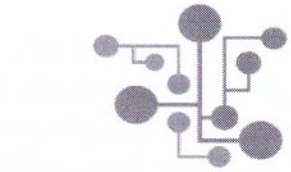
Test Plan

~~Final Version~~

Draft



for ~~DP~~
Send by Kalina (to everybody) and by Jiaqi (only to me). This is Kalina's version.

Group B

Todor Tsekov (2228777)

Wen Zhang (2351420)

Jiaqi Ni (2217774)

Yidi Wu (2197301)

Kalina Petrova (2221667)

Jianfei Feng (2204529)

April 27, 2015

Class: Ei8s1/Ei8s2

Course: Project C-phase

Mentor: Bert Van Gestel

If final version : mark 6.

Introduction

The purpose of this test-plan is to see whether all functionalities described in the use-cases are working correctly and whether certain undesired actions affect the program's workflow. Instead of constant warning pop-up when a certain action is not permitted we focused on preventing the user from creating mistakes as much as possible, meaning certain functions will be disabled when they're not supposed to be accessed.

We will conduct a small test with a handful of testers and conduct a final reconfiguration before the final acceptance test with the client.

for you as testers?

Test Action:

This test action are based on URS document we already made.

Scenarios:

- Add crossing.
- Delete crossing.
- Change crossing.
- Rotate crossing.
- Modify traffic Light System.
- Change traffic Light Setup.
- Alter Flow.
- Navigate.
- Play Simulation.
- Pause Simulation.
- Stop Simulation.
- Create New Project.
- Load Project and Statistics.
- Save Project and Statistics.
- Exit Application.
- Go to Main Screen.
- Undo.
- Redo.
- Reset.

~~explaining tables on page 7 to 15.~~

Table content: " "

Purpose: = testname?

Explains what the purpose of the taken test is. In our case we want to check if we can redirect our user to the main menu screen under certain conditions.

Target on screen:

The actual screen commands the user will interact with.

Test Data/Simulation:

Test our actions under different kinds of conditions and with different kinds of data to check if we have captured all the exceptions and if we take necessary precautions to prevent the action from crashing.

Expected Result:

What is the expected result in each different case we ran the test.

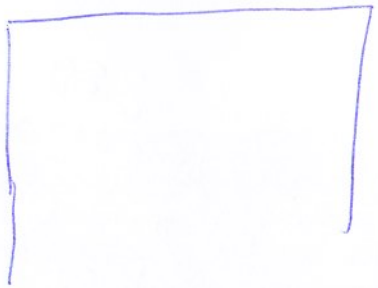
Actual Result:

The actual result that occurred during the test.

Outcome and actions required:

Compare the Expected results and the actual results to come to conclusions what kind of actions are to be taken to fix the inaccuracies.

Test Tables:



test1
test2
test3

I don't like "if" in this test table.

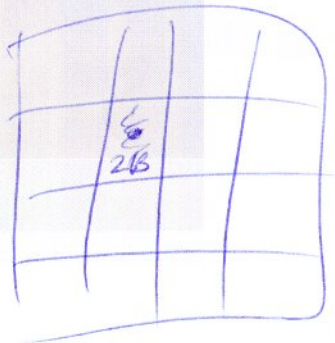
No.	Test Name	Target on screen	Test Data/Simulation	Expected Result	Actual Result	Outcome and Actions required
1.	Add crossing	User choose a crossing→user drags a crossing to a cell→user drop the crossing	1. User chooses a crossing (either type A or B) by left clicking the mouse, then drag it to a cell (cell2B). 2. User chooses a crossing (either type A or B) by left clicking the mouse, then drag it to a cell (cell3B). 3. User chooses a crossing (either type A or B) by left clicking the mouse, then drag it to a cell (cell2B).	1.1 if the simulator is not running/paused, which means user can only see the "start" button enabled in the control panel instead of "stop" button. Then system will place that crossing in cell 2B. 1.2 if the simulator is running/paused, which means user can only see the "stop" button enabled in the control panel instead of "start" button. Then system should give an error indicating that adding crossing is only available while setting up the simulation. 2.1 Same as 1.1. 2.2 Same as 1.2. 3. The system will give an error indicating that it is not possible to place a crossing where there already exists an old one.		
2.	Delete crossing	User right click on a crossing →choose the	1. Delete B3 under initial or paused state and	1. When click delete, the crossing user chosen B3		

