ITCS231 Data Structures and Algorithm analysis

PROJECT 1

E-commerce Cart System

Summitted by

Mr.Phusit Mongkhonwatcharaphun

6588059

Summitted to

Asst. Prof. Ananta Srisuphab

Faculty of ICT, Mahidol University

Introduction

This is project about creating e-commerce cart system that user can view products, add them to cart, remove item, and simulate proceed checkout. Manager can add and remove product in their store.

Content

Class **Page**

Contents

ListNode

Attribute

- Private T item : Value in ListNode
- Private ListNode<T> next : next node that link with this node

Constructors

- public ListNode(T item)
  - Parameter: generic datatype item
  - This constructor set node value to item.

- public ListNode(T item, ListNode<T> next)
  - Parameter: generic datatype item, ListNode

  - This constructor set node value to item and set next npde to next.

Methods

- public T getItem()
  - Return value of this node.

- public ListNode<T> getNext()
  - Return next node of this node.

- public void setItem(T item)
  - Change value in this node to new value item.

- public void setNext(ListNode<T> next)
  - Change next node of this node to new node next.

- public String toString()
  - Return string that is node value

LinkedList

Attribute

- private ListNode<T> front : front of this list/ first node of this list.
- private int size : size of this list

Constructor

- public LinkedList()
  - Set LinkedList front equal to null
  - Set size to 0

Methods

- public boolean isEmpty()
  - return true if front equal to null.
  - else return false.

- public void add(T item)
  - create new ListNode with value of item.
  - this method will add Node to the list.
  - Size of list +1.
  - If isEmpty() equal true : front = Node
    Else: its loop Node.getNext() until it's last Node of the list then set next node to Node.

- public void remove()
  - this method is remove last index of the list.
  - if isEmpty() == true will throw error because no item in the list.
  - Set temporary equal to front.
  - Loop until it is one before the last node in list and set it next to null.
  - Size of list -1

- public void remove(int index)
    - parameter : index of item in list.
    - Remove the specific value in the list by index.
    - Throw index out of bounds, If index < 0 or Index > size.
    - Size-1 after remove.

- public int size()
    - return size of this list.

- public T get(int index)
    - parameter : index of item
    - Throw index out of bounds, If index < 0 or Index > size.
    - Loop until node is that index.
    - Return value od that node.

- public boolean contain(T item)
    - parameter : item
    - loop through every item in list
    - if node value equal to item return true.
      if cannot find that item in list, Return false.

- public String toString()
    - loop through all members in the list and get the value.
    - return string with the format.
    - Ex. [0,1,2,3]

ArrayList

Attribute

- private Object[] array : array to store object
- private final int defaultSize = 10 : default size of arraylist
- private int Tsize : temporary size
- private int size : current size of arraylist

Constructor

- public ArrayList()
  - initialize arraylist size to 0.
  - Tsize = defaultSize which is 10.
  - Create new array size 10.

Methods

- public int Size()
  - return this current size.

- public boolean isEmpty()
  - check if this current size == 0 return true if yes else return false.

- public boolean checkFull()
  - check if this size == Tsize which is 10
  - return true if yes else return false

- public boolean contain(T item)
  - parameter : item
  - if this array is empty throw exception no element in array.
  - If element in list equal to item return True, else return False.

- public void set(int index, T item)
    - parameter : index , item
    - if index < 0 or index >= this current size throw IndexOutOfBounds.
    - Change item in array with specific index to new item.

- public T get(int index)
    - parameter : index
    - if index < 0 or index >= this current size throw IndexOutOfBounds.
    - Return value arraylist with specific index in the list.

- public void add(T item)
    - parameter : item
    - check if size is full or not if full create new temporary storage to store all value in list the. Create new array size *2 and copy value from temporary to new array
    - item = array with size index
    - size+1

- public void remove()
    - if list is empty print can not remove
    - size -1
    - copy value to temporary storage array then create new array with size -1 and copy value from temporary to array.

- public void remove(int index)
    - parameter : index
    - if index < 0 or index >= this current size throw IndexOutOfBounds.
    - Remove value in list with specific index by copy value to temporary array when i == index then skip that value. Copy from temporary to new array.

- public String toString()
    - return string with all items/elements in the arraylist.

Product

Attribute

- private int ID : id of product
- private String Name : Name of product
- private String Desc : Description of product
- private double Price : Price for each item
- private int Quantity : Quantity/Stock for product

Constructor

- public Product(int id, String name, String desc, double price, int quantity)
  - Set ID to id.
  - Set Name to name.
  - Set Desc to desc
  - Set Price to price
  - Set Quantity to quantity

- public Product(int id, String name, String desc, double price)
  - Set ID to id.
  - Set Name to name.
  - Set Desc to desc
  - Set Price to price
  - Set Quantity to 1

Methods

- public void setQuantity(int amount)
  - parameter : amount.
  - set Quantity equal to this amount.

