**Lab 2 : Machine d’état UML**

**SEG2506 - Construction de logiciels**

**Hiver 2020**

**École de Génie Électrique et Science Informatique**

**Université d'Ottawa**

Professeur : Hussein Al Osman

Étudiant 1 : Khang Nguyen 30007277

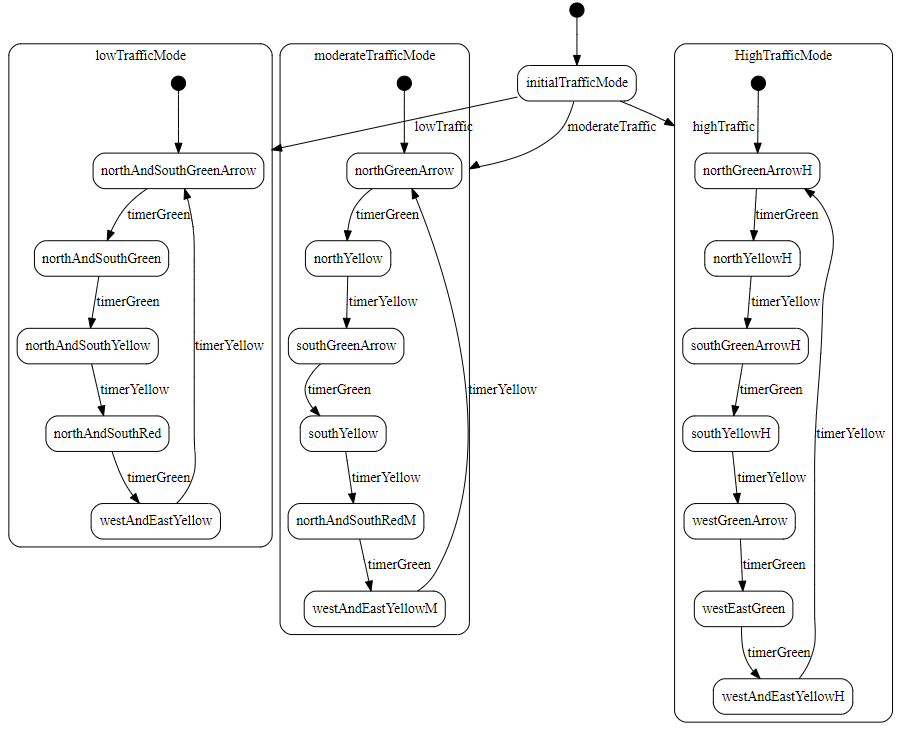
Étudiant 2 : Pascal Marleau-Proulx 300034507

Date de la soumission : le 7 Fevrier 2020

# Objectif:

L’objectif de ce labo est de développer avec succès une machine d’état UML pour le feu de circulation, générer un code java en utilisant UMPLE et l’intégrer dans un programme existant.

# State Diagram:



# UMPLE:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

class TrafficLight

{

status {

initialTrafficMode{

lowTraffic() -> lowTrafficMode;

moderateTraffic() -> moderateTrafficMode;

highTraffic() -> HighTrafficMode;

}

lowTrafficMode{

northAndSouthGreenArrow {

entry / { trafficLightManager.northArrow(); }

entry / { trafficLightManager.southArrow(); }

entry / { trafficLightManager.westRed(); }

entry / { trafficLightManager.eastRed(); }

timerGreen() -> northAndSouthGreen;

}

northAndSouthGreen {

entry / { trafficLightManager.northGreen(); }

entry / { trafficLightManager.southGreen(); }

entry / { trafficLightManager.westRed(); }

entry / { trafficLightManager.eastRed(); }

timerGreen() -> northAndSouthYellow;

}

northAndSouthYellow {

entry / { trafficLightManager.northYellow(); }

entry / { trafficLightManager.southYellow(); }

entry / { trafficLightManager.westRed(); }

entry / { trafficLightManager.eastRed(); }

timerYellow() -> northAndSouthRed;