Load test2...

and --- see usecase. testcases!

depends on former testcase

load test2



No.	Test Name	Target on screen	Test Data/Simulation	Expected Result	Actual Result	Outcome and Actions required
		"delete crossing"→confirm deletion→system deletes the crossing.	crossing are existing on grid. 2. Delete C3 under initial or paused state and crossing are existing on grid. 3. Delete under initial state, crossings are not existing on grid. 4. Delete under running state. 5. User cancels the operation when the system is asking confirmation.	will be deleted. Flow of C3 will be changed to default one. 2. C3 will deleted. 3. When user choose an empty cell, right-click menu will not appear. 4. User cannot see the right-click menu no matter on which cell user clicks. 5. System does not delete the crossing and give a proper message.		
3.	Change crossing	User right click on a crossing →choose the "change a crossing" option→choose a new crossing→set the traffic light options→system changes the crossing.	1. When simulation is running or paused. 2. User right clicks a cell with a crossing, for example B3 and chooses the option "Change crossing"(the simulations is not running and it is not in pause state) 3. User right clicks a blank cell, for example C4 (the simulations is not running and it is not in pause state) 4. User gives up the operation halfway.	1. The grid options are unavailable. 2. System will show the option panel o you can choose a new crossing. 3. System will show nothing. 4. System cancels the operation and gives a proper message.		

How do you know there is a crossing in B3?

Again : usecase ...
(input values for ...)

No.	Test Name	Target on screen	Test Data/Simulation	Expected Result	Actual Result	Outcome and Actions required
4.	Rotate crossing	Grid->Crossing->Right click menu->Rotate	1. Right click on cell B2 (with crossing) then choose rotate crossing option. 2. Right click on cell C4 (without crossing) then choose rotate crossing option. 3. Start simulation. Right click on any cell. 4. Pause simulation. Right click on any cell. 5. Stop simulation. Right click on cell B2 (with crossing). Choose rotate crossing option	1. Crossing on B2 now has rotated 90 degrees clockwise. 2. No such option should be available to user 3. Nothing happens and no options pop out 4. No Nothing happens and no options pop out 5. The crossing on B2 now has rotated 90 degrees clockwise		
5.	Modify traffic light	Hover over a crossing->click left corner output icon->setting window	1. Hover over cell B2 (with crossing) Click icon on top left corner. Change interval for state 1 to 40. 2. Hover over cell B2 (with crossing) Click icon on top left corner. Change interval for state 2 to 1000 3. Hover over cell C4	1. State 1 for the traffic light system on cell B2 now has interval of 40 seconds. 2. Interval won't change as indicated change is too high. 3. No icon should appear in the top left corner. 4. No icon should appear in		

How can you check it?

No.	Test Name	Target on screen	Test Data/Simulation	Expected Result	Actual Result	Outcome and Actions required
			5. Stop Simulation. Right click on cell B2 (with crossing). Choose Change traffic light setup option. Choose a traffic light setup to replace existing one.			
7.	Navigate	Project-grid screen->Navigation button	1. When simulation is not running or paused and crossings are existing on grid. 2. When simulation is not running or paused and crossings are not existing on grid. 3. When simulation is running.	1. Navigation button is unavailable for users 2. Navigation button is unavailable for users 3. User clicks the navigation start point button, then user selects start point ,then clicks the destination point button and selects end point and input the flow numbers, click confirm button and the data saved or back to default value. After save value, system calculates the new data.		
8.	Play Simulation	Project-grid screen ->Play simulation	1. When simulation is not running or paused and crossings are existing on	1. 1 There're crossing already on grid and connect to each other. User click Navigate button which in tool panel.		

