NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College, Affiliated to VTU | Approved by AICTE New Delhi & UGC
Accredited by NAAC with 'A' Grade & Accredited by NBA

Golden Jubilee
YEARS OF EXCELLENCE IN EDUCATION

A MINI PROJECT

REPORT

for

Mini Project in JAVA (19CSE48)

COVID PATIENT REGISTRATION

Submitted by

RAJESH G
USN: 1NH19CS139
Semester-Section: 4-SEC

In partial fulfillment for the award of the degree of

Bachelor of Engineering

in

COMPUTER SCIENCE AND ENGINEERING

NEW HORIZON
COLLEGE OF ENGINEERING

Autonomous College, Affiliated to VTU | Approved by AICTE New Delhi & UGC
Accredited by NAAC with 'A' Grade & Accredited by NBA

Certificate

This is to certify that the mini project work titled

COVID PATIENT REGISTRATION

Submitted in partial fulfillment of the degree of
Bachelor of Engineering in
Computer Science and Engineering by

RAJESH G
USN: 1NH19CS139

DURING
EVEN SEMESTER 2020-2021
for
COURSE CODE: 19CSE48

Ramesh 5/8/21
Signature of Reviewer

Signature of HOD

SEMESTER END EXAMINATION

Name of the Examiner	*Signature with date*
1. _____	_____
2. _____	_____

ABSTRACT

With advancements in technology, people are looking forward to make the works easy and in these days may queries and complaints are increasing day by day so many people give there complaints in respective departments so it will be difficult to store and maintain in paper so to avoid that thing we can directly register all those information and issues in online mode it will be easy for us to access them quickly. So we are using this application which can help to store the details which is covid patient registration.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success.

I have great pleasure in expressing gratitude to **Dr. Mohan Manghnani**, Chairman of New Horizon Educational Institutions for providing necessary infrastructure and creating good environment.

I take this opportunity to express my profound gratitude to **Dr. Manjunatha,** Principal, New Horizon College of Engineering, for his constant support and encouragement.

I am grateful to **Dr. Amarjeet Singh,** Dean - Academics, for his unfailing encouragement and suggestions, given to me in the course of my project work.

I would also like to thank **Dr. B. Rajalakshmi**, Professor and Head, Department of Computer Science and Engineering, for her constant support.

I also express my gratitude to **Dr. Pamela Vinitha Eric**, **Professor**, Department of Computer Science and Engineering, my project guide, for constantly monitoring the development of the project and setting up precise deadlines. Her valuable suggestions were the motivating factors in completing the work.

RAJESH G

USN: 1NH19CS139

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

As we all saw the todays world pandemic which is the most deadliest making the people not to do even a single job without the mask and the sanitizer .in order to make more effective work in case of the hospital we are making the covid patient registration which keeps the work more safety and keeps the information.

1.2 OBJECTIVES

To manage patient name raised by the people accordingly by accepting n number of issues and accepting there queries with different description by entering their ID and can update there queries status by giving the their ID

1.3 METHODOLOGY TO BE FOLLOWED

The methodology I am following in this mini project is based on the java programming language and I am also using the inheritance concept in this program. the software methodology followed in the mini project includes object-oriented programming methodology.

1.4 EXPECTED OUTCOMES

* entering the patient id

* enter the customer name

* entering the number of the orders

* enter the order id

* enter the order name

* enter the order description

* enter order status

CHAPTER 2

FUNDAMENTALS OF JAVA PROGRAMMING

2.1 COVID PATIENT REGISTRATION

- Java is a platform independent language. Compiler(javac) converts source code (.java file) to the …
- Java is an Object Oriented language. Object oriented programming is a way of organizing programs …
- Simple. Java is considered as one of simple language because it does not have complex features …
- Robust Language. Robust means reliable. Java programming language is developed in a way that ..

2.2 CLASS

Class is a keyword to declare that a new class is being defined. A class is a template for objects and an object is an instance of a class. When the individual objects are created, they inherit all the variables and methods from the class. Everything in java is associated with classes and objects along with its attributes and methods

A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

1. Modifiers: A class can be public or has default access .

2. class keyword: class keyword is used to create a class.

