Assignment 1

Name: Aditya Sawant

Roll No: 31302

Topic: Restaurant Management

Assignment - 1

Title : Collection and Generics

Problem Stmt : Develop java application for any system using collection framework and generics

Objective:
- To onderstand collection framework and generics
- To study different types of data structures availaible.

Outcome :
- Development of application which uses collection framework and generics.
- Implementation of primitive operations on data structure

Theory :
The data structures provided by the java utility packages are very powerful and perform a wide range of functions. The collection framework was designed to meet several goals, such as :

- the framework had to be high performance. The implementations of fundamental collections were to be highly effecient.

- the framework had to allow different types of collection to work in a similar manner and with high degree of interoperability.

- The framework had to extend/adapt a collection easily.

~~Collections used are~~.

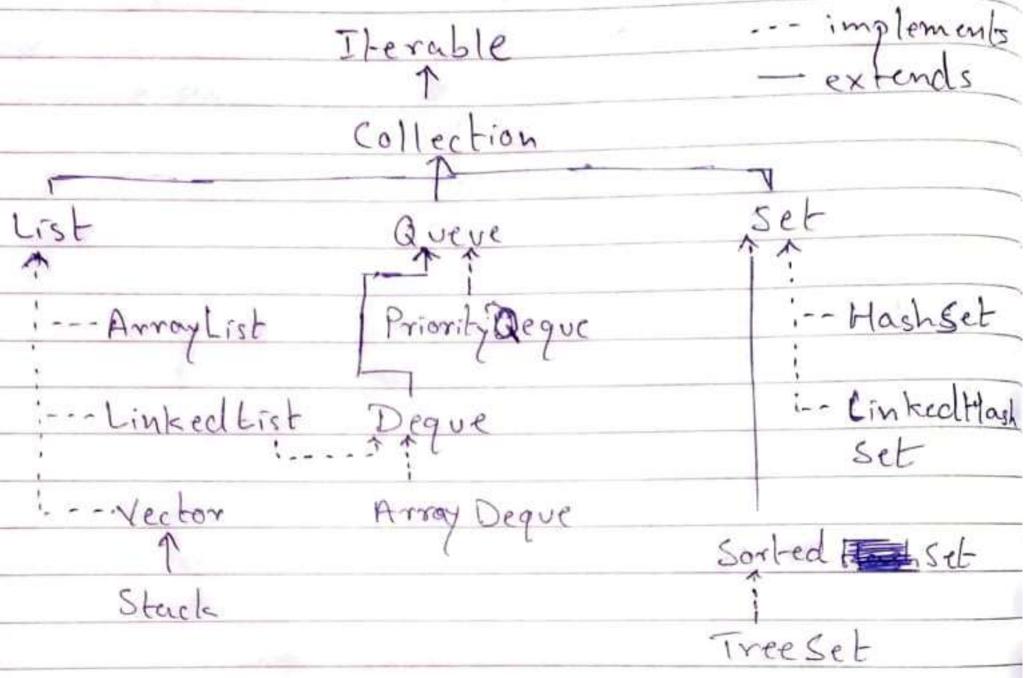
- Arraylist:
  It provides us with dynamic arrays in Java. Though, it maybe slower than standard arrays, it can be useful where lots of manipulation of ~~data~~ array is needed. It implements List interface and is present in java.util. package.

- Linkedlist:
  It is an implementation of linked list data structure which is linear structure where elements are not stored in contiguos locations and every element is seperate object.

- Queue Interface:
  It is a FIFO data structure used to store data where order of element matter. It is implemented in classes like Deque, ArrayQueue and PriorityQueue.

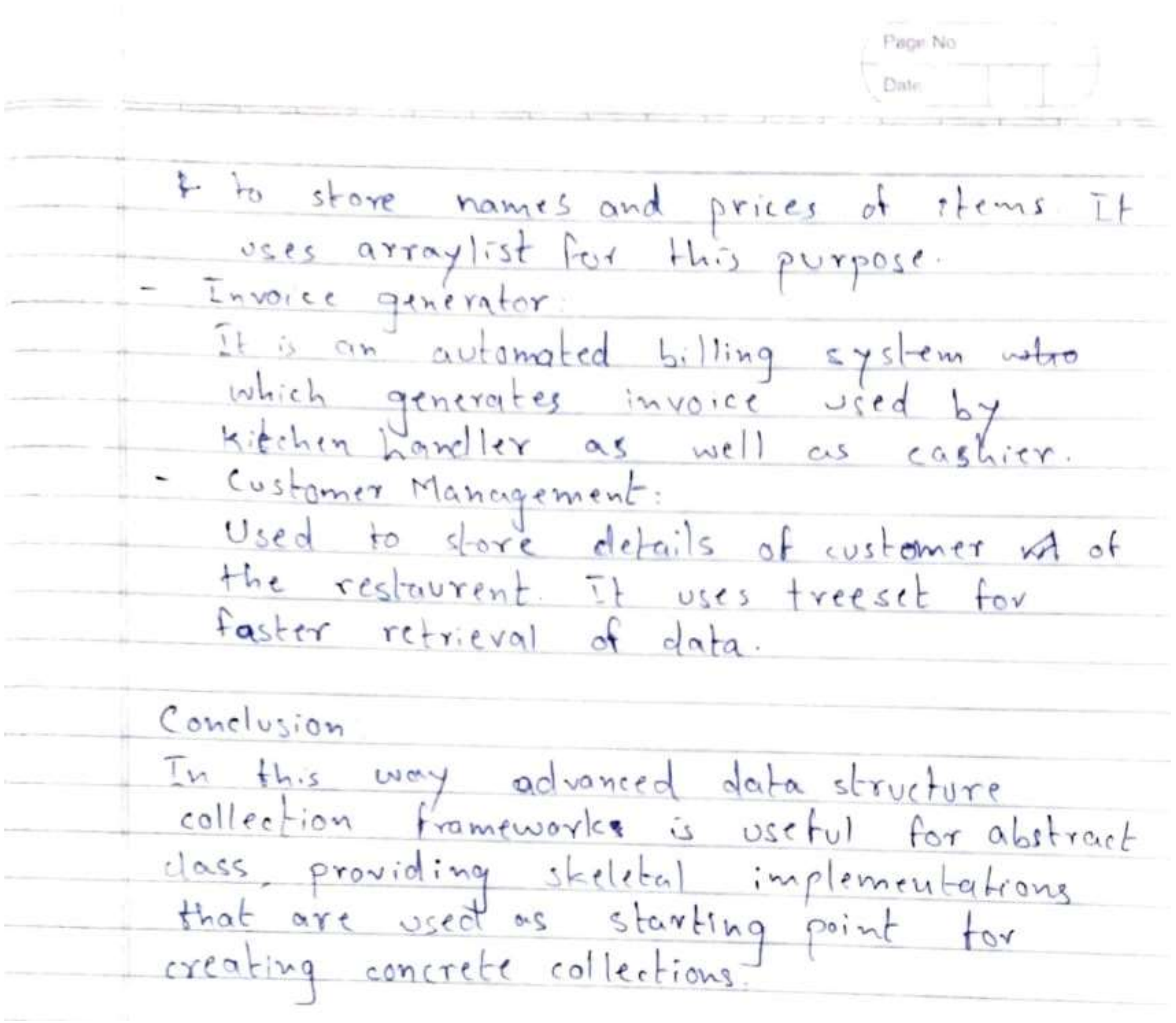
Page No.
Date :

Heirarchy of Collection Framework



Algorithm:

- Main function displays available options to the admin. The options include
  - Multi inventory system:
    Used to store name and quantity of available resources categorised by different types.
  - Employee management:
    It is used to store details of staff ~~an~~ and employees. It uses linkedlist from java collection framework for data manipulation
  - Menu generator:
    This is used for generating a menu for the restaurent. ~~me~~ It includes ability



Page No
Date

& to store names and prices of items It uses arraylist for this purpose.