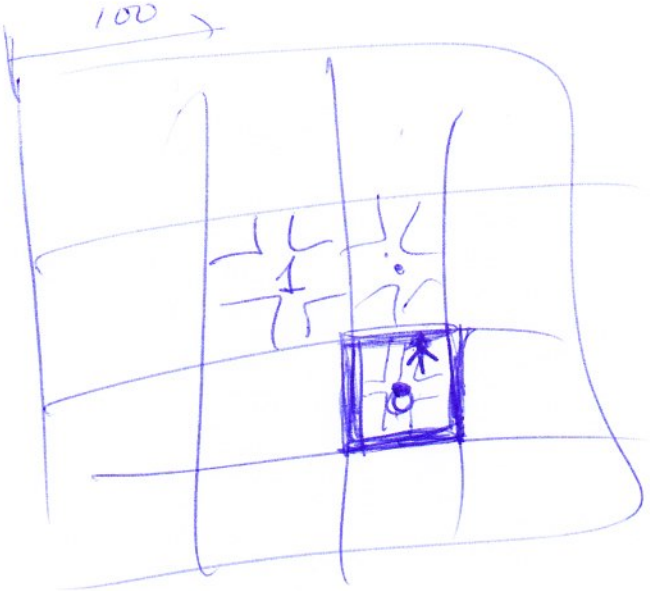
Use case 7: Alter flow ---

No.	Test Name	Target on screen	Test Data/Simulation	Expected Result	Actual Result	Outcome and Actions required
			screen, the user clicks main menu icon and a pop-up menu appears. The user chooses the Create a new project option. 3. The user is working on a file and the user is at Project-grid screen, the user clicks main menu icon and a pop-up menu appears. The user chooses the Create a new project option.	3. A pop-up message will be shown that asks user wants to save the existing file or not. After closing the existing file, the user clicks main menu icon and a pop-up menu appears. The user chooses the Create a new project option.		
13.	Load project and statistics	User goes to Main-Screen ->user clicks load button	1. When the user starts the application. The user open the project and is in the Main-Screen and clicks load button. 2. The user is at Project-grid screen, the user clicks main menu icon and a pop-up menu appears. The user chooses the Open a new project option.	1. The file is loaded, the system is at Project-grid screen. 2. The file is loaded, the system is at Project-grid screen. 3. A pop-up message will be shown that asks user wants to save the existing file or not. After closing the existing file, the user clicks main		

How can you check it?

No.	Test Name	Target on screen	Test Data/Simulation	Expected Result	Actual Result	Outcome and Actions required
			3. The user is working on a file and the user is at Project-grid screen, the user clicks main menu icon and a pop-up menu appears. The user chooses the Open a new project option.	menu icon and a pop-up menu appears. The user chooses the Open a new project option.		
14.	Save project and statistics	Power-grid screen->File->save button	1. 1.When simulation is not running or paused and crossings are existing on grid. 2. When simulation is not running or paused and crossings are not existing on grid. 3. When simulation is running.	1.1 If user has already simulated it, grid project and statistics will be saved to default location. 1.2 If user has not simulated it yet, grid project can be saved and statistics will be saved to an empty txt file. 2. Save button is not available. 3. Save button is not available.	-> 2 different test cases	
15.	Exit Application	Power-grid screen->File->close button	1. The user clicks close button and the simulation stops and is already save by user. 2. The user clicks close button and the simulation stops and is	1. The project is closed. 2. A message shows to notify user that his project has not been saved. Asks user if he'd like to save. 3. The button is disabled. A		

No.	Test Name	Target on screen	Test Data/Simulation	Expected Result	Actual Result	Outcome and Actions required
			successfully and set the traffic light setting then click the reset button. 4. Add some crossings successfully and set the traffic lights then start simulation, then click reset button. 5. Add some crossings successfully and set the traffic lights then start simulation, after a while, stop it, then click reset button.	all the crossing in the grid and give you a proper message indicating that the application is reset. 3. Same as 2. 4. System will give you an error message indicating that it is not possible to reset the application while it is running the simulation. 5. Same as 2.	{ exactly!	



r, c

c[r-1, c]

rowm colmm

Cell : rowNr
columnNr
crossing null

Cell
~~Crossing~~ [,] c;

c = new ~~Crossing~~ Cell [10, 20];

for c[rowm, colm] = null.

int

c[4, 5] = new
Crossi(-)

DD, April 27, 2015

Class Description: version 1.0

Form: the main form where user will do the operations.
Attributes:
Timer: a control component of the form.
Gird: a control component of the form. ← a picturebox?
Simulator: a concrete object of class Simulator.
Methods:
Form (): constructor.
StartSimulator(): a method which will start the simulation.
PauseSimulation(): a method which will pause the simulation.
StopSimulation(): a method which will stop the simulation.
Navigate (): a method which will let user allow a group of cars come from somewhere and go to somewhere.
SaveProject(): a method which will allow the user to save the current project.
LoadProject(): a method which will allow the user to load a project.
Timer_Enabled(): enable the timer, called in the StartSimulation method.
Timer_Disabled(): disabled the timer, called in the PauseSimulation or StopSimulation method.
UpdateInformation(): when the simulation is on, it update the information and show them on the form every second.

