OOPM Mini Project Report

TOPIC: BILL MANAGEMENT SYSTEM
SUBJECT: OBJECT ORIENTED PROGRAMMING METHODOLOGY
CLASS: SE-IT

MADE BY:

NAME:ATASHI KHATUA
BATCH: B
ROLL NO: 27
YEAR: 2016-2017

NAME:MADHULIKA TADAS
BATCH: C
ROLL NO: 56
YEAR: 2016-2017

INDEX

Sr no.	Topic	Page No.
1.	Introduction to JAVA	3
2.	OO Concepts Used	10
3.	About Bill Management System	10
3	Assumptions and protocols	11
4.	Project code	11
5.	Sequence Diagram	21
6.	Class Diagram	22
7.	Project Execution	23
8.	References	25

Introduction

What is JAVA?

Java is an entire programming language resembling C or C++. It takes a sophisticated programmer to create Java code. And it requires a sophisticated programmer to maintain it. With Java, you can create complete applications. Or you can attach a small group of instructions, a Java "applet" that improves your basic HTML. A Java Applet can also cause text to change color when you roll over it. A game, a calendar, a scrolling text banner can all be created with Java Applets. There are sometimes compatibility problems between Java and various browsers, operating systems or computers, and if not written correctly, it can be slow to load. Java is a powerful programming language with excellent security, but you need to be aware of the tradeoffs.

What is JSP?

Short for Java Server Page. A server-side technology, Java Server Pages are an extension to the Java servlet technology that was developed by Sun. JSPs have dynamic scripting capability that works in tandem with HTML code, separating the page logic from the static elements -- the actual design and display of the page -- to help make the HTML more functional(i.e. dynamic database queries). A JSP is translated into Java servlet before being run, and it processes HTTP requests and generates responses like any servlet. However, JSP technology provides a more convenient way to code a servlet. Translation occurs the first time the application is run. A JSP translator is triggered by the .jsp file name extension in a URL. JSPs are fully interoperable with servlets. You can include output from a servlet or forward the output to a servlet, and a servlet can include output from a JSP or forward output to a JSP. JSPs are not restricted to any specific platform or server. It was orignially created as an alternative to Microsoft's ASPs (Active Server Pages). Recently, however, Microsoft has countered JSP technology with its own ASP.NET, part of the .NET initiative.

What is JavaScript?

When new technologies start, they sometimes acquire names that will be confusing in the future. That's the case with JavaScript. JavaScript is not 'Java'. JavaScript is a simple programming language that writes commands to your browser when the HTML page is loaded. Note: you can have compatibility issues with Java Script, especially in newer versions of Browsers.

What is Java?

Java is a simple, distributed object oriented programming language which provides the security, High performance, robustness.
Java is a portable and Architectural neutral language which can be Interpreted.
Java is multithreaded and Dynamic language.

About Java

Java is a Programming language originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java Platform. The language derives much of its Syntax from c and C++ but has a simpler object Model and fewer low-level facilities. Java applications are typically compiled to bytecode(class file) that can run on any Java Virtual machine (JVM) regardless of computer architecture.

Why Software Developers Choose Java?

Java with its versatilty, efficiency, and portability, Java has become invaluable to developers by enabling them to:

• Write software on one platform and run it on virtually any other platform
• Create programs to run within a Web browser and Web services
• Develop server-side applications for online forums, stores, polls, HTML forms processing, and more
• Combine applications or services using the Java language to create highly customized applications or services
• Write powerful and efficient applications for mobile phones, remote processors, low-cost consumer products, and practically any other device with a digital heartbeat.

Goals in creation of Java

There were five primary goals in the creation of the Java language

1. It should be "simple, object oriented".
2. It should be "robust and secure".
3. It should be "architecture neutral and portable".
4. It should execute with "high performance".
5. It should be "interpreted, threaded, and dynamic".

Architecture of Java

Java's architecture arises out of four distinct but interrelated technologies:

- The Java programming language
- The Java class file format
- The Java Application Programming Interface
- The Java virtual machine

When you write and run a Java program, you are tapping the power of these four technologies. You express the program in source files written in the Java programming language, compile the source to Java class files, and run the class files on a Java virtual machine. When you write your program, you access system resources (such as I/O, for example) by calling methods in the classes that implement the Java Application Programming Interface, or Java API. As your program runs, it fulfills your program's Java API calls by invoking methods in class files that implement the Java API.

The Java Virtual Machine

At the heart of Java's network-orientation is the Java virtual machine, which supports all three prongs of Java's network-oriented architecture: platform independence, security, and network-mobility.