- public int getId()
  - return this product ID

- public String getName()
    - return this product Name.

- public double getPrice()
    - return this product Price.

- public int getQuantity()
    - return this product Quantity.

- public String getDesc()
    - return this product Description.

- public double getTotal()
    - return this price multiply with quantity.

- public String centerText(String text, int width)
    - parameter : text , width
    - this method return String that arrange in to the middle of that width.

- public String toString()
    - return String format of product in form of table and the price that show is calculate to total.

- public String toStringForManager()
    - return String format of product in form of table and the price for only 1 item.

ProductManager

Attribute

- private ArrayList<Product> store : arraylist to store all product.
- private ArrayList<Integer> Id : arraylist to store id of all product.
- private int size : size of product in arraylist.

Constructor

- public ProductManager()
  - initialize store,Id to new Arraylist.
  - Set size = 0.
  - Add id 1-8 product to store.
  - If error throw exception.

Methods

- public void add(Product item)
  - parameter : item
  - if size == 0 add item to store , add id to Id list.
  - size +1
  - if id is contain in Id list print Can not add this product because it's already exist.

- public void add(int id, int quantity)
  - parameter : id , quantity
  - this method is add stock to exist product in store
  - if id not contain in the list print Id product not exist.
  - If id contain add quantity to that product.

- public int IdtoIndex(int id)
  - parameter : id
  - If size of this store == 0 print out your store is empty.
  - If id not contain in store list print out id is not exist.
  - If have, return index of that ID.

- public ArrayList<Product> getItemList()
    - return all product in list of the store.

- public ArrayList<Integer> getIdList()
    - return all id in list of the store.

- public void displayProduct()
    - print out all product format in form of table.

- public void remove(int id)
    - c
    - if store size == 0 will throw exception “YOUR STORE IS ALREADY EMPTY”
    - if id not contain in idList will throw exception “INVALID ID OR DO NOT HAVE PRODUCT”
    - this method will remove the id in the list and remove product of that id in store.
    - Size -1

- public void reduce(int id, int quantity)
    - parameter : id , quantity
    - if store size == 0 will throw exception “YOUR STORE IS ALREADY EMPTY”
    - if id not contain in idList will throw exception “INVALID ID OR DO NOT HAVE PRODUCT”
    - if quantity > current quantity of that product throw exception “REDUCE QUANTITY IS MORE THAN OUR STOCK”
    - if input quantity <= 0 throw exception “INVALID QUANTITY”
    - This method will reduce quantity of current product.

- public void set(int id, Product item)
    - parameter : id , item
    - if id is not contain in list print out “Invalid or do not have product”
    - if product list size == 0 print out “You did not add any product yet.”
    - This method will set that product with that id to new product.

- public Product get(int id)
    - parameter : id
    - if id not contain in list throw exception
    - else return product that equal to id from parameter.

- public Product get(int id, int quantity)
    - parameter : id, quantity
    - this method will return new Product that equal this id and set quantity to quantity from parameter
    - if id not contain in list and quantity from parameter more than quantity of product in store will print out error.

- public int size()
    - return size of product in store.

- public void writeProductsToCSV(String fileName)
    - parameter : fileName .
    - this method will generate .csv file
    - header is id, name, price, quantity, description of each product in stock.

Cart

Attribute

- private LinkedList<Product> cart : arraylist to store all product.
- private LinkedList<Integer> idInCart : arraylist to store id of all product.
- private ProductManager stock : list of product in store.
- private String[] coupon = {“9ARM”, “LnwZA”} : coupon for discount
- public String discount = “” : default = null.
- public double discountPrice = 0 : default = 0.

Constructor

- public Cart()
  - initialize cart, idInCart to new Linkedlist.
  - Initialize stock = new ProductManager.

Methods

- public int IdtoIndex(int id)
  - parameter : id
  - this method will return index of Product id from parameter**.**
  - if cart == 0 throw exception.

- public Product get(int id)
  - parameter : id
  - return Product in cart with id from parameter.

- public void add(int id, int quantity)
  - parameter : id , quantity
  - this method will add product that equal id from parameter, and quantity = quantity from parameter.
  - If store not contain that id throw exception.
  - If quantity in store < quantity from parameter throw exception.
  - If quantity from parameter <=0 throw exception
  - If id already contain in cart will add quantity instead.

- public void add(int id)
    - parameter : id
    - same as add method before but the quantity to add is 1.

- public void remove(int id)
    - parameter : id
    - this method will remove product in cart that equal to id from parameter.
    - Will throw exception, if cart == 0 and cart not contain that id.

