Assignment
DEV 3 - March 8, 2017

The Dev TEAM

March 8, 2017

1 Goal and description

The goal of this assignment is to apply programming concepts you are learning during DEV 3 such as type systems and polymorphism. For this purpose you will develop a simulation game about logistical tasks.

In the simulation a series of factories are producing products that need to be carried out by trucks. Precisely the simulation features two factories (a Mine and an Ikea laboratory) that produce with different timings products (products are stacked right next to the factory). In front of every factory a truck is waiting until the stack of products reaches a certain amount. Once the amount is reached the products are loaded into the truck inside a container. Once loaded inside the truck, the container is transport outside the screen.

The skeleton of the application comes with a *template project*. The template is available on N@tschool.

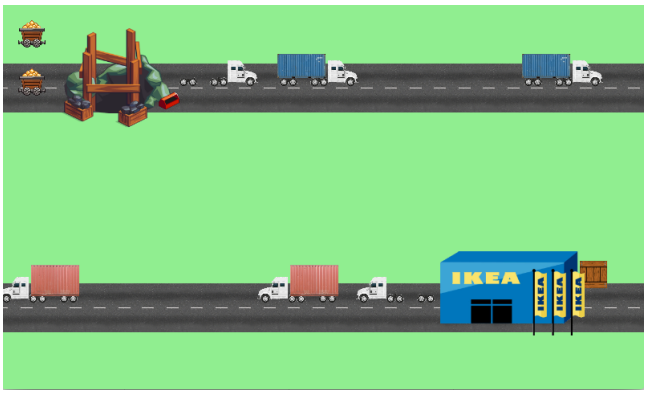

Figure 1: "Screenshot of the final version of the simulation."

2 Tasks

The assignment is divided in two parts (plus a warm-up):

Warming up: study the code Study all the data structures of the provided template. It's very important in order to design the solution. Study very well the IStateMachine interface and its implementations. For this purpose, we also provide you a class diagram of the solution, derived from the `Java` version of the simulation.

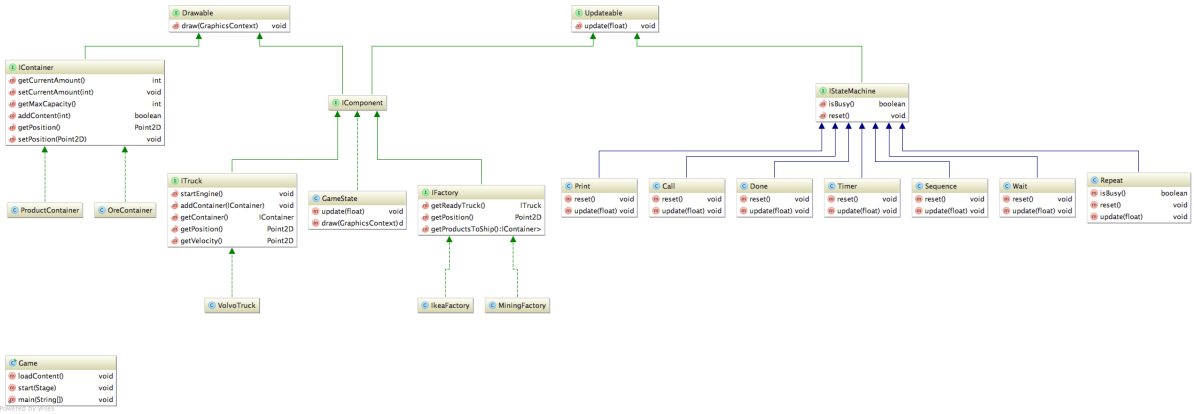

Figure 2: "Class diagram of the simulation derived from the `Java` solution of the simulation."

Part 1 We provide you with a series of interfaces and implemented classes. Your task is define proper implementations for each interface (except for IStateMachine). The final result for this part is having the factories producing an accumulating products next to them. We provide you with the following classes and interfaces:

- Game: the entry point of the simulation. Some parts are commented and taken from the working complete solution. Ideally once you finished the implementation you can uncomment them and test your application. Read such commented code and use them to define the signatures for your constructors.
- IStateMachine: a state machine interface. Along with such an interface a series of concrete classes each implementing a statemachine are provided. Compose instances of these classes to define the sequence of logistical steps. For instance: *keep accumulating ore until a certain amount is reached. Repeat this behavior.*
- IContainer: represents a generic container. Instances of such container can be either stacked next to the factory or carried inside a truck.
- IFactory: represents a Mine or an Ikea and generally any factory the produces goods.
- IDrawable: represents a generic drawable entity.
- IUpdateable: represents a generic updateable entity

- IComponent: represents a generic entity that is drawable and updateable.

We ask you to implement the following concrete classes:

- Mine: is a factory producing ore. It is an IComponent.
- Ikea: is a factory producing products. It is an IComponent.
- Ore and products are drawable containers.

Part 2 Extend the current simulation so to have tucks loading and transporting the stacked products outside the screen. Moreover we ask you to extend the actual state machines with a parallel IStateMachine operator that keeps running forever two given IStateMachines.

Optional: Refactor some actual state machine so to support Lambdas and integrate them into your code. **Optional**: Use skeletal implementations in order to reduce code duplication.

3 Software requirements

3.1 Software needed fo CSharp version

To implement the simulation you need `mono-project or .net` and `mono game`. You can download the `mono-project` from **click me** and `mono-game` from **click me**. Mono-game is yet another game engine. It is an open source implementation of the Microsoft XNA 4 Framework to develop a cross platform applications.

3.2 Software needed for Java version

Java 1.8 or later is required.

4 Submission and deadline

Contribution: *Groups of 2 students is allowed with individual responsibility*
What: *Upload a zip-file following the pattern provided below for naming:* ***first-name_lastname _studentNr.(zip)*** *for example:* ***Jan_Willem _0123456****.(zip).*
When: Both parts must be submitted *before the oral practicum.*
Where: *On N@tschool*

GOOD LUCK!!! The DEV TEAM ☺