A Java virtual machine's main job is to load class files and execute the bytecodes they contain. As you can see in Figure 1-3, the Java virtual machine contains a class loader, which loads class files from both the program and the Java API. Only those class files from the Java API that are actually needed by a running program are loaded into the virtual machine. The bytecodes are executed in an execution engine.

Coding standard

Java suggests set of coding standard to follow while writing java program. Coding standard helps author as well as others to better understand program. It reduce amount of debugging time considerably. Basically coding standard suggests how to name class, methods variables of different scope, package etc.

Writing a Java program

In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler. A .class file does not contain code that is native to your processor; it instead contains bytecodes — the machine language of the Java Virtual

Machine1 (Java VM). The java launcher tool then runs your application with an instance of the Java Virtual Machine.

Java OOPs Concepts

Simula is considered as the first object-oriented programming language. The programming paradigm where everything is represented as an object, is known as truly object-oriented programming language.

Smalltalk is considered as the first truly object-oriented programming language.

OOPs (Object Oriented Programming System)

Object means a real word entity such as pen, chair, table etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation
- Interfaces
- Pacakages

Object − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.

Class − A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.

A class can contain any of the following variable types.

- **Local variables** − Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
- **Instance variables** − Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
- **Class variables** − Class variables are variables declared within a class, outside any method, with the static keyword.

Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the use of inheritance the information is made manageable in a hierarchical order.

The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class).

extends Keyword

extends is the keyword used to inherit the properties of a class. Following is the syntax of extends keyword.

Syntax

```
class Super {
   .....
   .....
}
class Sub extends Super {
   .....
   .....
}
```

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

Any Java object that can pass more than one IS-A test is considered to be polymorphic. In Java, all Java objects are polymorphic since any object will pass the IS-A test for their own type and for the class Object.

It is important to know that the only possible way to access an object is through a reference variable. A reference variable can be of only one type. Once declared, the type of a reference variable cannot be changed.

The reference variable can be reassigned to other objects provided that it is not declared final. The type of the reference variable would determine the methods that it can invoke on the object.

A reference variable can refer to any object of its declared type or any subtype of its declared type. A reference variable can be declared as a class or interface type.

Example

Let us look at an example.

```
public interface Vegetarian{}
public class Animal{}
public class Deer extends Animal implements Vegetarian{}
```

Now, the Deer class is considered to be polymorphic since this has multiple inheritance. Following are true for the above examples −

- A Deer IS-A Animal
- A Deer IS-A Vegetarian
- A Deer IS-A Deer

- A Deer IS-A Object

When we apply the reference variable facts to a Deer object reference, the following declarations are legal −

Example

```
Deer d = new Deer();
Animal a = d;
Vegetarian v = d;
Object o = d;
```

All the reference variables d, a, v, o refer to the same Deer object in the heap.

Abstraction is the quality of dealing with ideas rather than events. For example, when you consider the case of e-mail, complex details such as what happens as soon as you send an e-mail, the protocol your e-mail server uses are hidden from the user. Therefore, to send an e-mail you just need to type the content, mention the address of the receiver, and click send.

Likewise in Object-oriented programming, abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.

In Java, abstraction is achieved using Abstract classes and interfaces.

Abstract Class

A class which contains the **abstract** keyword in its declaration is known as abstract class.

- Abstract classes may or may not contain *abstract methods*, i.e., methods without body (public void get();)
- But, if a class has at least one abstract method, then the class **must** be declared abstract.
- If a class is declared abstract, it cannot be instantiated.
- To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.
- If you inherit an abstract class, you have to provide implementations to all the abstract methods in it.

Encapsulation is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction.

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as **data hiding**.

To achieve encapsulation in Java −

- Declare the variables of a class as private.

- Provide public setter and getter methods to modify and view the variables values.

An interface is a reference type in Java. It is similar to class. It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

Along with abstract methods, an interface may also contain constants, default methods, static methods, and nested types. Method bodies exist only for default methods and static methods.

Writing an interface is similar to writing a class. But a class describes the attributes and behaviors of an object. And an interface contains behaviors that a class implements.

Unless the class that implements the interface is abstract, all the methods of the interface need to be defined in the class.

An interface is similar to a class in the following ways −

- An interface can contain any number of methods.
- An interface is written in a file with a **.java** extension, with the name of the interface matching the name of the file.
- The byte code of an interface appears in a **.class** file.
- Interfaces appear in packages, and their corresponding bytecode file must be in a directory structure that matches the package name.

However, an interface is different from a class in several ways, including −

- You cannot instantiate an interface.
- An interface does not contain any constructors.
- All of the methods in an interface are abstract.
- An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
- An interface is not extended by a class; it is implemented by a class.
- An interface can extend multiple interfaces.

Packages are used in Java in order to prevent naming conflicts, to control access, to make searching/locating and usage of classes, interfaces, enumerations and annotations easier, etc.

