Don Bosco School, Liluah

2018-2019

Computer Science Project

Inventory System for retail store

Nilava Metya

Class: XII C
Roll No.: 23
Reg. No.: 2006/040

January 14, 2019

Contents

Introduction **2**

Overview of program **2**

Source Code **3**

Input files **17**

Output **18**

Introduction

People visit retail stores almost regularly to buy items of regular need. It is not easy to manage a retail store, even though it seems quite simple standing in the shoes of the customer. There have to be a lot of paper-work to keep a record of the cost and the total sale incurred on a daily basis, for every purchase. Also a secure system is needed so that no customer can manipulate with the records. But keeping track of all these activities on a paper is difficult. The following problems arise in a pen and paper database:

- Pen-and-paper database does not allow to change data already in database. Even if one leaves some space for further input of data (to be done later), this might end up in having large piles of papers, most of which is unnecessary space. The space might also get exhausted. Cancellations make the paper untidy.
- Manual input of data is every error-prone.
- Searching for a particular customer details or an item in the stock is quite difficult as data is totally unsorted, and maintained in the order they are introduced into the shop.
- Manual tracking of transactions in one database is cumbersome, in order to make the list quite detailed.

This project is intended to write a working code in JAVA language to imitate a retail store, keeping in mind the need of automation of transactions and all the actions. The time used up for manual billing and updation of database can be devoted to a customer to optimize the sale and serving them fast, increasing their satisfaction level. The working of the program has been shown with a small sample data of input. It reads from and writes data into *.txt files so as to maintain a proper database.

Overview of program

The following classes have been implemented:

- class Item: This class characterizes an item in the retail store. The characteristics of an item include: int id, String name, int quantity, int price. The class also has the methods to (binary) search id in an ArrayList containing objects of type Item or linear search by name.
- class Customer: This class characterizes a customer. The attributes include: int id, String name, long phone, String pass, int shopped to provide the facility of logging into their account and also stores the total amount of shopping done in the store since beginning. There are methods to search a customer by id or by name in an ArrayList of objects of type Customer. Additionally, the method boolean login() validates the logging in by a customer giving upto three attemps to input correct password, otherwise program terminates.
- class PurchaseItem: Its objects are used in a single execution of the program. A single execution of the program can serve one customer only. The objects derived from this class are used to store the details of items purchased by the customer from the store.
- class Store: This class is where all the actual executions take place. It contains the following methods for **reading data** from database, the customer to **buy** items or for the shopkeeper to **add new customers** or **increase the stock**: void readCust(), void readStock(), int buy(), void addCust(String name), void addToStock(). Data is read from files and processed into ArrayLists of appropriate type. The program is user-interactive and the resultant actions are made into data and all updations are done in the required files accordingly.

