SWEN3145 - Software Modelling

Modelling Exercise 8

1 Purpose

The purpose of this session is for you to practice writing pre- and postconditions for operations.

2 Purpose

The purpose of this session is for you to:

1. derive attribute(s); and
2. add operations, including pre- and postconditions.

to UML class model.

3 Instructions

Some specific rules are given below.

1. The modelling exercise in this tutorial may be completed in groups of 2-3 students[1] – you may also choose to work on your own. The deliverables are the same whether you are in a group or working on your own.
2. Submit a *.zip* file with your answers to tasks 2 and 6 from Section 5 – it should contain a pdf file for your answer to tasks 2 and 6, a *.use* and its associated layout files with your answers for tasks 3 to 5, and object model command and their associated layout files for Task 6. Put the names of the people you worked with on the first lines of the pdf file — names that are missing from the file will not get the marks assigned to the submission or be sent feedback. Only one group member needs to make a submission on OurVLE.

4 Description

We have continued to discuss the *Robot* application during the lecture hours. The class diagram in Figure 1 depicts journeys. Since there are operations, other than query operations, in the diagram, it is now considered a design model and not a requirements model. The *VehicleRate* shows the rate for journeys for a particular vehicle. The rate is given per unit of distance. For example, if for a Robot $R1$, its rate is \$55 per 0.25 (km); when *roundDistance* is set to *true*:

1. the *unit* value indicates that the distance should be rounded to the nearest 0.25km; and
2. the cost will always be a whole number multiple of the *perUnitCost*.

When *roundDistance* is set to *false* there is no rounding of the distance, and the cost is not constrained to be a whole number multiple of the *perUnitCost*.

[1] This group can be different from your project group and be different from week to week during the semester.

