Seng201 Assignment 2016

Seng201 Team. Last updated May 24, 2016

Due 5pm Monday 30th May, 2016

1 Introduction

In thiſ aſſignment you are going to use a Java toolkit for use in traffic engineering applications. As well as writing some code yourself, you will also modify or use code written by others. Like all good Java software, the various classes you write may be used by a variety of clients such as graphical simulation software packages for modelling traffic flows under various conditions or by the software that controls the city's network of traffic lights.

There's quite a lot of reading here — don't be discouraged! Once you have a good understanding of the way the various aspects of the topic fit together, you will find writing your part a lot easier. We'll be releasing material in several parts: the focus of this part is domain knowledge and analysis — no Java coding is involved.

First some essential details:

- The assignment is worth 25% of your final grade and is due at 5pm on Monday 30th May, 2016.

- The drop dead date is 5pm on Friday 3rd June, 2016. Late submissions will be accepted between the due date and the drop dead date. There will be a penalty of 15% for late submission. No submissions will be accepted after the drop dead date.

This assignment is not just a programming exercise: it is an opportunity for you to demonstrate a range of software engineering skills. These might include analysis, modelling, design, testing and documentation as well as coding. It is typical in software engineering projects for the domain to be one you may not be expert in, and for requirements to be vague, incomplete or contradictory — we'll also exercise our communication skills!

2 How intersections work

Overview

We all spend a considerable part of our lives travelling on our country's roads. As city dwellers[1], most of us spend much of our travelling time at *intersections*. As you complete this assignment, you will get to think more deeply about intersections and learn a little[2] about how they are controlled. We hope your future journeys will be much more interesting as a result.

To begin with, we need to gather some terminology and definitions. Intersections — places where roads meet or cross — come in many different shapes and sizes (geometries). Figure 1 on the following page shows the two most common geometries — the familiar 'T' and '+' intersections.

There are many variations on these basic geometries. The roads may not meet at right angles or there may be more than two intersecting roads.

The simplest kind of intersections are *uncontrolled* and vehicles move through the intersection in accordance with the basic road rules governing right of way. Uncontrolled intersections are now quite rare in urban areas and are only suitable for low traffic volume locations. The next level of sophistication involves the use of signs to reinforce the rules (as on an intersection with Give Way signs at all approaches) or to override them (as in the case of Stop signs on a minor road where it joins a major one).

Other options include the use of roundabouts (also known as traffic islands or rotary intersections) and "free turn" lanes to increase traffic flow while relying on basic road rules. We will not consider these further in this assignment.

We will only be considering *controlled intersections* where electronic signalling devices —

[1] at least during term time

[2] but probably enough to convince you that computer scientist and not traffic engineer is the career for you!

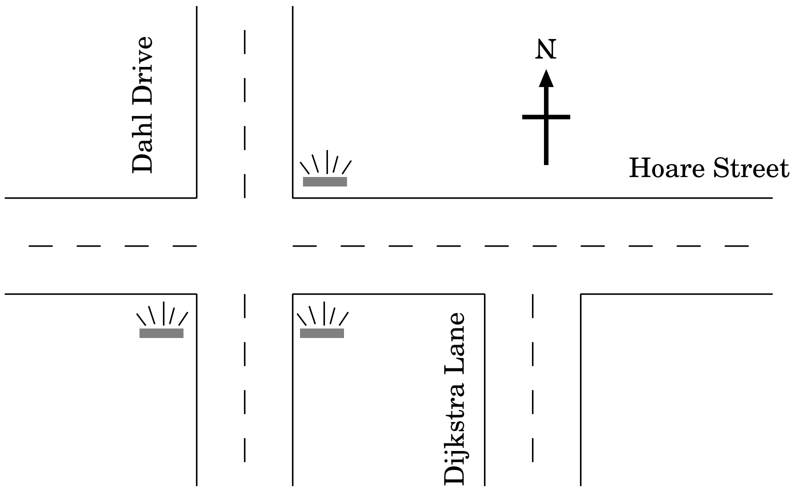


Figure 1: Common intersection geometries

traffic lights — are used to enforce a plan for giving the right of way in turn to vehicles approaching the intersection from different directions.