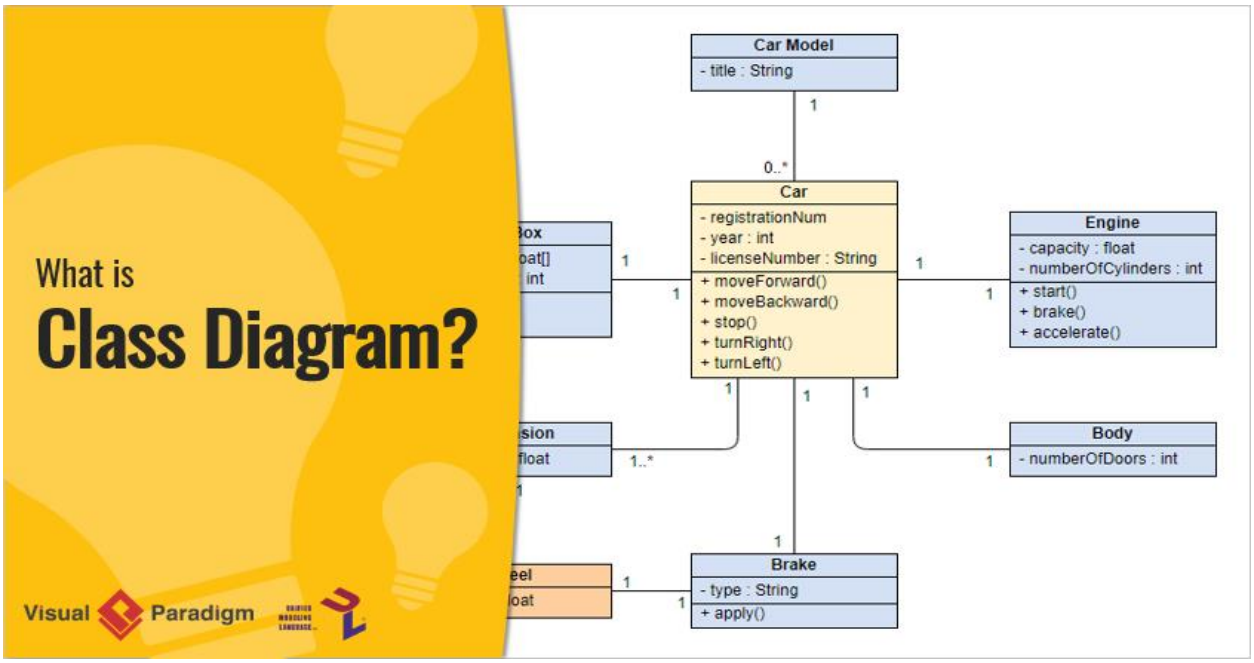
3. Class name:The name should begin with an initial letter .

// Class Declaration

```
public class Dog

{

        // Instance Variables

        String name;

        String breed;

        int age;

        String color;



        // Constructor Declaration of Class

        public Dog(String name, String breed,

                                int age, String color)

        {

                this.name = name;

                this.breed = breed;

                this.age = age;

                this.color = color;

        }



        // method 1
```

```
public String getName()
{
        return name;
}

// method 2
public String getBreed()
{
        return breed;
}

// method 3
public int getAge()
{
        return age;
}

// method 4
public String getColor()
{
```

```
        return color;
    }

    @Override
    public String toString()
    {
        return("Hi my name is "+ this.getName()+
            ".\nMy breed,age and color are " +
            this.getBreed()+"," + this.getAge()+
            ","+ this.getColor());
    }

    public static void main(String[] args)
    {
        Dog tuffy = new Dog("tuffy","papillon", 5, "white");
        System.out.println(tuffy.toString());
    }
}
```



2.3 OBJECT

It is a basic unit of Object-Oriented Programming and represents the real life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of :

State: It is represented by attributes of an object. It also reflects the properties of an object.

Behavior: It is represented by methods of an object. It also reflects the response of an object with other objects.

Identity: It gives a unique name to an object and enables one object to interact with other objects.