Gird: a component of the form, contains all the crossings, lanes, moving objects and traffic lights.
no, contains Simulator object?
Attributes:
Height: the height of the gird.
Width: the width of the gird.
Methods:
Gird(): the constructor.
AddCrossing(): add a crossing, with the type the user choose, to the gird.
RemoveCrossing(): remove a chosen crossing from the gird.
Clear: remove all the crossing in the gird.
Occupied(): check if there is already a crossing in some specific cell in the gird.

Simulator: the class which control the simulation.
Attributes:
Crossings: list of all the crossing.
currentStatus: current status.
Methods:
Simulator(): constructor.
Start(): start the simulation.
Pause(): pause the simulation.
Stop(): stop the simulation.
SetStartPoint(): when the user want to navigate from somewhere to another place, this is for setting the start position.
SetEndPoint():when the user want to navigate from somewhere to another place, this is for

setting the end position.
Navigate(): allow the user want to navigate from somewhere to another place.
AddCrossing(): Add a new crossing.
RemoveCrossing(): Remove a crossing.
RatateCrossing(): Rotate a crossing.
SetFlow(): set the number of flow for a specific land.
Connected(): check whether two lanes are connected. ← which lanes?
GetCurrentState(): get current state.
GetAllCorssing(): get all the crossings in the current simulation.
CalculateNextSecond(): calculate all the changes of the simulation for the next second.

Crossing: the class which contains the information for a crossing.
Attributes:
Lanes: the lanes surrounded around the crossing. ← Give me an example
TrafficLights: the traffic lights on each lane. ↳ TrafficLightStages?
Location: the position of the crossing.
Methods:
Crossing(): constructor.
RightRotate(): rotate the crossing 90 degree clockwise.
GetAllLanes(): get all lanes around this crossing.
GetAllTrafficLights(): get all traffic lights in the lanes those are around the crossing.

CrossingTypeA, CrossingTypeB: special type of crossing.

TrafficLight: the traffic light on the lane.
Attributes:
RedInterval: the interval for the red light.
GreenInterval: the interval for the green light.
YellowInterval: the interval for the Yellow light.
all of them?
Methods:
TrafficLight(): constructor.
SetInterval(): set the interval for red, green and yellow light of this traffic light. ← all the same?

