**DePaul University**

**SE 450: Object-Oriented Software Development**

**Project Report**

**Traffic Simulation**

**Name: <Athina Verroiopoulou>**

**Students ID: 1531109**

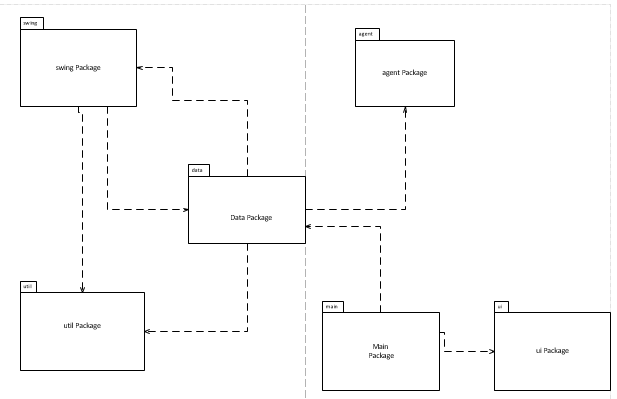
**November 16, 2015**

**Packages interaction:**

Data package is dependent to agent , util and swing

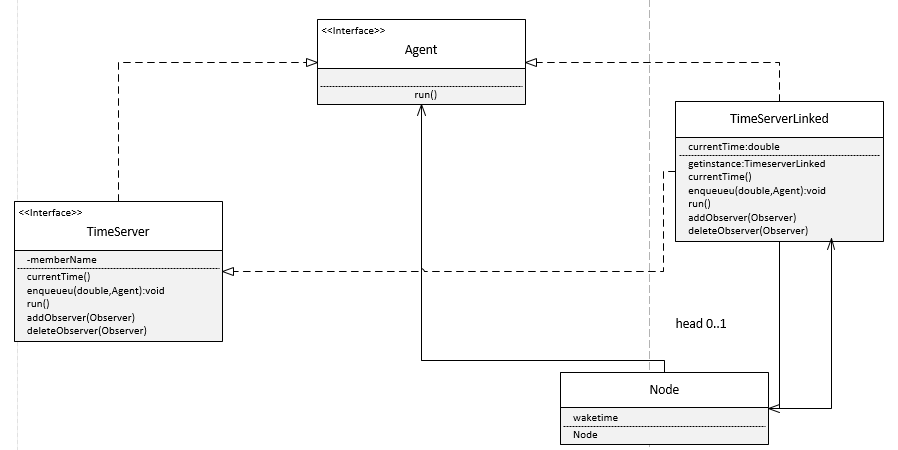
main to ui ,data

swing to data and util.

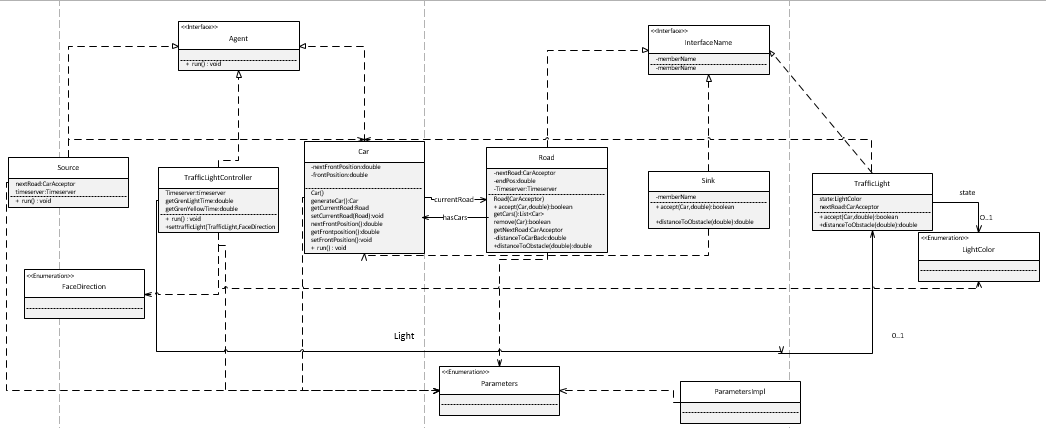


**Agent Package:**

TimeServerLinked and Node have associations to themselves for time and nextnode (0..1)



**Package Data:**



Car, Source, TrafficLightControler classes are dependent to Agent interface.

Road, Sink, TrafficLight classes are dependent to CarAcceptor interface.

Car is associated with Road (currentRoad) and Road with Car (hasCars).

Model, Car,Road, Source , trafficLightControler and ParametersImpl is dependent to parameters

Model is dependent to trafficLightControler that is associated with traffic light (Light) 0 to 1 and dependent to lightColor and FaceDirection.

TrafficLight is associated with lightColor.