- Invoice generator:
  It is an automated billing system ~~who~~ which generates invoice used by kitchen handler as well as cashier.
- Customer Management:
  Used to store details of customer ~~of~~ of the restaurent. It uses treeset for faster retrieval of data.

Conclusion

In this way advanced data structure collection frameworks is useful for abstract class, providing skeletal implementations that are used as starting point for creating concrete collections.

Code:

```
import java.util.*;

import java.io.*;

import java.net.*;


class InventoryManagement

{

        class Inventory

        {

                class InventoryItem

                {

                        String itemName;
```

```
		int itemQuantity;
		InventoryItem(){}
		InventoryItem(String name, int qty)
		{
			itemName = name;
			itemQuantity = qty;
		}
	}
	ArrayList<InventoryItem> inventory = new ArrayList<InventoryItem>();
	String inventoryName;
	Scanner sc = new Scanner(System.in);
	Inventory(String name)
	{
		inventoryName = name;
	}
	Inventory(){}
	void createList()
	{
		while(true)
		{
			System.out.println("Enter item name: ");
			String itemName = sc.nextLine();
			if(itemName.equals("stop"))
			{
				return;
			}
			System.out.println("Enter item quantity: ");
			int itemQuantity = sc.nextInt();
			sc.nextLine();
			inventory.add(new InventoryItem(itemName, itemQuantity));
		}
```

```
}
void addItem()
{
        InventoryItem obj = new InventoryItem();
        System.out.println("Enter item name: ");
        obj.itemName = sc.nextLine();
        System.out.println("Enter item quantity: ");
        obj.itemQuantity = sc.nextInt();
        inventory.add(obj);
}
void deleteItem()
{
        System.out.println("Enter item to be deleted: ");
        String name = sc.nextLine();
        for(int i=0; i<inventory.size(); i++)
        {
                if(name.equals(inventory.get(i).itemName))
                {
                        inventory.remove(i);
                }
        }
}
void updateInventory()
{
        System.out.println("Enter item to be updated: ");
        String name = sc.nextLine();
        System.out.println("Enter new quantity: ");
        int qty = sc.nextInt();
        for(int i=0; i<inventory.size(); i++)
        {
                if(name.equals(inventory.get(i).itemName))
                {
                    inventory.remove(i);
                    inventory.add(i, new InventoryItem(name, qty));
                    return;
                }
            }
        }
        void displayInventory()
        {
            System.out.println("------------------");
            System.out.println("|Name\t|Quantity|");
            System.out.println("------------------");
            for(InventoryItem item: inventory)
            {
                System.out.println("|" + item.itemName + "\t|" + item.itemQuantity
+ "      |");
                System.out.println("------------------");
            }
        }
        void displayMenu()
        {
            System.out.println("");
            System.out.println("****Inventory Options****");
            System.out.println("Current inventory: " + inventoryName);
            System.out.println("1. Add new item");
            System.out.println("2. Delete an item");
            System.out.println("3. Update inventory");
            System.out.println("4. Display inventory");
            System.out.println("5. Back");
            int choice = 0;
            while(true)
```

```
{
    try
    {
        System.out.println("Enter choice: ");
        choice = sc.nextInt();
        sc.nextLine();
        switch(choice)
        {
            case 1:
                addItem();
                break;
            case 2:
                deleteItem();
                break;
            case 3:
                updateInventory();
                break;
            case 4:
                displayInventory();
                break;
            case 5:
                return;
            default:
                System.out.println("Invalid choice.");
        }
    }
    catch(InputMismatchException e)
    {
        System.out.println("Invalid input. Please try again.");
        sc.nextLine();
    }
```

```
					}
				}
			}
			ArrayList<Inventory> list = new ArrayList<Inventory>();
			Scanner sc = new Scanner(System.in);
			int currentList;
			void createNewList()
			{
				System.out.println("Enter inventory name: ");
				String name = sc.nextLine();
				list.add(new Inventory(name));
				list.get(list.size()-1).createList();
			}
			void displayList()
			{
				if(list.size() == 0)
				{
					System.out.println("No list present. Please add a list.");
					return;
				}
				System.out.println("");
				System.out.println("****Current Lists****");
				for(int i=0; i<list.size(); i++)
				{
					System.out.println((i+1) + ". " + list.get(i).inventoryName);
				}
				System.out.println("Select list: ");
				currentList = sc.nextInt()-1;
				System.out.println("Selected " + list.get(currentList).inventoryName + ". Please 
select an operation.");
			}
```

```
void updateList()
{
        list.get(currentList).displayMenu();
}
void deleteList()
{
        list.remove(currentList);
}
void inventoryManagementMenu()
{
        System.out.println("");
        System.out.println("****Inventory Management****");
        System.out.println("1. Create new list");
        System.out.println("2. Display lists");
        System.out.println("3. Update list");
        System.out.println("4. Delete list");
        System.out.println("5. Back");
        int choice = 0;
        while(true)
        {
                try
                {
                        System.out.println("Enter choice: ");
                        choice = sc.nextInt();
                        sc.nextLine();
                        switch(choice)
                        {
                                case 1:
                                        createNewList();
                                        break;
                                case 2:
```

```
                        displayList();
                        break;
                    case 3:
                        updateList();
                        break;
                    case 4:
                        deleteList();
                        break;
                    case 5:
                        return;
                    default:
                        System.out.println("Invalid choice.");
                }
            }
            catch(InputMismatchException e)
            {
                System.out.println("Invalid input. Please try again.");
                sc.nextLine();
            }
            catch(IndexOutOfBoundsException e)
            {
                System.out.println("Error 404: List not found.");
            }
        }
    }
}
class EmployeeManagement
{
    class EmployeeDetails
    {
        String name;
        int age;
        long phoneNo;
        String designation;
        String address;
}
LinkedList<EmployeeDetails> employeeList= new LinkedList<EmployeeDetails>();
Scanner sc = new Scanner(System.in);
void insertRecord()
{
        EmployeeDetails obj = new EmployeeDetails();
        System.out.print("Name: ");
        obj.name = sc.nextLine();
        System.out.print("Age: ");
        obj.age = sc.nextInt();
        System.out.print("Phone No: ");
        obj.phoneNo = sc.nextInt();
        sc.nextLine();
        System.out.print("Designation: ");
        obj.designation = sc.nextLine();
        System.out.print("Address: ");
        obj.address = sc.nextLine();
        employeeList.add(obj);
}
void displayRecords()
{
        System.out.println("");
        System.out.println("****Employee List****");
        System.out.println("Name | Age | Phone No | Designation | Address");
        for(EmployeeDetails obj: employeeList)
        {
```

```
                                System.out.println(obj.name + " | " + obj.age + " | " + obj.phoneNo + " | " +
obj.designation + " | " + obj.address);
                        }
            }
            void updateRecord()
            {
                        System.out.print("Enter name of employee to be updated: ");
                        String name = sc.nextLine();
                        int i;
                        for(i=0; i<employeeList.size(); i++)
                        {
                                    if(name.equals(employeeList.get(i).name))
                                    {
                                                break;
                                    }
                        }
                        if(i == employeeList.size())
                        {
                                    System.out.println("Error 404: Record not found.");
                                    return;
                        }
                        System.out.println("Choose data to be updated: ");
                        System.out.println("1. Name");
                        System.out.println("2. Age");
                        System.out.println("3. Phone number");
                        System.out.println("4. Designation");
                        System.out.println("5. Address");
                        System.out.println("6. Back");
                        while(true)
                        {
                                    System.out.print("Enter choice: ");
```

```
                int choice = sc.nextInt();
                System.out.print("Enter updated data: ");
                sc.nextLine();
                String data = sc.nextLine();
                switch(choice)
                {
                        case 1:
                                employeeList.get(i).name = data;
                                break;
                        case 2:
                                employeeList.get(i).age = Integer.parseInt(data);
                                break;
                        case 3:
                                employeeList.get(i).phoneNo = Integer.parseInt(data);
                                break;
                        case 4:
                                employeeList.get(i).designation = data;
                                break;
                        case 5:
