Introduction to Java

CS9053 Section I2

Wednesday 6 PM – 8:30 PM

Prof. Dean Christakos

Assignment 7

March 14th, 2025

Due: March 21st 11:59 PM

**Part I: Design the Rental Agency Better**

1. Let’s return to our Rental Agency problem. In the previous problem sets, we used arrays and ArrayLists to keep track of book stock and what was lent out. You probably had the intuition that there was a better was to do this, and you’re correct—the data structures we learned in the Collections lecture can resolve most of our concerns.

You are going to do three things:

1. Reimplement the RentalAgency and RentalCar code (you can use your old code) to use collections that will allow lookups and removals to be achieved in O(1) time

A screenshot of a computer

Description automatically generated

1. Create a Patron class representing RentalAgency members who borrow RentalCar. Given a Customer id, getRentalCarsForCustomer should be able to show which items are borrowed by the Patron

A question marks on a box

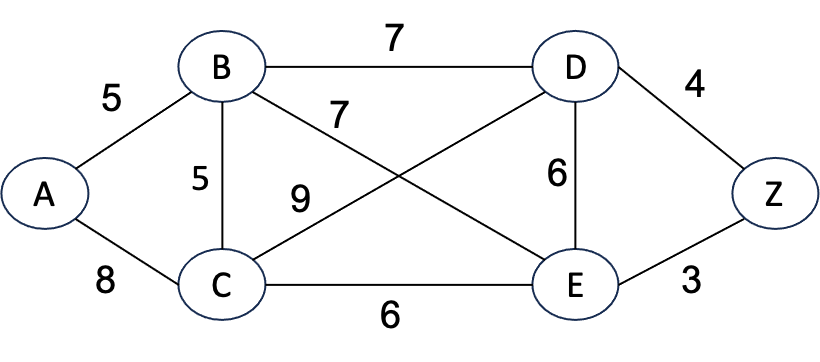
AI-generated content may be incorrect.

1. Finally, given that a RentalAgency has a getCars method and a carsInStock method, show how to display all the cars that have been rental out, using a single method **without** modifying the data held in the RentalAgency and/or Customer classes themselves

Hint: this means you will want to return new copies of any Collections when methods like getCars or carsInStock are called.

**Part II: Shortest path in a graph**

1. Here, I am going to give you a graph where you have to find the shortest path between node A and Z. The algorithm itself will be pretty straightforward, but the challenge is that you need to come up with the right Collection classes to represent the graph and to solve the problem as quickly as possible.



You will have to represent this graph in some way. My suggestion is to create an adjacency list that stores the edge weights and pick the correct Collections to implement an adjacency list.

To find the shortest path between A and Z, you will probably want to implement Dijkstra’s algorithm using a Breadth First Search (BFS). BFS means when you discover a node, you put the node one end of a list and take it from the other, and you have to do this in O(1) for each operation. There are a couple options about what data structure to use for this, but an ArrayList is not one of them, so don’t use that. You will also need a means of keeping track of which nodes you’ve already visited.

You can find Dijkstra’s algorithm described here: <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>

You will return the shortest from A to Z and the length. Obviously, a quick survey of the graph should make it clear what the correct answer is so it should be easy to check what the answer is.