A **Package** can be defined as a grouping of related types (classes, interfaces, enumerations and annotations) providing access protection and namespace management.

Some of the existing packages in Java are −

- **java.lang** − bundles the fundamental classes
- **java.io** − classes for input , output functions are bundled in this package

Programmers can define their own packages to bundle group of classes/interfaces, etc. It is a good practice to group related classes implemented by you so that a programmer

can easily determine that the classes, interfaces, enumerations, and annotations are related.
Since the package creates a new namespace there won't be any name conflicts with names in other packages. Using packages, it is easier to provide access control and it is also easier to locate the related classes.

Creating a Package

While creating a package, you should choose a name for the package and include a **package** statement along with that name at the top of every source file that contains the classes, interfaces, enumerations, and annotation types that you want to include in the package.
The package statement should be the first line in the source file. There can be only one package statement in each source file, and it applies to all types in the file.
If a package statement is not used then the class, interfaces, enumerations, and annotation types will be placed in the current default package.
To compile the Java programs with package statements, you have to use -d option as shown below.
javac -d Destination_folder file_name.java
Then a folder with the given package name is created in the specified destination, and the compiled class files will be placed in that folder.

OO concepts used:

The object oriented concept used in this project are

- Inheritance
- Abstraction
- Polymorphism

About BILL Management System:

- Nowadays all restaurants and lunch homes provides the customer with a bill which is generated by an automated software.
- Using Java such an interactive system can be developed efficiently to generate a bill within few seconds.
- The project of Bill Management System developed by us is a console based interactive system with some false datajust to examine the results.
- Here several factors like In-Hotel service, Home delivery service, discounts, tax, etc have been considered to satisfy a real life application.

Assumptions and protocols:

There are various assumptions made in Bill management system project.Some of them are as follows:

1. A strict menu is fixed and the item prices are predecided.
2. There are two services available, In-Hotel service and Home Delivery Service.
3. The delivery charges varies with the distance and is thus dependent on the user of the bill system.
4. Discount offers are available for festive seasons and the percentage of discount can be given by the user.
5. A tax of 2% is been applied on all the purchases of food items.
6. The bill contains the following details:
    - Header of the restaurant
    - Name of the employee generating the bill
    - Date of purchase automatically generated by the system
    - Name of the customer
    - Address of the customer
    - Contact number of the customer
    - Name of food items ordered
    - The prices of individual item
    - Discount offer(if any)
    - Deliverycharges(if any)
    - Tax amount
    - Total Bill
    - Payment details
    - Approval details
    - Closing comments

Project Code:

(BILL Management System)

```
//Package Billing
import java.io.DataInputStream;
import java.io.IOException;
import java.util.Calendar;
import java.util.GregorianCalendar;
```

```
class display
{
	static void str(String msg)
	{
		System.out.println(msg);
	}
}

class Customer
{
static String name;
static String address="--";
static double contact;
	static void getdata(String N,double C)
	{
		name=N;
		contact=C;
	}
	static void getdata(String N,double C,String A)
	{
		name=N;
		contact=C;
		address=A;
	}
	static void putdata()
	{
		display.str("Name:"+name);
		display.str("Address:"+address);
		System.out.printf("Contact:%.0f\n\n",contact);
	}
}

abstract class BillSystem
{
	static int order,quantity,foodn,i,j,input,rs,sum,discount;
	static float discount_amt,tax;
	static String food,name;
	static double phoneno;
	static String fooda[]=new String[13];
```

```
static int r[] = new int[13];
static int quan[]=new int[13];
GregorianCalendar date = new GregorianCalendar();
int day=date.get(Calendar.DAY_OF_MONTH);
int month=date.get(Calendar.MONTH);
int year=date.get(Calendar.YEAR);
static DataInputStream inp = new DataInputStream(System.in);
static int quan(int q,int r)
{
        r=r*q;
        return r;
}
abstract void createBill() throws Exception, IOException;
abstract void showBill() throws Exception;
static void DisplayMenu()
{
        display.str("\t\t\t\t\tMENU\n");
        display.str("\t\t\tItem\t\t\t\tRs.");
        display.str("\t\t1.Today's Special\t\t\t180");
        display.str("\t\t2.Chole Bhature\t\t\t\t90");
        display.str("\t\t3.Paneer Masala Special\t\t\t100");
        display.str("\t\t4.Paneer Tikka\t\t\t\t100");
        display.str("\t\t5.Chicken Soup\t\t\t\t150");
        display.str("\t\t6.Chicken Biryani\t\t\t250");
        display.str("\t\t7.Triple Noodles\t\t\t200");
        display.str("\t\t8.Chicken Lollypop\t\t\t100");
        display.str("\t\t9.Chicken Hakka Noodles\t\t\t80");
        display.str("\t\t10.Chicken Rice\t\t\t\t80");
        display.str("\t\t11.Dry Schezwan Chicken\t\t\t150");
        display.str("\t\t12.Chapati\t\t\t\t10");
        display.str("\t\t13.ColdDrinks\t\t\t\t15");
        display.str("\t\t14.Ice Cream\t\t\t\t15\n");
}
@SuppressWarnings("deprecation")
static void ItemList() throws Exception, IOException
{
        j=0;
        for(i=0;i<order;i++)
        {
```

```
display.str("Enter food Name (Number): ");
foodn=Integer.parseInt(inp.readLine());
display.str("Enter quantity: ");
quantity=Integer.parseInt(inp.readLine());
switch(foodn)
{
	case 1:
		food="Today's Special\t\t\t";
		rs=quan(quantity,180);
		break;
	case 2:
		food="Chole Bhature Special\t\t";
		rs=quan(quantity,90);
		break;
	case 3:
		food="Paneer Masala Special\t\t";
		rs=quan(quantity,100);
		break;
	case 4:
		food="Paneer Tikka\t\t\t";
		rs=quan(quantity,100);
		break;
	case 5:
		food="Chicken Soup\t\t\t\t";
		rs=quan(quantity,150);
		break;
	case 6:
		food="Chicken Biryani\t\t\t";
		rs=quan(quantity,250);
		break;
	case 7:
		food="Triple Noodles\t\t\t";
		rs=quan(quantity,200);
		break;
	case 8:
		food="Chicken Lollypop\t\t";
		rs=quan(quantity,100);
		break;
	case 9:
```

```
                        food="Chicken Hakka Noodles\t\t";
                        rs=quan(quantity,80);
                        break;
                case 10:
                        food="Chicken Rice\t\t\t";
                        rs=quan(quantity,80);
                        break;
                case 11:
                        food="Dry Shezwan Chicken\t\t";
                        rs=quan(quantity,150);
                        break;
                case 12:
                        food="Chapati\t\t\t\t";
                        rs=quan(quantity,10);
                        break;
                case 13:
                        food="Cold Drinks\t\t\t";
                        rs=quan(quantity,15);
                        break;
                case 14:
                        food="Ice Cream\t\t\t";
                        rs=quan(quantity,15);
                        break;
                default:
                        display.str("Invalid Food....");
            }
            fooda[j]=food;
            r[j]=rs;
            quan[j]=quantity;
            j++;
        }
    }
}


class InHotel extends BillSystem
{
    int tabno;
    DataInputStream in = new DataInputStream(System.in);
```

```
	@SuppressWarnings("deprecation")
	void createBill() throws Exception, IOException
	{
		try
		{
			display.str("Enter Table no.");
			tabno=Integer.parseInt(in.readLine());
			display.str("Enter Customer name");
			name=in.readLine();
			display.str("Enter Customer contact details");
			phoneno=Double.parseDouble(in.readLine());
			display.str("Enter the discount percent");
			Customer.getdata(name,phoneno);
			discount=Integer.parseInt(in.readLine());
			display.str("Enter How many items ordered: ");
			order=Integer.parseInt(in.readLine());
			DisplayMenu();
			ItemList();
		}
		catch(Exception e)
		{
		}
	}
	void showBill() throws Exception
	{
		display.str("/*-----------------------------------------------------------------------
----*/");
		display.str("\t\t\t\tHotel Sunshine");
		display.str("\t\t\tKoral Park, Dist. Bhiwandi 411222");
		System.out.println("Bill by Mr Ankur Singh\t\t\t\t\t\t" + day+"-
"+(month+1)+"-"+year+"\n");
		System.out.println("Table No.: " + tabno+"\n");
		Customer.putdata();
		System.out.println("\tOrder\t\t\tQuantity\t\tRupees\n");
		for(i=0;i<order;i++)
		{
			System.out.print(fooda[i]);
			System.out.print("\t"+quan[i]);
			System.out.print("\t\t"+r[i]+"\n");
```