Source Code

```
/*
 * Java Program to implement an inventory of items
 * Imitate a retail shop
 */
import java.util.*;
import java.io.*;
import java.nio.file.*;
import java.text.*;

class Store
{
    private static ArrayList<Item> stock; //arraylist to store all items
    private static ArrayList<Customer> cust; //arraylist to store details of customers
    //arraylist to store list of purchased items of a customer
    private static ArrayList<PurchaseItem> list;
    private static BufferedReader br;
    private static String custfolder;      //path of customer folder
    private static String stockfolder;     //path of stock folder
    private static File[] custfile;        //array to store files in customer folder
    private static File[] stockfile;       //array to store files in stock folder
    /*
     * method to remove the extension of parameter file name
     * file name (only) is id of Customer or Item
     * returns "x" if filename is "x.txt"
     */
    public static String removeExt(String s)
    {
        int posdot = s.indexOf('.');
        String fileNameOnly = s.substring(0,posdot);
        return fileNameOnly;
    }

    //read Customer details from customer folder and store in arraylist
    public static public void readCust()
    {
        File folder = new File(custfolder); //creates File object to access folder
        custfile = folder.listFiles();      //gets all files in folder
        int id, points;
        String name, pass;
        long ph;
        for(int i = 0 ; i < custfile.length ; i++)
        {
            try
            {
                //buffering input from file
                br = new BufferedReader(new FileReader(custfile[i]));
                //removing extension and adding to arraylist
                id = Integer.parseInt(removeExt(custfile[i].getName()));
                name = br.readLine();
                ph = Long.parseLong(br.readLine());
                pass = br.readLine();
                points = Integer.parseInt(br.readLine());
                Customer customer = new Customer(id,name,ph,pass,points);
                cust.add(customer);
            }
            catch(Exception e)
            {
                System.out.print("\nError in reading Customer file! Exception code: "
                    + e);
                System.exit(0);
            }
        }
    }

    //read Item details from stock folder and store in arraylist
    public static void readStock()
    {
        File folder = new File(stockfolder);
        stockfile = folder.listFiles();
        int id, q, p;
        String name;
        for(int i = 0 ; i < stockfile.length ; i++)
        {
            try
            {
                //buffering input from file
                br = new BufferedReader(new FileReader(stockfile[i]));
                //removing extension and adding to arraylist
                id = Integer.parseInt(removeExt(stockfile[i].getName()));
                name = br.readLine().toLowerCase();
                q = Integer.parseInt(br.readLine());
                p = Integer.parseInt(br.readLine());
                Item item = new Item(id,name,q,p);
                stock.add(item);
            }
            catch(Exception e)
            {
                System.out.print("Error in reading Items file. Exception code: " + e);
                System.exit(0);
            }
        }
    }

    /*
     * method to add to stock
     * may increase quantity of existing stock
     * or, may add a new item which does not exist in database
     */
    public static void addToStock()
    {
        System.out.print("\nEnter item name: ");
        Item s = new Item();
        String name = "";
        int q = 0, pos;
        try {name = br.readLine().toLowerCase();}
        catch(Exception e) {}
```

```
System.out.print("\nEnter item quantity: ");
do      //loop to verify correct input
{
    try{q = Integer.parseInt(br.readLine());}
    catch(Exception e)
    {
        System.out.print("\nError in input! Please re-enter: ");
        continue;
    }
} while(false);
pos = Item.search(stock,name);
if(pos < 0)      //if new item to add
{
    s.name = name;
    System.out.print("\nEnter unit price: ");
    do      //loop to verify correct input
    {
        try{s.price = Integer.parseInt(br.readLine());}
        catch(Exception e)
        {
            System.out.print("\nError in input! Please re-enter: ");
            continue;
        }
    } while(false);
    s.id = 0;
    if(stock.size() != 0) s.id = stock.get(stock.size()-1).id + 1;
    s.quantity = q;
    stock.add(s);
    Path file = Paths.get(stockfolder+"\\"+s.id+".txt");
    try
    {
        Files.createFile(file);
        PrintWriter pw = new PrintWriter(new BufferedWriter (new FileWriter
            (stockfolder+"\\"+s.id+".txt",true) ) );
        pw.println(name);
        pw.println(s.quantity);
        pw.println(s.price);
        pw.close();
        pw = new PrintWriter(new BufferedWriter (new FileWriter
            ("D:\\Stock_entry.txt",true) ) );
        pw.println(s.name + "\t" + s.quantity);
        pw.close();
    }
    catch(Exception e)
    {
        System.out.print("\nError in writing to Stock file! Exception code: "
            + e);
        System.exit(0);
    }
}
else        //if item already in stock
{
    s = stock.get(pos);
    s.quantity += q;
```

```
            try
            {
                PrintWriter pw = new PrintWriter(new BufferedWriter (new FileWriter
                    (stockfolder+"\\"+s.id+".txt") ) );
                pw.println(s.name);
                pw.println(s.quantity);
                pw.println(s.price);
                pw.close();
                pw = new PrintWriter(new BufferedWriter (new FileWriter
                    ("D:\\Stock_entry.txt",true) ) );
                pw.println(s.name +"\t" + s.quantity);
                pw.close();
            }
            catch(Exception e)
            {
                System.out.print("\nError in writing to Stock file! Exception code: "
                    + e);
                System.exit(0);
            }
        }
    }

    /*add a customer to database based on name
     * if already present in database, then no action is taken
     */
    public static void addCust(String name)
    {
        System.out.print("\nEnter customer phone:");
        boolean flag = true;
        long phone = 0;
        do      //loop to remove input error
        {
            try{phone = Long.parseLong(br.readLine());}
            catch(Exception e)
            {
                System.out.print("\nIncorrect Input! Please re-enter:");
                continue;
            }
        } while(false);
        int id = 1;
        id = cust.get(cust.size()-1).id + 1;
        System.out.print("Please set password. Password should be atleast 8 characters
            long.");
        String pass = "";
        do      //loop to verify correct password is input
        {
            try{pass = br.readLine();}
            catch(Exception e){}
            if(pass.length() < 8) System.out.print("\nPassword does not meet
                requirements. Please re-enter: ");
        } while(pass.length() < 8);
        Customer c = new Customer(id,name,phone,pass,0);
        cust.add(c);
        try     //block to write details to file
        {
            Path file = Paths.get(custfolder+"\\"+id+".txt");
            Files.createFile(file);
            PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter
                (custfolder + "\\" + id + ".txt",true)));
            pw.println(name);
            pw.println(phone);
            pw.println(pass);
            pw.println(c.shopped);
            pw.close();
        }
        catch(Exception e)
        {
            System.out.print("\nError in writng to Customer file! Exception code: " +
                e);
            System.exit(0);
        }
    }

    // method for customer to buy product and update details of stock
    public static int buy()
    {
        //print details of available items
        System.out.print("\nID\tQuantity\tPrice per unit\tItem");
        for(Item i : stock)
        {
            if(i.quantity > 0)
                System.out.print("\n"+i.id+"\t"+i.quantity+"\t\t"+i.price+"\t\t"+i.name);
        }
        int choose = -1;  //option to search item
        System.out.print("\nBuy by:\t1.ID\t2.Item name");
        do      //loop for proper input of data
        {
            System.out.print("\nEnter choice (0 to return): ");
            try{choose = Integer.parseInt(br.readLine());}
            catch(Exception e)
            {
                System.out.print("\nError in input! Please re-enter: ");
                continue;
            }
            if(choose == 0) return -1;
            if(choose < 1 || choose > 2) System.out.print("\nInvalid input! Please
                re-enter: ");
        } while(choose < 1 || choose > 2);
        int q = 0;
        int pos = -1;
        int price = 0;
        if(choose == 1)   //choose by item id
        {
            System.out.print("\nEnter id (-1 to quit) : ");
            do      //loop to input proper data
            {
                int id;
                try{id = Integer.parseInt(br.readLine());}
                catch(Exception e)
                {
                    System.out.print("\nInvalid input! Please re-enter: ");
                    continue;
                }
                if(id == -1) return -1;
                pos = Item.search(stock,id); //search in stock
                if(pos == -1)  //item not found in stock
                    System.out.print("\nItem not found. Please re-enter(-1 to quit): ");
            } while(pos < 0);
            Item s = stock.get(pos);
            System.out.print("\nItem: " + s.name + "\nQuantity in stock: " +
                s.quantity + "\nPrice per unit: " + s.price);
            System.out.print("\nEnter quantity to buy: ");
            do
            {
                try{q = Integer.parseInt(br.readLine());}
                catch(Exception e)
                {
                    System.out.print("\nInvalid input! Please re-enter: ");
                    continue;
                }
                if(q > s.quantity) System.out.print("\nNot enough supply. Re-enter: ");
            } while(q > s.quantity);
            list.add(new PurchaseItem(s,q)); //add to list of purchased items
            price = q * s.price;
            s.quantity -= q;      //reduce quantity in stock
            try    //block to update stock file
            {
                PrintWriter pw = new PrintWriter(new BufferedWriter (new FileWriter
                    (stockfolder + "\\" + s.id + ".txt")));
                pw.println(s.name);
                pw.println(s.quantity);
                pw.println(s.price);
                pw.close();
            }
            catch(Exception e)
            {
                System.out.print("\nError in writing to Stock file! Exception code: "
                    + e);
                System.exit(0);
            }
        }
        else        //choose by item name
        {
            System.out.print("\nEnter item (-1 to quit) : ");
            do
            {
                String item;
                try{item = br.readLine().toLowerCase();}
                catch(Exception e)
                {
                    System.out.print("\nInvalid input! Please re-enter: ");
                    continue;
                }
                if(item.equals("-1")) return -1;
                pos = Item.search(stock,item);  //search item name
                if(pos == -1) //item not in stock
                   System.out.print("\nItem not found. Please re-enter(-1 to quit): ");
            } while(pos < 0);
            Item s = stock.get(pos);
            System.out.print("\nItem: " + s.name + "\nQuantity in stock: " +
                s.quantity + "\nPrice per unit: " + s.price);
            System.out.print("\nEnter quantity to buy: ");
            do     //loop to input proper data
            {
                try{q = Integer.parseInt(br.readLine());}
                catch(Exception e)
                {
                   System.out.print("\nInvalid input! Please re-enter: ");
                   continue;
                }
                if(q > s.quantity) System.out.print("\nNot enough supply. Re-enter: ");
            } while(q > s.quantity);
            list.add(new PurchaseItem(s,q)); //add to list of purchased items
            price = q * s.price;
            s.quantity -= q;  //reduce quantity in stock
            try    //block to update stock file
            {
                PrintWriter pw = new PrintWriter(new BufferedWriter(new
                   FileWriter(stockfolder + "\\" + s.id + ".txt")));
                pw.println(s.name);
                pw.println(s.quantity);
                pw.println(s.price);
                pw.close();
            }
            catch(Exception e)
            {
                System.out.print("\nError in writing to Stock file! Exception code: "
                   + e);
                System.exit(0);
            }
        }
        return price;
}
//main method to begin execution
public static void main(String[] args)
{
    stock = new ArrayList<Item>();
    cust = new ArrayList<Customer>();
    custfolder = "D:\\Customer"; //path of customer folder
    stockfolder = "D:\\Stock";   //path of stock folder
    //file to store transaction details
    String fileTransaction = "D:\\Transactions.txt";
    list = new ArrayList<PurchaseItem>();
    readCust();
    readStock();
    //sort customers by id
```

```
Collections.sort(cust, (o1,o2) -> (o1.id - o2.id));
//sort items by id
Collections.sort(stock, (o1,o2) -> (o1.id - o2.id));
int id = -1;
int ind = -1;
System.out.print("\nEnter customer id: ");
br = new BufferedReader(new InputStreamReader(System.in));
do     //loop to ensure proper input
{
    try {id = Integer.parseInt(br.readLine());}
    catch(Exception e)
    {
        System.out.print("\nImproper input! Please Re-enter: ");
        continue;
    }
    if(id < 0 || id > cust.size())
        System.out.print("\nInvalid id! Please re-enter: ");
} while(id < 0 || id > cust.size());
Customer c = cust.get(id);
boolean flag = c.login();
if(!flag)
{
    System.out.print("\nUnsuccessful attempt! Program will exit.");
    System.exit(0);
}
if(id == 0)   //admin login
{
    int choice;
    do
    {
        choice = -1;
        System.out.print("\n1.Add Customer\n2.Add to stock\n Enter choice: ");
        do     //loop to ensure proper input
        {
            try{choice = Integer.parseInt(br.readLine());}
            catch(Exception e)
            {
                System.out.print("\nError occurred! Please re-enter: ");
                continue;
            }
            if(choice < 1 || choice > 2) System.out.print("\nInvalid Input!
                Re-enter choice: ");
        } while(choice < 1 || choice > 2);
        if(choice == 1)   //adding customer
        {
            System.out.print("\nEnter customer name: ");
            String name = "";
            try{name = br.readLine();}
            catch(Exception e)
            {
                System.out.print("\nError occurred! Please re-enter: ");
                continue;
            }
            if(Customer.search(cust,name) >= 0)
            System.out.print("\nCustomer already in database with id
               "+Customer.search(cust,name));
            else addCust(name);
        }
        else addToStock(); //add to stock
        //option to re-do
        System.out.print("\nDo you want to continue?(1 for YES, 0 for NO): ");
        do
        {
            try {ind = Integer.parseInt(br.readLine());}
            catch(Exception e)
            {
               System.out.print("\nError occurred! Please re-enter: ");
               continue;
            }
            if(ind < 0 || ind > 1) System.out.print("\nInvalid input! Please
               re-enter: ");
        } while(ind < 0 || ind > 1);
    } while(ind == 1);
    return;
}
//normal customer login
System.out.print("\nYou have successfully logged in!");
int bill = 0; //stores net amount of shopping
ind = -1;
do
{
    int temp = buy();
    if(temp != -1) bill += temp;
    //option to re-do
    System.out.print("\nDo you want to continue?(1 for YES, 0 for NO): ");
    do
    {
        try {ind = Integer.parseInt(br.readLine());}
        catch(Exception e)
        {
            System.out.print("\nError occurred! Please re-enter: ");
            continue;
        }
        if(ind < 0 || ind > 1) System.out.print("\nInvalid input! Please
           re-enter: ");
    } while(ind < 0 || ind > 1);
} while(ind == 1);
c.shopped += bill;
//write to transaction file and update customer points
try
{
    PrintWriter pw = new PrintWriter(new BufferedWriter(new
       FileWriter(custfolder + "\\" + c.id + ".txt")));
    pw.println(c.name);
    pw.println(c.phone);
    pw.println(c.pass);
    pw.println(c.shopped);
    pw.close();
```

```
            pw = new PrintWriter(new BufferedWriter(new
                FileWriter(fileTransaction,true)));
            if(bill > 0)
            {
                DateFormat df = new SimpleDateFormat("dd/MM/yy HH:mm:ss");
                Date dateobj = new Date();
                pw.println(df.format(dateobj) + "\t" + c.id + "(" + c.name + ")\tRs. "
                    + bill);
                for(PurchaseItem item : list)
                {
                    pw.println(item.item.name+": " + item.num + " x Rs " +
                        item.item.price + " = " + item.totprice);
                }
                pw.println();
                System.out.print("Payable price: Rs. " + bill);
            }
            pw.close();
        }
        catch(Exception e)
        {
            System.out.print("\nError in writing to file! Exception code: " + e);
        }
    }
}
```