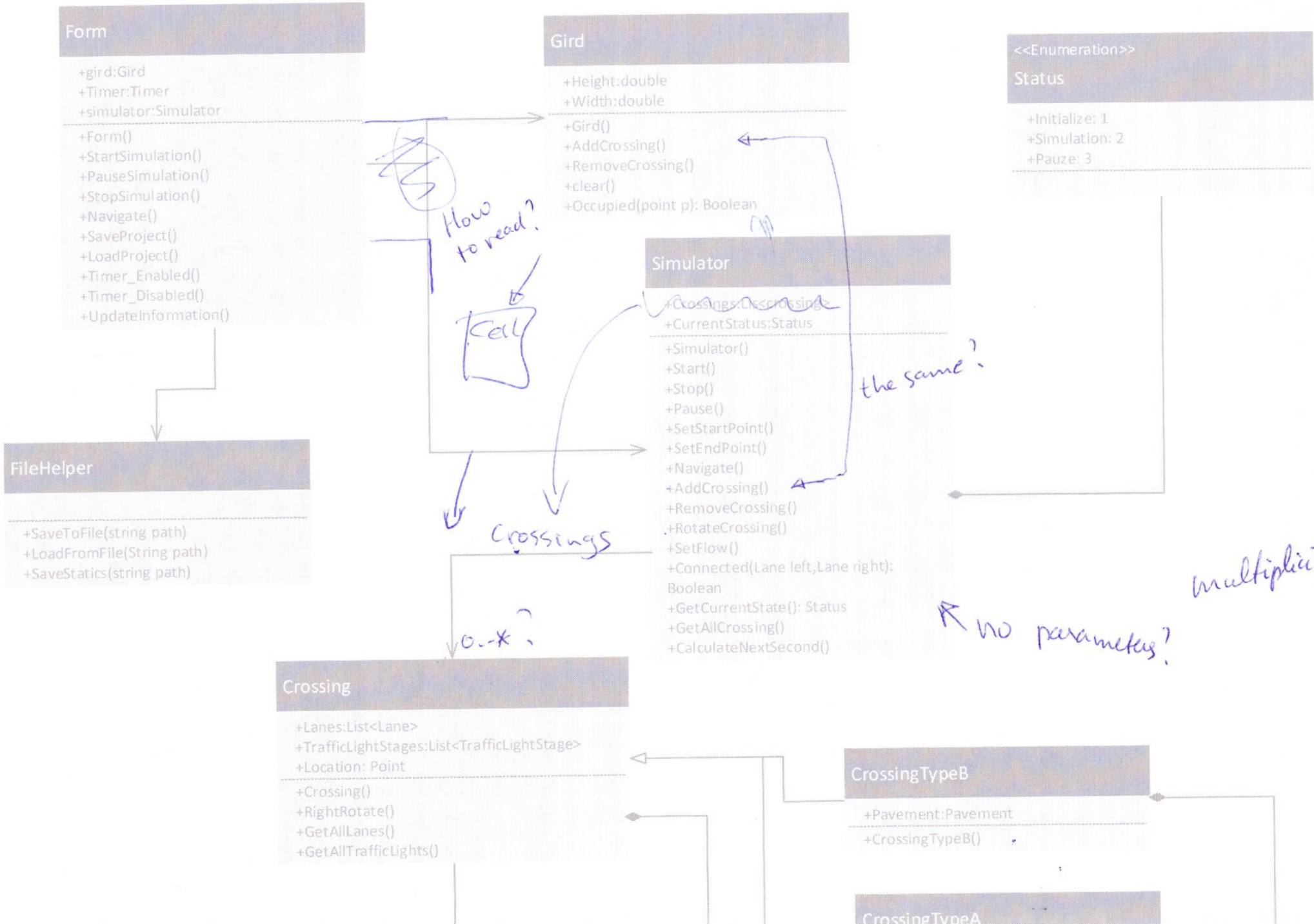
Lane: the lanes which contains all the moving objects.
Attributes:
Flow: the current number of moving objects on this lane.
MaxFlow: the capacity of this lane.
MovingObjects: all the moving objects on this lane.
Flow = MovingObjects.Count?
Methods:
Lane(): the constructor.
SetFlow(): set the number of moving objects on this lane.
GetFlow(): get the number of moving objects on this lane.
ObjectJoin(): a new moving object comes to this lane, flow increase by 1.
ObjectLeave(): a moving object leaves this lane, flow decrease by 1.