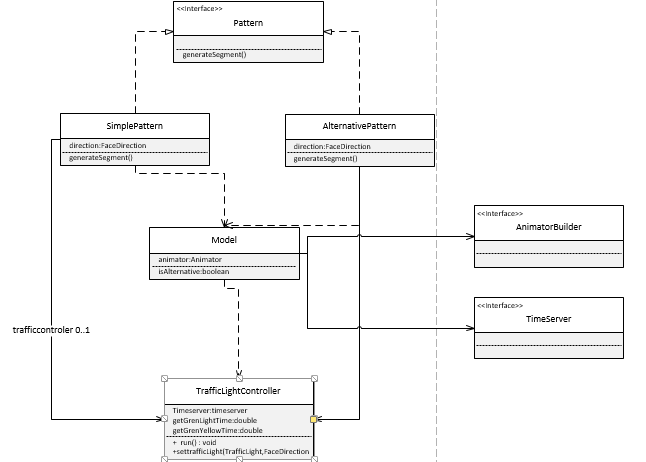
Model is dependent to Alternative and simple Pattern that are dependent to Pattern and are associated to TrafficLight and dependent with TrafficLight Controler

Sink and Source is dependent to Car.

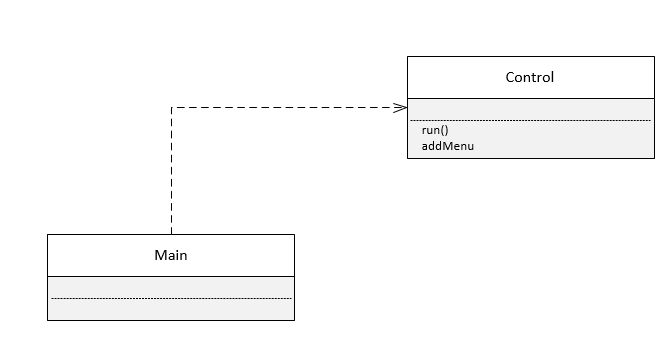
Alternative and simple Pattern are dependent to Road, Source, Sink,CarAcceptor

Model is associated with Animation Builder (builder) and timeserver (0..1)

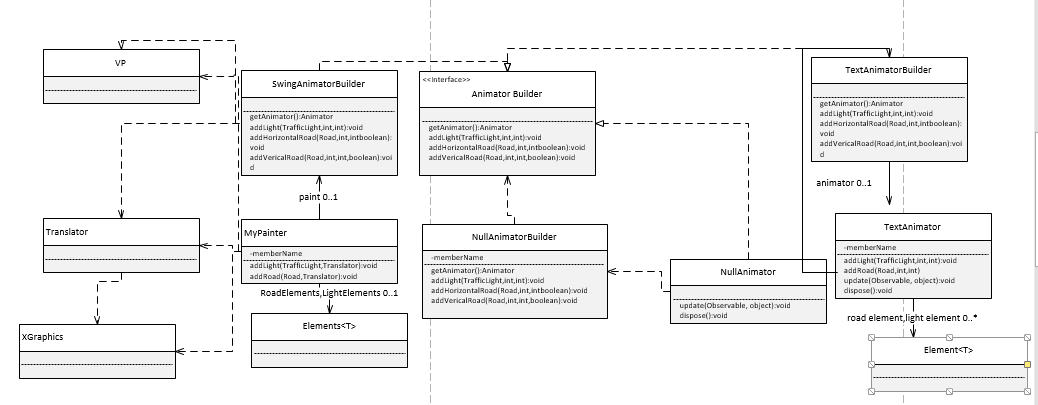
Alternative and simple Pattern are associated with Animation Builder.



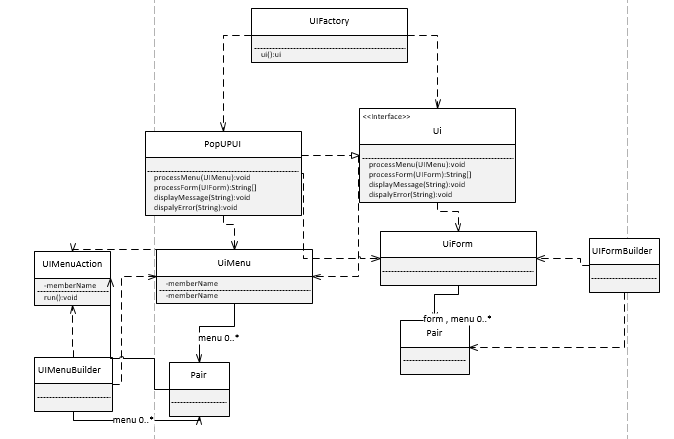
**Package Main:**



**Package Swing:**



**Package ui:**



popUpUi is Immutable class that displays a popup Swing UI that controls user input by processing the form and menu

Ui is Interface that establishes what the UI should incorporate whether it be Popup or text based