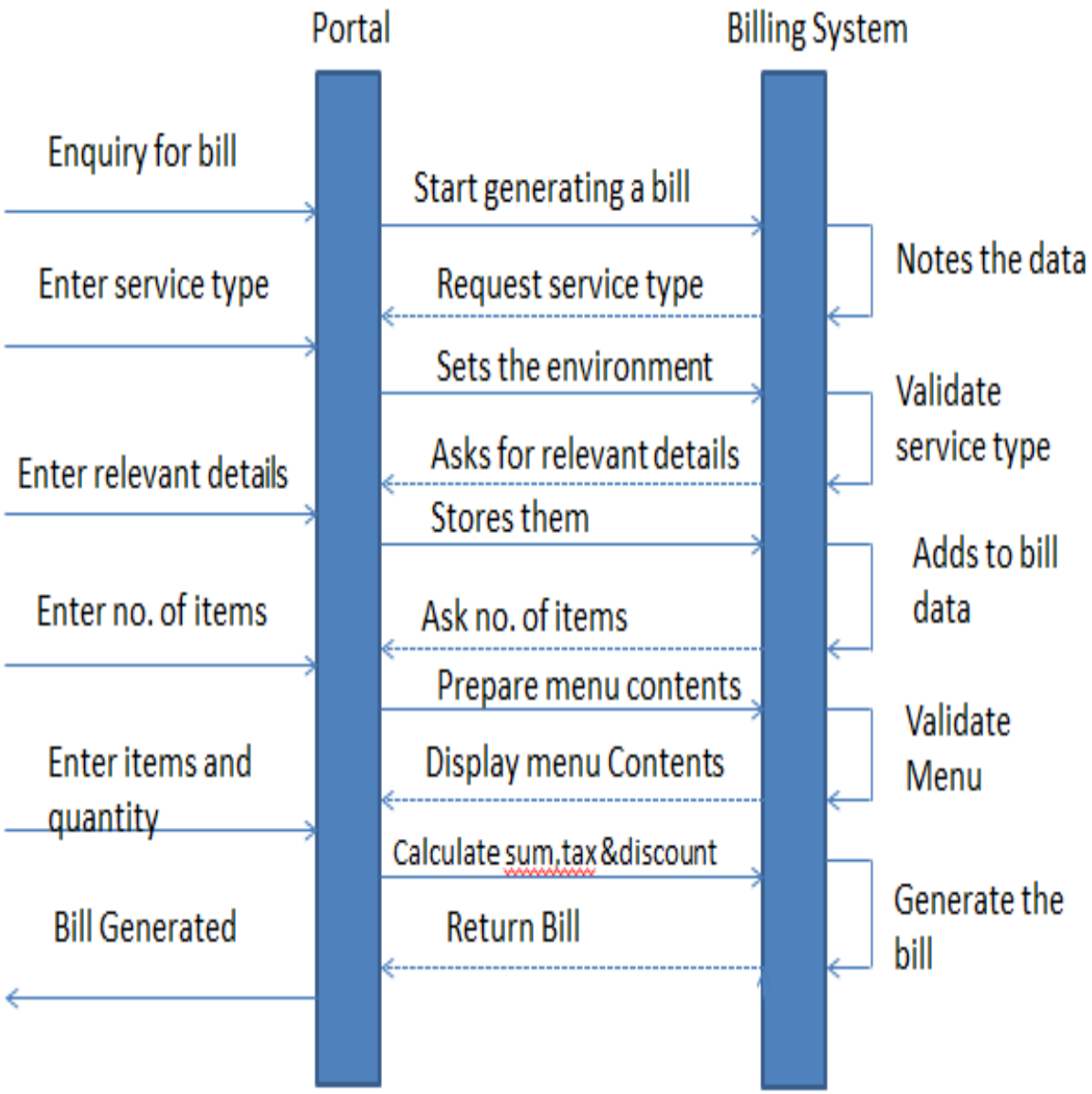
```
            }
            sum=0;
            for(i=0;i<order;i++)
            {
                  sum=sum+r[i];
            }
            tax=(float)2*sum/100;
            display.str("Tax\t\t\t\t\t\t\t"+(int)tax);
            sum=sum+(int)tax;
            discount_amt=(float)discount*sum/100;
            display.str("\t\t\t\t\t\t\tTotal: "+sum);
            sum=sum-(int)discount_amt;
            if(discount!=0)
                  display.str("\n\t\tTODAY'S DISCOUNT OFFER="+discount+"%
DISCOUNT");
            display.str("\n\t\t\t\t\t\t\t*Bill: "+sum+"*");
            display.str("Paid = Yes");
            display.str("\t\t\tApproved");
            display.str("\n\t\t~~~Thankyou for coming!!!Please visit us again~~~");
            display.str("\n--------------------------------------------------------------------
--------");
      }
}


class HomeDelivery extends BillSystem
{
      int dcharges;
      String location;
      DataInputStream in = new DataInputStream(System.in);
      @SuppressWarnings("deprecation")
      void createBill() throws Exception, IOException
      {
            try
            {
                  display.str("Enter Name of Receiver:");
                  name=in.readLine();
                  display.str("Enter Customer contact details");
```

```
                phoneno=Double.parseDouble(in.readLine());
                display.str("Enter address of customer");
                location=in.readLine();
                Customer.getdata(name,phoneno,location);
                display.str("Enter the discount percent");
                discount=Integer.parseInt(in.readLine());
                display.str("Enter the charges for Home Delivery:");
                dcharges=Integer.parseInt(in.readLine());
                display.str("Enter How many items ordered: ");
                order=Integer.parseInt(in.readLine());
                DisplayMenu();
                ItemList();
            }
            catch(Exception e)
            {
            }
        }
        void showBill() throws Exception
        {
            display.str("/*-------------------------------------------------------------------
----*/");
            display.str("\t\t\t\tHotel Sunshine");
            display.str("\t\t\tKoral Park, Dist. Bhiwandi 411222");
            System.out.println("Bill by Mr Ankur Singh\t\t\t\t\t\t" + day+"-
"+(month+1)+"-"+year+"\n");
            Customer.putdata();
            System.out.println("\tOrder\t\t\tQuantity\t\tRupees\n");
            for(i=0;i<order;i++)
            {
                System.out.print(fooda[i]);
                System.out.print("\t"+quan[i]);
                System.out.print("\t\t\t"+r[i]+"\n");

            }
            sum=0;
            for(i=0;i<order;i++)
            {
                sum=sum+r[i];
            }
```