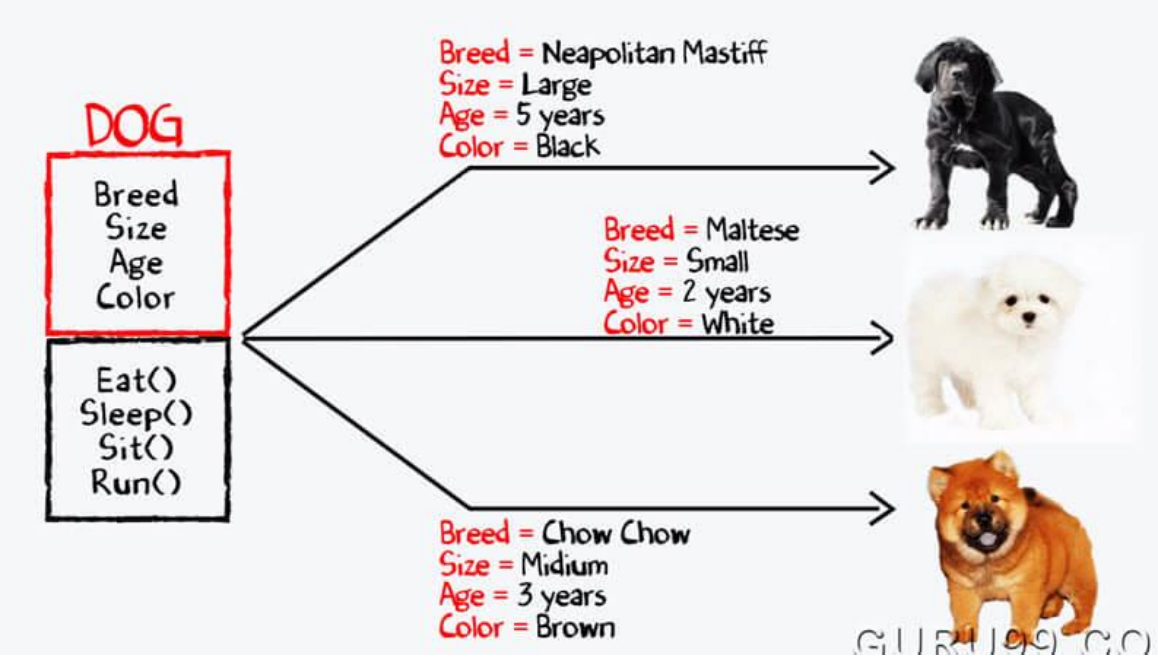
Example of an object: dog

Blank Diagram - Page 1 (5)

Objects correspond to things found in the real world. For example, a graphics program may have objects such as “circle”, “square”, “menu”. An online shopping system might have objects such as “shopping cart”, “customer”, and “product”.

Declaring Objects (Also called instantiating a class)

When an object of a class is created, the class is said to be instantiated. All the instances share the attributes and the behavior of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any number of instances.



I n this post, we will cover 5 different ways to created objects in Java and understand all essential concepts required to understand the methods.

Create objects using ‘new’ keyword

Create objects Using clone( ) method

Create objects using the newInstance( ) method of class

Create objects using deserialization

Create objects using the newInstance( ) method of constructor class

```
public class CreateObjectExample1
{
void show()
{
System.out.println("Welcome to javaTpoint");
}
public static void main(String[] args)
{
//creating an object using new keyword
CreateObjectExample1 obj = new CreateObjectExample1();
//invoking method using the object
obj.show();
}
}
```

Output:

```
Welcome to javaTpoint
```

2.4 INHERITANCE

Inheritance is one of the key features of OOP that allows us to create a new class from an existing class.

The new class that is created is known as **subclass** (child or derived class) and the existing class from where the child class is derived is known as **superclass** (parent or base class).

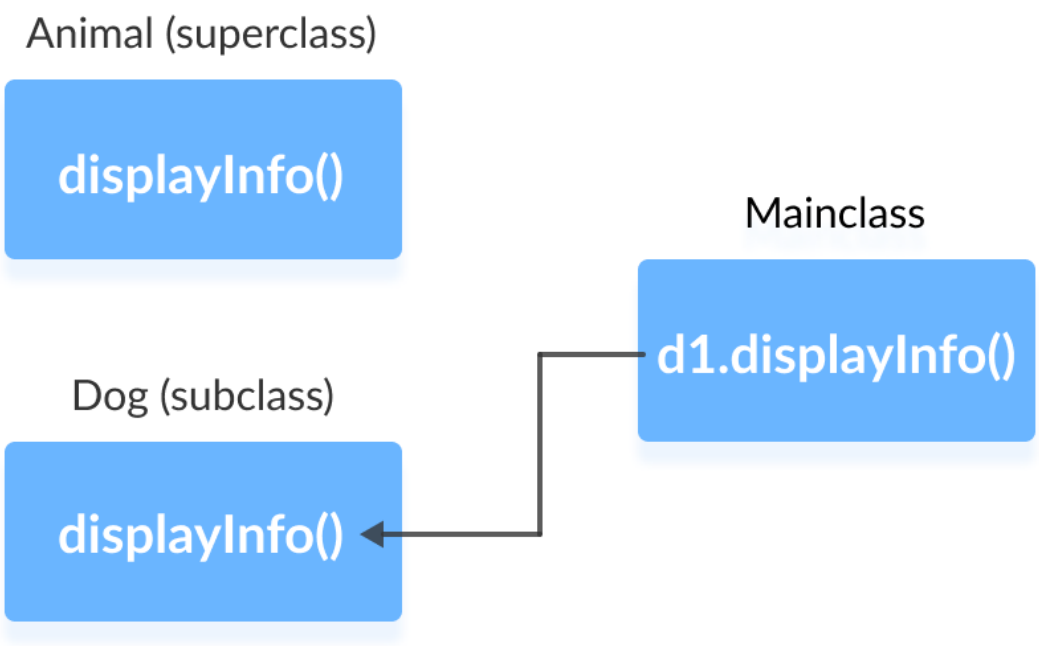
The `extends` keyword is used to perform inheritance in Java

Java Overriding Rules

- Both the superclass and the subclass must have the same method name, the same return type and the same parameter list.
- We cannot override the method declared as `final` and `static`.
- We should always override abstract methods of the superclass (will be discussed in later tutorials).

