Victor Mao (vtm160030)

CS 2336.003

GotG2 Ticket Reservation System

Project 5 Pseudocode

public abstract class Node

* 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 Node

* Package
  + package LinkList
* Variables
  + private DoubleLinkNode next
  + private DoubleLinkNode prev
* Overloaded Constructor
  + public DoubleLinkNode(int r, int s)
    - super(r, s);
    - next = null;
    - prev = null;
* 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()
    - head = null;
    - tail = null;
* 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 boolean isEmpty()
    - return head==null && tail==null;
  + public boolean contains(DoubleLinkNode dln)
    - loop through LinkedList
      * if (getRow() == dln.getRow() && getSeat() == dln.getSeat())
        + return true;
    - return false;
  + public void addNodeAtEnd(DoubleLinkNode dln)
    - if (isEmpty())
      * head = dln;
      * tail = dln;
    - else
      * Set next of tail to dln
      * Set prev of dln to tail
      * Set tail of LinkedList to dln
  + public void addNode(DoubleLinkNode dln)
    - if (contains(dln))
      * return;
    - else if (isEmpty())
      * head = dln;
      * tail = dln;
    - else if goes at head of list
      * dln.setNext(head);
      * head.setPrev(dln);
      * head = dln;
    - else if goes at tail of list
      * tail.setNext(dln);
      * dln.setPrev(tail);
      * tail = dln;
    - else goes in middle of list
      * loop until find correct place for new node
      * add node into correct place
  + public boolean removeNode(DoubleLinkNode dln)
    - if (isEmpty() or contains(dln))
      * return;
    - else if (dln == head)
      * remove head
    - else if (dln == tail)
      * remove tail
    - else is in middle
      * 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

public class Main

* Package
  + Outside of package LinkList
* Imports
  + import java.util.Scanner;
  + import java.util.\*;
  + import LinkList.\*;
* Variables
  + HashMap users, orders
  + LinkedList a1reserved, a2reserved, a3reserved, a1unreserved, a2unreserved, a3unreserved
  + Scanner scan
  + String username, password
* public static void main (String[] args)
  + users = readPasswords(); // hashmap with <username, password>
  + orders = readUsers(); // hashmap with <username, orders.
  + Scanner scan = new Scanner(System.in);
  + String username = “”;
  + String password = “”;
  + boolean loop = true;
  + while (loop)
    - System.out.print(“Enter username: “);
    - username = scan.nextLine();
    - for (int i=0; i<3; i++)
      * System.out.print(“\nEnter password: “);
      * try
        + password = scan.nextLine();
        + if <username, password> is not in HashMap users

throw new Exception();

* + - * + set i=4 to break out of for loop
        + set loop = false to break out of while loop
      * catch (Exception e)
        + System.out.println(“Password entered was incorrect. Try again.”);
        + continue;
  + call readReservedAuditorium() and readUnreservedAuditorium() to read auditorium information into LinkedList variables
  + if (username.equals(“admin”))
    - print admin main menu
      * 1. View Auditorium
      * 2. Print Report
      * 3. Exit
    - read user input for admin main menu
    - validate input for integer between 1-3
    - if “view auditorium”
      * print auditorium submenu
        + 1. Auditorium 1
        + 2. Auditorium 2
        + 3. Auditorium 3
      * read user input for auditorium submenu
      * validate input for integer between 1-3
      * call printAuditorium(reserved, unreserved) to print auditorium
    - else if “print report”
      * call printReport();
    - else // “exit”
      * save auditoriums back to files
      * exit program
  + else // user is not an admin
    - print customer main menu
      * 1. Reserve Seats
      * 2. View Orders
      * 3. Update Order
      * 4. Display Receipt
      * 5. Log Out
    - read user input for customer main menu
    - validate input for integer between 1-5
    - if “reserve seats”
      * print auditorium submenu
        + 1. Auditorium 1
        + 2. Auditorium 2
        + 3. Auditorium 3
      * read user input for auditorium submenu
      * validate input for integer between 1-3
      * call printAuditorium(reserved, unreserved) to print auditorium
      * ask for # of Adult Tickets
      * int numAdult = read user input for adult tickets
      * validate input for integer greater than 0
      * ask for # of Senior Tickets
      * int numSenior = read user input for senior tickets
      * validate input for integer greater than 0
      * ask for # of Child Tickets
      * int numChild = read user input for child tickets
      * validate input for integer greater than 0
      * for # of adult tickets
        + ask for row #
        + read user input for row #
        + validate input for row #
        + ask for seat #
        + read user input for seat #
        + validate input for seat #
      * for # of senior tickets
        + ask for row #
        + read user input for row #
        + validate input for row #
        + ask for seat #
        + read user input for seat #
        + validate input for seat #
      * for # of child tickets
        + ask for row #
        + read user input for row #
        + validate input for row #
        + ask for seat #
        + read user input for seat #
        + validate input for seat #
      * for all entered seats
        + checkAvailability()
      * if (all seats are available)
        + reserve every seat
        + add seats to appropriate <user, orders> in HashMap orders
        + return to main menu
      * else // not all seats are available
        + Search for best available
        + Ask if user wants to reserve best available
        + validate input
        + If yes

