Victor Mao (vtm160030)

CS 2 Project 3 Pseudocode

public abstract class BaseNode

* Package
  + package LinkList
* Variables
  + private int row
  + private int seat
* Overloaded Constructor
  + public Node(int r, int s)
    - row = r;
    - seat = s;
* Getters
  + public int getRow() { return row; }
  + public int getSeat() { return seat; }
* Setters
  + public void setRow(int r) { row = r; }
  + public void setSeat(int s) { seat = s; }

public class DoubleLinkNode extends BaseNode

* Package
  + package LinkList
* Variables
  + private DoubleLinkNode next
  + private DoubleLinkNode prev
* Overloaded Constructor
  + public DoubleLinkNode(DoubleLinkNode n, DoubleLinkNode p)
    - next = n;
    - prev = p;
* Getters
  + public DoubleLinkNext getNext() { return next; }
  + public DoubleLinkNext getPrev() { return prev; }
* Setters
  + public void setNext(DoubleLinkNext n) { next = n; }
  + public void setPrev(DoubleLinkNext p) { prev = p; }

public class LinkedList

* Package
  + package LinkList
* Variables
  + private DoubleLinkNode head
  + private DoubleLinkNode tail
* Overloaded Constructor
  + public LinkedList(Double h)
    - head = h;
    - tail = h;
* Getters
  + public DoubleLinkNode getHead() { return head; }
  + public DoubleLinkNode getTail() { return tail; }
* Setters
  + public void setHead(DoubleLinkNode h) { head=h; }
  + public void setTail(DoubleLinkNode t) { tail=t; }
* Methods
  + public void addNode(DoubleLinkNode dln)
    - Add a DoubleLinkNode dln to the end of LinkedList
    - Set next of last DoubleLinkNode in the LinkedList to dln
    - Set prev of dln to the last DoubleLinkNode in the LinkedList
    - Set tail of LinkedList to dln
  + public boolean removeNode(DoubleLinkNode dln)
    - Remove given DoubleLinkNode dln from the LinkedList
    - While loop to search when next is dln
    - Remove dln
      * Set next of DoubleLinkNode before dln to the DoubleLinkNode after dln
      * Set prev of the DoubleLinkNode after dln to the DoubleLinkNode before dln
    - Return true if successful
    - Return false if DoubleLinkNode dln does not exist in LinkedList

public class Main

* Package
  + Outside of package LinkList
* Imports
  + import java.util.Scanner;
  + import java.util.\*;
* Variables
  + LinkedLIst a1reserved, a2reserved, a3reserved, a1unreserved, a2unreserved, a3unreserved
* public static void main (String[] args)
  + Read Auditorium information into LinkedList variables
  + While user does not want to exit
    - printFirstMenu();
    - Read user input for first menu
    - Validate input for integer
    - If Reserve Seats
      * printSecondMenu();
      * Read user input for second menu
      * Ask user for number of tickets and the row and seat number of the first seat
      * Validate input for integers
      * Check if seats are available
        + If available, reserve seats
        + If not available, search for best available

Ask if user wants to reserve best available

If yes, then reserve best available seats

If no, then don’t reserve best available seats and return to main menu

* + - Else if View Auditorium
      * printSecondMenu();
      * Read user input for second menu
      * Validate input for integers
      * Print specific auditorium
    - Else exit
    - printReport();
    - saveAuditorium(reserved, unreserved, filePath) for all three auditoriums
* Methods
  + public LinkedList readReservedAuditorium(String filePath)
    - Open file at filePath
    - Create LinkedList
    - While not EOF
      * Read line
      * For 0 to line length
        + Create new DoubleLinkNode for every reserved seat
        + Add to LinkedList
    - Close file
    - Return LinkedList
  + public LinkedList readUnreservedAuditorium(String filePath)
    - Open file at filePath
    - Create LInkedList
    - While not EOF
      * Read line
      * For 0 to line length
        + Create new DoubleLinkeNode for every unreserved seat
        + Add to LinkedList
    - Close file
    - Return LinkedLIst
  + public void printFirstMenu()
    - Prints first menu
    - 1. Reserve Seats
    - 2. View Auditorium
    - 3. Exit
  + public void printSecondMenu()
    - Prints second menu
    - 1. Auditorium 1
    - 2. Auditorium 2
    - 3. Auditorium 3
  + public void printAuditorium(LinkedList reserved, LinkedList unreserved)
    - Create DoubleLinkNode res that points to the head of the LinkedList reserved
    - Create DoubleLinkNode unres that points to the head of the LinkedList unreserved
    - Print column headers
    - While res != null && unres != null
      * Print row number
      * Print appropriate reserved or unreserved character
      * Print new line
  + public boolean checkAvailability(LinkedList l, DoubleLinkNode dln)
    - Loop through LinkedList
    - Return true if dln is in LinkedList
    - Else return false
  + public boolean reserveSeats(LinkedList reserved, LInkedList unreserved, DoubleLinkNode dln)
    - Remove dln from LinkedList unreserved
    - Add dln to LinkedList reserved in correct order
  + public int countSeats(LinkedList l)
    - While loop through LInkedLIst
    - Increment counter for every seat in LInkedList l
    - Return counter
  + public void printReport()
    - Print header
    - For auditoriums 1 to 3
      * Call countSeats(LInkedList l)
      * Print out open seats, reserved seats, money earned
      * Increment total open seats, total reserved seats, total money earned
    - Print out total open seats, total reserved seats, total money earned
  + public DoubleLinkNode bestAvailable(LinkedList unreserved, int quantity)
    - Determine middle of auditorium
    - Keep a closest DoubleLinkNode variable
    - Loop through LinkedList unreserved
      * Determine the distance from each unreserved node
      * If it is closer than the closest DoubleLinkNode
        + Check if the seats to the right of the node are available
        + If all are available then set closest DoubleLinkNode
    - Return closest
  + public void saveAuditorium(LinkedList reserved, LinkedList unreserved, String filePath)
    - New PrintWriter
    - Call private recursive function writeToFile(head)
    - Print out contents in PrintWriter to filePath
  + private PrintWriter writeToFile(DoubleLinkNode head, PrintWriter pw)
    - if head == null
      * Return pw
    - Else
      * pw.append(‘.’ or ‘#’);
      * Recursively call writeToFile(head.getNext())