Traffic streams

We can describe what happens at an intersection in terms of a number of *traffic streams.* The traffic approaching an intersection along a particular road may be regarded conceptually as being made up of several separate streams. Whenever we approach an intersection we have (usually!) already selected a traffic stream that fits our intended movement. Examples might be "I'm in the right turn lane" or "I'm in the straight ahead or left turn lane". Figure 2 on the next page shows six common streams seen at intersections — see how many you can spot on your way home. These 6 streams are for traffic approaching from one particular direction. The streams for traffic approaching from other directions are simply those shown in Figure 2 on the following page rotated by the appropriate amount.

Many streets are divided into a number of parallel *lanes* by road markings and sometimes physical features such as median strips. Lane markings are usually designed so that they correspond to particular traffic streams (e.g. north-bound through traffic AND west-bound left turning traffic). Detectors embedded in the road are unable to tell whether vehicles are intending to continue straight through or make turns. In this case the stream is regarded as consisting of the combination of the possible options. Detectors are considered further in later sections.

For example, traffic approaching the intersection of Hoare Street and Dahl Drive from the south (see Figure 1) could consist of up to three traffic streams: through traffic continuing north on Dahl Drive, left-turning traffic continuing west on Hoare Street and right-turning traffic continuing east on Hoare Street. These could be accommodated in different ways depending on the number of lanes available at the intersection: by having three separate traffic streams (types (a), (b) and (c) in Figure 2 on the next page); by having two traffic streams (types (c) and (d)) or (types (b) and (e)); by having one traffic stream (type (f)).

Sometimes particular potential traffic streams are not available — such as a turn onto a one-way street or continuing straight ahead at a 'T' junction — at an intersection.

Traffic lights

At a controlled intersection, there are a number of *signal heads* mounted on poles or suspended above the roadway. Each signal head has a number of *signal faces*, each consisting of a number of individual *lights*. Each light has a single colour and may be on or off[3]. The lights are arranged in groups. For simplicity, we will assume that these are always groups of three:

[3]We'll ignore flashing states for now

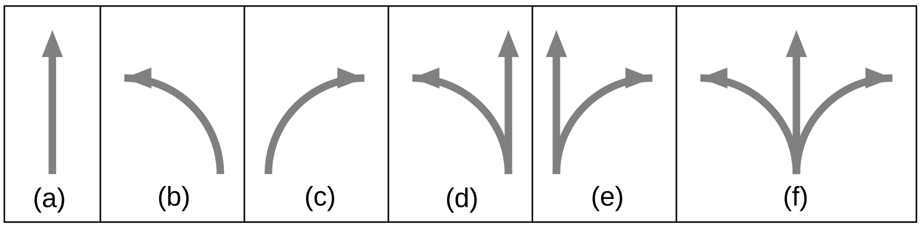
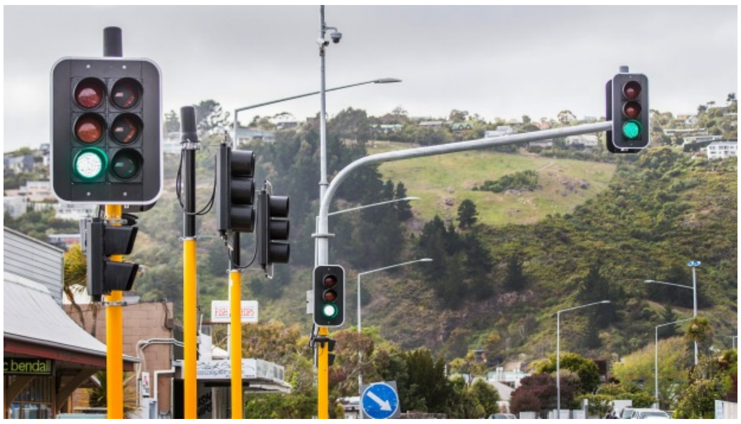
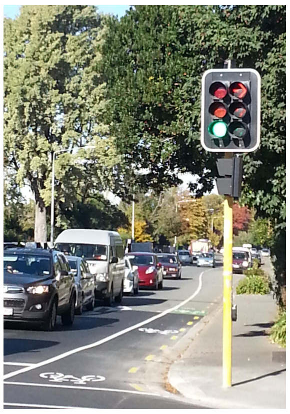


Figure 2: Some common traffic streams



(a)



(b)

Figure 3: Signal heads and signal faces

in practice other configurations (such as a single green right turn arrow light) occur. Some examples are shown in Figure 3.

