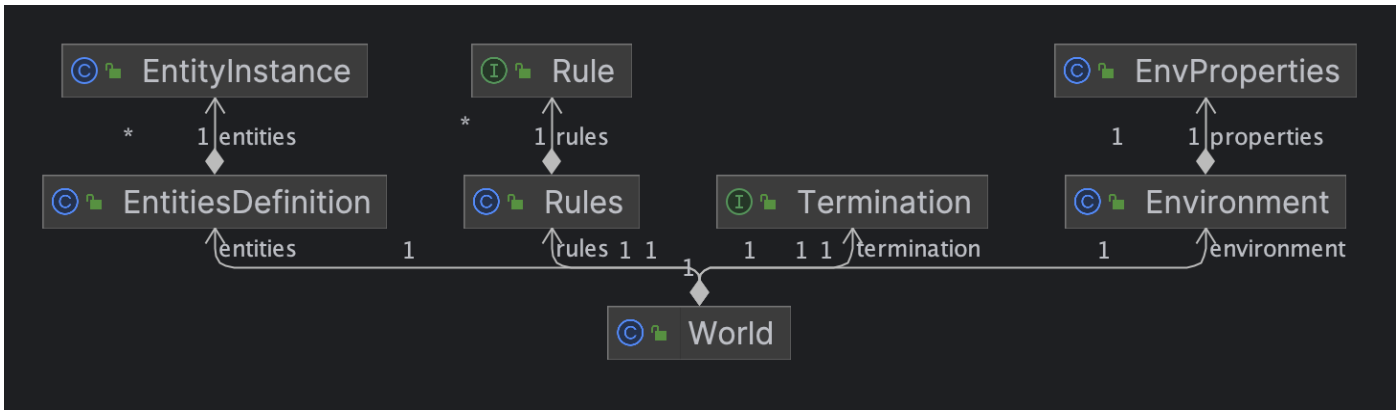
1. Engine Module – the “heart” of the project. The Engine runs all the logic of the program. From handling file and validating data to running the simulation.
2. Console UI – the user interface of the program. It manages the interaction with the user, and displaying the simulation results.
3. DTO – data transfer objects module, a module designed to be the “buffer” between the engine and the UI. Any data that the engine needs to provide for the user, it does that using the DTO, and the opposite as well.

Engine Module:

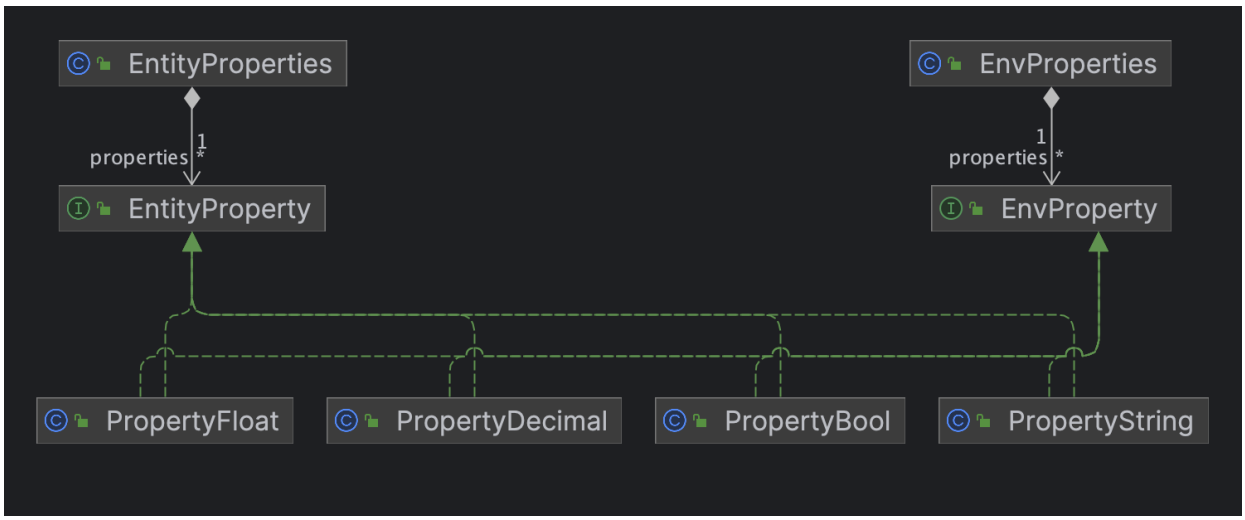
The Engine Has many roles and functions. In this section I will present the main components of the Engine module.