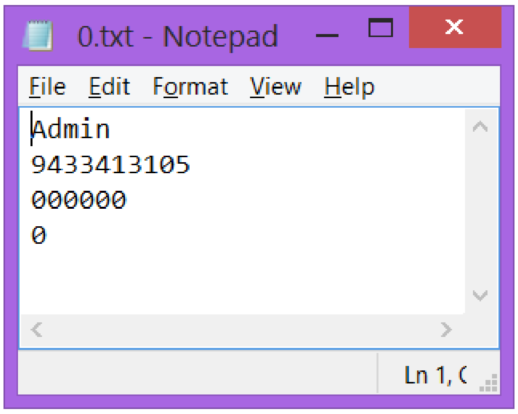
```java
// User defined datatype of Items in store
import java.util.*;
class Item
{
    int id;         //stores id of item
    String name;    //stores name of item
    int quantity;   //stores quantity of item in stock
    int price;      //stores unit price
    //non-parameterized constructor
    Item()
    {
        id = quantity = price = 0;
        name = "";
    }
    //parameterized constructor
    Item(int i, String n, int q, int p)
    {
        id = i;
        name = n;
        quantity = q;
        price = p;
    }
    //binary search for item id in list
    static int bsearch(ArrayList<Item> al, int a, int b, int k)
    {
        int m = (a + b) / 2;
        if(al.get(m).id == k) return m;
        if(al.get(m).id < k) return bsearch(al,m+1,b,k);
        if(al.get(m).id > k) return bsearch(al,a,m-1,k);
        return -1;
    }
    static int search(ArrayList<Item> al, int key)
    {
        return bsearch(al,0,al.size()-1,key);
    }
    //linear for item name in list
    static int search(ArrayList<Item> al, String key)
    {
        for(int i = 0 ; i < al.size() ; i++)
        {
            if((al.get(i).name).equals(key))
            return i;
        }
        return -1;
    }
}
```

```
// User defined datatype of Customer
import java.util.*;
class Customer
{
    int id;        //stores id of customer
    String name;   //stores name of customer
    long phone;    //stores phone number of customer
    String pass;   //stores password of customer
    int shopped;   //stores net worth of shopping
    //parameterized constructor
    Customer(int x, String y, long z, String p, int point)
    {
        id = x;
        name = y;
        phone = z;
        pass = p;
        shopped = point;
    }
    /* method to grant access to account
     * returns false if incorrect password input 3 times
     */
    boolean login()
    {
        System.out.print("\nPlease Enter Password: ");
        int c = 0;
        String input;
        Scanner sc = new Scanner(System.in);
        do
        {
            input = sc.next();
            c++;
            if(!input.equals(pass) && c<3)
                System.out.print("Incorrect Attempt! You have " +(3-c) +" attempts
                    remaining. Please re-enter password: ");
        } while(!input.equals(pass) && c<3);
        if(c == 3) return false;
        return true;
    }
    //binary search for customer id in list
    static int bsearch(ArrayList<Customer> al, int a, int b, int k)
    {
        int m = (a+b)/2;
        if(al.get(m).id == k) return m;
        if(al.get(m).id < k) return bsearch(al,m+1,b,k);
        if(al.get(m).id > k) return bsearch(al,a,m-1,k);
        return -1;
    }
    static int search(ArrayList<Customer> al,int id)
    {
        return bsearch(al, 0, al.size()-1, id);
    }
    //linear search for customer name in list
    static int search(ArrayList<Customer> al, String key)
    {
        for(int i = 0 ; i < al.size() ; i++)
        {
            if((al.get(i).name).equals(key))
            return i;
        }
        return -1;
    }
}
```