                                employeeList.get(i).address = data;
                                break;
                        case 6:
                                return;
                        default:
                                System.out.println("Invalid choice.");
                }
        }
}
/*<E> void newValue(int i, E data)
{
        if(i == 2 || i == 3)
        {


        }
}*/
void deleteRecord()
{
        System.out.print("Enter name of employee to be updated: ");
        String name = sc.nextLine();
        int i;
        for(i=0; i<employeeList.size(); i++)
        {
                if(name.equals(employeeList.get(i).name))
                {
                        employeeList.remove(i);
                }
        }
        if(i == employeeList.size())
        {
                System.out.println("Error 404: Record not found.");
                return;
        }
        else
        {
                System.out.println("Record deleted.");
        }
}
void employeeManagementMenu()
{
        System.out.println("");
        System.out.println("****Employee Management****");
        System.out.println("1. Insert record");
```

```
                System.out.println("2. Display records");
                System.out.println("3. Update record");
                System.out.println("4. Delete record");
                System.out.println("5. Back");
                while(true)
                {
                        System.out.print("Enter choice: ");
                        int choice = sc.nextInt();
                        sc.nextLine();
                        switch(choice)
                        {
                                case 1:
                                        insertRecord();
                                        break;
                                case 2:
                                        displayRecords();
                                        break;
                                case 3:
                                        updateRecord();
                                        break;
                                case 4:
                                        deleteRecord();
                                        break;
                                case 5:
                                        return;
                                default:
                                        System.out.println("Invalid choice.");
                        }
                }
        }
}
```

```
}

class MenuGenerator
{
	class MenuItem
	{
		String name;
		int price;
		MenuItem(String n,int p)
		{
			name = n;
			price = p;
		}
	}
	Scanner sc = new Scanner(System.in);
	ArrayList<MenuItem> menu = new ArrayList<MenuItem>();
	void createMenuList()
	{
		while(true)
		{
			System.out.print("Item name: ");
			String name = sc.nextLine();
			if(name.equals("stop"))
			{
				return;
			}
			System.out.print("Item price: ");
			int price = sc.nextInt();
			sc.nextLine();
			menu.add(new MenuItem(name, price));
		}
	}
	void displayMenuItems()
```

```
{
        System.out.println("****Food Menu****");
        System.out.println("Item Name    Price");
        for(MenuItem m : menu)
        {
                System.out.println(m.name + " " + m.price);
        }
}
void addMenuItem()
{
        System.out.print("Item name: ");
        String name = sc.nextLine();
        System.out.print("Item price: ");
        int price = sc.nextInt();
        sc.nextLine();
        menu.add(new MenuItem(name, price));
}
void deleteMenuItem()
{
        String name;
        System.out.print("Enter name of item to be deleted: ");
        name = sc.nextLine();
        for(int i=0; i<menu.size(); i++)
        {
                if(name.equals(menu.get(i).name))
                {
                        menu.remove(i);
                }
        }
}
void menuGenerator()
```

```
{
        System.out.println("");
        System.out.println("****Menu Generator****");
        System.out.println("1. Create menu");
        System.out.println("2. Display menu");
        System.out.println("3. Add menu item");
        System.out.println("4. Delete menu item");
        System.out.println("5. Back");
        while(true)
        {
                System.out.print("Enter choice: ");
                int choice = sc.nextInt();
                sc.nextLine();
                switch(choice)
                {
                        case 1:
                                createMenuList();
                                break;
                        case 2:
                                displayMenuItems();
                                break;
                        case 3:
                                addMenuItem();
                                break;
                        case 4:
                                deleteMenuItem();
                                break;
                        case 5:
                                return;
                        default:
                                System.out.println("Invalid choice.");
                        }
                }
        }
}
class Invoice
{
        String customerName;
        int tableNo;
        HashMap<String,Integer> orderDetails = new HashMap<String,Integer>();
        Invoice(){}
        Invoice(String name, int table, HashMap<String,Integer> obj)
        {
                customerName = name;
                tableNo = table;
                orderDetails = obj;
        }
}
class InvoiceGenerator
{
  Scanner sc = new Scanner(System.in);
  LinkedList<Invoice> customerList = new LinkedList<Invoice>();
  KitchenHandler kh;
  InvoiceGenerator(KitchenHandler obj)
  {
        kh = obj;
  }
  void createInvoice()
  {
    System.out.print("Enter customer name: ");
    String name = sc.nextLine();
    System.out.print("Enter table number: ");
    int tableNo = sc.nextInt();
    sc.nextLine();
    HashMap<String,Integer> orderDetails = new HashMap<String,Integer>();
    while(true)
    {
        System.out.print("Item name: ");
        String itemName = sc.nextLine();
        if(itemName.equals("stop"))
        {
            break;
        }
        System.out.print("Item quantity: ");
        int itemQty = sc.nextInt();
        sc.nextLine();
        orderDetails.put(itemName,itemQty);
    }
    Invoice invoice = new Invoice(name,tableNo,orderDetails);
    customerList.add(invoice);
    kh.addNewOrder(invoice);
}
void deleteInvoice()
{
    System.out.println("Enter table no: ");
    int tblNo = sc.nextInt();
    int i;
    for(i=0; i<customerList.size(); i++)
    {
        if(tblNo == customerList.get(i).tableNo)
        {
            customerList.remove(i);
        }
    }
    if(i == customerList.size())
    {
        System.out.println("Error 404: Customer not found.");
    }
}
void displayCustomerQueue()
{
    if(customerList.size() == 0)
    {
        System.out.println("No records present.");
        return;
    }
    for(Invoice obj : customerList)
    {
        System.out.println("Name: " + obj.customerName);
        System.out.println("Table No: " + obj.tableNo);
        for(Map.Entry m : obj.orderDetails.entrySet()){
            System.out.println(m.getKey()+" "+m.getValue());
        }
    }
}
KitchenHandler invoiceGeneratorMenu()
{
    while(true)
    {
        try
        {
            System.out.println("");
            System.out.println("****Invoice Generator****");
            System.out.println("1. New Invoice");
            System.out.println("2. Display Invoice");
            System.out.println("3. Delete Invoice");
            System.out.println("4. Back");
            System.out.print("Enter choice: ");
            int choice = sc.nextInt();
            sc.nextLine();
            switch(choice)
            {
                case 1:
                    createInvoice();
                    break;
                case 2:
                    displayCustomerQueue();
                    break;
                case 3:
                    deleteInvoice();
                    break;
                case 4:
                    return kh;
                default:
                    System.out.println("Invalid choice.");
            }
        }
        catch(InputMismatchException e)
        {
            System.out.println("Error: Wrong input type. Please try again.");
            sc.nextLine();
        }
    }
  }
}
```

```
class KitchenHandler
{
   Queue<Invoice> orderQueue = new LinkedList<Invoice>();
   Scanner sc = new Scanner(System.in);
void addNewOrder(Invoice obj)
   {
          orderQueue.add(obj);
   }
   void displayOrderQueue()
   {
      System.out.println("");
      System.out.println("****Order Queue****");
      for(Invoice obj : orderQueue)
      {
         System.out.println("Name: " + obj.customerName);
         System.out.println("Table No: " + obj.tableNo);
         for (Map.Entry m : obj.orderDetails.entrySet()) {
            System.out.println(m.getKey() + " " + m.getValue());
         }
      }
   }
   void kitchenHandlerMenu()
   {
          while(true)
          {
                    System.out.println("");
                    System.out.println("****Kitchen Handler****");
                    System.out.println("1. Display order queue");
                    System.out.println("2. Request service");
                    System.out.println("3. Back");
                    System.out.print("Enter choice: ");
```

```
        int choice = sc.nextInt();
        switch(choice)
        {
            case 1:
                displayOrderQueue();
                break;
            case 2:
                orderQueue.remove();
                break;
            case 3:
                return;
            default:
                System.out.println("Invalid choice.");
        }
    }
   }
class CustomerHandler extends CustomerDetails
{
    class CustomerDetails implements Comparable<CustomerDetails>
    {
        String custName;
        long custPhoneNo;
        String custAddress;
        CustomerDetails(){}
        CustomerDetails(String name, int phoneNo, String address)
        {
            custName = name;
            custPhoneNo = phoneNo;
            custAddress = address;
        }
        public int compareTo(CustomerDetails cd)
```

