CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

The people who are suffering from small diseases like fever, cold, cough etc will first approaches to pharmacy where the basic needs are provided. In this place if we develop a code which stores the data of customer and also data of the medicine storage and also calculate the total cost of the medicine issued.

Medical pharmacy is the first preferences where the people will approach for small problems.

It is mainly useful for kids during cold and cough etc.

So if we make a program that calculate the total amount of all products that are taken And also to know the stock available for issuing medicine and also to add the medicine which is not available in the store.

This program can also store the data of bills of the customer.

Because of this program we store all these things instead of remaining all the things or writing in the book

It also allocates the discount on the total bill of medicine which will be attracted to the Customer and this will also increase the business of the store.

1.2 OBJECTIVES

The main objectives are

- To issue the medicine requested by the customer.
- To add new medicine in to the medicine list.
- To decrese the burden of having trouble in the calculations.
- To increase the pharmacy business.
- To make things more easier.

- To print the bill of all medicines taken by the customer.
- Also to allocate the discount on the total bill.

1.3 METHODOLOGY TO BE FOLOWED

The various OOPS concepts used in this project are

- Various classes used for different purpose.
- Objects to call the classes
- Overloading concept
- Overriding to update the stock
- Inheritance to extend the classes
- Abstract classes to define the method in that class later.
- Exceptions to open the file and to read the file and to ride the file.
- I/O functions to store the data to read the data
- Different packages

  We are importing different packages to import different classes which are needed

1.4 EXPECTED OUTCOMES

- To get username and phone number which required during the billing process.
- To search weather the requested medicine is present or not.
- If present issue the medicine
- Else to update the stock of medicine which is not present.
- To calculate total cost of the medicine.
- To issues discount based on the cost.
- And also to store the bills of the customers in to the billsfile.

1.5 HARDWARE AND SOFTWARE REQUIRMENTS

Software requirements

- Core java
- Installed JDK
- Eclipse IDE

Hardware requirements

- Minimum windows 95 software.
- Hard drive and Minimum of 8MB memory.
- A CD-ROM drive.

CHAPTER 2

OBJECT ORIENTED CONCEPTS

2.1 CLASS

The class is at the core of Java. It is the logical construct upon which the entire Java language is built because it defines the shape and nature of an object. As such, the class forms the basis for object-oriented programming in Java. Any concept you wish to implement in a Java program must be encapsulated within a class. Because the class is so fundamental to Java, this and the next few chapters will be devoted to it. Here, you will be introduced to the basic elements of a class and learn how a class can be used to create objects. You will also learn about methods, constructors, and the this keyword. Class Fundamentals Classes have been used since the beginning of this book. However, until now, only the most rudimentary form of a class has been shown. The classes created in the preceding chapters primarily exist simply to encapsulate the main( ) method, which has been used to demonstrate the basics of the Java syntax. As you will see, classes are substantially more powerful than the limited ones presented so far. Perhaps the most important thing to understand about a class is that it defines a new data type. Once defined, this new type can be used to create objects of that type. Thus, a class is a template for an object, and an object is an instance of a class. Because an object is an instance of a class, you will often see the two words object and instance used interchangeably. The General Form of a Class When you define a class, you declare its exact form and nature. You do this by specifying the data that it contains and the code that operates on that data. While very simple classes may contain only code or only data, most real-world classes contain both. As you will see, a class' code defines the interface to its data. A class is declared by use of the class keyword. The classes that have been used up to this point are actually very limited examples of its complete form. Classes can (and usually do) get much more complex

In my project I have used three classes they are:

- Main class
- Availability class

- Customer class

Main class

First the main class extends customer

Here the customer details will be asked by calling the method called customers which is declared as an abstract method in the abstract class called customer

It also contain different files bills, medicinequantity, medicinecost

It also do the operation to add the medicine which we don’t have in the store.

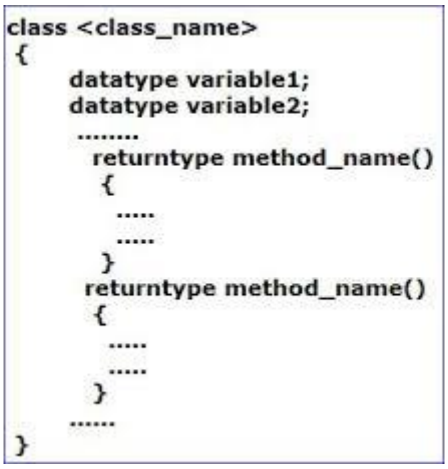
Finally it give the bill of the medicines we purchased.

Availability class

It checks the stock of the medicine and also updates the stock of the new medicine.

Customer class

It is the class that is declared as an abstract class. Abstract class is defined as the class that contains the undefined method which will be defined later



```
class <class_name>
 {
      datatype variable1;
      datatype variable2;
       ........
        returntype method_name()
         {
           .....
           .....
         }
       returntype method_name()
        {
          .....
          .....
        }
      ......
 }
```

2.2 OBJECT

As just explained, when you create a class, you are creating a new data type. You can use this type to declare objects of that type. However, obtaining objects of a class is a two-step process. First, you must declare a variable of the class type. This variable does not define an object. Instead, it is simply a variable that can refer to an object. Second, you must acquire an actual, physical copy of the object and assign it to that variable. You can do this using the new operator. The new operator dynamically allocates (that is, allocates at run time) memory for an object and returns a reference to it. This reference is, more or less, the address in memory of the object allocated by new. This reference is then stored in the variable. Thus, in Java, all class objects must be dynamically allocated. Let’s look at the details of this procedure.

