Victoria University of Wellington
School of Mathematics, Statistics and Computer Science

COMP205: Software Design and Engineering

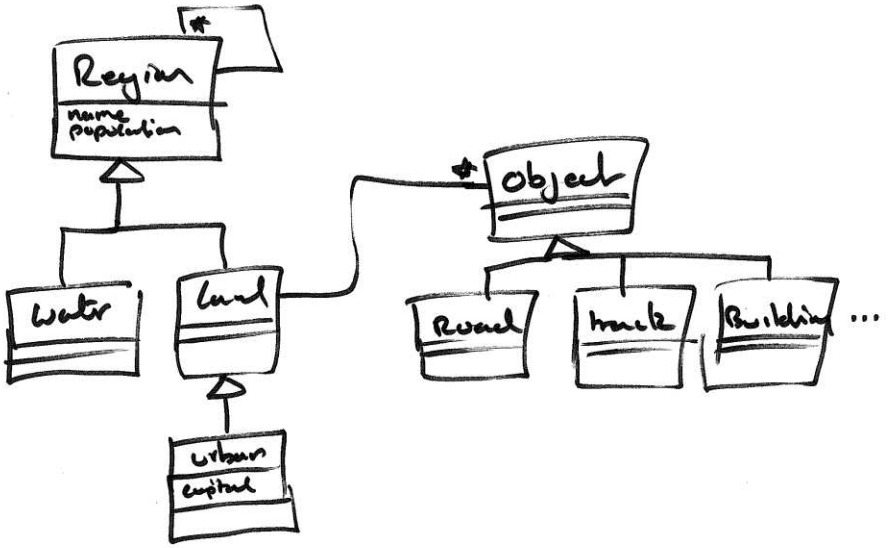
Terms Test Solutions

1 Question 1

a) The key nouns are: map, region, globe, continent, country, sea, area, water, urban, land, name, population, capital, object, road, railway, track, building, river. The nouns in brackets (i.e. town, city, North Island, Wellington) are not key nouns, since they are just used to illustrate the meaning of the text.

b) The top five candidate classes are: region, water, land, urban, object.

c) A good UML solution looks something like the following:



The key points are: areas are regions (i.e. they are not distinct concepts); regions can be in regions (to an arbitrary depth); land areas, water areas and urban areas are all subclasses of region; land areas have objects inside them; region and urban areas have attributes (note, you can implement the capital attribute as a subclass of urban area and there is not a lot of difference between the two).

2 Question 2

There really are a lot of problems with this code. The main ones are:

- The UML defines a "print" method, yet this does not appear in the code.
- The `Vehicle` class defines a `Person` field, yet this is not in the UML (and it's never initialised).
- Neither `Car` nor `Truck` are actually defined to be classes! (i.e. they are missing the `class` keyword).
- The `Car` class does not extend `Vehicle`, as indicated in the UML. Rather, it has a `Vehicle` field which is rather poorly named.
- The `Car` constructor passes four parameters into `super`. Firstly, `super` has no meaning since `Car` does not extend anything (hence this will not compile). Secondly, if `Car` did extend `Vehicle`, then only three (not four) parameters can be passed to `super`.
- The `ITEMS` field is inconsistently named. According to the COMP205 coding standard, only constants should be named all in capitals. Thus, the field should be named "items" instead.
- The `ITEMS` field is not initialised, which will result in a `NullPointerException` whenever the `addItem` method is called.
- The UML declares that a truck can contain many items. But, the `ITEMS` field does not look like a collection. Therefore, a collection should be used instead (e.g. `List<Item> items;`).
- The `Truck` constructor does not use its third parameter `nw`. Furthermore, it passes too few arguments to its super constructor, meaning the code will not compile.
- None of the methods in the `Truck` class are public, so it's impossible to create or use any instances of `Truck`!

A few people said that `numberOfDoors` was too long a name for a field. I do not agree with this as, in COMP205, we advocate field names that are descriptive (and these certainly are).

Some people also said that the `Truck` constructor has to be public. Again, this is not really true. A constructor does not have to be public and there are many cases where it makes sense to have private constructors (e.g. the singleton pattern). But, at least one method must be protected or public otherwise we can't really use the class!

People also said that the variables `nd`, `ns` and `nw` were poorly named. With this I do agree, but did not give marks for it, since I felt there were more important problems to mentione.

3 Question 3

The six problems with this code were:

Line 3 "`Map<TO,Set<FROM>> ins = new HashMap<TO,Set<TO>>()`"
should be "`ins = new HashMap<TO,HashSet<FROM>>()`".

This is apparent from the declared type of `ins` and from the way it is used throughout the code.

Line 8 "`Set<Object> from(FROM f)`" **should be** "`Set<TO> from(FROM f)`".

This is apparent from the element type of `outs` (i.e. `Set<TO>`), which is being returned by this method. Also, recall that `Set<FROM>` is **not** a subtype of `Set<Object>`.

Line 16 "`ins.put(t,new HashSet())`" **should be** "`ins.put(t,new HashSet<FROM,TO>())`". Again, this is apparent from the element type of `ins`.

Line 20 "`FROM size()`" **should be** "`int size()`".

The method clearly returns an `int` (not "anything" as declared above) since `c` is an `int` and is used as an `int`.

Line 28 "`class Attends extends HashRel`"
should be "`class Attends extends HashRel<Student,Course>`".

This is perhaps not entirely obvious. Certainly, `HashRel` requires two type parameters be provided (since it is declared to take two, namely `FROM` and `TO`). To conclude that the values are `Student` and `Course`, you need to look carefully at how the `add` method is implemented. This accepts a `Student` and a `Course` and makes uses `HashRel.add` on them.

Line 37 "`... printFrom(Object s, HashRel<Object,Object> r)`" needs to be fixed. To see why, recall that e.g. `HashRel<Integer,Integer>` is `not` a subtype of `HashRel<Object,Object>`. Thus, the print method does not work "for ANY HashRel" as the comment states it should. There are three solutions:

i. "`... printFrom(Object s, HashRel<? extends Object,? extends Object> r)`"

ii. "`<FROM,TO> static void printFrom(FROM s, HashRel<FROM,TO> r)`".

iii. You can declare the printer class to take two type parameters `FROM` and `TO`.

I also awarded marks for observing a typo in the line "`c += outs.size()`", which was meant to be "`c += out.size()`".

4 Question 4

To see the answer to this, type the code into Eclipse and run it!