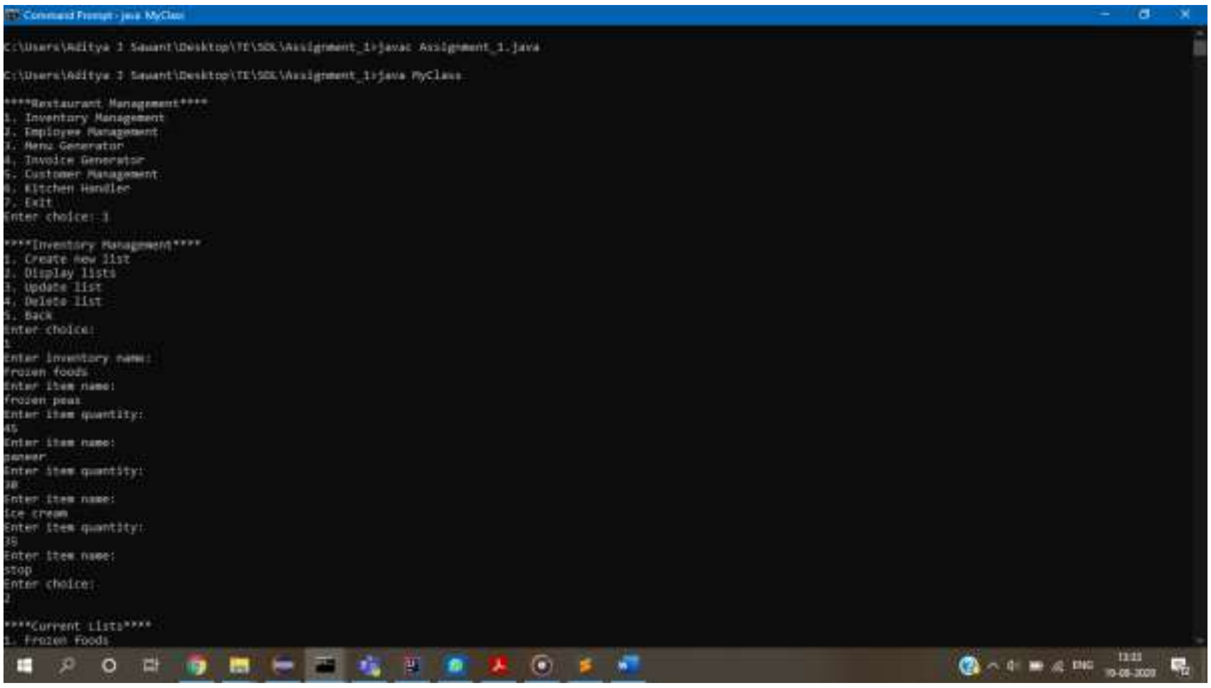
```
		{
			return this.custName.compareTo(cd.custName);
		}
	}
	TreeSet<CustomerDetails> customerList = new TreeSet<CustomerDetails>();
	Scanner sc = new Scanner(System.in);
	void newCustomer()
	{
		//CustomerDetails obj = new CustomerDetails();
		System.out.print("Name: ");
		String custName = sc.nextLine();
		System.out.print("Phone No: ");
		int custPhoneNo = sc.nextInt();
		sc.nextLine();
		System.out.print("Address: ");
		String custAddress = sc.nextLine();
		customerList.add(new CustomerDetails(custName, custPhoneNo, custAddress));
	}
	void updateCustomer()
	{
		System.out.print("Name: ");
		String name = sc.nextLine();
		CustomerDetails temp = new CustomerDetails();
		for(CustomerDetails obj : customerList)
		{
			if(obj.custName.equals(name))
			{
				temp = obj;
				customerList.remove(obj);
			}
		}
```

```
        System.out.print("1. Name");
        System.out.print("2. Phone No");
        System.out.print("3. Address");
        System.out.print("Enter choice: ");
        int choice = sc.nextInt();
        switch(choice)
        {
                case 1:
                        System.out.print("Enter new name: ");
                        temp.custName = sc.nextLine();
                        break;
                case 2:
                        System.out.print("Enter new phone number: ");
                        temp.custPhoneNo = sc.nextInt();
                        break;
                case 3:
                        System.out.print("Enter new address: ");
                        temp.custAddress = sc.nextLine();
                        break;
                default:
                        System.out.println("Invalid choice.");
        }
        customerList.add(temp);
}
void deleteCustomer()
{
        System.out.print("Name: ");
        String name = sc.nextLine();
        for(CustomerDetails obj : customerList)
        {
                if(obj.custName.equals(name))
```