Objects are used for class different method from different classes. Methods can be called with out creating an object using static keyword.

2.3 INHERITENCE

Inheritance is taken between two classes where there will be one super class and other will be sub classes where in sub classes we will give a keyword called extends. This keyword is used to inherit all the contents of the parent class to the remaining child classes.

There are many types of inheritance

1. Simple inheritance

   Example

   ```
   Class a {
   //content of the code
   }
   Class b extends a {
   //content of the code
   }
   ```

2. Multi-level inheritance Example

   ```
   Class a {
   ```

```
//content of the code
}
Class b extends a {
//content of the code
}
Class c extends b {
//content of the code
}
```

3. Multiple inheritance

   This can't be supported in java.

4. Hybrid inheritance

   This can't be supported in java

5. Hierarchical inheritance

   Example

```
Class a {
//content of the code
}
Class b extends a {
//content of the code
}
Class c extends a {
//content of the code
```

```
        }
```

Where in these five types hybrid and multiple inheritances java can't be supported.

In my project I am using multilevel inheritance.

2.4 POLYMORPHISM

Polymorphism is nothing but the combination of overloading and overriding concepts. Polymorphism simply means many ways. This concept is used to perform single operation in many ways.

There are two types of polymorphism

- Static/complietime/ earlybinding: using overloading concept we can implement complie time polymorphism.
- Dynamic/runtime/late binding: using overriding concept we can implement run time polymorphism.

In the project polymorphism is used to update the details of the stock available using overriding

2.4 ABSTRACT CLASS

First of all an abstract is a keyword that is used to create a method or a class which is just declared and defining of that method can be done later when ever needed.
We can use abstract keyword in two different ways
One is to create an abstract class and second is to create an abstract method
It has only declaration but not implementation.
If any one of the method in the class is declared as an abstract then the class must also be declared as an abstract.
We cannot call any method in the abstract class.
Example:

```
Abstract class vechile{
        Public abstract int no of wheels();
}
```

Here the class vechile is declared as an abstract because the method in that class is declared as an abstract. These can be implemented later.

2.5 MULTITHREADING

Java provides built-in support for multithreaded programming. A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread, and each thread defines a separate path of execution. Thus, multithreading is a specialized form of multitasking. You are almost certainly acquainted with multitasking because it is supported by virtually all modern operating systems. However, there are two distinct types of multitasking: process-based and thread-based. It is important to understand the difference between the two. For many readers, process-based multitasking is the more familiar form. A process is, in essence, a program that is executing. Thus, process-based multitasking is the feature that allows your computer to run two or more programs concurrently. For example, process-based multitasking enables you to run the Java compiler at the same time that you are using a text editor or visiting a web site. In process-based multitasking, a program is the smallest unit of code that can be dispatched by the scheduler. In a thread-based multitasking environment, the thread is the smallest unit of dispatchable code. This means that a single program can perform two or more tasks simultaneously. For instance, a text editor can format text at the same time that it is printing, as long as these two actions are being performed by two separate threads. Thus, process-based multitasking deals with the "big picture," and thread-based multitasking handles the details. Multitasking threads require less overhead than multitasking processes. Processes are heavyweight tasks that require their own separate address spaces. Interprocess communication is expensive and limited. Context switching from one process to another is also costly. Threads, on the other hand, are lighter weight. They share the same address space and cooperatively share the same heavyweight process. Interthread communication is inexpensive, and context switching from one thread to the next is lower in cost. While Java programs make use of process-based multitasking environments, process-based multitasking is not under Java's direct control. However, multithreaded multitasking is.

Multithreading enables you to write efficient programs that make maximum use of the processing power available in the system. One important way multithreading achieves this is by keeping idle time to a minimum. This is especially important for the interactive, networked environment in which Java operates because idle time is
common. For example, the transmission rate of data over a network is much slower than the rate at which the computer can process it. Even local file system resources are read and written at a much slower pace than they can be processed by the CPU. And, of course, user input is much slower than the computer. In a single-threaded environment, your program has to wait for each of these tasks to finish before it can proceed to the next one—even though most of the time the program is idle, waiting for input. Multithreading helps you reduce this idle time because another thread can run when one is waiting. If you have programmed for operating systems such as Windows, then you are already familiar with multithreaded programming. However, the fact that Java manages threads makes multithreading especially convenient because many of the details are handled for you.

2.6 I/O FUNCTIONS

Java programs perform I/O through streams. A stream is an abstraction that either produces or consumes information. A stream is linked to a physical device by the Java I/O system. All streams behave in the same manner, even if the actual physical devices to which they are linked differ. Thus, the same I/O classes and methods can be applied to different types of devices. This means that an input stream can abstract many different kinds of input: from a disk file, a keyboard, or a network socket. Likewise, an output stream may refer to the console, a disk file, or a network connection. Streams are a clean way to deal with input/output without having every part of your code understand the difference between a keyboard and a network, for example. Java implements streams within class hierarchies defined in the java.io package.
Java defines two types of streams: byte and character. Byte streams provide a convenient means for handling input and output of bytes. Byte streams are used, for example, when reading or writing binary data. Character streams provide a convenient

means for handling input and output of characters. They use Unicode and, therefore, can be internationalized. Also, in some cases, character streams are more efficient than byte streams. The original version of Java (Java 1.0) did not include character streams and, thus, all I/O was byte-oriented. Character streams were added by Java 1.1, and certain byte-oriented classes and methods were deprecated. Although old code that doesn't use character streams is becoming increasingly rare, it may still be encountered from time to time. As a general rule, old code should be updated to take advantage of character streams where appropriate. One other point: at the lowest level, all I/O is still byte-oriented. The characterbased streams simply provide a convenient and efficient means for handling characters. An overview of both byte-oriented streams and character-oriented streams is presented in the following sections.