The most common signal face configuration consists of red, amber & green circular lights controlled so that only one member of the group is on at any time (e.g. when the green light is on, the yellow and red lights in the same group are off). Another very common configuration is a group of three (red, amber and green) arrow lights: these are usually controlled so that at most one light is on at any time.

Variations include through arrows in red, amber & green as well as bus, tram, pedestrian and cycle lane signals.

Each signal face is positioned so that is is clearly visible to approaching vehicles in the relevant traffic stream(s). Each traffic stream may have several signal faces. Three is the usual number — check for yourself! Figure 1 on the preceding page shows the signal faces visible to the south-bound traffic streams approaching the intersection of Dahl Drive and Hoare Street from the north. Each face has a *location* relative to the intersection centre and an *orientation* indicating the direction it is facing. The locations of the signal faces in Figure 1 on the previous page are NE, SW and SE. All faces for a particular stream will usually face the same direction — all three face north in this case.

Signal faces can be shared between traffic streams. In practice, a "through and left" stream (type (d) in Figure 2) and a "through and right" stream (type (e)) would probably share a signal face for controlling through traffic movement. Figure 3(b) shows a signal face which can control types (c), (d) and (f) depending on the states of the round lights ((green, yellow or red) and right arrow (green, yellow, red, off).

Phases & phase plans

Each controlled intersection has a *cycle* consisting of a number of *phase*s. The intersection has a *phase plan* which specifies the order of the individual phases. Each phase is carried out in turn until the cycle is complete, at which time the cycle then begins again. You may have noticed that some intersections may have more than one phase plan. For example, a "rush hour" plan might include protected right turns and shorter phases while an "off peak" plan might omit protected turns and have longer phases.

In any particular phase the intersection's lights are switched into a pattern allowing one or more traffic streams to proceed (by giving them green lights) and preventing others from moving (by giving them red lights). For example, one phase might permit both north-bound and south-bound through traffic on Dahl Drive; another might permit north-bound and south-bound through traffic on Dahl Drive as well as left-turning traffic onto Hoare Street.

Figure 4 on the next page shows one possible cycle for the intersection of Dahl Drive and Hoare Street. The cycle consists of four phases. Each phase involves two traffic streams from the basic set shown in Figure 2 on the preceding page.

1. The appropriate combination of red and green lights is set to permit north-bound and south-bound through traffic on Dahl Drive as well as left turns onto Hoare Street. No right turns are allowed in this phase. No traffic on Hoare Street is permitted to move in this phase.

2. Now the appropriate lights are changed to stop the north and south bound through traffic and traffic turning left onto Hoare Street. In addition, further light changes permit right turns from Dahl Drive onto Hoare Street. Traffic on Hoare Street is still not permitted to move in this phase.

3. Now it is the turn of the traffic which has been waiting on Hoare Street. The appropriate lights are changed to prevent right turns onto Hoare Street and to prevent any traffic on Dahl Drive from moving. East-bound and west-bound traffic on Hoare Street is permitted as well as left turns onto Dahl Drive.

4. Finally, all streams are stopped except for right turns from Hoare Street onto Dahl Drive.

At the end of phase 4 the cycle begins again.

Four-phase intersections are quite common — see how many can you spot on your way home. A 2-phase intersection has a very simple plan but you may also encounter many more complex phase plans.

Now let's consider the details of the light changing process as one phase ends and another begins. The transition between phases is not immediate — there is a *change interval* which includes the *yellow interval* where amber lights are displayed to allow traffic already committed to crossing the intersection to be cleared and an *all red interval* where red lights are set to prevent any further incoming traffic from the traffic streams active in the phase and to provide a safety margin[4]. Only when the all red interval has elapsed can the next phase be permitted to begin allowing further traffic movement. Typical yellow intervals are 3–4 seconds long and typical all-red intervals are about 1 second. Is this consistent with your observations?