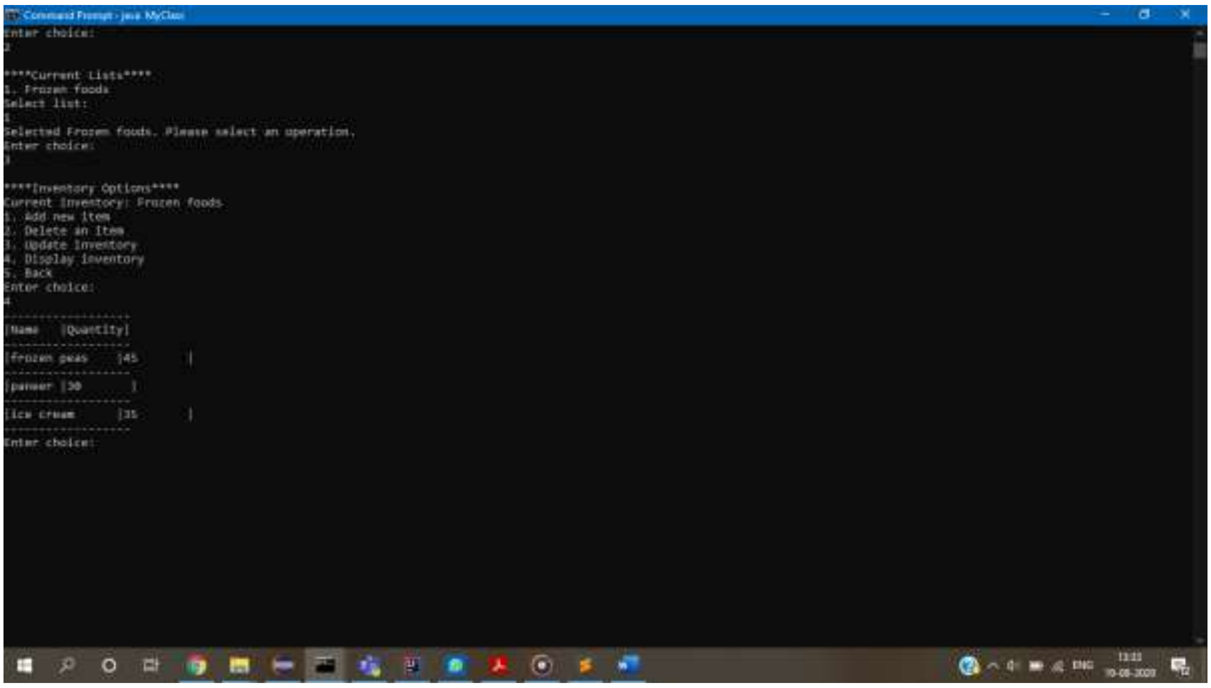
```
                    {
                              customerList.remove(obj);
                    }
          }
    }
    void displayCustomers()
    {
          for(CustomerDetails obj : customerList)
          {
                    System.out.println(obj.custName + " " + obj.custPhoneNo + " " +
obj.custAddress);
          }
    }
    void customerHandlerMenu()
    {
          while(true)
      {
        try
        {
          System.out.println("");
          System.out.println("****Customer Management****");
          System.out.println("1. New customer");
          System.out.println("2. Display customers");
          System.out.println("3. Update customer");
          System.out.println("4. Delete customer");
          System.out.println("5. Back");
          System.out.print("Enter choice: ");
          int choice = sc.nextInt();
          sc.nextLine();
          switch(choice)
          {
                case 1:
                    newCustomer();
                    break;
                case 2:
                    displayCustomers();
                    break;
                case 3:
                    updateCustomer();
                    break;
                case 4:
                    deleteCustomer();
                    break;
                case 5:
                        return;
                default:
                    System.out.println("Invalid choice.");
            }
        }
        catch(InputMismatchException e)
        {
            System.out.println("Error: Wrong input type. Please try again.");
            sc.nextLine();
        }
    }
          }
}
class MyClass
{
          public static void main(String[] args)
          {
                    InventoryManagement invMgmt = new InventoryManagement();
                EmployeeManagement empMgmt = new EmployeeManagement();
                MenuGenerator menuGen = new MenuGenerator();
                CustomerHandler custHandler = new CustomerHandler();
                KitchenHandler kh = new KitchenHandler();
                InvoiceGenerator invoiceGen = new InvoiceGenerator(kh);
                while(true)
        {
          try
          {
            System.out.println("");
            System.out.println("****Restaurant Management****");
            System.out.println("1. Inventory Management");
            System.out.println("2. Employee Management");
            System.out.println("3. Menu Generator");
            System.out.println("4. Invoice Generator");
            System.out.println("5. Customer Management");
            System.out.println("6. Kitchen Handler");
            System.out.println("7. Exit");
            System.out.print("Enter choice: ");
            Scanner sc = new Scanner(System.in);
            int choice = sc.nextInt();
            switch(choice)
            {
              case 1:
                invMgmt.inventoryManagementMenu();
                break;
              case 2:
                empMgmt.employeeManagementMenu();
                break;
              case 3:
                menuGen.menuGenerator();
                break;
              case 4:
                kh = invoiceGen.invoiceGeneratorMenu();
                break;
              case 5:
                    custHandler.customerHandlerMenu();
                    break;
              case 6:
                kh.kitchenHandlerMenu();
                break;
              case 7:
                return;
              default:
                System.out.println("Invalid input.");
            }
          }
          catch(InputMismatchException e)
          {
            System.out.println("Error: Wrong input type. Please try again.");
          }
        }
            }
}
```

Screenshots of output:

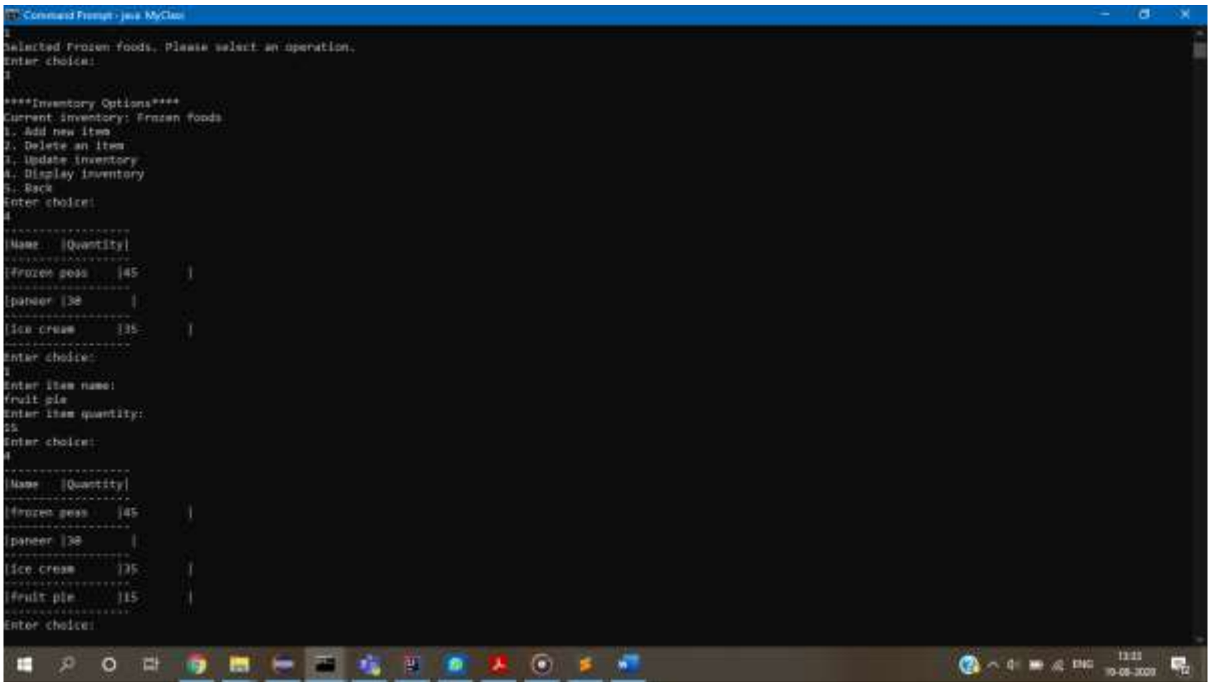
Inventory Management:



```
C:\Users\Aditya J Sawant\Desktop\TE\SDL\Assignment_1>javac Assignment_1.java

C:\Users\Aditya J Sawant\Desktop\TE\SDL\Assignment_1>java MyClass

****Restaurant Management****
1. Inventory Management
2. Employee Management
3. Menu Generator
4. Invoice Generator
5. Customer Management
6. Kitchen Handler
7. Exit
Enter choice: 1

****Inventory Management****
1. Create new list
2. Display lists
3. Update list
4. Delete list
5. Back
Enter choice:
1
Enter inventory name:
Frozen foods
Enter item name:
frozen peas
Enter item quantity:
45
Enter item name:
paneer
Enter item quantity:
30
Enter item name:
ice cream
Enter item quantity:
35
Enter item name:
stop
Enter choice:
2

****Current Lists****
1. Frozen foods
```

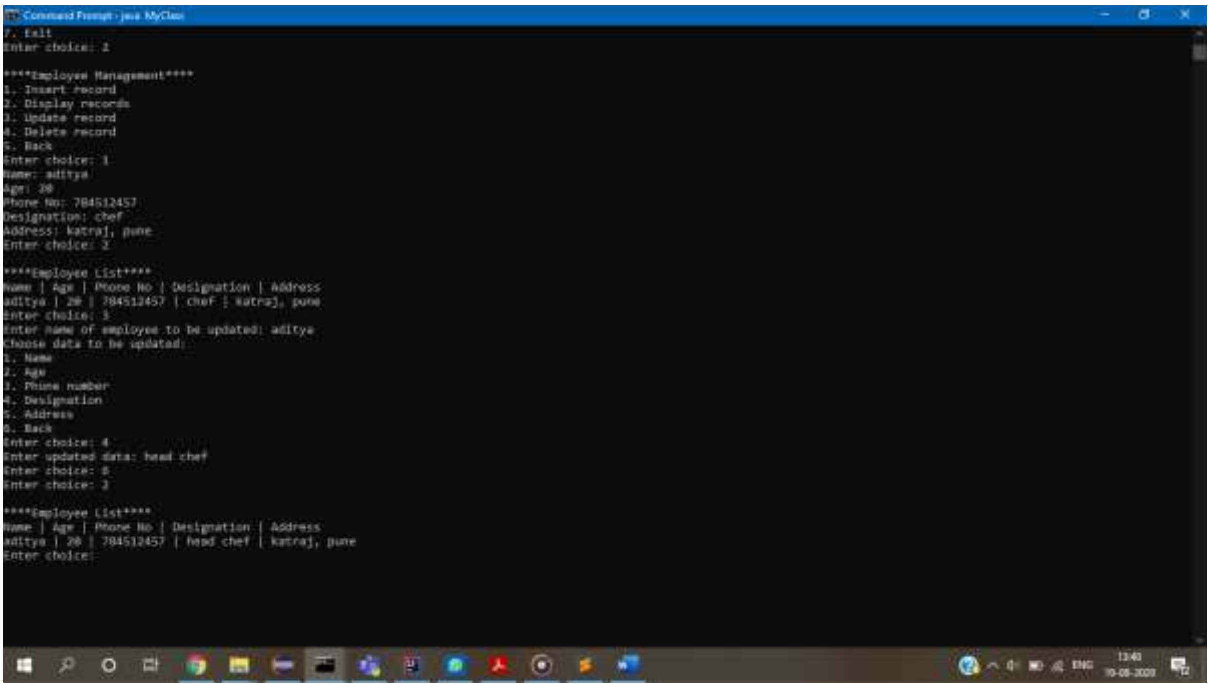


```
Enter choice:
2

****Current Lists****
1. Frozen foods
Select list:
1
Selected Frozen foods. Please select an operation.
Enter choice:
3

****Inventory Options****
Current Inventory: Frozen foods
1. Add new item
2. Delete an item
3. Update inventory
4. Display inventory
5. Back
Enter choice:
4
-------------------
|Name    |Quantity|
-------------------
|frozen peas     |45       |
-------------------
|paneer |30       |
-------------------
|ice cream       |35       |
-------------------
Enter choice:
```