The Byte Stream Classes

Byte streams are defined by using two class hierarchies. At the top are two abstract classes: InputStream and OutputStream. Each of these abstract classes has several. concrete subclasses that handle the differences among various devices, such as disk files, network connections, and even memory buffers.

The abstract classes InputStream and OutputStream define several key methods that the other stream classes implement. Two of the most important are read( ) and write( ), which, respectively, read and write bytes of data. Each has a form that is abstract and must be overridden by derived stream classes.

The Character Stream Class

The Character Stream Classes Character streams are defined by using two class hierarchies. At the top are two abstract classes: Reader and Writer. These abstract classes handle Unicode character streams. The abstract classes Reader and Writer define several key methods that the other stream classes implement. Two of the most important methods are read( ) and write( ), which read and write characters of data, respectively. Each has a form that is abstract and must be overridden by derived stream classes.

2.7 JAVA PACKAGES

To create a package is quite easy: simply include a package command as the first statement in a Java source file. Any classes declared within that file will belong to the specified package. The package statement defines a name space in which classes are stored. If you omit the package statement, the class names are put into the default package, which has no name. (This is why you haven't had to worry about packages before now.) While the default package is fine for short, sample programs, it is inadequate for real applications. Most of the time, you will define a package for your code.

This is the general form of the package statement:

package pkg;

Here, pkg is the name of the package. For example, the following statement creates a package called mypackage:

Java uses file system directories to store packages. For example, the .class files for any classes you declare to be part of mypackage must be stored in a directory called mypackage. Remember that case is significant, and the directory name must match the package name exactly. More than one file can include the same package statement. The package statement simply specifies to which package the classes defined in a file belong. It does not exclude other classes in other files from being part of that same package. Most real-world packages are spread across many files. You can create a hierarchy of packages. To do so, simply separate each package name from the one above it by use of a period. The general form of a multileveled package statement is shown here:

package pkg1[.pkg2[.pkg3]];

2.8 EXCEPTION HANDLINGS

There are five types of handlings i.e., try, catch, throw, throws, finally.

Try-catch:

Risky code with in the try block and corresponding handling code inside catch block

Within the try block if anywhere an exception occurs, then rest of the try block won't be executed even though we handled that exception

Throw:

It is used to explicitly throw an exception

Syntax:

Throw new Exception ("message");

Throws:

It is used to declare the exception, it provide information to the programmer that there may occur an exception so during call of that method, programmer must use exception handling mechanism

Finally:

Code is finally block always executed, whether or not exception has occurred Allows you to run any cleanup type statements that you want to execute.

CHAPTER 3

DESIGN

3.1 DESIGN GOALS

- The main aim of designing a project is to improve the pharmacy business
- To check the availability of medicine in the shop in all of that medicine present in the store.
- To make the bills easier and also to store the bills instead of writing in the book
- To add the medicine in to the store and check the availability.

3.2 ALGORITHM

Availability class

First declare the class availability to check availability and add the new medicine.

- Declare a method chechAvailability with the parameters taking mediciniequantity,count.
- If count is less than or equal to the medicine quantity return true
  Else return false.
- Declare a method named write with the parameres tablet and path p3
  Write the medicine name in to the path p3 by using write operation.
- Declare a method named updateStock to update the cost and quantity using random

Customer class

- Declare the class as an abstract class so that we can declare this part later in the main class.
- Declare an abstract method named customers in the abstract classs customer and leave it. We can declare it later.
- Also extend the customer class with Availability class.

Main class

- Extend the main class with customer class.

- create a method write with a parameter file.
- Use exceptions try.
- create the two file objects

  one file object is create using the path of file named medicinequantity.txt.

  another file object is created using the path of file named medicinecost.txt.
- read the content of the file using FileReader.
- Count the no of lines using LineNumberReader.
- To add new medicine go to the end of the file and add in the next line of the file
- All medicine name, medicine quantity, medicine cost will updated.
- Then close the files
- If there is any exception then catch an exception called printsStackTrace().
- End the method write
- Now since we declared the method customers as an abstract we will define it now

  Enter the customer name

  Enter the customer phone number

  End the method
- Now declare the main method

  Take a file

  Check weather the file exist or not

  If file exists

  Read the content of the file using filereader

  No of lines using linereader

  Print “WELCOME TO MEDISTORE”

  Check the bills file

  If the file is empty

  Generate the bills and cost

  Else make them fixed

  While true

  Enter the medicine

If medicine present enter how many tablets

If medicine is not present call the updatestock to add medicine

Print sorry medicine is out of stock please visit onr day later.

