Practical Exercise 9.2 – Data Structures Case Studies

Overall Objective

To suggest appropriate data structures to be used in the given case studies and develop the solutions.

Background

You will need to know:

1. basic Java programming knowledge
2. classes and interfaces
3. generics
4. data structures

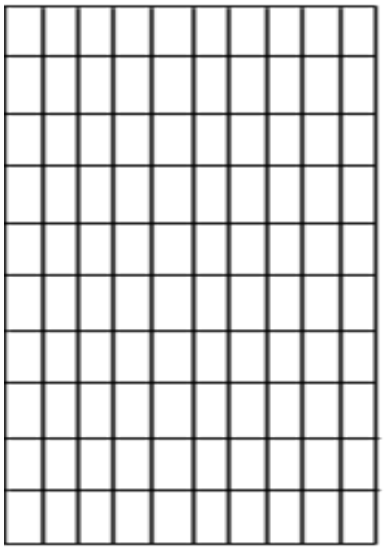
Description

Discuss and suggest suitable data structure(s) and its implementation to model and store objects for the followings:

Case Study 1

Container storage regions are usually set up in a port as shown in Fig. 1(a) where a crane is assigned to each region. Each region is a rectangular area with stacks of containers. A top-down view of one particular region is shown in Fig. 1(b) where each rectangle represents a stack of containers. It is assumed that there are at most 10 containers in one stack due to the height of a crane. In addition, each container is given a unique identity number and a description of the content.

(a) Storage region with a crane



(b) Top-down view of region.

Fig.1: Container storage regions in port.

You are requested to **simulate** the port operation by providing the following:

a. Design a **Container** object.

b. Based on Fig.1, design a suitable data structure to represent a ***container storage region*** with 10 rows by 10 columns of container stacks.

c. Provide the idea for an **unload**(int *id*, int *r*, int *c*) method for the ***container storage region*** to search for a particular container with identity number *id* at row *r* and column *c*. The method should take the container out onto a truck. It is assumed that one cannot move any container temporarily outside the storage region.

d. Give a suggestion that supports fast searching of a container within a container storage region by quickly computing the row and column values.

Case Study 2

You have been hired to write part of the employee database for Haddock Shipping, a global package shipping and delivery company. Employees are added to the database via the following method call:

AddEmployee(InfoRecord newRecord, String theBoss, int hiringDate);

where **newRecord** is the new employee's information record, **theBoss** is the new employee's boss, and **hiringDate** is an integer representing the date the employee was hired (the higher the number, the later the date). You might use different data structures for the following parts (a) and (b).

a. Your task is to store information for the union workers who deliver the packages around the city. Each worker is assigned a route, and every four months, each worker gets to choose a new route if he or she wants. However, these choices are made in order of seniority, i.e., the worker who has been with Haddock Shipping the longest has highest seniority and gets to pick first.

   The company is growing quickly, so new workers are frequently added to the collection of union workers employed by Haddock Shipping. Workers quitting (and thus being removed from the database) occur far less quickly, our greater concern is speeding up the addition of new workers and the traversal of the collection of workers in the order of greatest to lowest seniority. Explain which data structure you would use to implement the collection of workers and how you would implement insertion, removal, and traversal-in-seniority-order on this structure.

b. Your second task is to store information about the company managers. Managers get points based on good performance. When a manager leaves the company, it opens up a vacancy, and the people who had that manager as a boss, the one with the most points gets promoted into the position. This leaves a new vacancy - the newly promoted employee's old job - which is filled in the same way, until finally a lowest-level manager, who was only the boss of the union workers but no managers, gets promoted and the company then hires a new manager to fill that vacancy (union workers have no desire to be promoted to management). Explain which data structure you would use to implement this collection of managers and how you would use that implementation in the promotion algorithm.