For simplicity, we will treat change intervals and all-red intervals as separate phases in this assignment.

Controlling intersections

Many intersections have detectors buried beneath the road surface in order to detect the approach of vehicles in the corresponding lanes (and hence traffic streams). Some cycle lanes also feature detectors. Detectors come in two types. Modern (active) detectors notify the intersection controlling software when a vehicle arrives. The intersection may then decide to switch to a phase which will *service* the corresponding traffic stream. Older style (passive) detectors record vehicle arrivals but the intersection must *poll* them to interrogate them about the data they have recorded. Polling involves asking each detector in turn if it has detected any vehicles.

Controlled intersections can be classified into three major groups according to the way they (*via* their phase plans) use information provided by detectors.

Pre-timed (unactuated) intersections have a pre-determined *phase plan*. The order and length of each phase, and hence the total cycle length, is fixed in advance. The phase plan may depend on factors such as time of day (e.g. shorter phases at rush hour) or day of the week (e.g. longer phases in weekends). However, no adjustment is made for

[4]allowing for people who go through red lights or are slow to clear the intersection area

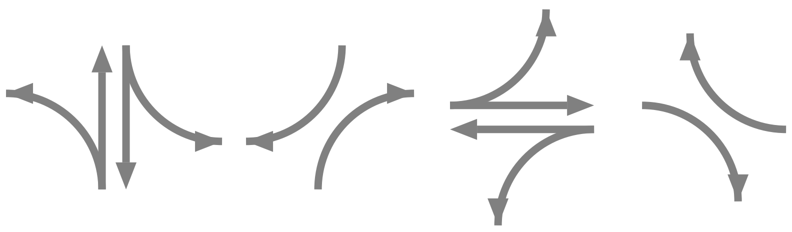


Figure 4: Four phase cycle for Dahl Drive and Hoare Street intersection

actual traffic conditions (as reported by detectors).

Semi-actuated intersections are often found where a minor road crosses a major road. The phase plan for such an intersection will typically remain in a *default phase* giving priority to the major road, until a vehicle is detected in a traffic stream on the minor road. The intersection responds by switching to a phase where the traffic streams on the minor road are serviced. This approach maximises throughput on the major road while giving traffic on the minor road a fair go. Both the cycle length and green times may vary from cycle to cycle in response to demand.

Fully-actuated intersections have all phase changes initiated in response to detector actuations. The phase sequence is specified, as are minimum (and sometimes maximum) green times for each phase. Cycle lengths and green times may vary from cycle to cycle. Some phases may be optional (i.e. minimum green time = 0) and may be omitted if no demand is sensed by the corresponding detectors. An example would be skipping right turn only phases (such as phases 2 & 4 in Figure 4) if no vehicles have entered right turn only lanes.

The ultimate in sophistication is linking intersections into *zones* with the same cycle lengths. This enables *platoons* of vehicles to travel through a sequence of intersections without being stopped. You may have noticed this technique used on the one-way street system around Christchurch.

A local example

Figure 5 shows an example of how we'll think about intersections. It shows the intersection of Clyde & Creyke Roads with Kotare[5] Street — this is at the north-east corner of the block containing the campus so (as we did) you can observe it for yourself.

Traffic entering from each direction sees signal faces consisting of 3 round lights. Traffic arriving from the north also sees a signal face consisting of 3 right arrow lights (see Figure 3(b). Overhead lights are ignored, as is the "free turn" for traffic turning left from Creyke Road into Clyde Road.

The active phase plan consists of seven phases. In the first phase all traffic arriving from the east and west may proceed (i.e. traffic streams of type (f) in Figure 2), with right-turning vehicles giving way as usual, while traffic arriving from the north or south must wait. In Figure 5 the state of each signal face is indicated by the corresponding colours[6]. For example, the black "lights" indicate that the arrows are all off in phases 6 and 7.