Now it is time for billing

It displays all the medicine we buyed

And cost of each medicine , total cost of the medicine, discount on total bill

CHAPTER 4

IMPLEMENTATION

4.1 MODULE 1 FUNCTIONALITIY

In this functionality I have implemented the Availability class where it has three methods in it

public class Availability{

this method is used to check weather the medicine is there or not If there it returns true else it return false

```
            public static boolean checkAvailability(int medicinequantity,int count)
      {
            if(count<=medicinequantity)
            {
                  return true;
            }
            return false;
      }
```

Here overloading happence because they are two methods having same name write

```
      public static void write(String tablet,Path p3) throws IOException
      {
            String s=System.lineSeparator()+tablet;
            try {
              Files.write(p3, s.getBytes(), StandardOpenOption.APPEND);
            } catch (IOException e) {
              System.err.println(e);
            }
      }
```

This method is used to update the stock i.e if medicine is not available it is used to add the medicine.

```
    public static void UpdateStock(String tablet,Path p,Path p1)
        {
                Random rand=new Random();
                Path
p2=Paths.get("D:\\PROJECT\\MediStore\\src\\medi\\medicines.txt");
                try
                {
                        write(tablet,p2);
                        write(Integer.toString(rand.nextInt(25)+5),p);
                        write(Integer.toString(rand.nextInt(100)),p1);
                }
                catch(IOException e)
                {
                        e.printStackTrace();
                }
        }
}
```

4.2 MODULE 2 FUNCTIONALITY

This is very small functionality

It used to just enter the coustomer details

This is declared as an abstract class since there is an abstract method in the class

This will be defined later in the up coming modules

This class is extended by the class Availability

```
package medi;
abstract public class customer extends Availability{
                public static String name;
                public static long r;
                abstract public void customers();
}
```

4.3 MODULE 3 FUNCTIONALITY

This is the main functionality of all the functionalities

This class extends the customer class

By this we can say that we have use the multilevel inheritance in the project

```
package medi;

import java.io.*;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.util.Scanner;
import java.util.Random;

public class Main extends customer{
this method is used to add new medicinie
	public static void write(File file) throws IOException
	{
		File medicineq=new
File("D:\\PROJECT\\MediStore\\src\\medi\\medicinequantity.txt");
		File medicinec=new
File("D:\\PROJECT\\MediStore\\src\\medi\\medicinecost.txt");
		try
		{
			FileReader fr1=new FileReader(file);
			LineNumberReader lr1=new LineNumberReader(fr1);
			try
			{
				String stme=lr1.readLine();
```

```
                                FileOutputStream fos = new FileOutputStream(medicineq);
                                BufferedWriter bw1 = new BufferedWriter(new
OutputStreamWriter(fos));
                                FileOutputStream fos2 = new
FileOutputStream(medicinec);
                                BufferedWriter bw2 = new BufferedWriter(new
OutputStreamWriter(fos2));
                                while(stme!=null)
                                {
                                        Random rand1=new Random();
                                        stme=lr1.readLine();
                                        if(stme!=null)
                                        {

        bw1.write(Integer.toString(rand1.nextInt(100)));
                                                bw1.newLine();

        bw2.write(Integer.toString(rand1.nextInt(25)+5));
                                                bw2.newLine();
                                        }
                                        else
                                        {

        bw1.write(Integer.toString(rand1.nextInt(100)));

        bw2.write(Integer.toString(rand1.nextInt(25)+5));
                                        }
                                }
                                bw2.close();
                                bw1.close();
```

```
                    lr1.close();

                }
                catch(IOException e)
                {
                    e.printStackTrace();
                }
            }
            catch(IOException e)
            {
                e.printStackTrace();
            }

        }
```

Further implementation of this class is done in the next functionality

4.3 MODULE 4 FUNTIONALITY

This functionality is the continuation for the main class

The below method is defined now which is declared before as an abstract method

Here user customer name and phone number asked which is used for further in implementing billing process

```
    public void customers()
    {
            Scanner s=new Scanner(System.in);
            System.out.println("Enter coustomer namme");
      name=s.next();
            System.out.println("Enetr coustomer phone number:");
      r=s.nextLong();
    }
```