}

northAndSouthRed {

entry / { trafficLightManager.northRed(); }

entry / { trafficLightManager.southRed(); }

entry / { trafficLightManager.westGreen(); }

entry / { trafficLightManager.eastGreen(); }

timerGreen() -> westAndEastYellow;

}

westAndEastYellow{

entry / { trafficLightManager.northRed(); }

entry / { trafficLightManager.southRed(); }

entry / { trafficLightManager.westYellow(); }

entry / { trafficLightManager.eastYellow(); }

timerYellow() -> northAndSouthGreenArrow;

}

}

moderateTrafficMode{

northGreenArrow {

entry / { trafficLightManager.northGreenAndArrow(); }

entry / { trafficLightManager.southRed(); }

entry / { trafficLightManager.westRed(); }

entry / { trafficLightManager.eastRed(); }

timerGreen -> northYellow;

}

northYellow{

entry / { trafficLightManager.northYellow(); }

entry / { trafficLightManager.southRed(); }

entry / { trafficLightManager.westRed(); }

entry / { trafficLightManager.eastRed(); }

timerYellow() -> southGreenArrow;

}

southGreenArrow {

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() -> northAndSouthRedM;

}

northAndSouthRedM {

entry / { trafficLightManager.northRed(); }

entry / { trafficLightManager.southRed(); }

entry / { trafficLightManager.westGreen(); }

entry / { trafficLightManager.eastGreen(); }

timerGreen() -> westAndEastYellowM;

}

westAndEastYellowM{

entry / { trafficLightManager.northRed(); }

entry / { trafficLightManager.southRed(); }

entry / { trafficLightManager.westYellow(); }

entry / { trafficLightManager.eastYellow(); }

timerYellow() -> northGreenArrow;

}

}

HighTrafficMode{

northGreenArrowH {

entry / { trafficLightManager.northGreenAndArrow(); }

entry / { trafficLightManager.southRed(); }

entry / { trafficLightManager.westRed(); }

entry / { trafficLightManager.eastRed(); }

timerGreen -> northYellowH;

}

northYellowH{

entry / { trafficLightManager.northYellow(); }

entry / { trafficLightManager.southRed(); }

entry / { trafficLightManager.westRed(); }

entry / { trafficLightManager.eastRed(); }

timerYellow() -> southGreenArrowH;

}

southGreenArrowH {

entry / { trafficLightManager.northRed(); }

entry / { trafficLightManager.southGreenAndArrow(); }

entry / { trafficLightManager.westRed(); }

entry / { trafficLightManager.eastRed(); }

timerGreen() -> southYellowH;

}

southYellowH {

entry / { trafficLightManager.northRed(); }

entry / { trafficLightManager.southYellow(); }

entry / { trafficLightManager.westRed(); }

entry / { trafficLightManager.eastRed(); }

timerYellow() -> westGreenArrow;

}

westGreenArrow {

entry / { trafficLightManager.northRed(); }

entry / { trafficLightManager.southRed(); }

entry / { trafficLightManager.westGreenAndArrow(); }

entry / { trafficLightManager.eastRed(); }

timerGreen() -> westEastGreen;

}

westEastGreen {

entry / { trafficLightManager.northRed(); }

entry / { trafficLightManager.southRed(); }

entry / { trafficLightManager.westGreen(); }

entry / { trafficLightManager.eastGreen(); }

timerGreen() -> westAndEastYellowH;

}

westAndEastYellowH{

entry / { trafficLightManager.northRed(); }

entry / { trafficLightManager.southRed(); }

entry / { trafficLightManager.westYellow(); }

entry / { trafficLightManager.eastYellow(); }

timerYellow() -> northGreenArrowH;

}

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# TrafficLight Class:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//%% NEW FILE TrafficLight BEGINS HERE %%

package SEG2106.core;

/\*PLEASE DO NOT EDIT THIS CODE\*/

/\*This code was generated using the UMPLE 1.29.1.4753.5a97eca04 modeling language!\*/

