ELEC 2543 Object-Oriented Programming and Data Structures

Homework Assignment 2

Due Date: 9:30am, May 10, 2021 (Monday)

Overview: In this assignment, you will first develop the one-directional ring data structure and then use it to simulate the round robin mechanism of time sharing in finishing a list of jobs.

One-dimensional Ring Data Structure

A 1-D ring is a circular linked list that the last node points to the head node as shown in Figure 1.

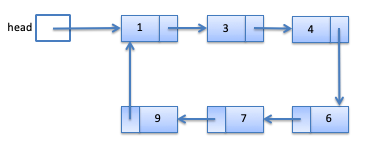


Figure 1: Ring Example

The skeleton of class Ring is provided. Class RingNode is defined as inner class in class Ring. Operations in class Ring are:

public void addObj(Object obj)

This method adds obj in the ring. That is, a new RingNode containing obj is created and is added after the head node. Suppose we add 5 to the ring in Figure 1, the new ring becomes the one in Figure 2.

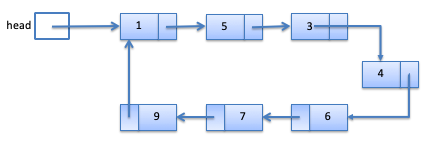


Figure 2: after adding 5

public int size()

This method returns the number of nodes in the ring. The size of the ring in Figure 1 is 6.

Apart from the head reference, RingNode curr is used to reference the node that is being examined. When the first node is added to the ring, curr points to the head node. There are three operations related to curr.

public Object getCurrObj()

This method returns the object being pointed by curr without removing it.

public void removeCurrObj()

This method removes the object being pointed by curr. curr then points to the next node. If the node being removed is also the head, move the head to point to the original next node of head. For example, if curr is pointing to 1 in Figure 2 when we are about to remove 1, the new ring becomes the one in Figure 3. curr also points to 5.

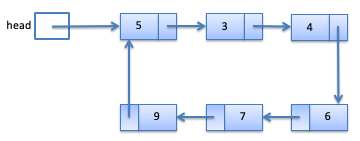


Figure 3: ring after removing the 1

public void advance()

Advance curr to the next node.

Use TestRing.java to test your Ring implementation before proceeding to the next part.

Round Robin Time Sharing Mechanism

A job is some task that needs to be served by a server for a certain amount of time. When there are many jobs waiting to be served, we can arrange the jobs in a logical ring, and the server serves each job for some fixed amount of time one by one, following the ring order. This is called the round robin time sharing mechanism. The amount of time needed of a job may be more than the time a server serves each time. If the total time served on this job is less than the amount of time needed, the job has not finished and remains in the ring. Otherwise, the job is removed from the ring. You are going to write a program that simulates the executing of a list of jobs using the round robin mechanism.

class Job represents the jobs to be executed. Each job has an ID, amount of service time needed, and the total time served so far. ID is determined by order of appearance. That is, the first created job has ID 1, and the second created job has ID 2, etc. Develop class Job with the appropriate instance variables and the following methods. To simplify the work, you can assume all time needed are in integral units of hours.

constructor public Job (int servedTimeNeeded)

This method instantiates a new Job object with the served time needed specified in the parameter.

public int served(int servedTime)

This method simulates the job calling this method to be served for servedTime amount of time. The method returns how much more time the job needed to be served before it is done. If the time allocated is more than the job needs, the method returns the amount of excessive time. For example, if the job still needs 1 hour but servedTime = 3, the method returns -2.

public String toString()

Each job is printed in format ID:(served time needed,amount of time has been served).

Use TestJob.java to test your Job implementation before proceeding to the next part.

class RRSimulation simulates a list of jobs to be served in a round robin fashion. The private instance variable and the constructor has been defined. A list of jobs is created (GenJob.java has been provided for that purpose) and put in a Ring object in the constructor. Develop the public void run(int unit) method that performs the following:

* serve each job in the ring for unit hours until all jobs are done. A message is printed out to indicate the job being served.
* If a job is finished, remove it from the ring and print out a message.
* After the service, the new job list is printed.
* After all jobs are done, print out a message telling how many units of time are needed.

Test your program using SimulationDriver.java. A sample output is provided in output.txt.

Handin

Follow the instructions on Moodle.