Final Project Phase II

1 Your Tasks

1.1 Task 1 - Update your CRC Model to accomodate client's new requirements

To keep up with the competition, your client has increased the number of products available for sale and in addition, they are building distribution centers across main cities. Further, the company has its own transportation capabilities between distribution centers, so they can move merchandise quickly from one distribution center to another. They would like to introduce a new delivery model as follows: the ordered items would ship from the delivery center which is closest to the buyer's address.

Please update your model so for each product, you add the capability to maintain the stock across multiple distribution centers. Each distribution center will be labelled by the city where is located. Also the buyers (customers) have a delivery address where one of the attributes is certainly the city.

There is one restriction however. The final delivery to a buyer is done via courier, and the courier servers all the cities through his fleet of cargo aeroplanes which fly certain routes that connect all cities, however there are cities who have no direct flight. So our task is to ship from the distribution center closest in distance to the buyer, using the courier fly routes. For example, consider the following scenario. We have two distribution centers, one in the city A, and another one in the city B. The buyer is located at C. The courier however has cargo service on the following routes: [A,C], [A,D], [B,D], [D,E], [E,C]. Clearly there is more than one route from A to C and also there is more than one route from B to C. Due to the specifics of the flights, even there is a direct route from A to C, it may not necessarily be the shortest one. Assuming the delivery cost is proportional to the distance (you may assume, for example, the delivery costs 1 cent per kilometer; that is an 1000 km distance would translate to a cost of 10 dollars), your task is to compute the optimal delivery charge for each order and include it in the invoice.

In order accomplish this task, you should make use of undirected graphs. You can assume if there is a flight from A to B, there is also a flight from B to A and the distance is the same. You may represent the graph in the way that is easiest for you. That is you can make useof the Graph interface we have developed for E2, but certainly you don't have to. It is important that you do not brute force the optimal route - brute force solution will not receive full marks. Pick your favorite shortest path algorithm and implement it.

In addition, you should add new features:

- A logged in administrator mustbe able to add new distribution centres.
- If a new distribution centre is added, every product must automatically add the distribution center, with an available quantity zero for that particular center.
- A customer must be able to save his shopping cart when he decides to exit the application before using the checkout feature.
- Meanwhile the quantity in the shopping cart must NOT be available to the other potential buyers.
- A customer must be able to cancel his shopping cart.
- A logged in administrator must be able to maintain the shipping routes.

Your group must update your CRC model for the problem described. Instead of handing in your index cards, create a `pdf` file named `crc.pdf` in your team's subversion repository in folder `PII`. This file should contain a collection of your diagrams that define each of your CRC cards just as you did for the sprint I.

Keep in mind that you are designing the backend, not GUI screens. None of the CRC cards should mention buttons, text fields, or other graphical components. In later phases of the project, you will implement both the backend and the frontend, but for now you are designing justthe backend.

1.2 Task 2 - Implement the back end

In this sprint, you must complete the development of the back end. Add new files if necessary, create new classes, update existing classes, according to you needs. Everything should be synchronized with your CRC model.

1.3 Task 3 - API

In this sprint, your will develop an Application Program Interface (API) so we can test using a common set of methods. The starter code is available on the course web site and balckboard as well. The class name is `Project` (file `Project.java`). Please not add/remove anything from `Project.java`. Strictly complete the code where indicated and make this class part of your main package.

1.4 Task 4 - The software development process

Your team should meet regularly while working on the project. You are required to have two types of meetings: **planning** meetings and **status** meetings.
You need to have two planning meetings: one in the beginning of the project and one mid-way through the project phase. During a planning meeting, the team will (a) recap on the current state of the project (if mid-way meeting), (b) decide on a set of tasks the team will accomplish before the next planning meeting, and (c) decide who will perform which tasks.
In addition to the planning meetings, the team will meet for weekly status meetings. (You may have more frequent status meetings if you wish.) During status meetings, each member will report on (a) what (s)he has accomplished since the last meeting, (b) what (s)he plans to accomplish before the next meeting, and (c) if there are any problems or obstacles that prevent him/her from making progress. To demonstrate the software development process the team followed, you need to maintain a plain text fi

le called `meetings.txt`, where the team will record all meeting minutes. A sample of meeting minutes is posted on the course web site and on blackboard.
On the day of each meeting, commit this

file into the directory for this phase in your team repository. The contents of this

file must match the state of the rest of your repository! If a member repeatedly does not meet deadlines or misses meetings, contact your instructor immediately.

2 Marking

All of these items will affect your grade:

- Coding style and quality (1%)
  - Simple and clear code, clear logic.
- Functionality and usability of the application (2%)
  - All functions from the feature list (starting from the `ProjectInfo` up to this document) must be implemented.
  - Stability of the application (e.g. should not crash on invalid input).
- Javadoc (1%)
  - The quality of the documentation of the code.
- Quality of the software development process (1%)
  - The file `meetings.txt` must be committed according to the schedule.

  - The contents of the repository and the state of the code must match the contents of `meetings.txt`.
  - The file `crc.pdf` must reflect the state of the software.
  - Subversion commit history must show participation by all team members and frequent commits.

- Please note the whole phase II is worth 4% of your final grade.

3 Checklist

Have you ...

- committed your work to your team repository?
- committed updated `crc.pdf`?
- committed `meetings.txt`?
- used `svn list` and `svn status` to verify that your changes were committed?