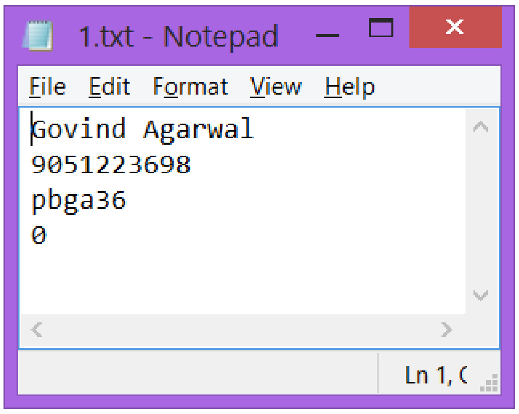
```
// User defined datatype of each purchase
class PurchaseItem
{
    Item item;     //sotres item purchased
    int num;       //stores number of items purchased
    int totprice; //stores net amount for current object
    //non parameterized constructor
    PurchaseItem()
    {
        item = null;
        num = 0;
        totprice = 0;
    }
    //parameterized constructor
    PurchaseItem(Item i, int n)
    {
        item = i;
        num = n;
        totprice = i.price * num;
    }
}
```

Input files

Customer files

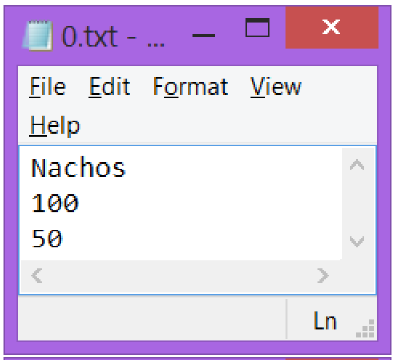


```
Admin
9433413105
000000
0
```

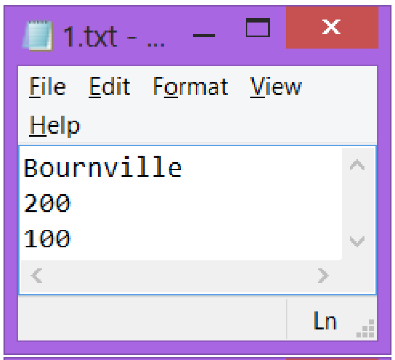


```
Govind Agarwal
9051223698
pbga36
0
```