This the main method of all classes all methods will be called here

Billing process is done here

Code is written to store the all the bills in the file

```
	public static void main(String args[])
	{
		Scanner sc=new Scanner(System.in);
		File f = new File("D:\\PROJECT\\MediStore\\src\\medi\\medicines.txt");
		int c=0;
		if(f.exists())
		{
			try
			{
				FileReader fr=new FileReader(f);
				LineNumberReader lr=new LineNumberReader(fr);
				try
				{
					String strm3=lr.readLine();
					while(strm3!=null)
					{
						c++;
						strm3=lr.readLine();
					}
					//System.out.println(c);
				}
				catch(IOException e)
				{
					e.printStackTrace();
				}
			}
			catch(IOException e)
			{
```

```
                        e.printStackTrace();
                }


        }
        System.out.println("           WELCOME TO MEDICAL STORE          ");
        customer cust=new Main();
        cust.customers();
        String cart[]=new String[c];
        int cartitemcosts[]=new int[100];
        int nooftablets[]=new int[100];
        int tabletcost[]=new int[100];
        File f1=new File("D:\\PROJECT\\MediStore\\src\\medi\\bills.txt");
        int c1=0;
        if(f1.exists())
        {
                try
                {
                        FileReader fri=new FileReader(f1);
                        LineNumberReader lr=new LineNumberReader(fri);
                        try
                        {
                                String strm2=lr.readLine();
                                while(strm2!=null)
                                {
                                        c1++;
                                        strm2=lr.readLine();
                                }
                        }
                        catch(IOException e)
                        {
```

```
                        e.printStackTrace();
                    }
                    if(c1==0)
                    {
                        try
                        {
                            write(f);
                        }
                        catch(IOException e)
                        {
                            e.printStackTrace();
                        }
                    }
                }
                catch(IOException e)
                {
                    e.printStackTrace();
                }

            }
            System.out.println("Buy Medicine");
            int cost=0;
            int h=0;
            while(true)
            {
                System.out.println("Enter the medicine u need");
                String tablet=sc.next();
                String z = "no";
                int n=0;
                int count1=0;
            try
            {
                        File filemedicine=new
File("D:\\PROJECT\\MediStore\\src\\medi\\medicines.txt");
                        FileReader fr=new FileReader(filemedicine);
                        LineNumberReader lr=new LineNumberReader(fr);
                        try
                        {
                                    String strm=lr.readLine();
                                    while(strm!=null)
                                    {
                                                count1++;
                                                if(strm.contains(tablet))
                                                {
                                                            n=count1;
                                                            z="yes";
                                                            break;
                                                }
                                                else
                                                {
                                                            strm=lr.readLine();
                                                }
                                    }
                        }
                        catch(IOException e)
                        {
                                    e.printStackTrace();
                        }
            }
            catch(IOException e)
```

```
                        {
                                e.printStackTrace();
                        }
                        File medicineq=new
File("D:\\PROJECT\\MediStore\\src\\medi\\medicinequantity.txt");
                        File medicinec=new
File("D:\\PROJECT\\MediStore\\src\\medi\\medicinecost.txt");
                        System.out.println(z);
                        if(z.equals("yes"))
                        {
                                System.out.println("How many Tablets do you want to add
to cart");
                                int count=sc.nextInt();
                                int count2=0;
                                int medicinequantity=0;
                                try
                                {
                                        FileReader fr3=new FileReader(medicineq);
                                        LineNumberReader lr=new
LineNumberReader(fr3);
                                        try
                                        {
                                                String stm1=lr.readLine();
                                                while(stm1!=null)
                                                {
                                                        count2++;
                                                        if(count2==n)
                                                        {

        medicinequantity=Integer.parseInt(stm1);
```

```
                                        break;
                                    }
                                    stm1=lr.readLine();
                                }
                                lr.close();
                            }
                            catch(IOException e)
                            {
                                e.printStackTrace();
                            }
                        }
                        catch(IOException e)
                        {
                            e.printStackTrace();
                        }
                        boolean status=checkAvailability(medicinequantity,count);
                        if(status==true)
                        {
                            int lnc=0;
                            try
                            {
                                FileReader fr4=new FileReader(medicinec);
                                LineNumberReader lrc=new
LineNumberReader(fr4);
                                try
                                {
                                    String stmk=lrc.readLine();
                                    while(stmk!=null)
                                    {
                                        lnc++;
```

```
                                                        if(lnc==n)
                                                        {
                                                                cart[h]=tablet;

        cost=cost+(count*Integer.parseInt(stmk));
                                                                nooftablets[h]=count;

        tabletcost[h]=Integer.parseInt(stmk);

        cartitemcosts[h]=(count*Integer.parseInt(stmk));
                                                                h=h+1;
                                                        }
                                                        stmk=lrc.readLine();
                                                }
                                                lrc.close();
                                        }
                                        catch(IOException e)
                                        {
                                                e.printStackTrace();
                                        }
                                }
                                catch(IOException e)
                                {
                                        e.printStackTrace();
                                }
                        }
                        else
                        {
                                System.out.println("Medicine is out of
stock.\nplease come back later");
```

```
                                        }
                              }
                              else
                              {
                                        System.out.println("Medicine is out of stock.\nPlease
come back later");
                                        Path
p=Paths.get("D:\\PROJECT\\MediStore\\src\\medi\\medicinecost.txt");
                                        Path
p1=Paths.get("D:\\PROJECT\\MediStore\\src\\medi\\medicinequantity.txt");
                                        UpdateStock(tablet,p,p1);
                              }
                              System.out.println("Do you want to add other medicine:(choose
1/0)");
                              int choice=sc.nextInt();
                              if(choice==0)
                              {
                                        break;
                              }
                    }
                    if(cost==0)
                    {
                              System.out.println("Thanks for visiting");
                    }
                    else
                    {
                              Path
p=Paths.get("D:\\PROJECT\\MediStore\\src\\medi\\bills.txt");
                              String main="BILLING FOR YOUR MEDICINE";
                              String name1="Name :"+name;
                              String phno="Phone number :"+r;
                              String totbill="The Bill is:"+cost;
                              String discount="Discount:"+cost/10;
                              String finalbill="The Final Bill:"+(cost-(cost/10));
                              System.out.println(main);
                              System.out.println(name1);
                              System.out.println(phno);
                              String dupmain=System.lineSeparator()+main;
                              String dupname1=System.lineSeparator()+name1;
                              String dupphno=System.lineSeparator()+phno;
                              String duptotbill=System.lineSeparator()+totbill;
                              String linesep=System.lineSeparator()+" ";
                              String dupdiscount=System.lineSeparator()+discount;
                              String dupfinalbill=System.lineSeparator()+finalbill;
                              try {
                                Files.write(p, dupmain.getBytes(),
StandardOpenOption.APPEND);
                                Files.write(p, dupname1.getBytes(),
StandardOpenOption.APPEND);
                                Files.write(p, dupphno.getBytes(),
StandardOpenOption.APPEND);
                              } catch (IOException e) {
                                System.err.println(e);
                              }
                              for(int k=0;k<h;k++)
                              {
                                        String str2=cart[k]+"  "+nooftablets[k]+"x"+tabletcost[k]+" :
"+cartitemcosts[k];
                                        String dupstr2=System.lineSeparator()+str2;
                                        try
```

```
                                    {
                                            Files.write(p, dupstr2.getBytes(),
StandardOpenOption.APPEND);
                                    }
                                    catch(IOException e)
                                    {
                                            System.err.println(e);
                                    }
                                    System.out.println(str2);
                            }

                            System.out.println(totbill);
                            System.out.println(discount);
                            System.out.println(finalbill);
                            try {

                               Files.write(p, duptotbill.getBytes(),
StandardOpenOption.APPEND);
                               Files.write(p, dupdiscount.getBytes(),
StandardOpenOption.APPEND);
                               Files.write(p, dupfinalbill.getBytes(),
StandardOpenOption.APPEND);
                               Files.write(p, linesep.getBytes(), StandardOpenOption.APPEND);
                               Files.write(p, linesep.getBytes(), StandardOpenOption.APPEND);
                            } catch (IOException e) {
                               System.err.println(e);
                            }
                    }
            }
}
```