- public void reduceQuantity(int id, int quantity)
    - parameter : id , quantity
    - this method will reduce product quantity of that id in cart.
    - After reduce if that product quantity = 0 will remove from cart automatically.
    - Throw exception if cart == 0 , quantity > in cart quantity , parameter quantity <= 0, id not contain in cart.

- public double getTotalPrice()
    - this method will calculate total item price
    - if have discount total price – 10%

- public void getCart()
    - this method will print all item in cart.

- public void getReceipt()
    - this method will print receipt.

- public String centerText(String text, int width)
    - this method receive text and width and return that text in center of that width.

- public boolean isdiscountValid(String code)
    - this method will check if code is equal to discount coupon that initialize in attribute.
    - If valid return true, else return false.

- public boolean cartEmpty()
    - return true if this cart is empty == 0.

- public LinkedList<Product> getList()
    - get list of all item in cart.

- public ProductManager getStock()
    - get Stock in store.

UserInterface

Attribute

- private Cart userCart : for use cart
- private Boolean manager = false (set default manager to false)

Constructor

- public UserInterface()
  - initialize userCart to new Cart.

Methods

- public void welcome()
  - create csv file stock default.
  - this method will show interface of welcome and show list of all product in store.

- public void interactLogin()
  - every time go back to this interface will set manager to false.
  - this method have 3 cases
  - case 1 : Exit from the system.
  - case 2 : go to customer instruction menu.
  - case 3 : go to admin instruction menu.(set manager to true)
  - If input is incorrect will print “INVALID INPUT PLEASE TRY AGAIN”

- public void customerInstruction()
  - this method is interface for customer have 4 case
  - case 0 is back to login menu.
  - case 1 is go to adding session.
  - case 2 is go to removing/reducing session.
  - case 3 is viewing item in your cart.

- public void addingSession()
    - will show product in stock
    - ask user to input 1 or 0
    - if 0 go back to customer menu / admin menu if manager true
    - if 1 will add product in your / if manager is true will add new product to store or add stock to product.

- public void addProcess()
    - ask user input to add product.
    - If 0 exit to menu.
    - if its customer will add product to cart.
    - If manager will add quantity if id already exist else add new product.

- public void removingSession()
    - if manager true show product stock else show user cart.
    - case 0 : go back to menu
    - case 1: going removing process.
    - case 2: going reducing process.

- public void removeProcess()
    - ask user input id.
    - if 0 exit to menu.
    - if manager true remove product in the store
    - else remove product in cart

- public void reduceProcess()
    - ask user input id.
    - if 0 exit to menu.
    - if manager true reduce product quantity in the store.
    - else reduce product quantity in cart.

- public void viewingCart()
    - this method view show user cart
    - ask user input 2 cases
    - case 0 : exit to menu
    - case 1 : go to check out

- public void checkingoutSession()
    - this method ask user to input.
    - Have 3 cases.
    - case 0 : exit to menu.
    - case 1 : confirming your order
    - case 2 : adding discount code
    - if cart is empty can not check out and go back to menu.

- public void confirmProcess()
    - this method will simulate checking out process
    - ask user to input 0,1,2
    - 0 : go back to checking out session.
    - 1 : pay by cash.
    - 2 : pay by mobile banking.
    - Create csv file of stock after purchasing.

- public void discountProcess()
    - ask user to input discount coupon.
    - Type 0 to exit.
    - If coupon can use go to confirm process , else print code invalid.

- public void adminInstruction()
    - this method will show instruction for admin/manager.

App

```java
public class App {
    public static void main(String[] args) throws Exception {
        UserInterface u = new UserInterface();
        u.welcome();
        u.interactLogin();

    }

}
```

-App is main class for running program.

-Create new userinterface object.

-call method from user interface.

Challenge Faced

In the first when I create program I started from creating listNode, LinkedList and ArrayList by myself. Very challenge problem is that I need to test and find bug that sometimes I don't know where is incorrect and I thought it is correct , but actually there is mistake in my code that I did not see which is very small so I need to debug it step by step to find why it's can not work properly. Example I forget to set node = node.getNext() so it will not go looping to all members and lead to error in my test. When I step up to do Product, ProductManager and Cart the program become more complex I need to think all possible case that user can made mistake to my program and I need create code to handle that problem.

Lesson Learned

For me I learn a lot of new thing that can applied to use in next project. First is I forget that Java have static keyword that you can use this to other method outside the class and change value of item real time ,but in my project I initialize object in Cart and calling method from that object which if it's reality problem these 2 class need to separate from each other. If you want to get data you need to get from static way. Next is I don't need to write many methods in 1 class just write only important and think about implement in the other class so it's will not wasting so much time as you see in some of my classes there are many method that I created but I did not use. And last is I need time to understand what is the question what is the problem what is requirement for program. If I take more time to comprehend it so I can see way to implement it more systematic.