PLD Agile

Duration of the PLD: 8 sessions of 4 hours

1 Description of the application

You must design and implement an application for optimising delivery tours in cities. For more sustainable cities, these deliveries are done with bicycles.

When launching the application, the user loads an XML file which describes a city map. This file gives the list of all intersections (such that each intersection has a latitude and a longitude) and the list of all road segments (such that each road segment has an origin intersection, a destination intersection, a name and a length in meters). The XML file also specifies the address of the warehouse, from which couriers start their tours. Three XML files are available on Moodle. When a map is loaded, the application displays it.

By default, the number of couriers is set to 1, and the user may change this number. Initially, there is no delivery request.

When the user receives a new delivery request (by e-mail, SMS, phone, etc), he selects a courier, the location of the delivery and its time-window. All time-windows are one hour long time intervals that start at 8, 9, 10, or 11 a.m. Each time a new delivery request is entered, the system computes the best possible tour corresponding to the current requests associated with the selected courier. This tour must start from the warehouse at 8 a.m., visit each delivery during its time-window, and return back to the warehouse. If the courier reaches a delivery point before the beginning of its time window, he has to wait for the beginning of the time-window; if he arrives after the end of the time-window, the tour is not valid. We assume that the travel speed of all couriers is constant and equal to 15 kilometres per hour. We also assume that the time needed to perform each delivery is equal to five minutes. The best tour is the tour that minimises the arrival time on the warehouse. If there is no tour that satisfies all time-window constraints, then the application asks the user to select another courier; if there is no other courier, then the delivery request is rejected.

The application displays the tour of each courier on the map; it also displays for each delivery, the address, the arrival time and the departure time.

The user may save the current tours in a file, and he may also restore a set of tours from a file.

2 Organisation

The project is done by groups of five to seven students, using an iterative development process. The project will be composed of at least two iterations, and the first iteration will be 4 sessions long. For the next four sessions, you may choose to do between one and four iterations.

First iteration: This iteration corresponds to the inception phase. The goal is to identify the main use cases, design a first architecture of your application, and implement some use cases in order to have a first minimal but operational software.

Deliverables that are required at the end of the first iteration are:

- Glossary;
- Use case diagram;
- Brief format description of the main success scenario of all identified use cases;
- Structured description of all use cases that have been analysed and/or implemented during the first iteration;
- Class and package diagrams designed during the first iteration;
- Actual planning of the first iteration (time spent by each member of the team on each activity).

You will organise a demo with the client at the end of this first iteration (during the fourth session).

Next iterations: At the beginning of each iteration, you will choose some use cases (or use case scenarios) that will be analysed, designed and /or implemented, and you will estimate the time needed for each of these activities. At the end of each iteration, you will compare this planning with the actual time spent on each activity.

The deliverables that are required at the end of the PLD are:

- Document where you explain architectural choices and used design patterns;
- State-Transition diagram of your application (where transitions correspond to user events);
- Code of your application, including unit tests;
- Javadoc documentation (or a similar documentation if you don't use Java);
- Class and package diagrams (reverse engineered from your code), with an explanation of the differences between these diagrams and those designed during the first iteration;
- A report on test coverage (automatically generated);
- Initial planning and actual planning of each iteration;
- A short discussion (at most one page) on social and environmental issues related to your application;
- Technical and human review of the PLD.

Development environment: We recommend using Java for implementing the application. If you wish to use another object oriented language, you must first discuss your choice with us. You must use an IDE and tools for automating unit tests, for evaluating test coverage, for controlling versions, and for automatically generating online code documentation. You may use the Eclipse IDE[1] and its plug-ins: ObjectAid[2] (for reverse engineering diagrams from code), and JUnit[3] (for unit tests).

You may use the demo version of StarUML[4] to draw UML diagrams.

The online documentation of your code will be generated with JavaDoc, and you will apply the style guide of Oracle[5].

To compute a tour, you may use the Java code available on Moodle. The class *TSP1* implements the most basic (and naive) variant of algorithms studied in AAIA. You may implement more efficient variants by overriding methods *bound* and *iterator*. You may also implement another solving approach (dynamic programming or tabu search, for example), or use open source libraries (provided that you credit authors of these libraries!).

[1] `http://help.eclipse.org/juno/index.jsp` (see the Java Development User Guide)
[2] `http://www.objectaid.com/`
[3] `http://www.junit.org/`
[4] `http://staruml.io/`
[5] `http://www.oracle.com/technetwork/java/codeconventions-150003.pdf`