// line 2 "model.ump"

// line 149 "model.ump"

// line 154 "model.ump"

// line 159 "model.ump"

// line 164 "model.ump"

// line 169 "model.ump"

// line 174 "model.ump"

// line 179 "model.ump"

// line 184 "model.ump"

// line 189 "model.ump"

// line 194 "model.ump"

// line 199 "model.ump"

// line 204 "model.ump"

// line 209 "model.ump"

// line 214 "model.ump"

// line 219 "model.ump"

// line 224 "model.ump"

// line 229 "model.ump"

// line 234 "model.ump"

// line 239 "model.ump"

// line 244 "model.ump"

// line 249 "model.ump"

// line 254 "model.ump"

// line 259 "model.ump"

// line 264 "model.ump"

// line 269 "model.ump"

// line 274 "model.ump"

// line 279 "model.ump"

public class TrafficLight implements EventHandler

{

//------------------------

// MEMBER VARIABLES

//------------------------

//TrafficLight State Machines

public enum Status { initialTrafficMode, lowTrafficMode, moderateTrafficMode, HighTrafficMode }

public enum StatusLowTrafficMode { Null, northAndSouthGreenArrow, northAndSouthGreen, northAndSouthYellow, northAndSouthRed, westAndEastYellow }

public enum StatusModerateTrafficMode { Null, northGreenArrow, northYellow, southGreenArrow, southYellow, northAndSouthRedM, westAndEastYellowM }

public enum StatusHighTrafficMode { Null, northGreenArrowH, northYellowH, southGreenArrowH, southYellowH, westGreenArrow, westEastGreen, westAndEastYellowH }

private Status status;

private StatusLowTrafficMode statusLowTrafficMode;

private StatusModerateTrafficMode statusModerateTrafficMode;

private StatusHighTrafficMode statusHighTrafficMode;

//------------------------

// CONSTRUCTOR

//------------------------

private TrafficLightManager trafficLightManager;

public TrafficLight(TrafficLightManager trafficLightManager)

{

this.trafficLightManager = trafficLightManager;

setStatusLowTrafficMode(StatusLowTrafficMode.Null);

setStatusModerateTrafficMode(StatusModerateTrafficMode.Null);

setStatusHighTrafficMode(StatusHighTrafficMode.Null);

setStatus(Status.initialTrafficMode);

trafficLightManager.addEventHandler(this);

}

//------------------------

// INTERFACE

//------------------------

public String getStatusFullName()

{

String answer = status.toString();

if (statusLowTrafficMode != StatusLowTrafficMode.Null) { answer += "." + statusLowTrafficMode.toString(); }

if (statusModerateTrafficMode != StatusModerateTrafficMode.Null) { answer += "." + statusModerateTrafficMode.toString(); }

if (statusHighTrafficMode != StatusHighTrafficMode.Null) { answer += "." + statusHighTrafficMode.toString(); }

return answer;

}

public Status getStatus()

{

return status;

}

public StatusLowTrafficMode getStatusLowTrafficMode()

{

return statusLowTrafficMode;

}

public StatusModerateTrafficMode getStatusModerateTrafficMode()

{

return statusModerateTrafficMode;

}

public StatusHighTrafficMode getStatusHighTrafficMode()

{

return statusHighTrafficMode;

}

public boolean lowTraffic()

{

boolean wasEventProcessed = false;

Status aStatus = status;

switch (aStatus)

{

case initialTrafficMode:

setStatus(Status.lowTrafficMode);

wasEventProcessed = true;

break;

default:

// Other states do respond to this event

}

return wasEventProcessed;

}

public boolean moderateTraffic()

{

boolean wasEventProcessed = false;

Status aStatus = status;

switch (aStatus)

{

case initialTrafficMode:

setStatus(Status.moderateTrafficMode);

wasEventProcessed = true;

break;

default:

// Other states do respond to this event

}

return wasEventProcessed;