3 Some simplifying assumptions

We will make a few assumptions in order to make the problem simpler.

- All intersections involve one north-south road meeting an east-west road.

[5] Kingfisher

[6] In reality, these are magnets on Neville's whiteboard. Our colleague Richard Green suggests that Smarties^{TM} or Pebbles^{TM} would be a convenient, portable and edible alternative.

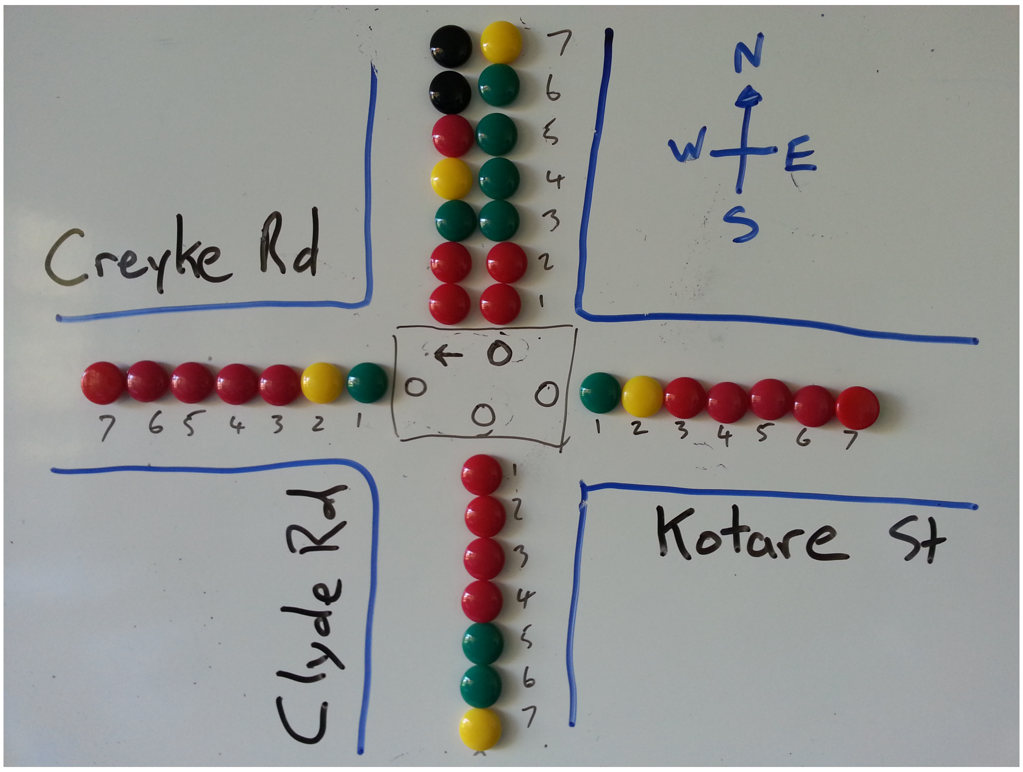


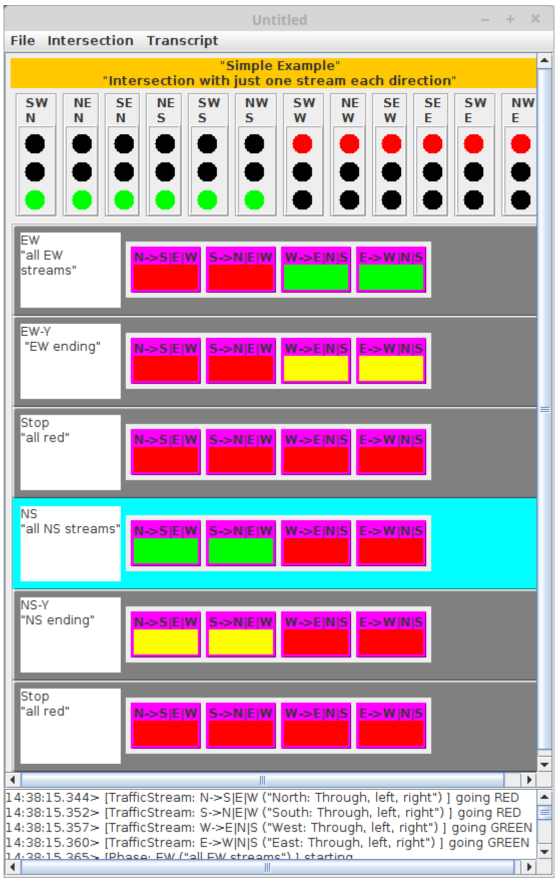
Figure 5: Phase plan for intersection of Clyde Road, Creyke Road and Kotare Street