reserve best available seats

add seats to appropriate <user, orders> in HashMap orders

return to main menu

* + - * + If no

don’t reserve best available seats

return to main menu

* + - else if “view orders”
      * userOrders = getValue(username)
      * for each order in userOrders
        + print auditorium, seats, and number of tickets per ticket type
      * return to main menu
    - else if “update order”
      * userOrders = getValue(username)
      * for each order in userOrders
        + print number + order
      * ask user to enter the number of an order to update
      * validate user input for an integer within bounds
      * print update order sub menu
        + 1. Add tickets to order
        + 2. Delete tickets from order
        + 3. Cancel Order
      * if “add tickets to order”
        + print auditorium submenu

1. Auditorium 1

2. Auditorium 2

3. Auditorium 3

* + - * + read user input for auditorium submenu
        + validate input for integer between 1-3
        + call printAuditorium(reserved, unreserved) to print auditorium
        + ask for # of Adult Tickets
        + int numadult = read user input for adult tickets
        + validate input for integer greater than 0
        + ask for # of Senior Tickets
        + int numSenior = read user input for senior tickets
        + validate input for integer greater than 0
        + ask for # of Child Tickets
        + int numChild = read user input for child tickets
        + validate input for integer greater than 0
        + for # of adult tickets

ask for row #

read user input for row #

validate input for row #

ask for seat #

read user input for seat #

validate input for seat #

* + - * + for # of senior tickets

ask for row #

read user input for row #

validate input for row #

ask for seat #

read user input for seat #

validate input for seat #

* + - * + for # of child tickets

ask for row #

read user input for row #

validate input for row #

ask for seat #

read user input for seat #

validate input for seat #

* + - * + for all entered seats

checkAvailability()

* + - * + if (all seats are available)

reserve every seat

add seats to appropriate <user, orders> in HashMap orders

return to main menu

* + - * + else // not all seats are available

Search for best available

Ask if user wants to reserve best available

validate input

If yes

reserve best available seats

add seats to appropriate <user, orders> in HashMap orders

return to main menu

If no

don’t reserve best available seats

return to main menu

* + - * else if “delete tickets from order”
        + for every seat in order

print number and row, seat and ticket type

* + - * + ask user to enter a number for row, seat
        + validate integer input
        + if selected seat

remove seat from order

update auditoriums linkedlists to unreserved

if no tickets left in order

remove order from user’s account

return to main menu

else

loop back and ask user for another number

* + - * + else if selected exit

exit to main menu

* + - * else // “cancel order”
        + for all seats in order

update auditorium linkedlists to unreserved

* + - * + remove order from user’s account
        + return to main menu
    - else if “display receipt”
      * print auditorium, seats, and number of tickets per ticket type
      * print amount of each order
      * print overall amount of all orders
      * return to main menu
    - else // “log out”
      * exit to starting point
* Methods
  + public HashMap readPasswords()
    - Scanner scan = new Scanner(new File(“userdb.dat”));
    - HashMap users = new HashMap();
    - While scan not EOF
      * Read line
      * Split into username and password
      * Insert new <username, password> into HashMap
    - scan.close();
    - Return HashMap
  + public HashMap readUsers()
    - Scanner scan = new Scanner(new File(“userdb.dat”));
    - HashMap users = new HashMap();
    - While scan not EOF
      * Read line
      * Split into username and password
      * Insert new <username, null> into HashMap
    - scan.close();
    - Return HashMap
  + 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 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 void printReport()
    - Print header
    - For auditoriums 1 to 3
      * Call countSeats(LinkedList l)
      * Print out open seats, total reserved seats, money earned
      * print adult seats, senior seats, children seats
      * Increment total open seats, total reserved seats, total money earned
    - Print out total open seats, total reserved seats, total money earned
  + 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 DoubleLinkNode bestAvailable(LinkedList unreserved, int quantity)
    - Determine middle of auditorium
    - Keep a closest DoubleLinkNode variable
    - Loop through LinkedList unreserved
      * Determine the distance from the middle of the auditorium to the middle of the chain of seats
      * 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())