SEG2106

FORMAL LAB REPORT

LAB 2 - UML STATE MACHINES

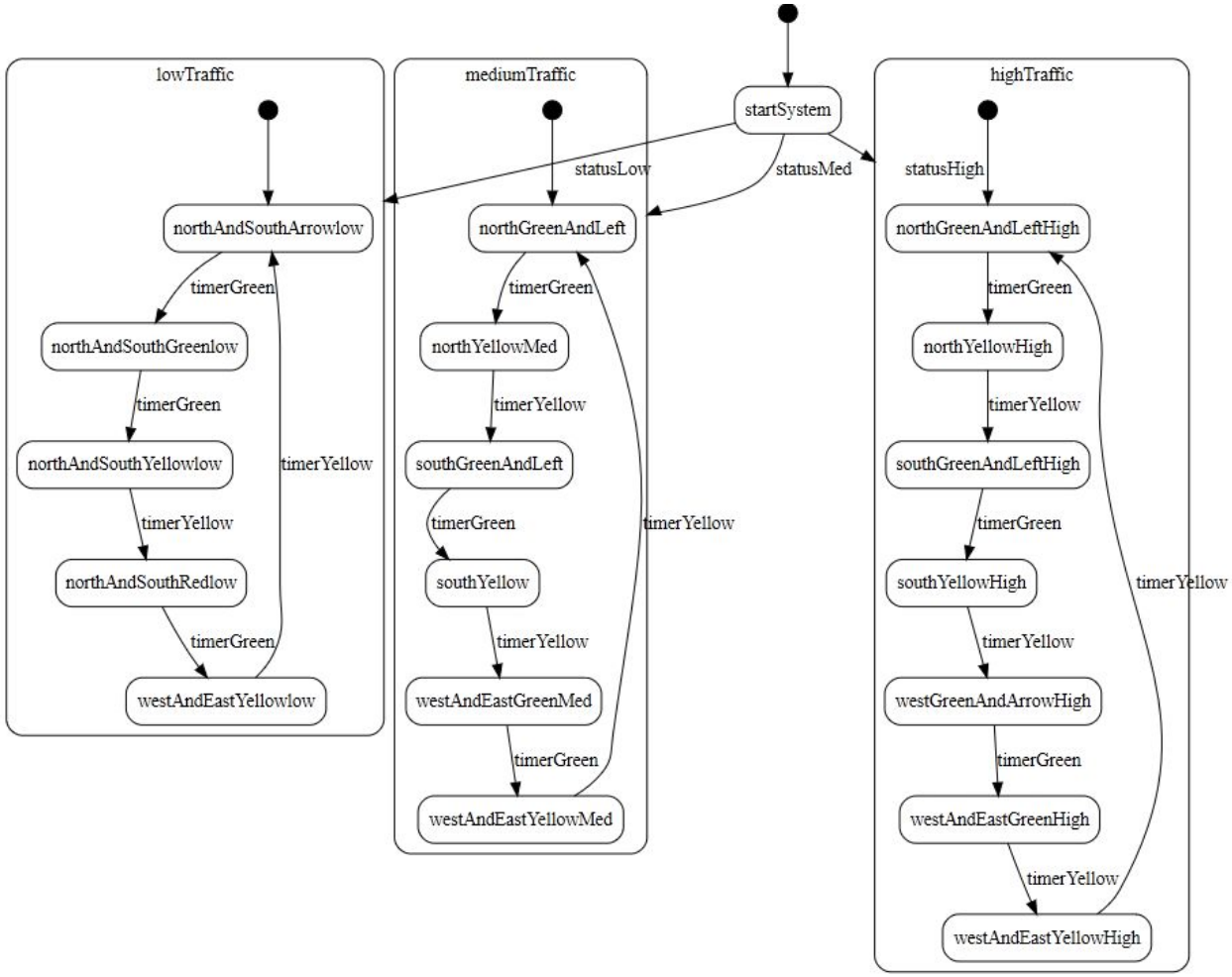
Completed by:
Gaurav Kalsi - 8498770
Arsham Jalayer - 8321782
Ryan Fleck - 8276723

Lab Objectives:

- Analyze traffic light system.
- Develop UML state machine for traffic lights.
- Generate java code from UML.
- Modify generated code to utilize existing GUI.

Design:

Umple State Machine Model:



Modified Umple Source:
The source is displayed in two columns for brevity. The columns are read left to right.

```
class TrafficLight
{
status {
  startSystem{
  statusLow() -> lowTraffic;
  statusMed() -> mediumTraffic;
  statusHigh() -> highTraffic;
  }
  lowTraffic{
     northAndSouthArrowlow{
     entry / { trafficLightManager.northArrow();}
     entry / { trafficLightManager.southArrow(); }
     entry / { trafficLightManager.westRed(); }
     entry/ { trafficLightManager.eastRed(); }
     timerGreen() -> northAndSouthGreenlow;
     }
     northAndSouthGreenlow {
     entry / { trafficLightManager.northGreen(); }
     entry / { trafficLightManager.southGreen(); }
     entry / { trafficLightManager.westRed(); }
     entry / { trafficLightManager.eastRed(); }
     timerGreen() -> northAndSouthYellowlow;
     }
     northAndSouthYellowlow {
     entry / { trafficLightManager.northYellow(); }
     entry / { trafficLightManager.southYellow(); }
     entry / { trafficLightManager.westRed(); }
     entry / { trafficLightManager.eastRed(); }
     timerYellow() -> northAndSouthRedlow;
     }
     northAndSouthRedlow {
     entry / { trafficLightManager.northRed(); }
     entry / { trafficLightManager.southRed(); }
     entry / { trafficLightManager.westGreen(); }
     entry / { trafficLightManager.eastGreen(); }
     timerGreen() -> westAndEastYellowlow;
     }
     westAndEastYellowlow{
     entry / { trafficLightManager.northRed(); }
     entry / { trafficLightManager.southRed(); }
     entry / { trafficLightManager.westYellow(); }
     entry / { trafficLightManager.eastYellow(); }
     timerYellow() -> northAndSouthArrowlow;
     }
     }

  mediumTraffic{
     northGreenAndLeft{
     entry / {
trafficLightManager.northGreenAndArrow(); }
     entry / { trafficLightManager.southRed(); }
     entry / { trafficLightManager.westRed(); }
     entry / { trafficLightManager.eastRed(); }
     timerGreen() -> northYellowMed;
     }
     northYellowMed{
     entry / { trafficLightManager.northYelllow(); }
     entry / { trafficLightManager.southRed(); }
     entry / { trafficLightManager.westRed(); }
     entry / { trafficLightManager.eastRed(); }
     timerYellow() -> southGreenAndLeft;
     }
     southGreenAndLeft{
     entry / { trafficLightManager.northRed(); }
     entry / {
trafficLightManager.southGreenAndArrow(); }
     entry / { trafficLightManager.westRed(); }
     entry / { trafficLightManager.eastRed(); }
     timerGreen() -> southYellow;
     }
     southYellow{
     entry / { trafficLightManager.northRed(); }
     entry / { trafficLightManager.southYellow(); }
     entry / { trafficLightManager.westRed(); }
     entry / { trafficLightManager.eastRed(); }
     timerYellow() -> westAndEastGreenMed;
     }
     westAndEastGreenMed{
     entry / { trafficLightManager.northRed(); }
     entry / { trafficLightManager.southRed(); }
     entry / { trafficLightManager.westGreen(); }
     entry / { trafficLightManager.eastGreen(); }
     timerGreen() -> westAndEastYellowMed;
     }
     westAndEastYellowMed{
     entry / { trafficLightManager.northRed(); }
     entry / { trafficLightManager.southRed(); }
     entry / { trafficLightManager.westYellow(); }
```

```
    entry / { trafficLightManager.eastYellow(); }
    timerYellow() -> northGreenAndLeft;
    }
    }


  highTraffic{
    northGreenAndLeftHigh{
    entry / {
trafficLightManager.northGreenAndArrow(); }
    entry / { trafficLightManager.southRed(); }
    entry / { trafficLightManager.westRed(); }
    entry / { trafficLightManager.eastRed(); }
    timerGreen() -> northYellowHigh;
    }
    northYellowHigh{
    entry / { trafficLightManager.northYelllow(); }
    entry / { trafficLightManager.southRed(); }
    entry / { trafficLightManager.westRed(); }
    entry / { trafficLightManager.eastRed(); }
    timerYellow() -> southGreenAndLeftHigh;
    }
    southGreenAndLeftHigh{
    entry / { trafficLightManager.northRed(); }
    entry / {
trafficLightManager.southGreenAndArrow(); }
    entry / { trafficLightManager.westRed(); }
    entry / { trafficLightManager.eastRed(); }
    timerGreen() -> southYellowHigh;
    }
    southYellowHigh{
    entry / { trafficLightManager.northRed(); }
    entry / { trafficLightManager.southYellow(); }
    entry / { trafficLightManager.westRed(); }
    entry / { trafficLightManager.eastRed(); }
    timerYellow() -> westGreenAndArrowHigh;
    }
    westGreenAndArrowHigh{
    entry / { trafficLightManager.northRed(); }
    entry / { trafficLightManager.southRed(); }
    entry / {
trafficLightManager.westGreenAndArrow(); }
    entry / { trafficLightManager.eastRed(); }
    timerGreen() -> westAndEastGreenHigh;
    }
    westAndEastGreenHigh{
    entry / { trafficLightManager.northRed(); }
    entry / { trafficLightManager.southRed(); }
    entry / { trafficLightManager.westGreen(); }
    entry / { trafficLightManager.eastGreen(); }
    timerYellow() -> westAndEastYellowHigh;
    }
    westAndEastYellowHigh{
    entry / { trafficLightManager.northRed(); }
    entry / { trafficLightManager.southRed(); }
    entry / { trafficLightManager.westYellow(); }
    entry / { trafficLightManager.eastYellow(); }
    timerYellow() -> northGreenAndLeftHigh;
    }
    }

    }
    }//$?[End_of_model]$?

class TrafficLight
{
  position 50 30 109 45;
}
```

Modified source code:

We have included the modified code in this zip, with filename “TrafficLight.java”

Discussion:

Our group did not run into any issues with the first component of the lab, where we modified the umple file to adhere to the provided behaviours.

Initially, after modifying the umple file, we only copy-pasted the bottom component of the generated java code, re-using the provided constructor hoping that it would lead to less modification. Taking this approach resulted in a handful of null pointer

exceptions, and after a few attempts, we decided that modifying the full generated code instead would yield better results.

After re-copying and modifying the umple code, we were able to make our traffic lights change colors in the sequence indicated in the lab manual. We tested multiple times, and confirmed that the behaviour matched.

Conclusion:

We were able to meet all of the guidelines of the lab and create a working traffic light system model.