CHAPTER 5

RESULT

RESULT VERIFICATION OF INHERITANCE

In the program I have used three class each class extends to another class

Class Availability

Abstract Class customer extends Availability

Class main extends customer

These are the classes that I have defined in the project. Since inheritance is implemented by the extends keyword. I need to access the information in Availability class to main class so I haved used extends keyword. Since one class extends other class this type of inheritance is known as multilevel inheritance.

Hence the verification of multilevel inheritance in my project is done

RESULIT VERIFICATION OF POLYMORPHISM

Polymorphism is noting but the overloading and overriding concepts. In my program I have used overloading concept in the program there are two methods of same name write it differs based on the parameters passed

One write has the parameters : write(File file)

Another write has the parameters: write`(String tablet, path p3)`

it goes to the method based on the parameters given

hence the verification of the polymorphism in my project is done

RESULT VERIFICATION OF ABSTRACT CLASS

When any one method in the class is defined as an abstract method then the class should be declared as an abstract class even though remaining methods are not abstract

In my project I have used the class customer as an abstract because the method customers in the class customer is declared as abstract

And it is later defined in the main class by creating the object

Hence the verification of the abstract class in my project is done

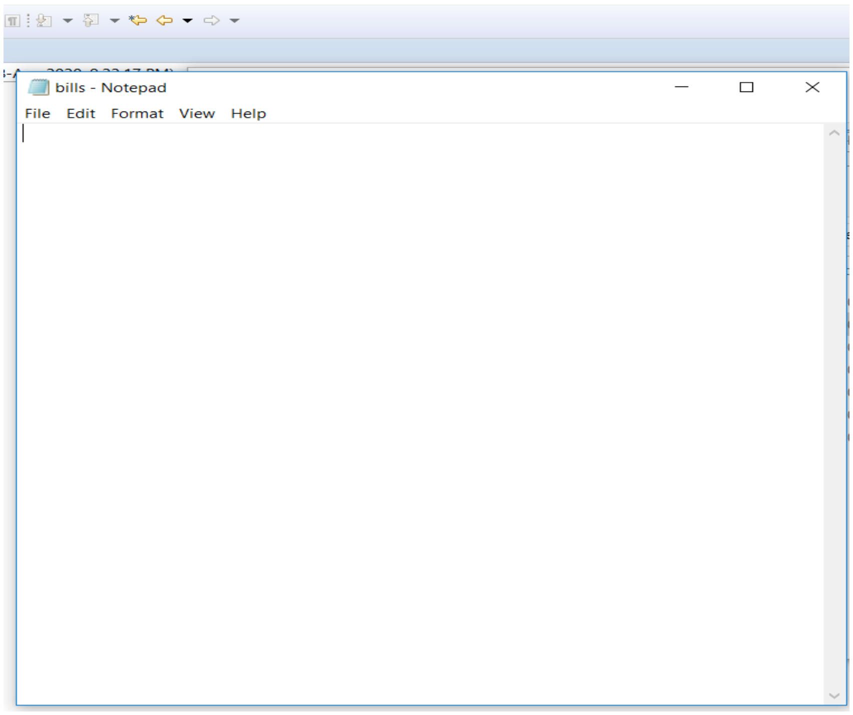
RESULT VERIFICATION OF EXCETION HANDLING

Exception handlings are used when the program is running successfully if they is any interrupt it throws the exception

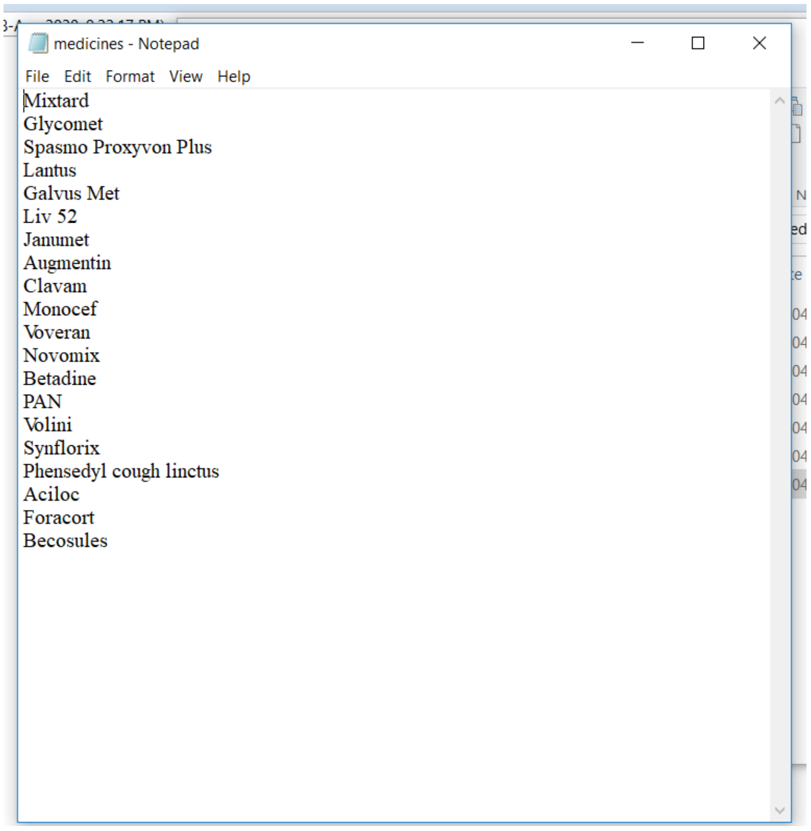
In my program I have used exceptions while reading and writing the content from the file. The exceptions I have used are try,catch,throws.