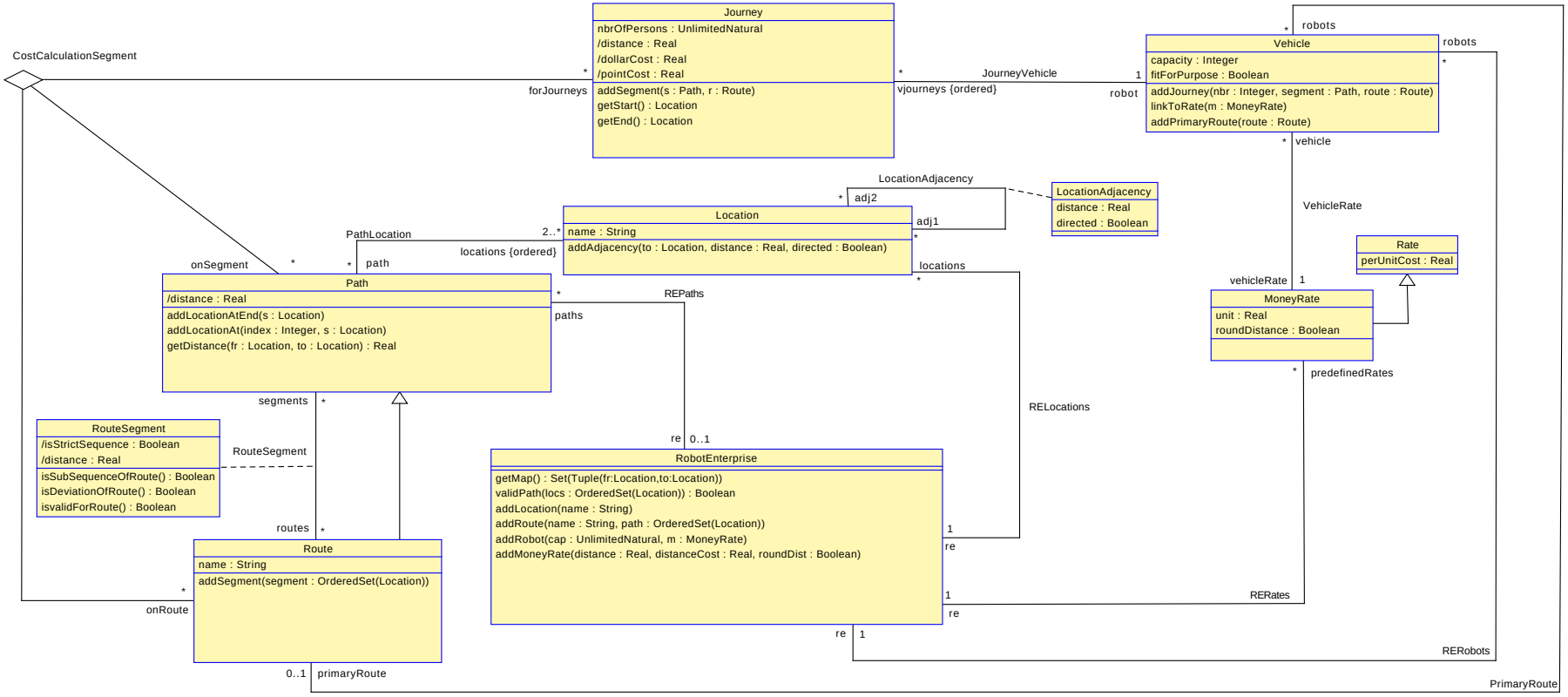

Figure 1: Robot Design Class Diagram for Journeys

5 Tasks

1. Download the *.use* and specification from OurVLE.
2. Construct a description of the Robot application as modelled in Figure 1, including additional structural invariants that have not been specified in the starting specifications. In your description you must make a decision on:
   a) whether vehicles are allowed to make journeys along routes that are not their primary routes – you should note that specifying a primary route is currently optional, but if one is specified for a vehicle, what impact should it have on what is allowed – it should have an impact, for example, you could say:
      i. after they have made x number of journeys for the month, they can go to other routes;
      ii. for the first half of the month they must stick to their primary route;
      iii. if they have defined a primary route, they must stick to it at all times; or
      iv. some other operational constraint as you determine.
   b) how to deal with journeys that are deviations from the route they are on – some additional information is below:
      i. a deviation, when made by the customer, could be for many reasons and it does not have to be that all the passengers for that journey disembark on a deviation;
      ii. currently, even for deviations, the distance is calculated as if no deviation is made; and
      iii. one thought for payment is to include who "requested" the deviation, customer or driver, and if it is customer, some additional modelling need to be done to capture this and what should be paid.

   Take some time to see what is already modelled so that your description is in keeping with what is already done. Save your answer in a *.pdf* file.
3. In the *Journey* class, write an OCL expression to correctly derive the dollar figure for the *cost* of the journey; since there is an expression to derive the *distance*, you do not need to recalculate it, so if there is a valid distance in the the *distance* attribute of the *RouteSegment* class, you may use it in your calculations – your decision for task 2b will also influence the cost calculation.
4. Move the $isSubSequenceOfRoute(...)$ and the $isDeviationOfRoute(...)$ query operations from the *RouteSegment* association class to the *Route* class and update the $notSequenceAndDeviationAtTheSameTime$ invariant and the $isvalidForRoute(...)$ query operation to still use them.
5. Choose two (2) of the non side-effect free operations, except $addLocation(...)$, $addRoute(...)$, or $addJourney(...)$ add and for each write:
   a) an English description of its precondition - you may separate the precondition into subclauses;
   b) an English description of its postcondition - you may separate the postcondition into subclauses;
   c) an OCL expression for each of its precondition subclauses; and
   d) an OCL expression for each of its postcondition subclauses.

   Enclose the English description of the pre- or postcondition in comments with the OCL expression. If there is no pre- or postcondition for an operation specify *true*, i.e.,

   $pre\ noPre : true$

   or

   $post\ noPost : true$

   to specify that there is no precondition or postcondition respectively - the $noPre$ and $noPost$ are labels and are similar to naming invariants.

6. Choose one of the operations from Task 5 and outline the test criteria (predicate, clause, an active for each of the clauses, i.e., GACC, CACC, or RACC, and an inactive for each of the clauses, i.e., GICC or RICC) and concrete tests cases (object models) to test its pre- and postconditions.

   Keep in mind that if for your operation you have:

   pre pre1: a or b
   pre pre2: c implies d
   pre pre3: e

   the predicate for the precondition should be:

   (a or b) and (c implies d) and e

   i.e., it contains five (5) clauses with the parentheses used to enforce the priority of the logic operators.