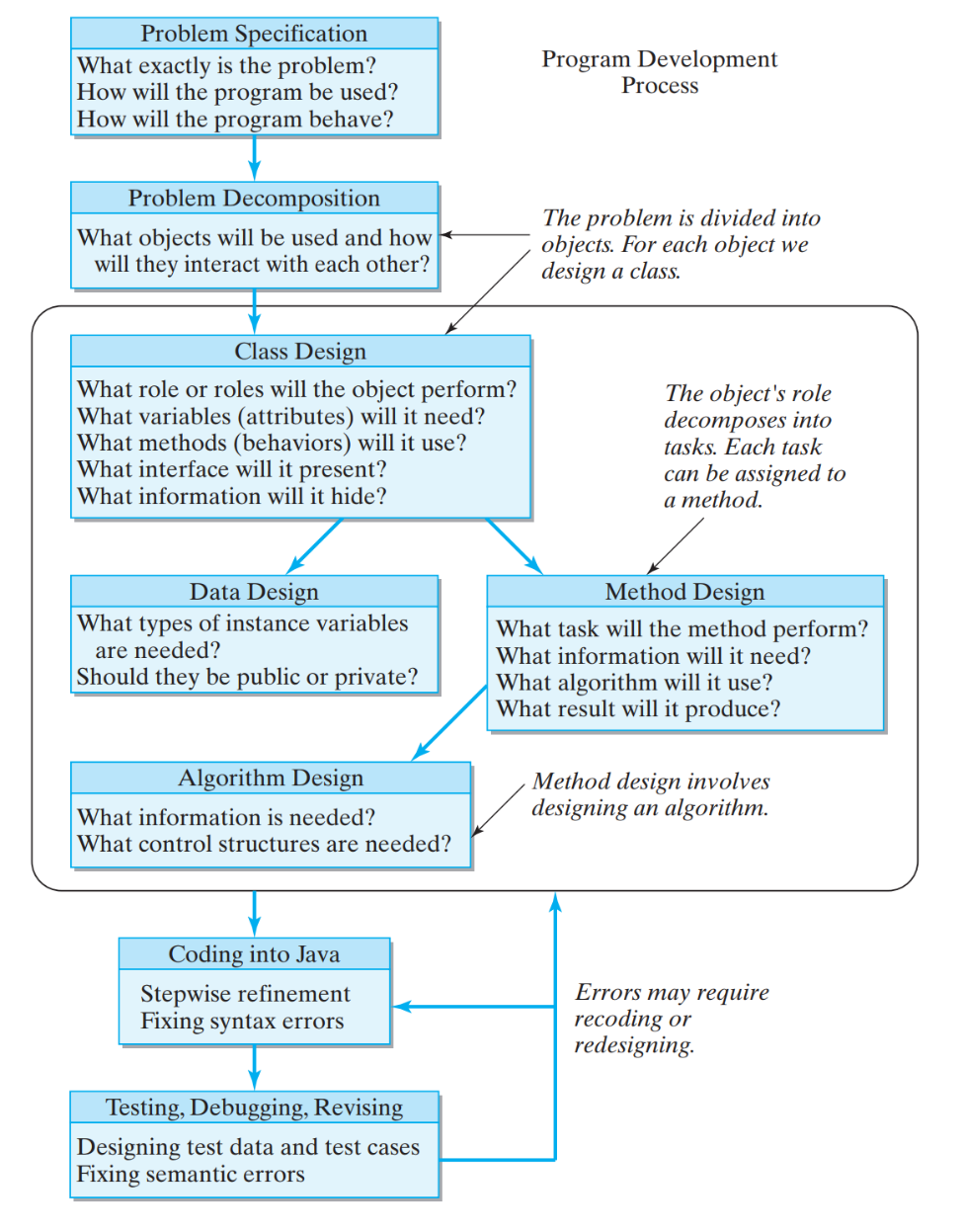
CP2406 2020 SP53 – Initial Project Brief

# Overview

Using the software development process illustrated below, students plan, design and develop a **Car Traffic Simulator** in Java using IntelliJ IDEA connected to a GitHub repo (**share this with your practical class supervisors!**).



# Expected Outcomes

* Each student incrementally produces their own project plan, design and solution
* The project plan consists of a **working document** based on the software development process – the document incrementally answers questions about the problem specification, decomposition, class design, data design, method design and algorithm design
* The design is represented as UML class diagrams using **Violet UML editor**: <http://alexdp.free.fr/violetumleditor>
* Each student uses a GitHub repo to incremental record changes to the project plan, UML design, and Java source code

# The working document

To plan out the software development process, your working document is expected to describe the following:

* **Problem specification:** What exactly is the problem? How will the program be used? How will the program behave?
* **Problem decomposition using UML class diagrams:** What objects will be used and how will they interact?
  + Divide the problem into objects
  + The UML class diagrams should answer the following design questions

**Class design:**

* + 1. What role(s) do objects of this class perform?
    2. What member fields do objects of this class need? Should they be public or private?
    3. What methods do objects of this class need? Should they be public or private?

**Method design:**

1. What should its method signature be?
2. What task will it perform? What algorithm will it use?

You are not expected to produce a complete development report – that’s why it’s called a “working document”. However, you need to at least produce the following outcomes for this first part of the project work:

1. Produce a working document that illustrates your designs for approximately 4-6 classes – one of which will be the **Main** class that contains the **main()** method
2. The classes should be drawn in a UML class diagram – the diagram should include information about class names, member field access control, method access control and class relationships (you get marks for using the correct UML notation for these things)
3. **Pick at least 3 classes and start implementing them** – include appropriate test classes that check and verify that these 3 production classes are working as expected - there is no need to create a test class for the **Main** class!
4. You get marks for ensuring that your class code matches your UML class diagrams – there is no need to include the test classes in your UML class diagrams!

For example, let’s say that you create some classes that represent objects such as a car, a traffic light and two road objects (this is a very simplified version of the entire problem domain). Each road might start out as one-way only. The outcome of your program could be to have the car object move from the first road object to the second road object. You could place the traffic light object at the end of the first road object.

You might treat the **Main** class as the simulator for the time being. The simulator needs to run a sequence of looped updates to change the state of the car and the traffic light. While the simulator is running, the car moves along the road and the traffic light changes colour. If the car reaches the traffic light at the end of the first road then it must stop. When the traffic light turns green, then the car moves onto the second road and continues to move until it reaches the end of that road.