Hence the verification of exception handling in my program is done

OUTPUT :

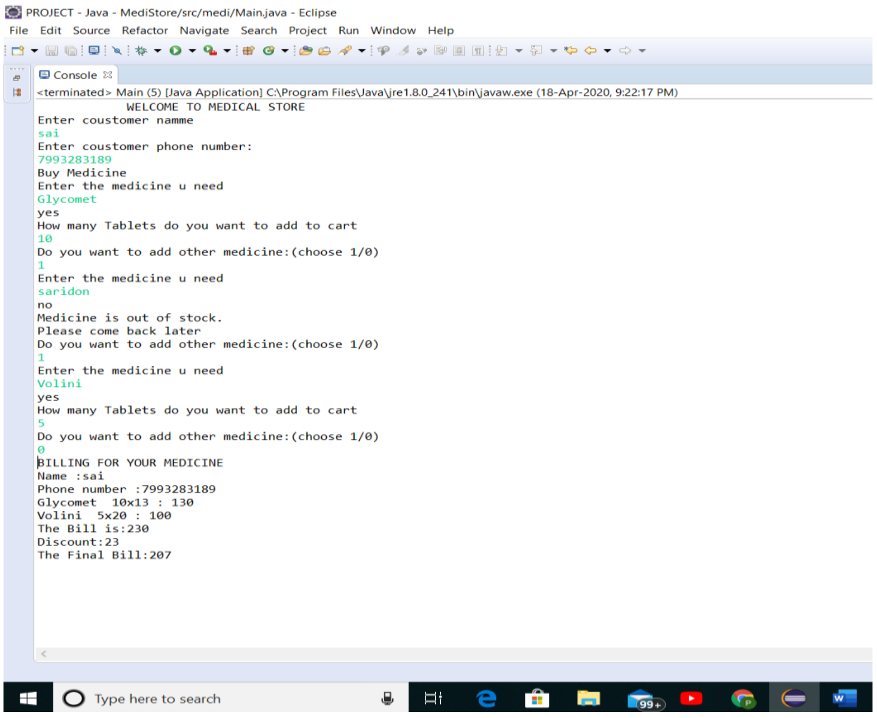


Here the bills file is empty since there is no bill done till now



```
Mixtard
Glycomet
Spasmo Proxyvon Plus
Lantus
Galvus Met
Liv 52
Janumet
Augmentin
Clavam
Monocef
Voveran
Novomix
Betadine
PAN
Volini
Synflorix
Phensedyl cough linctus
Aciloc
Foracort
Becosules
```