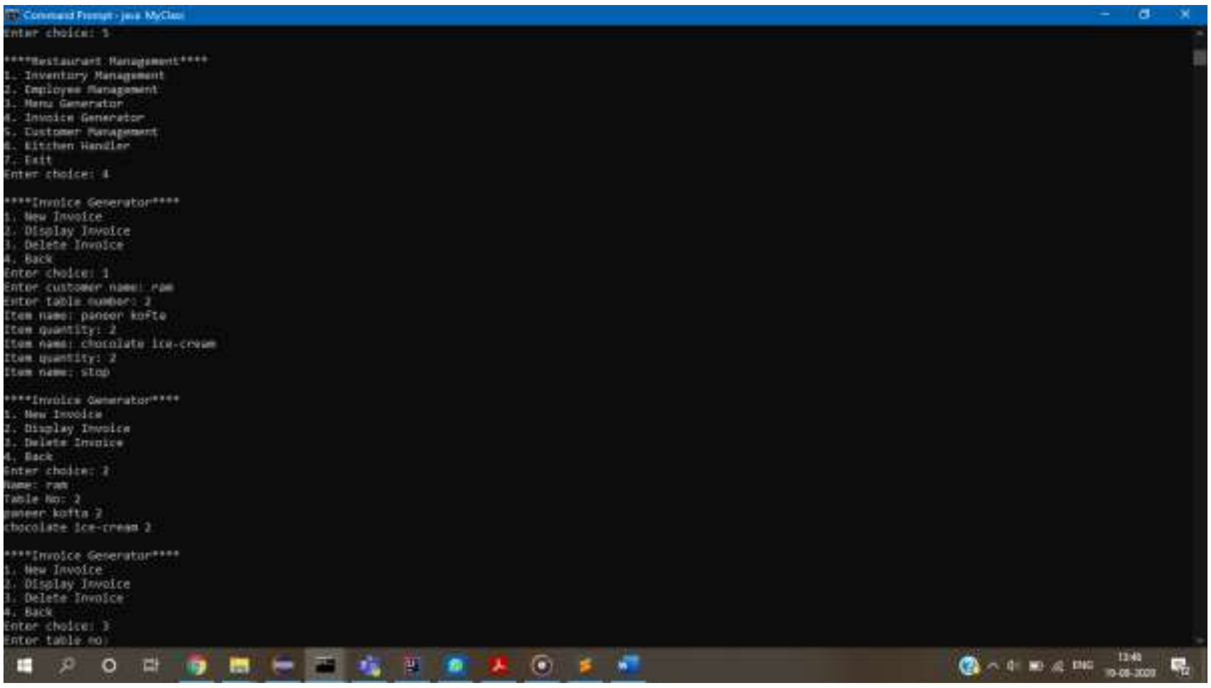
```
Command Prompt - java MyClass
1
Selected Frozen foods. Please select an operation.
Enter choice:
3

****Inventory Options****
Current inventory: Frozen foods
1. Add new item
2. Delete an item
3. Update inventory
4. Display inventory
5. Back
Enter choice:
4
------------------
|Name   |Quantity|
------------------
|frozen peas    |45       |
------------------
|paneer |30       |
------------------
|ice cream      |35       |
------------------
Enter choice:
1
Enter item name:
fruit pie
Enter item quantity:
15
Enter choice:
4
------------------
|Name   |Quantity|
------------------
|frozen peas    |45       |
------------------
|paneer |30       |
------------------
|ice cream      |35       |
------------------
|fruit pie      |15       |
------------------
Enter choice:
```

Employee Management:

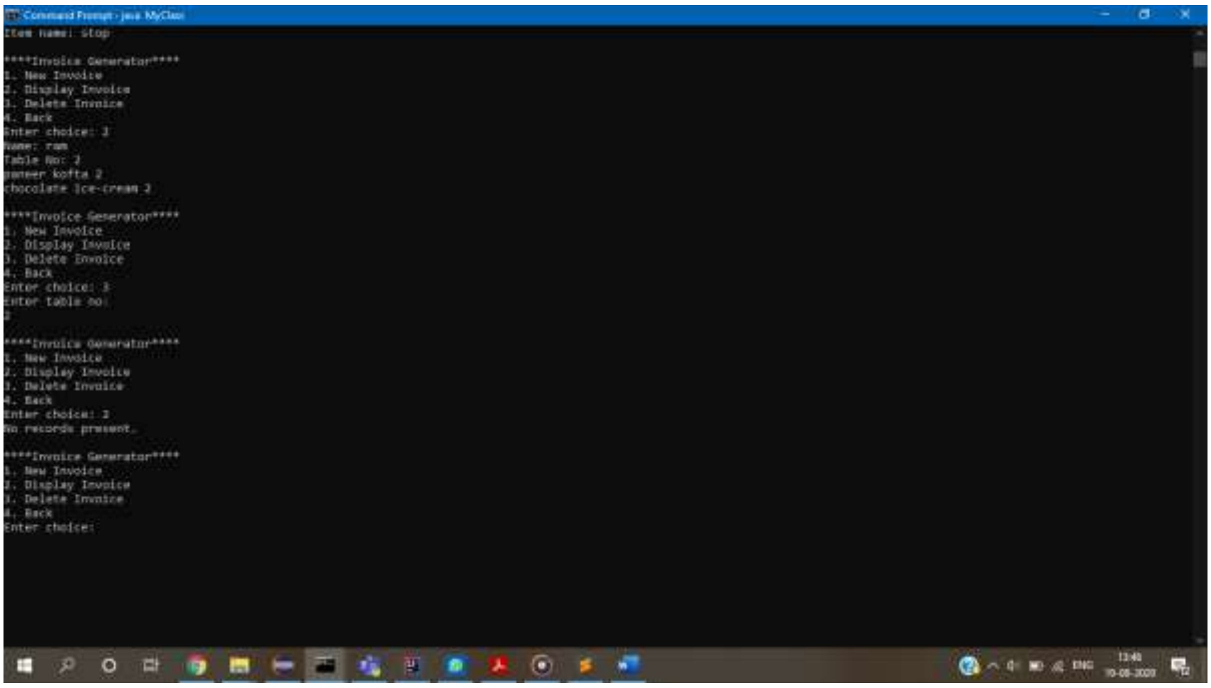


```
Command Prompt - java MyClass
7. Exit
Enter choice: 2

****Employee Management****
1. Insert record
2. Display records
3. Update record
4. Delete record
5. Back
Enter choice: 1
Name: aditya
Age: 20
Phone No: 784512457
Designation: chef
Address: katraj, pune
Enter choice: 2

****Employee List****
Name | Age | Phone No | Designation | Address
aditya | 20 | 784512457 | chef | katraj, pune
Enter choice: 3
Enter name of employee to be updated: aditya
Choose data to be updated:
1. Name
2. Age
3. Phone number
4. Designation
5. Address
6. Back
Enter choice: 4
Enter updated data: head chef
Enter choice: 6
Enter choice: 2

****Employee List****
Name | Age | Phone No | Designation | Address
aditya | 20 | 784512457 | head chef | katraj, pune
Enter choice:
```

Invoice Generator:

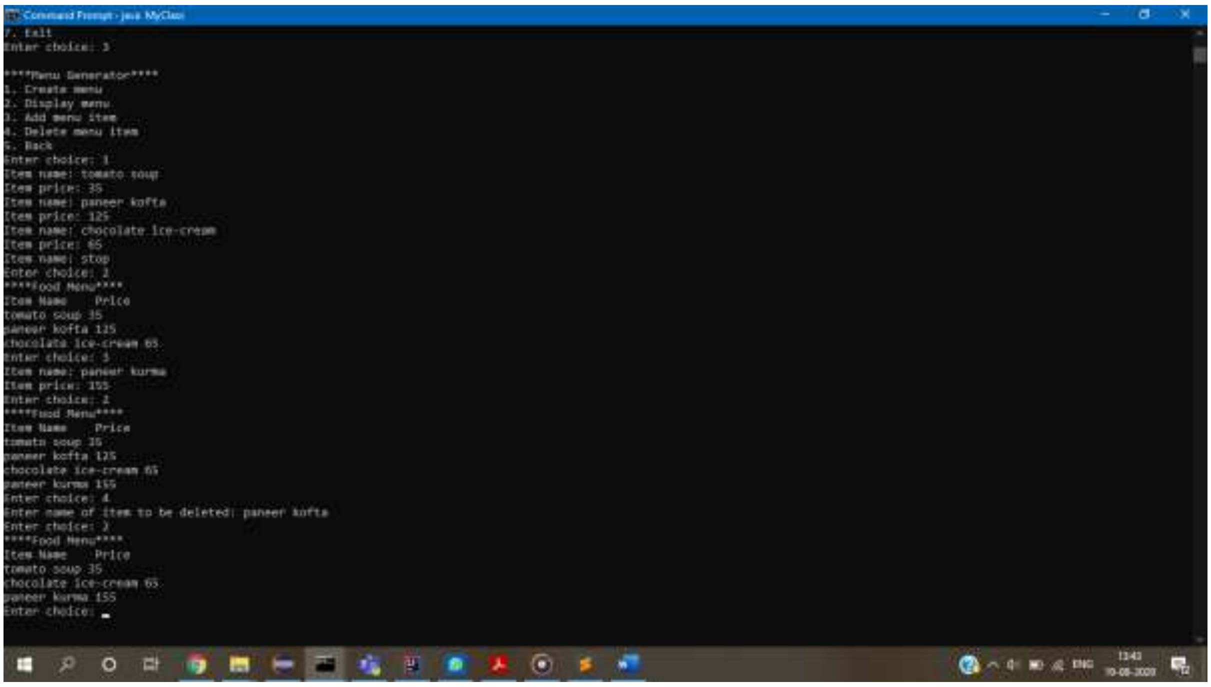


```
Enter choice: 5

****Restaurant Management****
1. Inventory Management
2. Employee Management
3. Menu Generator
4. Invoice Generator
5. Customer Management
6. Kitchen Handler
7. Exit
Enter choice: 4

****Invoice Generator****
1. New Invoice
2. Display Invoice
3. Delete Invoice
4. Back
Enter choice: 1
Enter customer name: ram
Enter table number: 2
Item name: paneer kofta
Item quantity: 2
Item name: chocolate ice-cream
Item quantity: 2
Item name: stop

****Invoice Generator****
1. New Invoice
2. Display Invoice
3. Delete Invoice
4. Back
Enter choice: 2
Name: ram
Table No: 2
paneer kofta 2
chocolate ice-cream 2

****Invoice Generator****
1. New Invoice
2. Display Invoice
3. Delete Invoice
4. Back
Enter choice: 3
Enter table no:
```



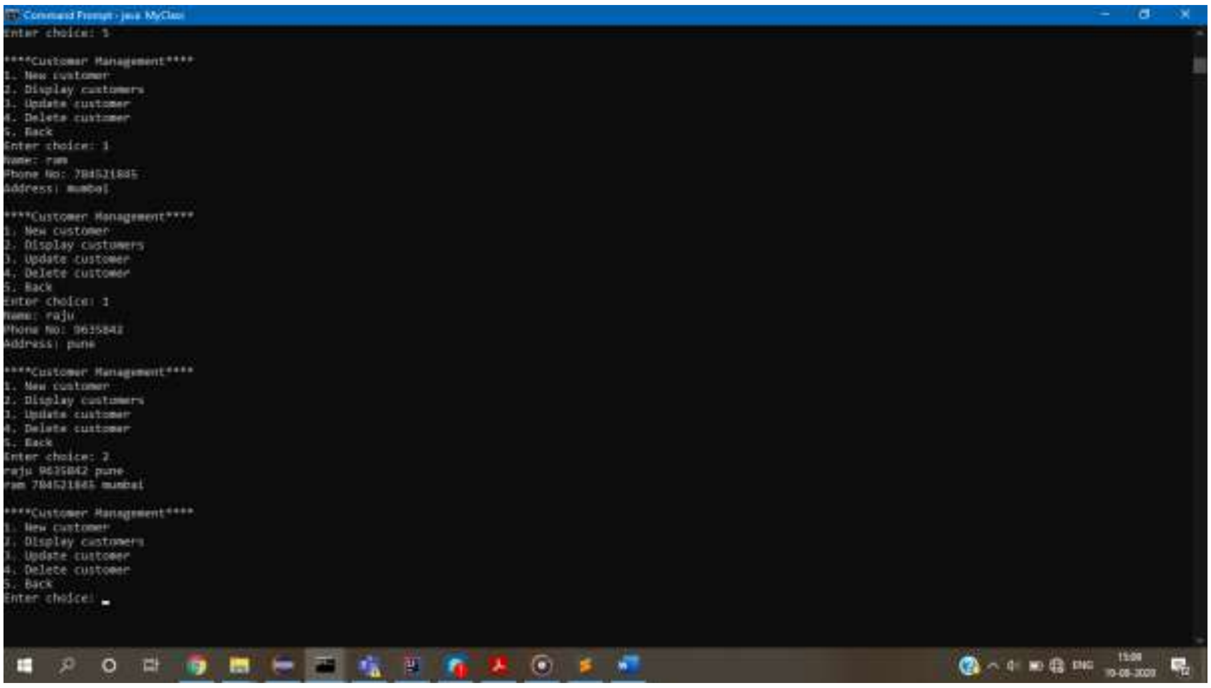
```
Item name: stop

****Invoice Generator****
1. New Invoice
2. Display Invoice
3. Delete Invoice
4. Back
Enter choice: 2
Name: ram
Table No: 2
paneer kofta 2
chocolate ice-cream 2

****Invoice Generator****
1. New Invoice
2. Display Invoice
3. Delete Invoice
4. Back
Enter choice: 3
Enter table no:
2

****Invoice Generator****
1. New Invoice
2. Display Invoice
3. Delete Invoice
4. Back
Enter choice: 2
No records present.

****Invoice Generator****
1. New Invoice
2. Display Invoice
3. Delete Invoice
4. Back
Enter choice:
```

Menu Generator:



```
7. Exit
Enter choice: 3

****Menu Generator****
1. Create menu
2. Display menu
3. Add menu item
4. Delete menu item
5. Back
Enter choice: 1
Item name: tomato soup
Item price: 35
Item name: paneer kofta
Item price: 125
Item name: chocolate ice-cream
Item price: 65
Item name: stop
Enter choice: 2
****Food Menu****
Item Name    Price
tomato soup 35
paneer kofta 125
chocolate ice-cream 65
Enter choice: 3
Item name: paneer kurma
Item price: 155
Enter choice: 2
****Food Menu****
Item Name    Price
tomato soup 35
paneer kofta 125
chocolate ice-cream 65
paneer kurma 155
Enter choice: 4
Enter name of item to be deleted: paneer kofta
Enter choice: 2
****Food Menu****
Item Name    Price
tomato soup 35
chocolate ice-cream 65
paneer kurma 155
Enter choice: _
```

Customer Management:



```
Enter choice: 5

****Customer Management****
1. New customer
2. Display customers
3. Update customer
4. Delete customer
5. Back
Enter choice: 1
Name: ram
Phone No: 784521885
Address: mumbai

****Customer Management****
1. New customer
2. Display customers
3. Update customer
4. Delete customer
5. Back
Enter choice: 1
Name: raju
Phone No: 9635842
Address: pune

****Customer Management****
1. New customer
2. Display customers
3. Update customer
4. Delete customer
5. Back
Enter choice: 2
raju 9635842 pune
ram 784521885 mumbai

****Customer Management****
1. New customer
2. Display customers
3. Update customer
4. Delete customer
5. Back
Enter choice: _
```