}

public boolean highTraffic()

{

boolean wasEventProcessed = false;

Status aStatus = status;

switch (aStatus)

{

case initialTrafficMode:

setStatus(Status.HighTrafficMode);

wasEventProcessed = true;

break;

default:

// Other states do respond to this event

}

return wasEventProcessed;

}

public boolean timerGreen()

{

boolean wasEventProcessed = false;

StatusLowTrafficMode aStatusLowTrafficMode = statusLowTrafficMode;

StatusModerateTrafficMode aStatusModerateTrafficMode = statusModerateTrafficMode;

StatusHighTrafficMode aStatusHighTrafficMode = statusHighTrafficMode;

switch (aStatusLowTrafficMode)

{

case northAndSouthGreenArrow:

exitStatusLowTrafficMode();

setStatusLowTrafficMode(StatusLowTrafficMode.northAndSouthGreen);

wasEventProcessed = true;

break;

case northAndSouthGreen:

exitStatusLowTrafficMode();

setStatusLowTrafficMode(StatusLowTrafficMode.northAndSouthYellow);

wasEventProcessed = true;

break;

case northAndSouthRed:

exitStatusLowTrafficMode();

setStatusLowTrafficMode(StatusLowTrafficMode.westAndEastYellow);

wasEventProcessed = true;

break;

default:

// Other states do respond to this event

}

switch (aStatusModerateTrafficMode)

{

case northGreenArrow:

exitStatusModerateTrafficMode();

setStatusModerateTrafficMode(StatusModerateTrafficMode.northYellow);

wasEventProcessed = true;

break;

case southGreenArrow:

exitStatusModerateTrafficMode();

setStatusModerateTrafficMode(StatusModerateTrafficMode.southYellow);

wasEventProcessed = true;

break;

case northAndSouthRedM:

exitStatusModerateTrafficMode();

setStatusModerateTrafficMode(StatusModerateTrafficMode.westAndEastYellowM);

wasEventProcessed = true;

break;

default:

// Other states do respond to this event

}

switch (aStatusHighTrafficMode)

{

case northGreenArrowH:

exitStatusHighTrafficMode();

setStatusHighTrafficMode(StatusHighTrafficMode.northYellowH);

wasEventProcessed = true;

break;

case southGreenArrowH:

exitStatusHighTrafficMode();

setStatusHighTrafficMode(StatusHighTrafficMode.southYellowH);

wasEventProcessed = true;

break;

case westGreenArrow:

exitStatusHighTrafficMode();

setStatusHighTrafficMode(StatusHighTrafficMode.westEastGreen);

wasEventProcessed = true;

break;

case westEastGreen:

exitStatusHighTrafficMode();

setStatusHighTrafficMode(StatusHighTrafficMode.westAndEastYellowH);

wasEventProcessed = true;

break;

default:

// Other states do respond to this event

}

return wasEventProcessed;

}

public boolean timerYellow()

{

boolean wasEventProcessed = false;

StatusLowTrafficMode aStatusLowTrafficMode = statusLowTrafficMode;

StatusModerateTrafficMode aStatusModerateTrafficMode = statusModerateTrafficMode;

StatusHighTrafficMode aStatusHighTrafficMode = statusHighTrafficMode;

switch (aStatusLowTrafficMode)

{

case northAndSouthYellow:

exitStatusLowTrafficMode();

setStatusLowTrafficMode(StatusLowTrafficMode.northAndSouthRed);

wasEventProcessed = true;

break;

case westAndEastYellow:

exitStatusLowTrafficMode();

setStatusLowTrafficMode(StatusLowTrafficMode.northAndSouthGreenArrow);

wasEventProcessed = true;

break;

default:

// Other states do respond to this event

}

switch (aStatusModerateTrafficMode)