Here the observe carefully the medicine there are 20 medicines

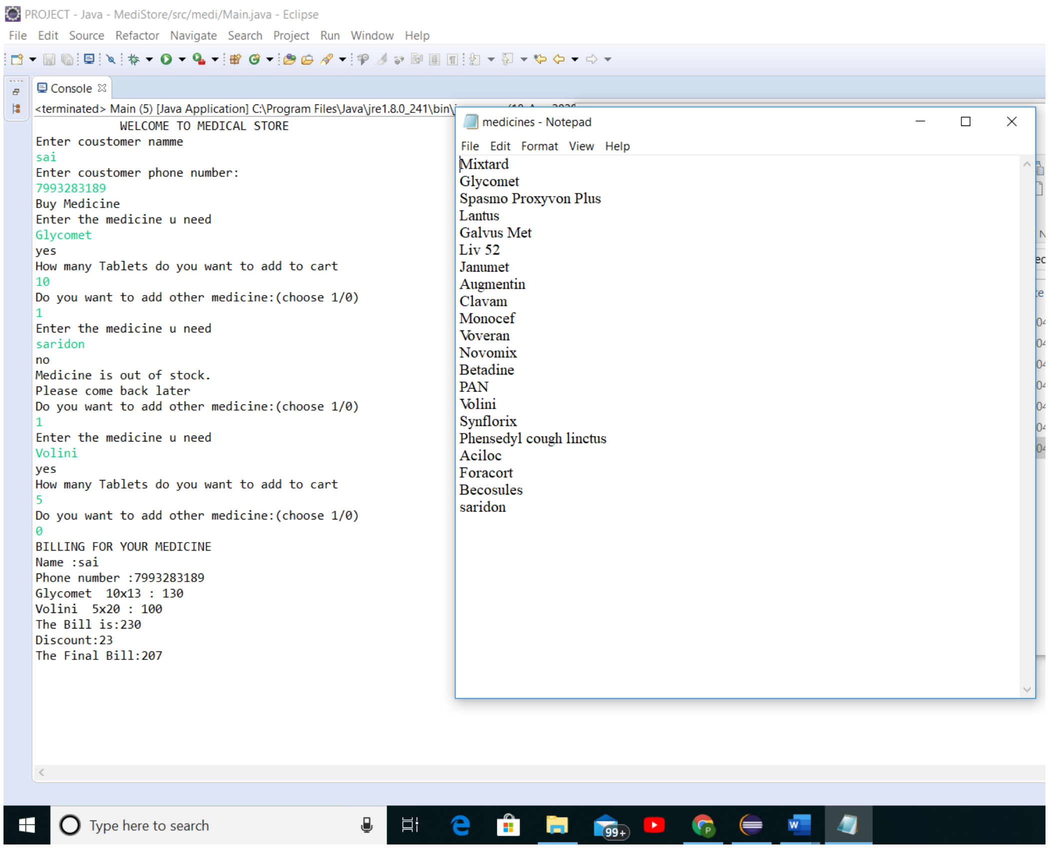

PROJECT - Java - MediStore/src/medi/Main.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Console
Type here to search

```
<terminated> Main (5) [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (18-Apr-2020, 9:22:17 PM)
                WELCOME TO MEDICAL STORE
Enter coustomer namme
sai
Enter coustomer phone number:
7993283189
Buy Medicine
Enter the medicine u need
Glycomet
yes
How many Tablets do you want to add to cart
10
Do you want to add other medicine:(choose 1/0)
1
Enter the medicine u need
saridon
no
Medicine is out of stock.
Please come back later
Do you want to add other medicine:(choose 1/0)
1
Enter the medicine u need
Volini
yes
How many Tablets do you want to add to cart
5
Do you want to add other medicine:(choose 1/0)
0
BILLING FOR YOUR MEDICINE
Name :sai
Phone number :7993283189
Glycomet  10x13 : 130
Volini  5x20 : 100
The Bill is:230
Discount:23
The Final Bill:207
```

Here when we enter saridon it is not there in the list so after giving

Medicine is out of stock

It directly updates the saridon in to the stock



```
<terminated> Main (5) [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\
            WELCOME TO MEDICAL STORE
Enter coustomer namme
sai
Enter coustomer phone number:
7993283189
Buy Medicine
Enter the medicine u need
Glycomet
yes
How many Tablets do you want to add to cart
10
Do you want to add other medicine:(choose 1/0)
1
Enter the medicine u need
saridon
no
Medicine is out of stock.
Please come back later
Do you want to add other medicine:(choose 1/0)
1
Enter the medicine u need
Volini
yes
How many Tablets do you want to add to cart
5
Do you want to add other medicine:(choose 1/0)
0
BILLING FOR YOUR MEDICINE
Name :sai
Phone number :7993283189
Glycomet  10x13 : 130
Volini  5x20 : 100
The Bill is:230
Discount:23
The Final Bill:207
```

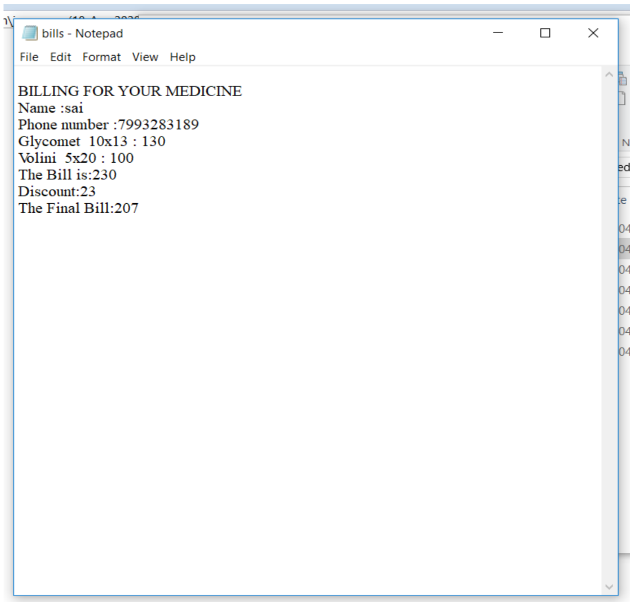
```
Mixtard
Glycomet
Spasmo Proxyvon Plus
Lantus
Galvus Met
Liv 52
Janumet
Augmentin
Clavam
Monocef
Voveran
Novomix
Betadine
PAN
Volini
Synflorix
Phensedyl cough linctus
Aciloc
Foracort
Becosules
saridon
```

After execution when you open the file the new item saridon is added in to the file

And the cost and stock also will be updated

In the above screen short we can observe that the billing of medicine is done were the

10 percent discount is allotted on total bill



```
BILLING FOR YOUR MEDICINE
Name :sai
Phone number :7993283189
Glycomet  10x13 : 130
Volini  5x20 : 100
The Bill is:230
Discount:23
The Final Bill:207
```

Previously we saw the bills file is empty now the bills file is updated with the new bill.

CHAPTER 6

CONCLUSION

Medistore is one of the basic java application in java created using different OOPS concepts available.

Medistore has the basic options that is needed

- To store the data of billing
- To store the medicines that are present in the store
- To update the new medicines.
- To allocates the billing including discount.

This project is done successfully and also executed successfully without any errors
It first ask the customer name and the mobile number
Then once you enter it ask for medicine u need if medicine is present it ask for how many tablets you need else it says the stock is empty and updates the stock. Next it asks if want another medicine or not if u want it issues another medicine else it print the bill of your medicine. The bill includes customer name, phone number, name of medicines you bought and cost of the medicine, quantity you required and total cost of all medicine, finally final cost of bill after discount.
Many OOPS concepts are used in this project.

REFERENCES

1. Herbert Schildt-Java_ The Complete Reference-McGraw-Hill Education (2017)
2. Head First Java: A Brain-Friendly Guide, 2nd Edition - Kathy Sierra, Bert Bates(2003)
3. PROGRAMMING WITH JAVA- PRIMER A
4. https://www.w3schools.com/java/java_oop.asp
5. https://www.javatpoint.com/java-tutorial

These are the books and the online links that I have reffered to complete my mini project.