- Signal faces are oriented to the north, south, east or west and are located at the north-west, north-east, south-west or south-east corners of intersections. No overhead signal faces need be considered.
- Semi-actuated intersections need not be considered. If necessary, we can always add a SemiActuatedPhasePlan class later to cope with these.
- Signal faces contain 3 lights: one of each colour and all of the same shape.
- For "round" signal faces, green lights are always followed by yellow lights which are followed in turn by red lights in each cycle. We need not consider other sequences — such as those in countries where another yellow interval occurs between red and green.
- Change intervals and all-red intervals are treated as separate phases.
- Only passive detectors are used. These can report, when asked, whether they have seen any vehicles pass over them.
- No phases are optional.

4 Getting started

You should be able to begin the following activities already and you will find them helpful preparation for the next part which will be available soon.

- Make sure you read the description (§ 2 on page 1) of the way intersections work and that you understand it fully.
- Observe some real intersections in action. You can start this right away. Identify the signal faces, traffic streams and phases.
- For the active phase plan, draw up a table listing the phases and the state of each stream in the phase.
- Note which signal faces are associated with each traffic stream. Remember that signal faces might shared between streams.
- Using Smarties^{TM}, Pebbles^{TM} or other tools, construct a diagram such as that shown in Figure 5 to show the lights displayed in the signal faces in each phase.



```
14:38:15.344> [TrafficStream: N->S|E|W ("North: Through, left, right") ] going RED
14:38:15.352> [TrafficStream: S->N|E|W ("South: Through, left, right") ] going RED
14:38:15.357> [TrafficStream: W->E|N|S ("West: Through, left, right") ] going GREEN
14:38:15.360> [TrafficStream: E->W|N|S ("East: Through, left, right") ] going GREEN
14:38:15.365> [Phase: EW ("all EW streams") ] starting
```

Figure 6: Simple intersection monitor

- Time some phases. Do you think your intersection is unactuated or fully-actuated?
- Do you think it has more than one phase plan? For example, is there any difference between rush hour and off peak plans?

5 Our toolkit

Figure 6 shows a very simple intersection monitor — an instance of IntersectionMonitor. Near the top is a panel showing the intersection's signal faces, together with their locations and orientations.

The monitor displays a "row" for each phase — the active phase is highlighted. As well as the phase description, there is some information about each stream and a colour chip showing the state of each stream (remember that signal faces can be shared between streams) in the corresponding phase.

A transcript provides a log of the activity at the intersection. Each entry has a time stamp (traffic.misc.TimeStamp) and the transcript can be saved in a file for subsequent analysis.

Given an Intersection, an IntersectionMonitor can display information about its state. The traffic package contains classes (such as TrafficStream which can be used to assemble an intersection.

Intersections may also be constructed by reading intersection descriptions from a file. Figure 7 shows the description file corresponding to the intersection of Figure 6.

- The file contains a number of *tags*, such as `<SignalFaces>` — valid tags are found in traffic.Tags.
- Lines beginning with `//` are treated as comments and empty lines are ignored.
- Within a tag body each line contains one or more fields. The fields are separated by tabs (which are shown underlined in Figure 7)[7]. Why tabs? The idea is to make it easy to use a Scanner to parse the content of the line. Using commas as separators seems appealing but ultimately the hassle of escaping commas inside field content is frustrating...

The file structure is as follows:

Intersection Label and description for the intersection.