{

case northYellow:

exitStatusModerateTrafficMode();

setStatusModerateTrafficMode(StatusModerateTrafficMode.southGreenArrow);

wasEventProcessed = true;

break;

case southYellow:

exitStatusModerateTrafficMode();

setStatusModerateTrafficMode(StatusModerateTrafficMode.northAndSouthRedM);

wasEventProcessed = true;

break;

case westAndEastYellowM:

exitStatusModerateTrafficMode();

setStatusModerateTrafficMode(StatusModerateTrafficMode.northGreenArrow);

wasEventProcessed = true;

break;

default:

// Other states do respond to this event

}

switch (aStatusHighTrafficMode)

{

case northYellowH:

exitStatusHighTrafficMode();

setStatusHighTrafficMode(StatusHighTrafficMode.southGreenArrowH);

wasEventProcessed = true;

break;

case southYellowH:

exitStatusHighTrafficMode();

setStatusHighTrafficMode(StatusHighTrafficMode.westGreenArrow);

wasEventProcessed = true;

break;

case westAndEastYellowH:

exitStatusHighTrafficMode();

setStatusHighTrafficMode(StatusHighTrafficMode.northGreenArrowH);

wasEventProcessed = true;

break;

default:

// Other states do respond to this event

}

return wasEventProcessed;

}

private void exitStatus()

{

switch(status)

{

case lowTrafficMode:

exitStatusLowTrafficMode();

break;

case moderateTrafficMode:

exitStatusModerateTrafficMode();

break;

case HighTrafficMode:

exitStatusHighTrafficMode();

break;

}

}

private void setStatus(Status aStatus)

{

status = aStatus;

// entry actions and do activities

switch(status)

{

case lowTrafficMode:

if (statusLowTrafficMode == StatusLowTrafficMode.Null) { setStatusLowTrafficMode(StatusLowTrafficMode.northAndSouthGreenArrow); }

break;

case moderateTrafficMode:

if (statusModerateTrafficMode == StatusModerateTrafficMode.Null) { setStatusModerateTrafficMode(StatusModerateTrafficMode.northGreenArrow); }

break;

case HighTrafficMode:

if (statusHighTrafficMode == StatusHighTrafficMode.Null) { setStatusHighTrafficMode(StatusHighTrafficMode.northGreenArrowH); }

break;

}

}

private void exitStatusLowTrafficMode()

{

switch(statusLowTrafficMode)

{

case northAndSouthGreenArrow:

setStatusLowTrafficMode(StatusLowTrafficMode.Null);

break;

case northAndSouthGreen:

setStatusLowTrafficMode(StatusLowTrafficMode.Null);

break;

case northAndSouthYellow:

setStatusLowTrafficMode(StatusLowTrafficMode.Null);

break;

case northAndSouthRed:

setStatusLowTrafficMode(StatusLowTrafficMode.Null);

break;

case westAndEastYellow:

setStatusLowTrafficMode(StatusLowTrafficMode.Null);

break;

}

}

private void setStatusLowTrafficMode(StatusLowTrafficMode aStatusLowTrafficMode)

{

statusLowTrafficMode = aStatusLowTrafficMode;

if (status != Status.lowTrafficMode && aStatusLowTrafficMode != StatusLowTrafficMode.Null) { setStatus(Status.lowTrafficMode); }

// entry actions and do activities

switch(statusLowTrafficMode)

{

case northAndSouthGreenArrow:

// line 12 "model.ump"

trafficLightManager.northArrow();

// line 13 "model.ump"

trafficLightManager.southArrow();

// line 14 "model.ump"

trafficLightManager.westRed();

// line 15 "model.ump"

trafficLightManager.eastRed();

break;

case northAndSouthGreen:

// line 19 "model.ump"

trafficLightManager.northGreen();

// line 20 "model.ump"

trafficLightManager.southGreen();

// line 21 "model.ump"

trafficLightManager.westRed();

// line 22 "model.ump"

trafficLightManager.eastRed();

break;

case northAndSouthYellow:

// line 26 "model.ump"

trafficLightManager.northYellow();

