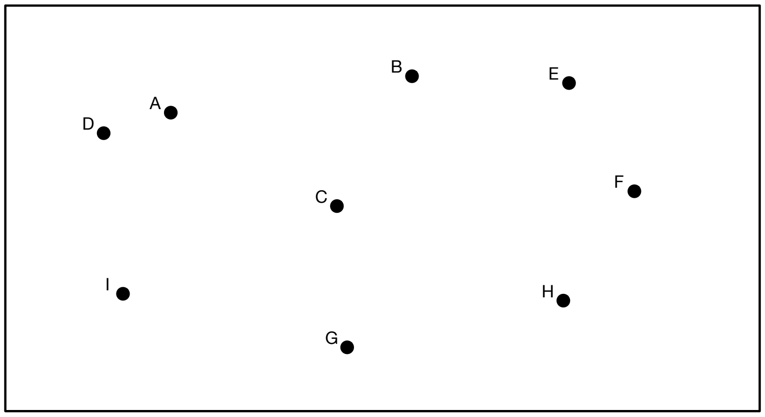
CS150: Programming Project 2 (Fall 2021) corrected

Contents

1 Project Description

Create a transport simulation on a two-dimensional map measured in miles. The simulation will model trucks of different sizes moving between different locations, picking-up shipments in one location and dropping them off in another location. There are no roads specified, instead trucks can move from one point to another in a direct and incremental fashion.



The simulation is broken into hours of time, such that, each hour, trucks are either moving towards a destination, picking up a shipment at a specified source location, unloading a shipment of goods at a specified destination, or waiting for an empty loading dock at a destination to unload a shipment. Similarly, each hour the different locations will be tracking the situation at the loading docs and picking the next truck for empty loading docs.

Once the simulation is configured from a file, a central clock object will tick off the hours executing the truck move method in turn, pulling from a queue storing the trucks. To ensure that the trucks implement the correct methods the following interface will be implemented. Similarly, every location will be visited to perform any needed actions and record any required information.

```
public interface Schedule {

  // Called each hour, allowing the object to perform an action.
  public void action();

  // Will store the object's current information into a log file.
  public int log_status();
}
```

Both trucks and warehouses will be represented as an object of the appropriate class. When the simulation starts, warehouses and trucks will be randomly created. Warehouses will have between one and three loading docs, randomly chosen, where trucks can drop-off and receive shipments. Trucks will come in three different sizes for carrying one, two, or three loads of cargo. But trucks that can move three loads will only move 3 miles per hour, two-load trucks travel 6 miles per hour, and one-load trucks move 9 miles per hour. At the start of the simulation, trucks are created randomly around the map empty and with a manifest of shipments. The manifest will be randomly determined with each entry containing a source of the shipment and a destination. When a truck picks up a shipment it will consist of a single load, and trucks are allowed to carry loads for multiple destinations as long as they do not exceed the carrying capacity.

The simulation will be finished when the last truck has made its last delivery. When the simulation is finished the program needs to perform an analysis of the data and indicate how efficient that particular model run was. The simulation will need to run with 4 different configurations, each with 10 different random seeds. Each configuration should be greatly different from the others.

When a truck reaches a warehouse for pickup or drop-off it must wait for a loading dock to become empty. If there are other trucks waiting for a loading dock, then the truck must wait its turn. Thus, trucks must access loading docks in the order of arrival.

A class will need to be created for drawing a dynamic map showing both the warehouse and truck locations. How to draw to a graphics window is discussed in the book. The dynamic map will keep each object type in a generic linked-list data structure. Both the truck and warehouse objects will implement the following interface so that the dynamic-map can store the items in a common data structure.

```
public interface Render {

  // This will draw the object on the canvas.
  public void draw(Graphics g);
}
```

Once complete, your simulation will need to generate a report on truck and warehouse activity recording what actions were taken and how efficiently the system ran.

2 Functional Design Requirements

Your program must implement the following requirements.

1. All specified interfaces must be implemented.
2. All data structures will be designed and implemented by the programmer and must be implemented as **generic** linked-list. Thus, the Java API can not be used for your datastructure.
3. There will be three different types of trucks that will inherit from a parent class.
4. The design must be well divided across several classes where each class has one specific functionality. These will be specified two weeks into the assignment period.
5. Along with the class specification, a report will be provided specifying what data will be tracked in the simulation and how the system will define simulation efficiency.
6. Input configuration will be stored in a file and read into the simulation at the beginning of the model run. There should be a logging file, capturing all activity coming out of the system and can be used for debugging. All output data will be written to data files for later examination.
7. A final report will be provided that will compare and contrast the 40 different model runs.

3 Some Advice

- **Design first!**
- **Get something small to work, then make it more complex.**

- **Break the functionality into parts and test those parts.**
- **For the multiple model runs, this can be automated with a class that runs the model for different configurations.**

4 Project Calendar

Date	Week	Day		Description
10/13	7	1	wed	Project Distributed
10/14	7	2	thr	
10/15	7	3	fri	
10/16	7	4	sat	
10/17	7	5	sun	
10/18	7	6	mon	
10/19	7	7	tue	
10/20	8	8	wed	Three questions on project submitted to Moodle.
10/21	8	9	thr	
10/22	8	10	fri	Question FAQ
10/23	8	11	sat	
10/24	8	12	sun	
10/25	8	13	mon	
10/26	8	14	tue	
10/27	9	15	wed	Java Document API design in BlueJ project, plus a document describing the ways your simulation will track itself and how you will compute efficiency.
10/28	9	16	thr	
10/29	9	17	fri	Design feedback.
10/30	9	18	sat	
10/31	9	19	sun	
11/1	9	20	mon	
11/2	9	21	tue	
11/3	10	22	wed	Functional Code Submission (compiles and some functional code).
11/4	10	23	thr	
11/5	10	24	fri	Code submission feedback.
11/6	10	25	sat	
11/7	10	26	sun	
11/8	10	27	mon	
11/9	10	28	tue	
11/10	11	29	wed	
11/11	11	30	thr	
11/12	11	31	fri	
11/13	11	32	sat	
11/14	11	33	sun	Final Submission to Moodle (varied by used submission delays).

October 13th – The project will be available on Moodle, with Moodle submissions for each stage.

October 20th – Submit three questions on the project, and these will be used to create a Frequently Asked Questions (FAQ). Once the FAQ has been created the professor will answer questions on the project during lecture. **Worth 5%** of grade.

October 27th – **Java Document API Design** will be submitted as a BlueJ project and generated from the classes contained within the project. Also a document describing the different statistics that will be tracked and how you will compute efficiency. **Worth 10%** of grade.

November 3rd – **Functional Code Submission** will be submitted as a BlueJ project. The project will compile and will appear to work, to a reasonable degree. What this means is that the program may not run perfectly, but it will compile and run without error with a provided text file. An explanation of how the program will be run must be provided in the README file, at least 75% of the program will be unit-test, Javadoc will provide useful documentation, the code will be organized, and there will not be any useless methods or comments created by BlueJ. **Worth 10%** of grade.

November 14th – The rest of the deliverables will be provided on this day and will have the following grade break down: Program Functionality (**worth 15%**), Unit-testing (**worth 10%**), Design Requirements (**worth 20%**), Final Report (**worth 15%**) and Comments/Organization (**worth 15%**).