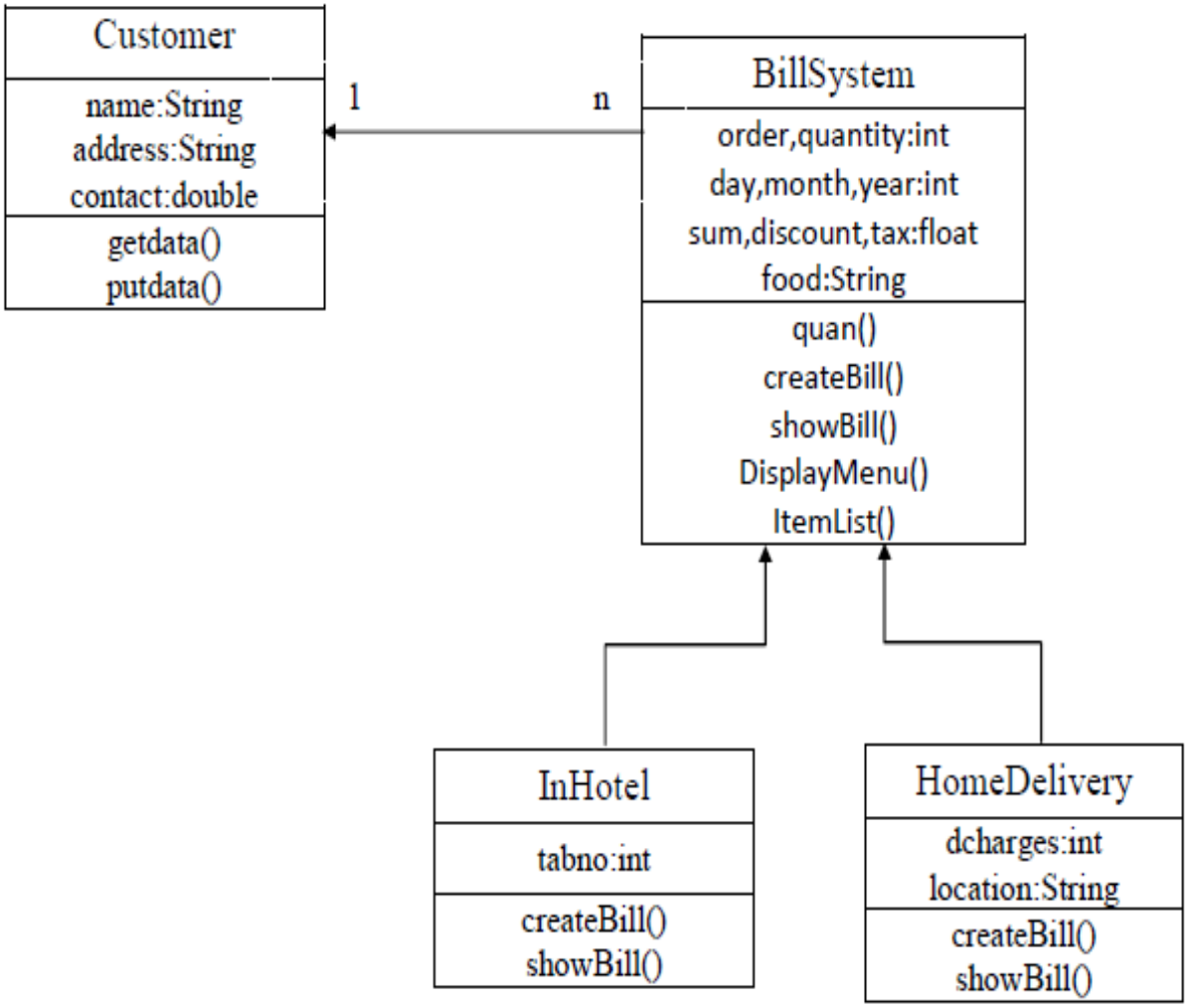
```
                display.str("Delivery Charges\t\t\t\t\t"+dcharges);
                tax=(float)2*sum/100;
                display.str("Tax\t\t\t\t\t\t\t"+(int)tax);
                sum=sum+dcharges;
                sum=sum+(int)tax;
                display.str("\t\t\t\t\t\t\tTotal: "+sum);
                discount_amt=(float)discount*sum/100;
                sum=sum-(int)discount_amt;
                if(discount!=0)
                        display.str("\n\t\tTODAY'S DISCOUNT OFFER="+discount+"%
DISCOUNT");
                display.str("\n\t\t\t\t\t\t\t*Bill: "+sum+"*");
                display.str("\n\t\t\t\t\tReceiver's Sign:_____\n");
                display.str("\t\t\t\t Delivery person's Sign:_____");
                display.str("Paid = Yes");
                display.str("\n\t\t  ~~~Thankyou for ordering online!!~~~");
                display.str("\n--------------------------------------------------------------------
--------");
        }
}


public class MainBillSystem3
{
        @SuppressWarnings("deprecation")
        public static void main(String args[])
        {
                int value,choice;
                DataInputStream in = new DataInputStream(System.in);
                try
                {
                        do{
                                display.str("\n\n");
                                display.str("Enter your choice:\n\t1.Bill\n\t2.Exit\n");
                                value=Integer.parseInt(in.readLine());
                                switch(value)
                                {
                                        case 1:
                                                display.str("\n\n");
```

```
                                        display.str("Enter your choice:\n\t1.In
Hotel Service\n\t2.Home Delivery\n");
                                        choice=Integer.parseInt(in.readLine());
                                        switch(choice)
                                        {
                                        case 1:
                                                InHotel b1 = new InHotel();
                                                b1.createBill();
                                                b1.showBill();
                                                break;
                                        case 2:
                                                HomeDelivery b2 = new
HomeDelivery();
                                                b2.createBill();
                                                b2.showBill();
                                                break;
                                        default:
                                                display.str("Invalid type of Bill");
                                        }
                                        break;
                                case 2:
                                        System.exit(1);
                                default:
                                        display.str("Invalid Choice");
                                }
                        }while(value!=2);
                }
                catch(Exception e)
                {
                        //Do nothing
                }
        }
}
```