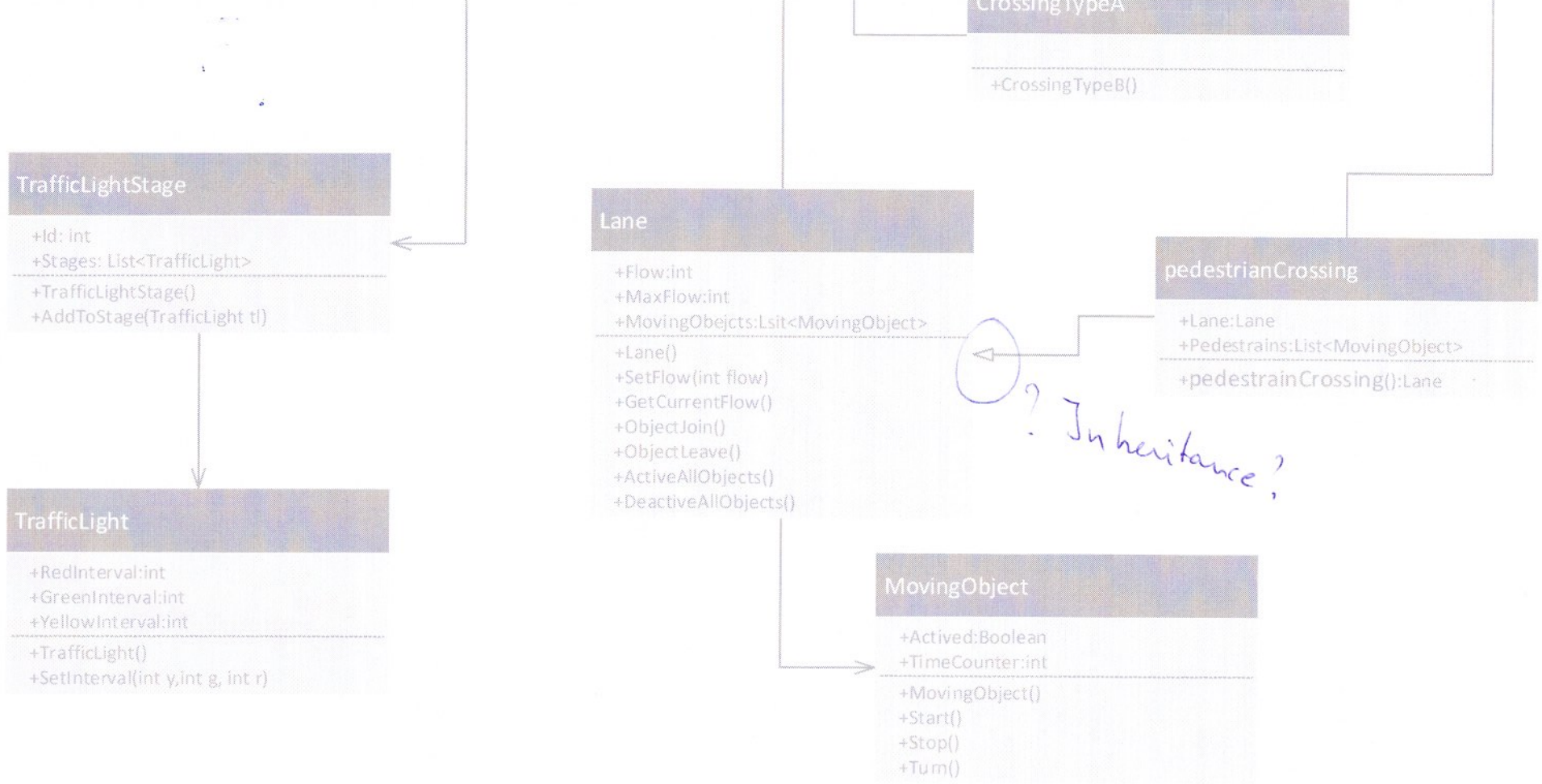
ActiveAllObjects(): activate all the moving objects on this lane.
DeActiveAllObjects(): deactivate all the moving objects on this lane.

MovingObject: the class for the moving objects (cars or pedestrians).
Attribute:
TimerCounter: to decide whether this object should turn to another new lane or stay on the current lane.
Actived: the status of the car.
Methods:
MovingObject(): constructor.
Start(): activate the car.
Stop(): deactivate the car.
Turn(): turn to another lane.

Sequence diagrams:

Add crossing
Delete crossing
Play
Pause
Stop
Rotate
Navigate
Alter flow


Form
+gird:Gird
+Timer:Timer
+simulator:Simulator
+Form()
+StartSimulation()
+PauseSimulation()
+StopSimulation()
+Navigate()
+SaveProject()
+LoadProject()
+Timer_Enabled()
+Timer_Disabled()
+UpdateInformation()
Gird
+Height:double
+Width:double
+Gird()
+AddCrossing()
+RemoveCrossing()
+clear()
+Occupied(point p): Boolean
<<Enumeration>>
Status
+Initialize: 1
+Simulation: 2
+Pauze: 3
Simulator
+Crossings:List<crossing>
+CurrentStatus:Status
+Simulator()
+Start()
+Stop()
+Pause()
+SetStartPoint()
+SetEndPoint()
+Navigate()
+AddCrossing()
+RemoveCrossing()
+RotateCrossing()
+SetFlow()
+Connected(Lane left,Lane right): Boolean
+GetCurrentState(): Status
+GetAllCrossing()
+CalculateNextSecond()
FileHelper
+SaveToFile(string path)
+LoadFromFile(String path)
+SaveStatics(string path)
Crossing
+Lanes:List<Lane>
+TrafficLightStages:List<TrafficLightStage>
+Location: Point
+Crossing()
+RightRotate()
+GetAllLanes()
+GetAllTrafficLights()
CrossingTypeB
+Pavement:Pavement
+CrossingTypeB()
CrossingTypeA
How to read?
Call
the same?
Crossings
0..*?
multiplicity's
No parameter?



+ : public. So everything is public.