Project "Object-Oriented Programming"
Part 2 of 3

Prof. Eric Steegmans

Academic Year 2018–2019

1 Introduction

This text describes the second part of the project of the course 'Object-Oriented Programming'. Some aspects of part 1 of the assignment will change in this second part. This has to be reflected in your code.

Preferably, the project is worked out in the same team as the first part, but it is allowed to change the composition of your team. All changes to team composition must be reported to Prof. Steegmans before April 1, 12 am. One of the team members needs to send the mail with his/her partner in CC. A student that breaks up a team is responsible for notifying his/her ex team member! If you work in the same team as in the first part, you don't have to report anything.
After the deadline mentioned above, we no longer accept team changes. However, if problems between team members arise, the team can split. In this case, Prof. Steegmans needs to be notified as soon as possible and each team member must work out the rest of the project individually.

If you work individually, some parts of the assignment do not have to be worked out. This is indicated at the relevant places throughout the assignment.

2 Assignment

The second part of the assignment contains a number of changes to the class of roads and adds concepts like locations and routes. Elements of part 1 that are not explicitly mentioned by this part of the assignment do not change, although they can be influenced indirectly.

2.1 Locations

As we have seen in Part 1, each road spans between two separate end points. In Part 2, these points are described more elaborately by their own class *Location*. Locations are characterized by the following properties.

Students working alone on the project must not work out locations. They may stick to end points as introduced in the assignment for the 1st part of the project.

2.1.1 Coordinates

Similar to the specification in Part 1, a location is described with two coordinates (latitude and longitude) in order to express its geographical location. The coordinates are represented by *decimal degrees* using two double precision floating-point numbers (Java's `double` type). Both the latitude and the longitude must be finite numbers. They do not change once a location has been created.

The methods related to coordinates are worked out nominally.

2.1.2 Address

Every location has a particular address, which consists of at least two characters and can only contain letters, digits, commas and spaces. The address of each location must start with a capital letter. For example, *"Celestijnenlaan 200A, 3001 Heverlee"* is a valid address. It is possible that in future versions of the application the spelling rules will have to be extended to allow additional characters in addresses. However, the characters mentioned above will never be omitted as valid characters. The address of a location may change during its lifetime.

The methods related to addresses are worked out totally. **Tip**: Study the method `matches` in class `String`, and regular expressions used in it.

2.1.3 Adjoining Roads

Each location can have adjoining roads. These are all the roads for which the given location is one of the end points. At any point in time, a location can exist without any adjoining roads. Furthermore, new roads can be constructed and existing ones can be demolished. Moreover, locations themselves can also be destroyed.

All public methods regarding adjoining roads should be worked out defensively. The non-public methods are worked out nominally. Your class must offer at least a method to get all adjoining roads for a location, as well as a method to check whether a given road is one of the adjoining roads for a given location. That method must return its result in (nearly) constant time.

2.2 Road

The most important change to roads with respect to the first part is related to the introduction of locations. The uniqueness of road identifications also changes slightly.

2.2.1 Identification

In this version, there is no longer a limit on the total number of roads. Moreover, as soon as a road has been destroyed, its identification can be re-used for other roads.

2.2.2 End points

Every road still spans in between two end points, lying on the surface of the earth. From now on, these end points are represented by two locations. The area in which the coordinates of the end points of roads are located is still limited to the positive quadrant and their latitude nor their longitude exceed a maximum of 70 degrees. It should not be possible to change this maximum value during the run-time of the application. However, it must be possible to change the maximum in future versions of the application without having to change any other classes. The maximum value always applies to all coordinates of any point.

The methods related to the end points are worked out nominally. Your class must include a method that returns both end points at the same time.

2.3 Route

A route is a traversable trajectory that connects a start location with an end location. Routes are characterized by the following properties. You are free to work out the methods for routes in a nominal way, in a total way or in a defensive way. You may also work out some methods (or parts of them) in one way, and other methods (or parts of them) in another way.

2.3.1 Start Location

Each route has a start location. Once a route has been created, its start location cannot be changed anymore.

2.3.2 Segments

Each route has one or more segments that represent the different roads that define the complete trajectory. In order for a route to be a valid interconnected chain of roads, each segment must share one mutual location with the previous segment and one with the next segment. However, there are two exceptions. For the first segment, one of its two locations is the start location of the route. The first segment therefore has only one mutual location, namely, with the second segment. Likewise, the last segment only has a mutual location with the second-to-last one, as its other location is the end location of the route.

For an existing route, segments can be added, removed or replaced. However, the set of segments resulting of these alterations must still form a valid route as described above. Furthermore, it must be possible (1) to ask for the total length of a route, (2) to ask for the start location and the end location of a route, (3)

to ask whether the route is traversable, i.e. whether all roads are unblocked in the direction to be taken, and (3) to ask for the sequence of locations visited by a route when traveled from the start location to the end location.

2.4 Main Program

The functionality of the developed classes needs to be demonstrated by means of the following scenario:

1. Instantiate four distinct locations and create three roads that connect these locations. One of the roads is a narrow country road, where the average speed is only 8 m/s. At this point, none of these roads are blocked or have any delays.

2. Create an extra parallel road between the locations that are already connected by the country road. This redundant road is a motorway. Make sure that it is longer, but faster than the country road, because of a higher average speed. For comparison, print the properties and travelling time of the two parallel roads to the standard output stream.

3. Create a route involving all four locations. The route must use the motorway to go from one of the locations to the other. Print the length of this route, and all visited locations on the standard output stream.

4. Block the motorway in the direction taken by the route. Show on the standard output stream that the given route is no longer traversable.

When you find it useful and/or necessary, you may write more details to the standard output stream. However, all write operations should be located in the main program.

3 Practical Guidelines

1. Only the imposed requirements need to be fulfilled. You will not make a better impression by developing extra elements that are not part of the assignment.

2. Details that are not explicitly specified or mentioned in the assignment may be worked out according to your own inspiration. When doing so, always go for the simplest solution.

3. The project needs to be worked out in Java 10 or higher.

4. When the assignment specifies that a particular property needs to be worked out nominally, totally or defensively, the manipulation of this property in more complex methods such as constructors also needs to be worked out accordingly.

5. All aspects of the class of *roads* must be documented both formally and informally. All aspects of the class of *locations* must be documented only in a formal way. If you want to get a score of 16 or more, you must document all aspects of the class of *routes* in a formal way. If you do not want to go for such a high score, you must not work out any documentation for the class or routes.

6. In addition to the definition of all classes, you must also work out a UML diagram of the design of your navigation system. This diagram should contain all classes, except for self-defined exception classes and test classes. The diagram must contain the name of each class and all its attributes. The diagram must not contain any methods. You may choose any tool for making the diagram (a hand-drawn diagram is also allowed). You must submit this diagram (in PDF format) along with your code.

3.1 Handing in

1. The second part of the project must be handed in **by both students of each team before May 5 at 9:00 am**. Deviations from this deadline will only be allowed in exceptional cases.

2. Submitting your solution is done via Toledo. You have to submit your solution as a **JAR** file. Creating a JAR file in the Eclipse environment is performed via the 'Export' function that is found in the 'File' menu. Be sure to select the source files of all classes that you have developed together with the class diagram in PDF. Mention your name and study programme in the comment field when submitting.

If certain aspects of this assignment are still unclear, further details will be posted on Toledo. Specific questions concerning the assignment can only be sent to Prof. Steegmans.

Good luck!