Sequence Diagram:



Class Diagram:

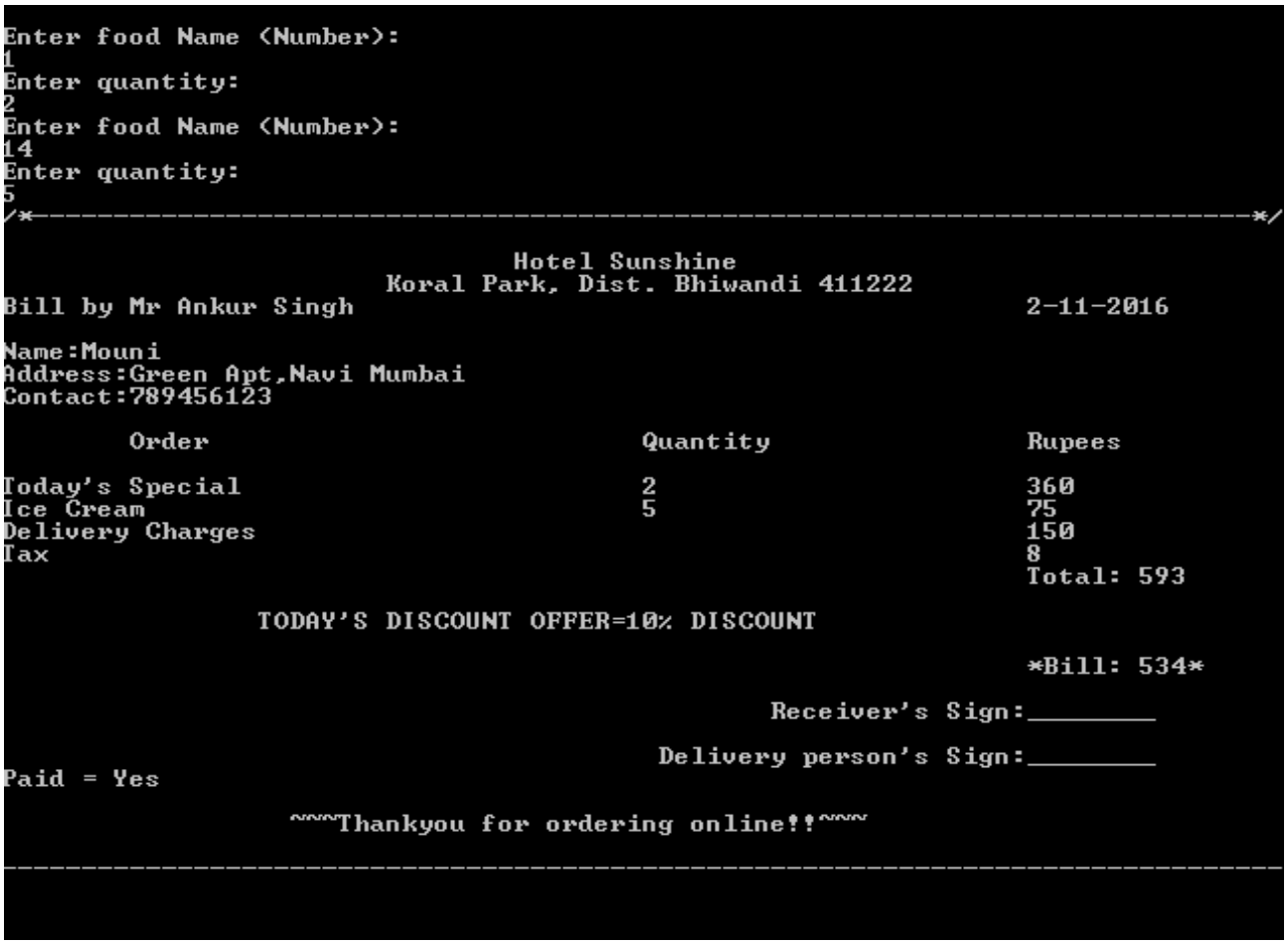


Project Execution:

```
C:\Users\Lenovo\Desktop\Final>java MainBillSystem3



Enter your choice:
        1.Bill
        2.Exit

1



Enter your choice:
        1.In Hotel Service
        2.Home Delivery

1
Enter Table no.
2
Enter Customer name
Arjun
Enter Customer contact details
9876543210
Enter the discount percent
20
Enter How many items ordered:
2
                                        MENU

                Item                                    Rs.
             1.Today's Special                          180
             2.Chole Bhature                            90
             3.Paneer Masala Special                    100
             4.Paneer Tikka                             100
             5.Chicken Soup                             150
             6.Chicken Biryani                          250
             7.Triple Noodles                           200
             8.Chicken Lollypop                         100
             9.Chicken Hakka Noodles                    80
             10.Chicken Rice                            80
             11.Dry Schezwan Chicken                    150
             12.Chapati                                 10
             13.ColdDrinks                              15
             14.Ice Cream                               15
```

```
Enter food Name (Number):
1
Enter quantity:
2
Enter food Name (Number):
13
Enter quantity:
2
/*--------------------------------------------------------------------------*/

                            Hotel Sunshine
                        Koral Park, Dist. Bhiwandi 411222
Bill by Mr Ankur Singh                                      2-11-2016

Table No.: 2

Name:Arjun
Address:--
Contact:9876543210

        Order                           Quantity                Rupees

Today's Special                         2                       360
Cold Drinks                             2                       30