UIFactory created new pop up ui

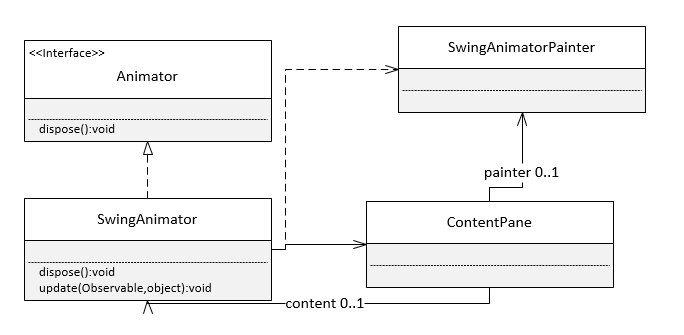
Ui form is an Immutable class establishes the model behind a Swing form

Ui form builder is the Builder that builds an immutable form based on the parameters.

ui menu is Immutable class that creates the UI Menu

ui menu action is an interface for the individual menu actions

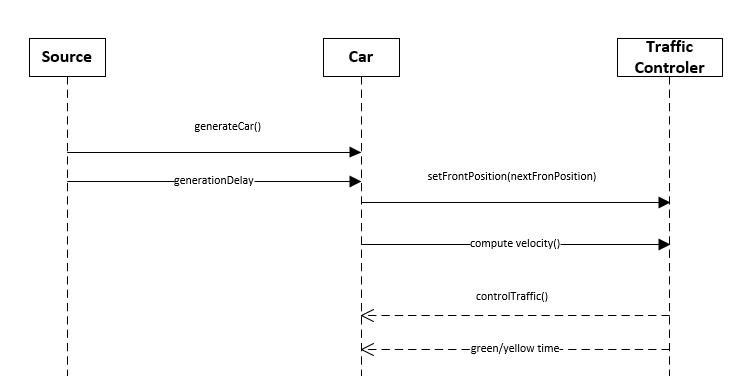
ui menu builder is the builder that creates an immutable menu with the specified parameters

**Package util:**

**Sequence Diagram:**

**How car update its position:**

Car movement is working with the interaction of Agent that has a run method. Source Generate car based on the source delay that connects with the time server and added to the current time. Car Acceptor accept the car that has a front position 0.0 in the start. Car next front position is computed based on the velocity. Velocity cannot be negative or more than max velocity. When an obstacle is very far we set max velocity, when it’s in a brake distance the car slow down and stop when it it’s in the stop distance. In the car run method we set as front position the result of next front position method. Car is in the Road and associated with it. When there is a traffic light in the road, traffic controller notify the car with the run method. Depending on light color the car stop or pass.



**Time Summary;**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Week** | **1** | **2** | **3** | **4** | **Total** |
| **Design** | **1.50 hours** | **8 hours** | **1hour** | **8 hours** | **18.50 hours** |
| **Code** | **9 hours** | **25 hours** | **16 hours** | **20 hours** | **70 hours** |
| **Debug** | **3 hours** | **4 hours** | **16 hours** | **10 hours** | **33 hours** |
| **Big Bug** |  |  |  | **3hours** | **3 hours** |
|  |  |  |  |  | **124.50 hours** |

**Notes on Patterns:**

Parameters is a singleton pattern that uses enumeration.

I used a static factory for car, road, and traffic light.

I used a strategy pattern for simple pattern and alternative one to generate the segments.

Traffic light is a state pattern determines what cars passes.

Car Acceptor is a proxy pattern that is used from road, sink, and traffic light.

Composite pattern for storing cars in the road.

Observer Pattern is the Animator in util package that is an observer and timeserverlinked is an observable.

MVC pattern: model is the data package, view is the ui and swing and controller is the Animator, util package.

**Successes and Failures:**

Program compiles with the Gui. Cars move in the road. Grid and Road segments are created. Traffic light works with the traffic controller. Cars stop and pass depending the color of the light. Car generated depending in source time delay and end up to sink. The choices one for run the simulation and 2 works and indicates the appropriate results.

The cars movement in the gui has problems. Car move fast in the beginning and slow down as time passes. Car stops far from light. The program slow down as more cars generated. In the Gui there are a lot of bugs such as cancel bottom that not working and there is no implement for exception in case of wrong input of values. There is also a problem with simple traffic pattern that not works but it’s implemented.

About the design decisions I change the project design many times. At the beginning I wants to make the parameters as a static class that implements an interface that will be useful for testing too. I consider to implement an intersection that has inside the traffic light. I wanted to make 2 traffic lights in graphic one for each road and use the provided states for traffic controller.

After implement all of them they didn’t work so I try to make the light as an intersection that set the flow of the cars and make the states for traffic controller depending on the light color in a very simple way. The major difficulty I had was with the parameters that had so many things inside and I was lost in the process. In the 3 week I had finished with the implementation but my code didn’t compile so I change the parameters into an enumeration to be clearer. A static class which is a singleton pattern use this enumeration.