TrafficStreams Label and description for each TrafficStream

PhasePlan One or more `<Phase>` tags

Phases Each line contains

1. Phase label
2. Phase description
3. The state of each traffic stream in this phase. An X indicates that the stream is not included and R, Y, and G indicate the corresponding colours
4. The duration (in seconds) of the phase[8].

SignalFaces Each line contains

[7]Some editors will sneakily replace tabs with spaces so you might need to check for that.

[8]Here we're assuming a PretimedPhasePLan — later you can add some suitable variation to allow FullyActuatedPhasePlans to be included

```
<Intersection>
Simple Example␣Intersection with just one stream each direction
</Intersection>

// TrafficStreams
<TrafficStreams>
N->S|E|W␣␣North: Through, left, right
S->N|E|W␣␣South: Through, left, right
W->E|N|S␣␣West: Through, left, right
E->W|N|S␣␣East: Through, left, right
</TrafficStreams>

// One or more PhasePlans. Each involves all intersection streams
<PhasePlan>
// Phases, stream states during phase, and phase duration
<Phases>
EW␣all EW streams␣RRGG␣␣5
EW-Y␣␣ EW ending␣RRYY␣␣2
Stop␣␣all red␣RRRR␣␣1
NS␣all NS streams␣GGRR␣␣2
NS-Y␣␣NS ending␣YYRR␣␣2
Stop␣␣all red␣RRRR␣␣1
</Phases>
</PhasePlan>

// SignalFaces and the streams they observe.
// location, orientation, kind, streams
<SignalFaces>
SW␣N␣␣STANDARD␣␣N->S|E|W
NE␣N␣␣STANDARD␣␣N->S|E|W
SE␣N␣␣STANDARD␣␣N->S|E|W

NE␣S␣␣STANDARD␣␣S->N|E|W
SW␣S␣␣STANDARD␣␣S->N|E|W
NW␣S␣␣STANDARD␣␣S->N|E|W

SW␣W␣␣STANDARD␣␣W->E|N|S
NE␣W␣␣STANDARD␣␣W->E|N|S
SE␣W␣␣STANDARD␣␣W->E|N|S

SE␣E␣␣STANDARD␣␣E->W|N|S
SW␣E␣␣STANDARD␣␣E->W|N|S
NW␣E␣␣STANDARD␣␣E->W|N|S
</SignalFaces>
```

Figure 7: Intersection description file

1. The location of the corresponding SignalFace. Legal values are defined by traffic.TrafficDirection.
2. The orientation of the corresponding SignalFace. Legal values are defined by traffic.TrafficDirection.
3. The kind (`STANDARD`, `LEFT_ARROW` or `RIGHT_ARROW`) of the signal face.
4. The labels of the streams (tab separated) that the face is associated with.

6 Next steps

Now that you have completed the activities in § 4 you are ready to start working with intersection data.

- Obtain a copy of the intersection monitor demonstrated in class — from the Learn assessment page where you found this document.
- Run it and familiarise yourself with the way it displays information about elements such as phases, streams and signal faces.
- The File menu contains an item to run a working demo, plus one to read files, such as the one in Figure 7, in the format described in § 5.
- The Intersection menu can be used to start and stop the intersection's phase plan cycle. The signal faces will change to reflect the current phase.
- Information about the changing states of the intersection appears in the transcript pane. The Transcript menu includes an item which allows the transcript content to be saved for later analysis.
- Write an intersection file for each intersection you observed as described in § 4 and check that it is correct by loading and running it.

7 What to do now

Now that you understand how intersections work, and how to describe them in files, it's time to write some Java code to implement some features of an intersection monitor.

The classes you write will be based on the description in Section 2 on page 1 and individual class specifications provided in the form of `javadoc` documentation as described below. Together, they should form the basis of a toolkit for traffic engineering.

You are provided with `javadoc` documentation for the classes that you must complete as well as for the other classes in the system.

You are also provided with skeleton/incomplete versions of the classes which you will need to complete.

You will need to implement private properties and methods in some classes,but do not change the public API of any class. If you think you need to do this then please discuss it with us.

You will find it helpful to use the analysis & modelling techniques covered in class, and the various kinds of UML diagrams will help you firm up your designs.