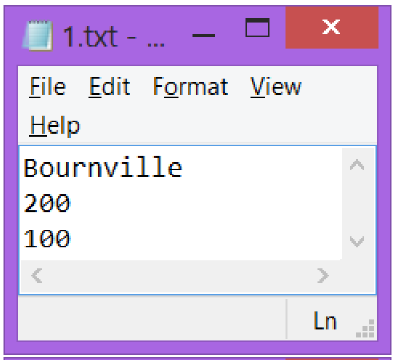
Items files



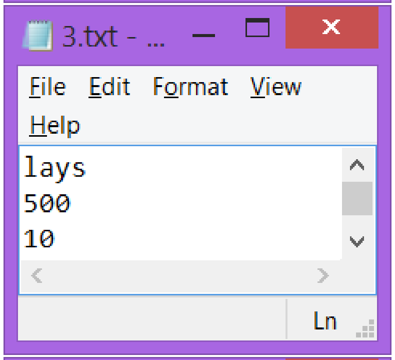
```
Nachos
100
50
```



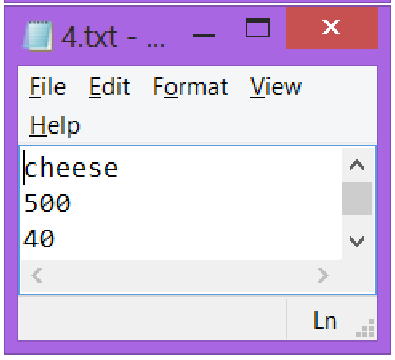
```
Bournville
200
100
```



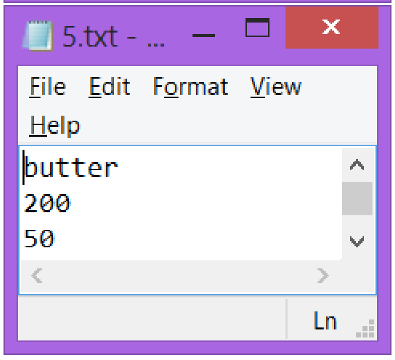
```
Bournville
200
100
```



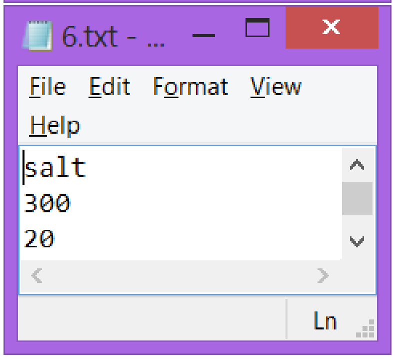
```
lays
500
10
```



```
cheese
500
40
```

