CSCI 1933 Project 4

International Shipping Simulation

Note: The project is due on **Friday, April 21st** by **11:55 PM**.

Instructions

Please read and understand these expectations thoroughly. Failure to follow these instructions could negatively impact your grade. Rules detailed in the course syllabus also apply but will not necessarily be repeated here.

- **Due:** The project is due on **Friday, April 21st** by **11:55 PM**.
- **Identification:** Place you and your partner's x500 in a comment in all files you submit. For example, `//Written by shino012 and hoang159`.
- **Submission:** Submit a `zip` or `tar` archive on Moodle containing all your `java` files. You are allowed to change or modify your submission, so submit early and often, and *verify that all your files are in the submission.*

  Failure to submit the correct files will result in a score of zero for all missing parts. Late submissions and submissions in an abnormal format (such as `.rar` or `.java`) will be penalized. Only submissions made via Moodle are acceptable.
- **Partners:** You may work alone or with *one* partner. **Failure to tell us who is your partner is indistinguishable from cheating and you will both receive a zero**. Ensure all code shared with your partner is private. Only one project per group.
- **Code:** You must use the *exact* class and method signatures we ask for. This is because we use a program to evaluate your code. Code that doesn't compile will receive a significant penalty. Code should be compatible with Java 8, which is installed on the CSE Labs computers. We recommend to not use IDEs for your implementations.
- **Questions:** Questions related to the project can be discussed on Moodle in abstract. This relates to programming in Java, understanding the writeup, and topics covered in lecture and labs. **Do not post any code or solutions on the forum**. Do not e-mail the TAs your questions when they can be asked on Moodle.
- **Grading:** Grading will be done by the TAs, so please address grading problems to them *privately.*

Code Style

Part of your grade will be decided based on the "code style" demonstrated by your programming. In general, all projects will involve a style component. This should not be intimidating, but it is fundamentally important. The following items represent "good" coding style:

- Use effective comments to document what important variables, functions, and sections of the code are for. In general, the TA should be able to understand your logic through the comments left in the code.

  Try to leave comments as you program, rather than adding them all in at the end. Comments should not feel like arbitrary busy work - they should be written assuming the reader is fluent in Java, yet has no idea how your program works or why you chose certain solutions.

- Use effective and standard indentation.

- Use descriptive names for variables. Use standard Java style for your names: `ClassName`, `functionName`, `variableName` for structures in your code, and `ClassName.java` for the file names.

Try to avoid the following stylistic problems:

- Missing or highly redundant, useless comments. `int a = 5; //Set a to be 5` is not helpful.

- Disorganized and messy files. Poor indentation of braces (`{` and `}`).

- Incoherent variable names. Names such as `m` and `numberOfIndicesToCount` are not useful. The former is too short to be descriptive, while the latter is much too descriptive and redundant.

- Slow functions. While some algorithms are more efficient than others, functions that are aggressively inefficient could be penalized even if they are otherwise correct. In general, functions ought to terminate in under 5 seconds for any reasonable input.

The programming exercises detailed in the following pages will be evaluated for code style. This will not be strict – for example, one bad indent or one subjective variable name are hardly a problem. However, if your code seems careless or confusing, or if no significant effort was made to document the code, then points will be deducted.

In further projects we will continue to expect a reasonable attempt at documentation and style as detailed in this section. If you are confused about the style guide, please talk with a TA.

Introduction

Due to increasing tuition prices, Goldy Gopher has decided that he needs to make a little extra cash along side being the school mascot. Being the aspiring business-gopher he is, he has decided to found Gopher Shipping Inc. for all your international shipping needs. After talking to his friends and fellow Big Ten mascots, he has come up with quite the array of vessels to transport shipments. These include:

- 100 Canoes from Coach P.J. Fleck
- 10 Yachts from Brutus Buckeye
- 20 Galleons from Sparty
- 15 Barges from Purdue Pete
- 10 Freighters from Herbie Husker
- 5 Airplanes from Willie the Wildcat
- 30 Carrier Pigeon Teams from Herky the Hawk
- 10 Rockets from Elon Musk

However, having never done this before, he doesn't know which transports to use or how often they should leave port. He has tasked you with writing a simulation in order to figure this out.

For each type of vessel, Goldy wants you to figure out to what capacity a vessel should be filled before departing (**C**) and how long it should wait at port before departing if it is not full (**W**). For example, a **C** value of 0.9 and a **W** value of 2 would mean a vessel would wait until it is 90 percent full or for 2 days to pass before departing.