// line 27 "model.ump"

trafficLightManager.southYellow();

// line 28 "model.ump"

trafficLightManager.westRed();

// line 29 "model.ump"

trafficLightManager.eastRed();

break;

case northAndSouthRed:

// line 33 "model.ump"

trafficLightManager.northRed();

// line 34 "model.ump"

trafficLightManager.southRed();

// line 35 "model.ump"

trafficLightManager.westGreen();

// line 36 "model.ump"

trafficLightManager.eastGreen();

break;

case westAndEastYellow:

// line 40 "model.ump"

trafficLightManager.northRed();

// line 41 "model.ump"

trafficLightManager.southRed();

// line 42 "model.ump"

trafficLightManager.westYellow();

// line 43 "model.ump"

trafficLightManager.eastYellow();

break;

}

}

private void exitStatusModerateTrafficMode()

{

switch(statusModerateTrafficMode)

{

case northGreenArrow:

setStatusModerateTrafficMode(StatusModerateTrafficMode.Null);

break;

case northYellow:

setStatusModerateTrafficMode(StatusModerateTrafficMode.Null);

break;

case southGreenArrow:

setStatusModerateTrafficMode(StatusModerateTrafficMode.Null);

break;

case southYellow:

setStatusModerateTrafficMode(StatusModerateTrafficMode.Null);

break;

case northAndSouthRedM:

setStatusModerateTrafficMode(StatusModerateTrafficMode.Null);

break;

case westAndEastYellowM:

setStatusModerateTrafficMode(StatusModerateTrafficMode.Null);

break;

}

}

private void setStatusModerateTrafficMode(StatusModerateTrafficMode aStatusModerateTrafficMode)

{

statusModerateTrafficMode = aStatusModerateTrafficMode;

if (status != Status.moderateTrafficMode && aStatusModerateTrafficMode != StatusModerateTrafficMode.Null) { setStatus(Status.moderateTrafficMode); }

// entry actions and do activities

switch(statusModerateTrafficMode)

{

case northGreenArrow:

// line 50 "model.ump"

trafficLightManager.northGreenAndArrow();

// line 51 "model.ump"

trafficLightManager.southRed();

// line 52 "model.ump"

trafficLightManager.westRed();

// line 53 "model.ump"

trafficLightManager.eastRed();

break;

case northYellow:

// line 57 "model.ump"

trafficLightManager.northYellow();

// line 58 "model.ump"

trafficLightManager.southRed();

// line 59 "model.ump"

trafficLightManager.westRed();

// line 60 "model.ump"

trafficLightManager.eastRed();

break;

case southGreenArrow:

// line 64 "model.ump"

trafficLightManager.northRed();

// line 65 "model.ump"

trafficLightManager.southGreenAndArrow();

// line 66 "model.ump"

trafficLightManager.westRed();

// line 67 "model.ump"

trafficLightManager.eastRed();

break;

case southYellow:

// line 71 "model.ump"

trafficLightManager.northRed();

// line 72 "model.ump"

trafficLightManager.southYellow();

// line 73 "model.ump"

trafficLightManager.westRed();

// line 74 "model.ump"

trafficLightManager.eastRed();

break;

case northAndSouthRedM:

// line 78 "model.ump"

trafficLightManager.northRed();

// line 79 "model.ump"

trafficLightManager.southRed();

// line 80 "model.ump"

trafficLightManager.westGreen();

// line 81 "model.ump"

trafficLightManager.eastGreen();

break;

case westAndEastYellowM:

// line 85 "model.ump"

trafficLightManager.northRed();

// line 86 "model.ump"

trafficLightManager.southRed();

// line 87 "model.ump"

trafficLightManager.westYellow();

// line 88 "model.ump"

trafficLightManager.eastYellow();

break;

}

}

private void exitStatusHighTrafficMode()

{

switch(statusHighTrafficMode)