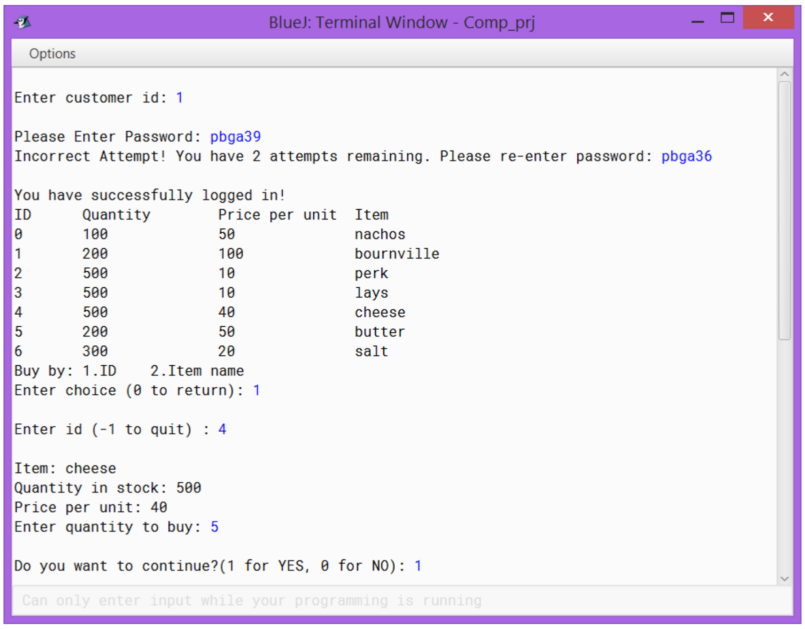


```
butter
200
50
```

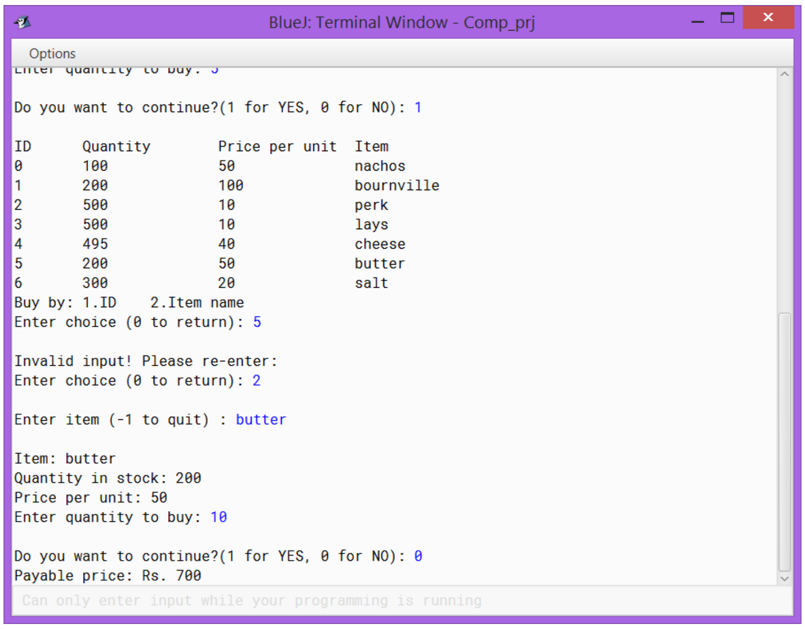


```
salt
300
20
```

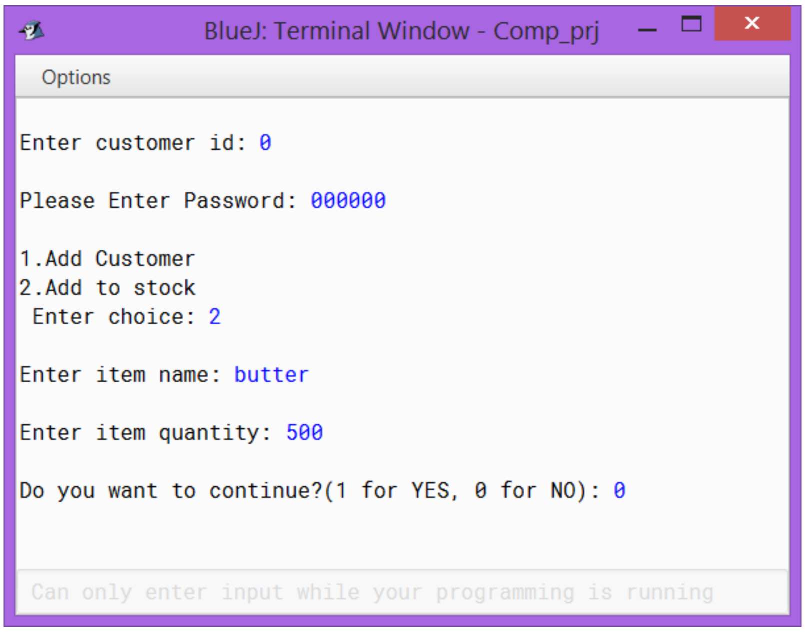
Output



```
Enter customer id: 1

Please Enter Password: pbga39
Incorrect Attempt! You have 2 attempts remaining. Please re-enter password: pbga36

You have successfully logged in!
ID      Quantity        Price per unit  Item
0       100             50              nachos
1       200             100             bournville
2       500             10              perk
3       500             10              lays
4       500             40              cheese
5       200             50              butter
6       300             20              salt
Buy by: 1.ID    2.Item name
Enter choice (0 to return): 1

Enter id (-1 to quit) : 4

Item: cheese
Quantity in stock: 500
Price per unit: 40
Enter quantity to buy: 5

Do you want to continue?(1 for YES, 0 for NO): 1
Can only enter input while your programming is running
```

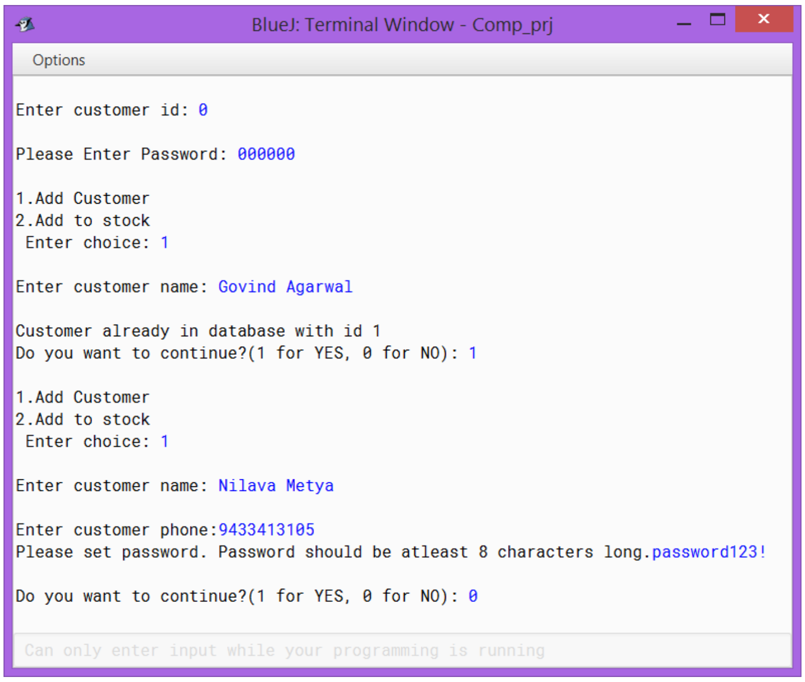


```
Enter quantity to buy: 5

Do you want to continue?(1 for YES, 0 for NO): 1

ID      Quantity        Price per unit  Item
0       100             50              nachos
1       200             100             bournville
2       500             10              perk
3       500             10              lays
4       495             40              cheese
5       200             50              butter
6       300             20              salt
Buy by: 1.ID    2.Item name
Enter choice (0 to return): 5

Invalid input! Please re-enter:
Enter choice (0 to return): 2

Enter item (-1 to quit) : butter

Item: butter
Quantity in stock: 200
Price per unit: 50
Enter quantity to buy: 10

Do you want to continue?(1 for YES, 0 for NO): 0
Payable price: Rs. 700
Can only enter input while your programming is running
```