In order to simplify his business model, Goldy has told you that he only wants one type of vessel and that a vessel will only hold shipments to one location. This way vessels only have to worry about traveling to one destination and his employees only need to be trained how to drive a single type of transport vessel.

Setup

We will assume the following adjustable system parameters:

- A variable type of transport vessel.

- A variable load percentage before departure.
- A variable total wait time before departure.

We will assume the following constraints:

- Shipments will be passive, but they will contain data such as:
  - Time they arrived at the port
  - Destination/Port they need to be shipped to
  - Weight (1-1000Kg)
- Shipments have a value of 10 dollars per Kilogram
- Arrival of a shipment at a port will be provided by an arrival class that will generate arrivals for a single port (determined statistically, see below), each port will have an instance of this class
- If a shipment cannot be put onto the current vessel, it will wait for the next one

Simulation of the system is achieved by creating classes to model the **behavior of each element** of the system as well as **each activity (event) that occur**s in the system. There is NOT an overall controller that calls for actions to happen at particular times except for a main driver loop that runs queued events until time runs out. Each element in the system models its own behavior and interactions with other elements in the system. Each NON-passive element class has an `Event` class associated with it. The associated event defines a `run()` method (as discussed in lecture) that simulates the specific behavior of that event.

Some of the classes that we will want are:

- Ports (`Port`) - contains `Shipment` queues
- Agenda (`PQ`)
- `Event` - an interface for all our events
- `VesselEvent` - occurs each time a `Vessel` arrives at a port
- `ShipmentMaker` - one occurs for each `Shipment` arriving at a port

Let's look at each of these components, and see what they should contain. Please note that you may need to include other information, but what is given here is considered fundamental.

ShipmentMaker

This will implement the `Event` interface similar to how the `CarMaker` class was implemented as described in lecture. This will be instantiated once for each `Shipment` creation event. One `ShipmentMaker` will be made for each port, allowing you to have some ports that are more popular than others. `ShipmentMaker` will reschedule itself (using the agenda), create a `Shipment`, decide where the shipment is going to and place the `Shipment` in the appropriate queue at the current stop. If a vessel going to the correct destination is currently at the port, the shipment will be added right away if there is space. If this causes the ship to reach departure capacity, it will cause the vessel to depart.

Shipment

This is a passive class that represents a shipment. It will be created with a `ShipmentMaker` and placed into the appropriate queue at the `Port` where it was created. It will contain its weight, its creation time, and its destination.

Port

This is instantiable once for each port. Each `Port` object will contain its location, its name, and a set of queues (one for each port).

There are exactly 10 ports:

Table 1: Ports

Name	Location	Shipments per Day
Minneapolis	(0, 0)	5
Saint Paul	(0, 10)	5
Antarctica	(0, -6000)	1
Japan	(4000, 4000)	10
Korea	(6000, 5000)	5
China	(5000, 6000)	100
Moon	(0, 1000000)	0 **(Cargo only sent to the Moon)**
Panama	(1000, 3000)	5
Hawaii	(2000, 2000)	5
Pirate Town	(3000, 3000)	10 **(Random chance of cargo being stolen)**

Vessel

There are 8 types of Transport Vessel:

Table 2: Vessels

Name	Capacity in Kilograms	Speed in Km/H	Cost per Km
Canoe	1000	10	1
Yacht	2000	60	2000
Galleon	15000	15	100
Barge	1000000	10	1000
Freighter	2000000	5	1000
Airplane	50000	850	10000
Carrier Pigeon Team	1000	10	0
Rocket	10000	2000	100000

VesselEvent

This implements the `Event` interface. A `VesselEvent` is created for every arrival of a vessel at a port. When a vessel arrives at a port, the `VesselEvent` causes the vessel associated with it to look at its shipments list to calculate how much money was made off of the current shipment, and then it removes them. The `VesselEvent` will then look at where the oldest shipment is going to, and start loading the vessel with Shipments going to that `Port`. If the vessel reaches its departure capacity, the `VesselEvent` will create a new `VesselEvent` and schedule it (via the agenda) for the arrival at the next stop at a time in the future depending on the distance to that port from the current one. If the vessel hasn't reached its departure capacity by the time its max wait time has occurred, the vessel will depart on its own.

PQ

This is the priority queue (likely to be called the agenda within your code). This code is provided. You will need one instance of it. This one instance will be used to schedule all events for your simulation.