{

case northGreenArrowH:

setStatusHighTrafficMode(StatusHighTrafficMode.Null);

break;

case northYellowH:

setStatusHighTrafficMode(StatusHighTrafficMode.Null);

break;

case southGreenArrowH:

setStatusHighTrafficMode(StatusHighTrafficMode.Null);

break;

case southYellowH:

setStatusHighTrafficMode(StatusHighTrafficMode.Null);

break;

case westGreenArrow:

setStatusHighTrafficMode(StatusHighTrafficMode.Null);

break;

case westEastGreen:

setStatusHighTrafficMode(StatusHighTrafficMode.Null);

break;

case westAndEastYellowH:

setStatusHighTrafficMode(StatusHighTrafficMode.Null);

break;

}

}

private void setStatusHighTrafficMode(StatusHighTrafficMode aStatusHighTrafficMode)

{

statusHighTrafficMode = aStatusHighTrafficMode;

if (status != Status.HighTrafficMode && aStatusHighTrafficMode != StatusHighTrafficMode.Null) { setStatus(Status.HighTrafficMode); }

// entry actions and do activities

switch(statusHighTrafficMode)

{

case northGreenArrowH:

// line 94 "model.ump"

trafficLightManager.northGreenAndArrow();

// line 95 "model.ump"

trafficLightManager.southRed();

// line 96 "model.ump"

trafficLightManager.westRed();

// line 97 "model.ump"

trafficLightManager.eastRed();

break;

case northYellowH:

// line 101 "model.ump"

trafficLightManager.northYellow();

// line 102 "model.ump"

trafficLightManager.southRed();

// line 103 "model.ump"

trafficLightManager.westRed();

// line 104 "model.ump"

trafficLightManager.eastRed();

break;

case southGreenArrowH:

// line 108 "model.ump"

trafficLightManager.northRed();

// line 109 "model.ump"

trafficLightManager.southGreenAndArrow();

// line 110 "model.ump"

trafficLightManager.westRed();

// line 111 "model.ump"

trafficLightManager.eastRed();

break;

case southYellowH:

// line 115 "model.ump"

trafficLightManager.northRed();

// line 116 "model.ump"

trafficLightManager.southYellow();

// line 117 "model.ump"

trafficLightManager.westRed();

// line 118 "model.ump"

trafficLightManager.eastRed();

break;

case westGreenArrow:

// line 122 "model.ump"

trafficLightManager.northRed();

// line 123 "model.ump"

trafficLightManager.southRed();

// line 124 "model.ump"

trafficLightManager.westGreenAndArrow();

// line 125 "model.ump"

trafficLightManager.eastRed();

break;

case westEastGreen:

// line 129 "model.ump"

trafficLightManager.northRed();

// line 130 "model.ump"

trafficLightManager.southRed();

// line 131 "model.ump"

trafficLightManager.westGreen();

// line 132 "model.ump"

trafficLightManager.eastGreen();

break;

case westAndEastYellowH:

// line 136 "model.ump"

trafficLightManager.northRed();

// line 137 "model.ump"

trafficLightManager.southRed();

// line 138 "model.ump"

trafficLightManager.westYellow();

// line 139 "model.ump"

trafficLightManager.eastYellow();

break;

}

}

public void delete()

{}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 

# Discussion:

Une problème durant notre labo est que nous avons essayer de réutiliser les codes umple (par exemple si le trafic légère avais un state en commun avec heavy traffic), nou ne savions pas assez sur les codes umple pour fair un tâche si compliquée. Nous avons donc copier coller les parties qui etait en commun et nous avons créé 3 machines différent pour chaque type the traffic. Nous avons ajouter un sélecteur au debut pour piger quelle type de trafic nous avons (merci a l’assistance des TAs). Le labo était très difficile comme nous étions pas confortable avec UMPLE. Après le labo nous somme maintenant plus confortable.

# Conclusion:

Le labo était vraiment le fun et nous a montré comment utiliser UMPLE et comment integrer le code de java generer. Nous somme satisfait avec la création de code UMPLE pour les traffic lights.