BlueJ: Terminal Window - Comp_prj
Options

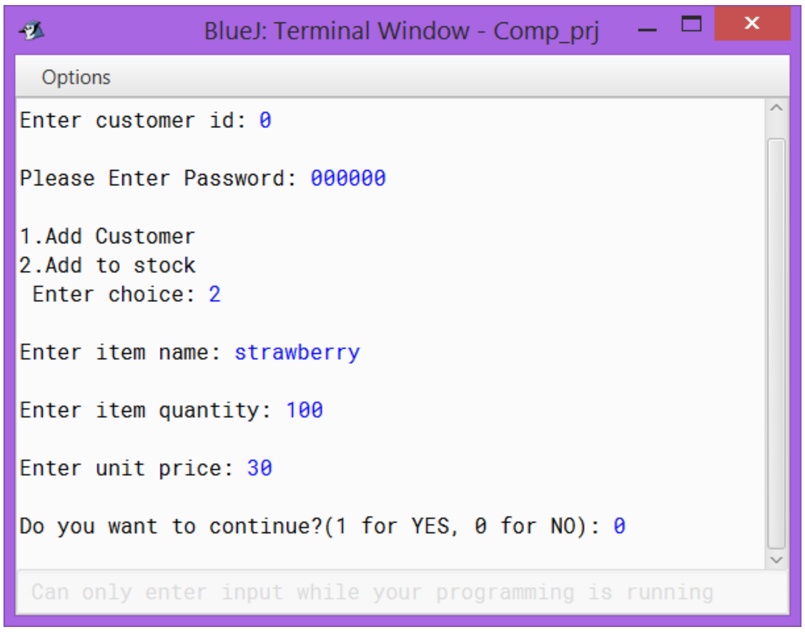
```
Enter customer id: 0

Please Enter Password: 000000

1.Add Customer
2.Add to stock
 Enter choice: 2

Enter item name: butter

Enter item quantity: 500

Do you want to continue?(1 for YES, 0 for NO): 0
```

Can only enter input while your programming is running



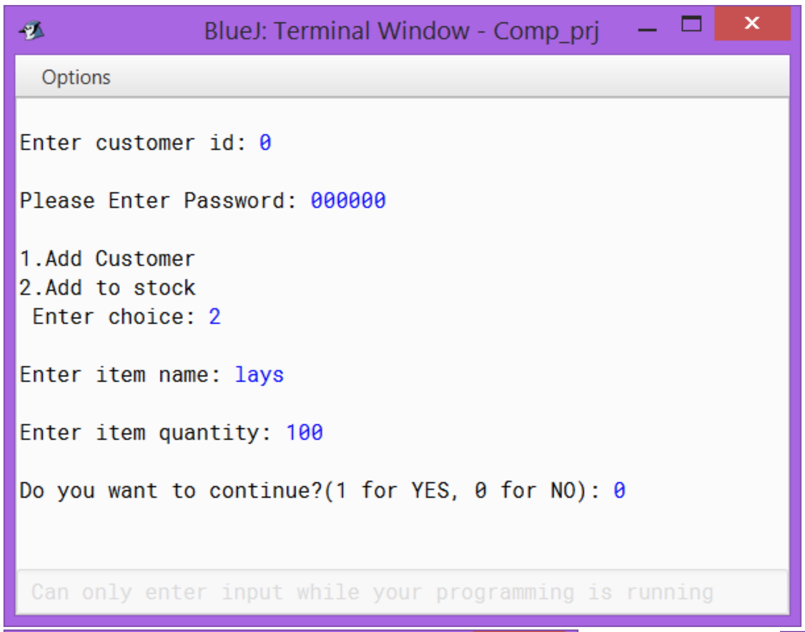
```
Enter customer id: 0

Please Enter Password: 000000

1.Add Customer
2.Add to stock
 Enter choice: 1

Enter customer name: Govind Agarwal

Customer already in database with id 1
Do you want to continue?(1 for YES, 0 for NO): 1

1.Add Customer
2.Add to stock
 Enter choice: 1

Enter customer name: Nilava Metya

Enter customer phone:9433413105
Please set password. Password should be atleast 8 characters long.password123!

Do you want to continue?(1 for YES, 0 for NO): 0
```

Can only enter input while your programming is running


BlueJ: Terminal Window - Comp_prj
Options

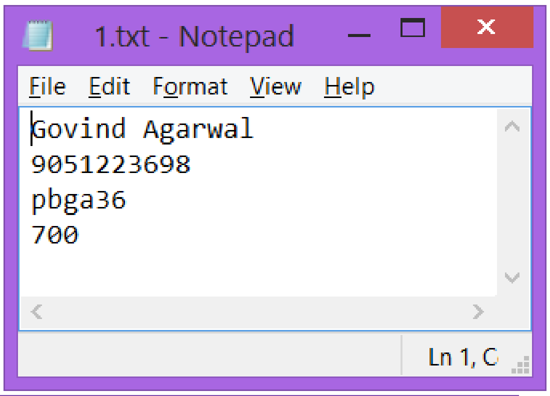
```
Enter customer id: 0

Please Enter Password: 000000

1.Add Customer
2.Add to stock
 Enter choice: 2

Enter item name: strawberry

Enter item quantity: 100

Enter unit price: 30

Do you want to continue?(1 for YES, 0 for NO): 0
```

Can only enter input while your programming is running

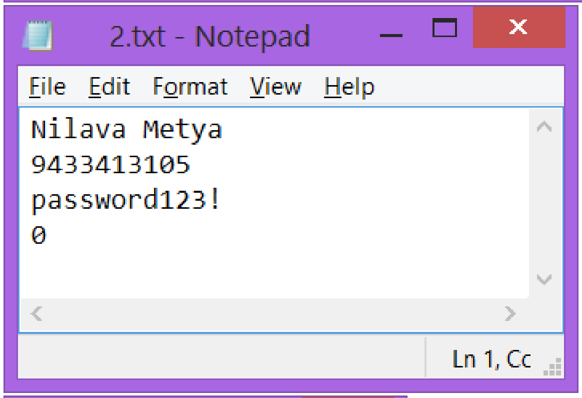

BlueJ: Terminal Window - Comp_prj
Options

```
Enter customer id: 0

Please Enter Password: 000000

1.Add Customer
2.Add to stock
 Enter choice: 2

Enter item name: lays

Enter item quantity: 100

Do you want to continue?(1 for YES, 0 for NO): 0
```