- traffic.diy.ModelIntersection.java — skeleton for the ModelIntersection class. You should complete its methods to show how a complete Intersection is made from the other provided classes (such as TrafficStream and Phase).
- traffic.diy.MyIntersectionMonitor.java — incomplete intersection monitor. This is like the monitor you have seen already, but has several menu items (highlighted in pink) on the File menu for you to complete.
- Save intersection... allows the Intersection being monitored to be saved in serialised form. It should be stopped before being saved.
- Similarly, Load saved intersection... allows the an Intersection previously saved saved in serialised form to be loaded.
- My Demo causes the monitor to display an Intersection built using a method of your ModelIntersection class.
- Open... causes the monitor to load an intersection description from a file in the format shown in Figure 7 and described in § 5. You will need to complete MyIntersectionLoader to implement this feature fully.

7.1 Using the resources provided

- You will find the javadoc in the api folder.
- Create an Eclipse project (e.g. myassignment).
- Create a package traffic.diy in the `src` directory. This is where the code you will write/modify goes.
- Add the Java source files from the skeleton folder to the package you created. These contain the sources for classes MyIntersectionMonitor,ModelIntersection and MyIntersectionLoader.
- In the Project → Properties dialog, add the JAR file `traffic-library-1.2.jar` to the build path (using the Add External JARs... button).
- You should now be able to run the application (there's a main method in MyIntersectionMonitor).

8 Submission & Assessment

This section covers the final details including submission instructions. Please make sure you have read it, and clarified anything you're not sure about with us, well before the due date.

9 Assessment

Marks will be awarded for successful completion of features as well as general software engineering quality.

- Even if some of your code doesn't compile it may still be worth some marks so submit all your work. If you know some features don't work then you could note that in your report.
- We'll test each of the features according to the specifications. We'll use our own data files as well as those you have submitted.
- Having determined how well your classes *work*, we'll examine your source code to see how well it is designed and written. Relevant factors might include:
  - Meaningful identifiers, naming conventions.
  - Documentation (e.g. javadoc and other comments).
  - Technical quality (use of properties, methods, algorithms, ...).

9.1 Extensions

If you have completed more than the basic tasks then you may have used more than one PhasePlan type and wondered how best to demonstrate your work.

For example, the "My Demo..." menu item in MyIntersectionMonitor calls one of *`preTimedIntersection()`* or *`ullyActivatedIntersection()`* from the ModelIntersection class. Similarly, maybe you can load both kinds from intersection description files.

In such cases it's fine to modify the monitor to demonstrate both. For example, you could add another menu item so that you have "My PreTimed Demo ..." and "My FullyActuated Demo..." menu items that call the appropriate methods. Alternatively, you could make a submenu (as we saw in class) under "My Demo ..." with an entry for each type.

It isn't expected that you write any additional elements (classes, interfaces, enums) — and we encourage you to discuss your plans with us before going ahead — but if you do then these should be placed in the traffic.diy package.

9.2 Submission checklist

- The code you have written (and perhaps also modified or extended) should all be in the traffic.diy package. This should be exported from your Eclipse project (using File →Export →General →FileSystem ...).
- Intersection description files should be placed in a folder called *data*. Each file should have an appropriate name and extension (it doesn't matter what it is — I just used `.dat`).
- Similarly, saved (i.e. serialised) intersections should be placed in the data folder — an appropriate extension (e.g. `.ser`) should be used to distinguish them from intersection description files.
- If you have implemented JUnit tests for the code you have written then these should be included in the traffic.diy package.

- Include a brief report explaining what you have done. This shouldn't be more than a page and it's main purpose is to make sure we don't overlook anything you have done that we should know about. You might include:
  - Your name.
  - Brief description of any extensions (such as those mentioned in § 9.1
  - instructions for using your system
  - explanations of the way you designed your intersection description files and chose the intersections they model.
  - thoughts on design challenges you faced (e.g. choice of collection types)
  - UML or other diagrams that helped you etc.
  - ...
- ZIP up all these files to a suitably-named archive and submit the result (e.g. `nic11-neville-churcher-seng201.zip`) via Learn.