1. **File Handling:**
    a. The engine gets a path to a file and uses JAXB to unmarshall it to JAXB classes.
    b. After the unmarshalling, the engine runs a “static” validation system that validates the data loaded to the JAXB classes. If it fails – it throws Exceptions from the engine to the UI, using a class in the DTO called ErrorDTO.
    If the “static” validation succeeded, we move on.
    c. Mapping – the Engine maps the JAXB classes to the domain classes (That will be presented in this file).
    d. After mapping validation – the engine runs another validation, that requires the domain objects to be “alive”. For example – this is where the “byExpression” validation happens.

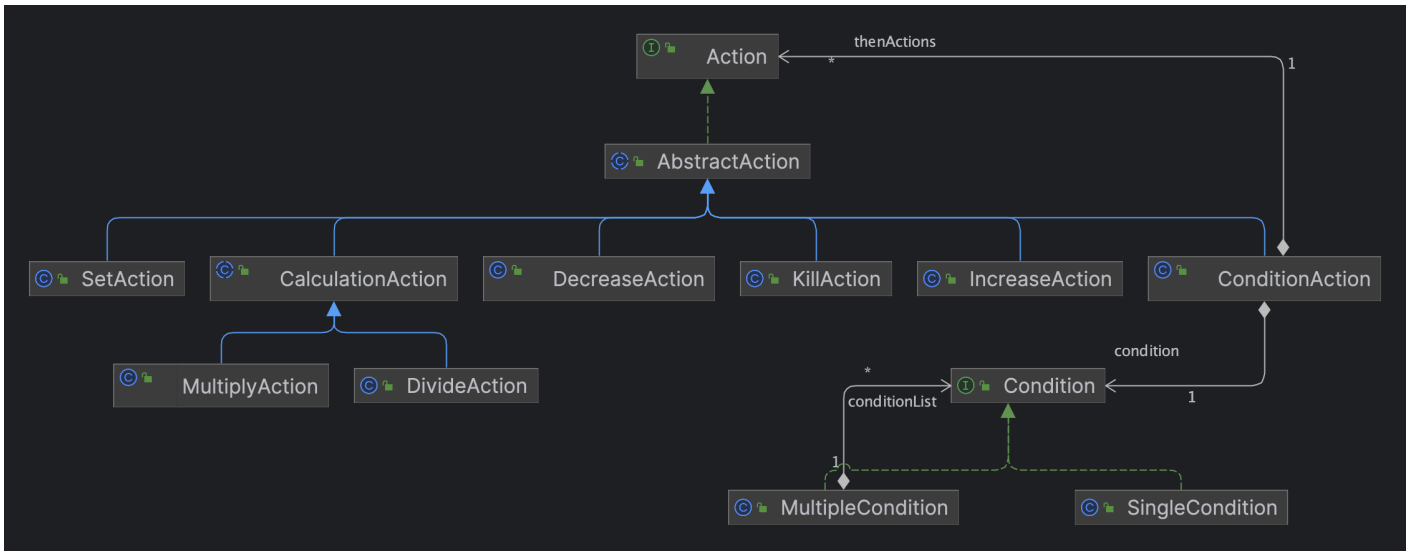
2. **WORLD:**
    a. The engine holds and manages all the domain classes of the “world”. Here Is a class diagram to represent the classes and relations to “World”:



3. **Entities, Environment and properties:**
    a. In my implementation, I have created a class called EntitiesDefinition, which holds the name of the entity, the population and a Map that holds all the EntityInstance-s.
    b. The Environment class holds a list of EnvProperties.
    c. For the properties – I’ve created different interfaces for entity properties and environment properties. For the implementation I’ve used the same classes for them:

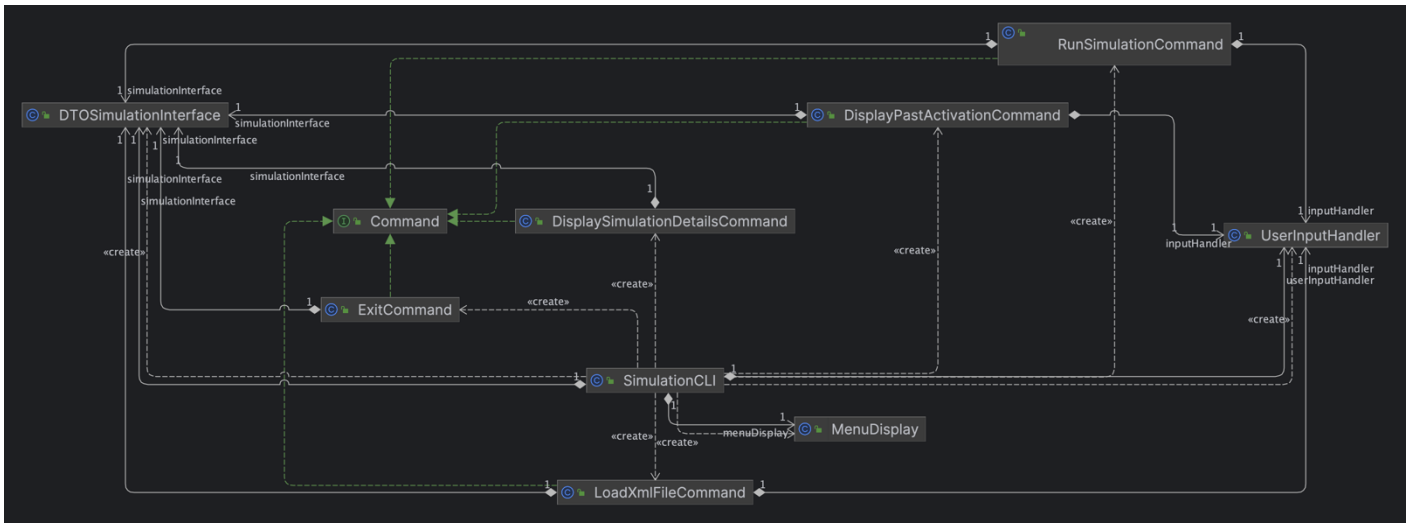


4. **Rules and actions:**
    a. Rules and Actions where one of the most challenging implementation in the domain class and logics.
    b. I managed to use interfaces and abstract classes to simplify it and make it work.
    c. Here is a diagram that represents the relations between them:



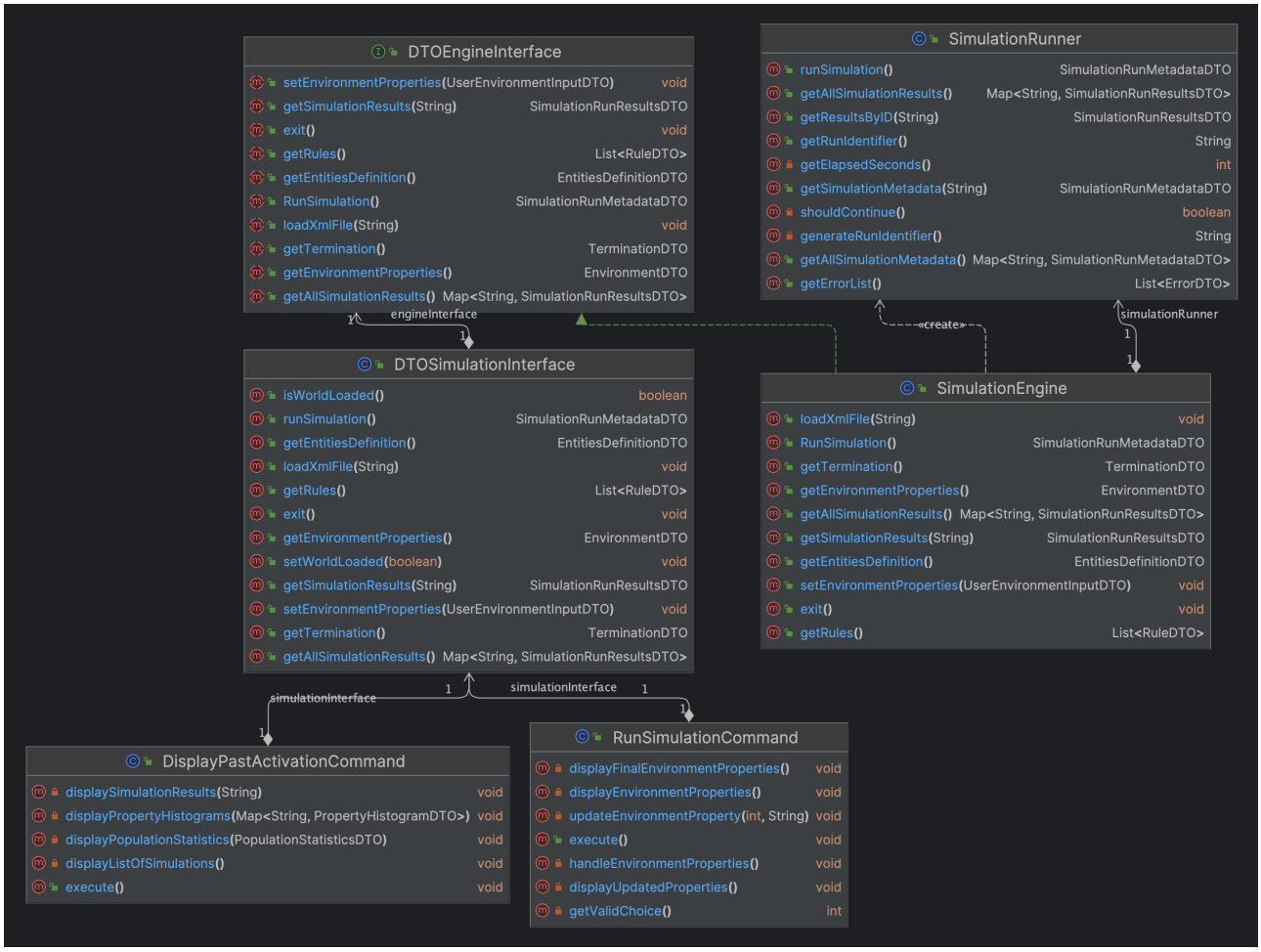
UI

1. The UI is the front of the Predictions program. In this case it's a Console Application UI.
2. The main is in the UI, Under SimulationCLI class.
3. SimulationCLI is the main class that manages the "front" of the simulation.
4. There are auxiliary classes, like UserInputHandler (who handles input) and MenuDisplay.
5. There in an interface called Command who represents all the actions can be made in the program:
    a. Load XML File
    b. Display Simulation Data
    c. Run Simulation
    d. Display Past Results
    e. Exit
6. The connection of the UI to the Engine Goes through DTOSimulationInterface Class that is in the DTO module. More about the interfaces between the UI and the Engine are in the next page.
7. Diagram of the UI classes and relations:



Simulation Run

In the design of the simulation run, I designed a separated classes that interfaces between each other in a way that keeps the engine separated from the UI, and the opposite. Here is a diagram that shoes an example of how the RunSimulation Command and the



DisplayPastResults Command are happening in “Predictions”:

1. The SimulationCLI class (Not in this scheme) in the ConsoleUI module, has the Main function that starts the program running. It presents the menu to the user and handles input.
2. Once the user loaded a valid xml file, and the user decided he wants to run the Simulation, SimulationCLI invokes RunSimulationCommand.execute().

3. RunSimulationCommand is a class in the ConsoleUI Module. It receives a DTO of the environment properties, and lets the user change them. It sends back a DTO to the engine so it can update the property values.
4. The RunSimulationCommand interacts with DTOSimulationInterface, which is a class in the DTO.
5. The DTOSimulationInterface is working with DTOEngineInterface in the DTO module.
6. The DTOEngineInterface is implemented by SimulationEngine in the Engine module.
7. SimulationEngine is the manager of the Engine, which he can invoke actions like loadXmlFile, runSimulation and more.