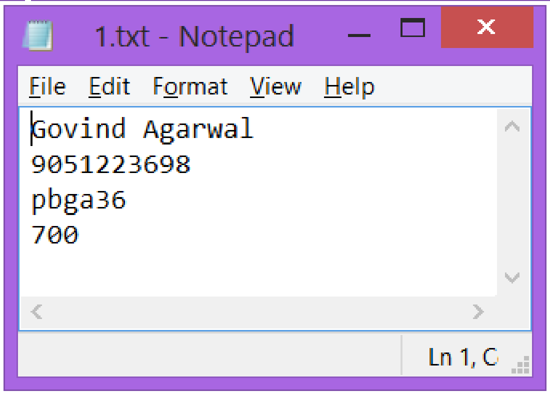
Can only enter input while your programming is running



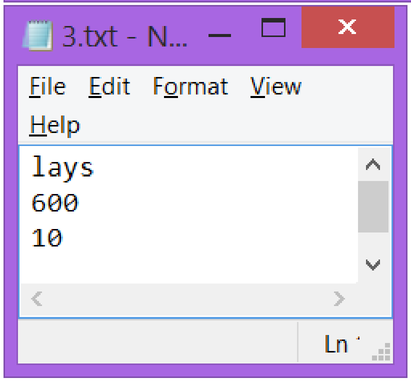
```
Govind Agarwal
9051223698
pbga36
700
```



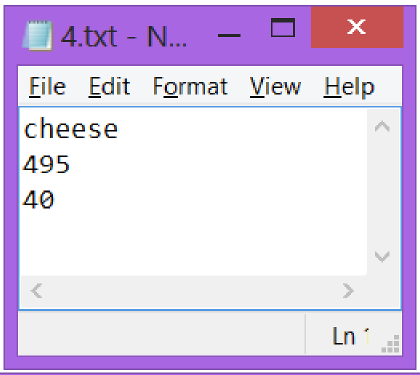
```
Nilava Metya
9433413105
password123!
0
```



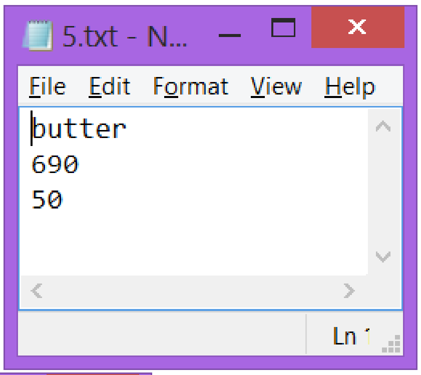
```
Govind Agarwal
9051223698
pbga36
700
```



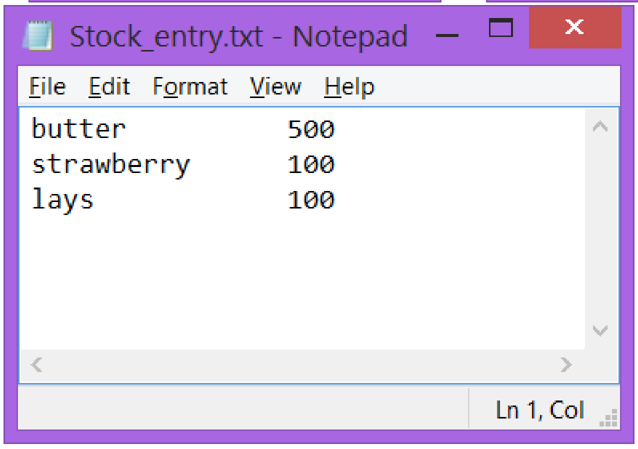
```
lays
600
10
```



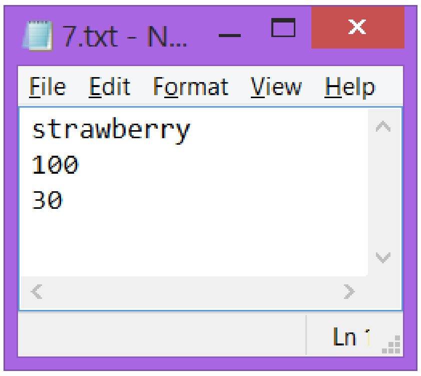
```
cheese
495
40
```



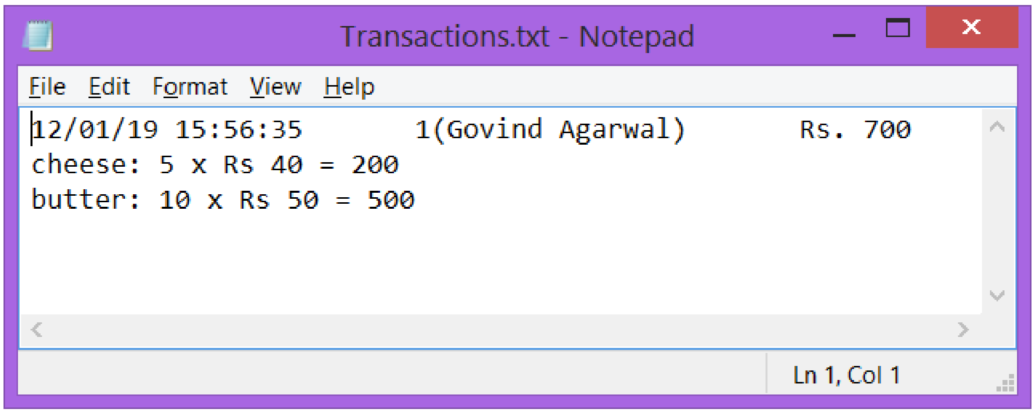
```
butter
690
50
```



```
butter          500
strawberry      100
lays            100
```



```
strawberry
100
30
```



```
12/01/19 15:56:35       1(Govind Agarwal)       Rs. 700
cheese: 5 x Rs 40 = 200
butter: 10 x Rs 50 = 500
```