Statistics

You will want to keep track of relevant `Statistics` such as:

- How full each vessel is
- The maximum time a shipment spends waiting at a port

- The average profit per shipment
- Profit per day
- etc.

Randomness

You will need to use a random distribution to model the arrival of shipments at each port. This is determined by method calls made in the `run()` method for the `ShipmentMaker`. You will also need to randomly generate which port each `Shipment` is bound for and how much the `Shipment` weighs.

Each stop will have an average number of shipments per day. To more realistically represent the pattern of arrival of Shipments, we will introduce some randomness according to the table below:

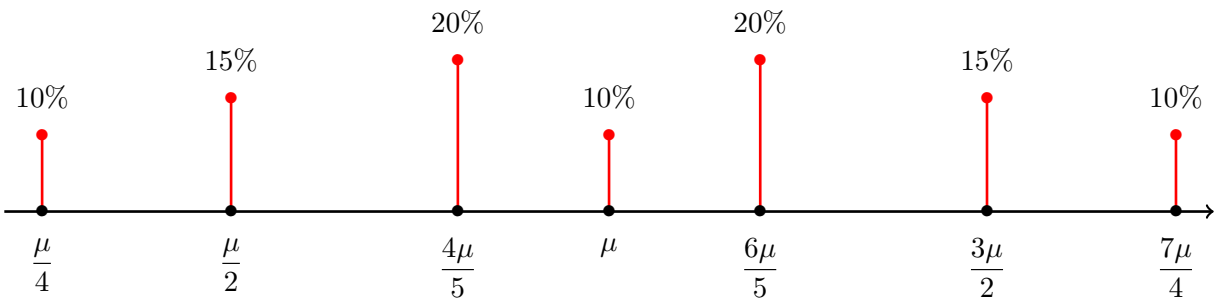

Figure 1: PMF of interarrival times with mean μ.

In this case, μ is the number of minutes in a day divided by the number of shipments a port receives in a day. Destination ports are all equally as likely and should be chosen at random when the Shipment arrives.

Important: In addition, when a ship is sent to Pirate Town, there is a 10% chance that your entire cargo is stolen, and you make no profit.

Determining processing time

A ship will take the shortest path between ports. The shortest distance is defined as

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{1}$$

To calculate how many minutes the ship takes to travel you will use the euclidean distance divided by the speed of the vessel times 60.

Assumptions

- Assume that shipment arrival distribution and destination port will use the distributions given.
- Use the rules above for calculating the amount of time between each port.

Variables

- C: the percentage a ship needs to be full before departing
- W: the amount of time a ship will wait to reach capacity before leaving with the current cargo
- Transport Vessel Type

What to do

First, get the simulation system to work, but start small and verify each feature works before proceeding to the next one. Then, produce a detailed report that gives a convincing presentation for what Goldy needs to do to maximize his profits. In order to provide a convincing argument, you will need to collect good statistics, verify them, and present them in a clear manner.

The problem is purposely vague so that you can decide what statistics you need to gather and what simulation tests you should run. But, you need to present enough information so that Goldy can make an educated decision. A significant portion of the grading will be based on your write-up. This means that you will need to have a working simulator in order to have results to write about.

The Write-up

The write-up must be submitted as a `.pdf` document along with the rest of your source code. 50% of your project grade will be somehow connected to the write-up. When deciding on statistics to maintain and the simulations to run, think about the need to support your conclusions in your write-up. The write-up should be in the form of a report–such as a consulting report–that you might submit to Goldy. The report should include data (such as average travel time and average profit per trip) summarized in a concise and readable fashion along with specific data that proves that the simulation results are correct.

For example, do the numbers make sense? Are your simulation runs at equilibrium? (That is, as you run the same model with the same parameters for longer amounts of time, do the stats tend to stay constant?) Certainly, some statistics are more difficult to properly calculate than others. So, don't try to do it all at once. Get some stats going, and verify them, before moving on.

In order for this to be a successful project, you need to start NOW, and have your simulator working about 5-7 days before the project is due. That will give you enough time to do all the tests and produce the report. Remember, Goldy will be interested in what type of vessel he should use and what conditions should be used for departure. You will need to run your simulation multiple times. It will take some thinking to know what runs to make and how to effectively present your findings.

Important:
Please include an appendix at the end of your write-up that includes the raw output of all of your simulations.

The write-up needs to be a minimum of 2 pages long, and double spaced with 12 point font. Charts, figures, and the appendix do not count toward the page count.