Tax                                                             7
                                                                Total: 397

                TODAY'S DISCOUNT OFFER=20% DISCOUNT

                                                                *Bill: 318*
Paid = Yes
                                Approved

                ~~~~Thankyou for coming!!!Please visit us again~~~~
--------------------------------------------------------------------------
```

```
Enter your choice:
        1.Bill
        2.Exit

1



Enter your choice:
        1.In Hotel Service
        2.Home Delivery

2
Enter Name of Receiver:
Mouni
Enter Customer contact details
0789456123
Enter address of customer
Green Apt,Navi Mumbai
Enter the discount percent
10
Enter the charges for Home Delivery:
150
Enter How many items ordered:
2
                                        MENU

                        Item                            Rs.
                1.Today's Special                       180
                2.Chole Bhature                         90
                3.Paneer Masala Special                 100
                4.Paneer Tikka                          100
                5.Chicken Soup                          150
                6.Chicken Biryani                       250
                7.Triple Noodles                        200
                8.Chicken Lollypop                      100
                9.Chicken Hakka Noodles                 80
                10.Chicken Rice                         80
                11.Dry Schezwan Chicken                 150
                12.Chapati                              10
                13.ColdDrinks                           15
                14.Ice Cream                            15
```



References:

www.tutorialspoint.com
http://www.javatpoint.com/payment-billing-product-project
http://www.muengineers.in/computer-project-list/java-projects-list/billing-management-system