The overriding takes place where the same methods were present in the both the base class and the derived class .there the method calling occurs with the help of the creating the object (as the default constructor) where it access only the derived methods not the base class .so here the method overriding happening where base class wont execute the method which is called.

In such cases we have the keyword called a 'super' by that we can access the method of the same name which is present in the base class .



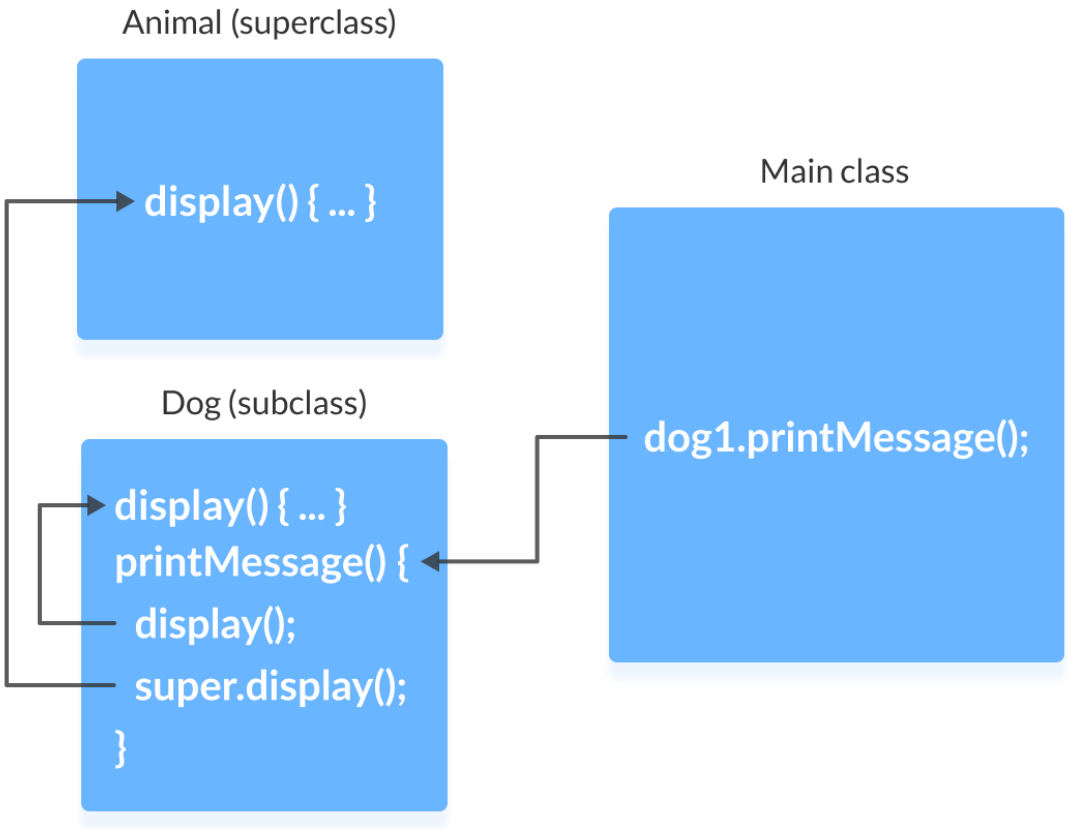
the displayInfo() method is present in both the Animal superclass and the Dog subclass. When we call displayInfo() using the d1 object (object of the subclass), the method inside the subclass Dog is called. The displayInfo() method of the subclass overrides the same method of the superclass.

SUPER KEYWORD

Uses of super keyword

1. To call methods of the superclass that is overridden in the subclass.

2. To access attributes (fields) of the superclass if both superclass and subclass have attributes with the same name.
3. To explicitly call superclass no-arg (default) or parameterized constructor from the subclass constructor.



We then created an object `dog1` of the `Dog` class. Then, the `printType()` method is called using this object.

Inside the `printType()` function,

- `type` refers to the attribute of the subclass `Dog`.
- `super.type` refers to the attribute of the superclass Animal.

Use of super() to access superclass constructor

Here, when an object dog1 of Dog class is created, it automatically calls the default or no-arg constructor of that class.
Inside the subclass constructor, the super() statement calls the constructor of the superclass and executes the statements inside it. Hence, we get the output I am an animal.

Call Parameterized Constructor Using super()

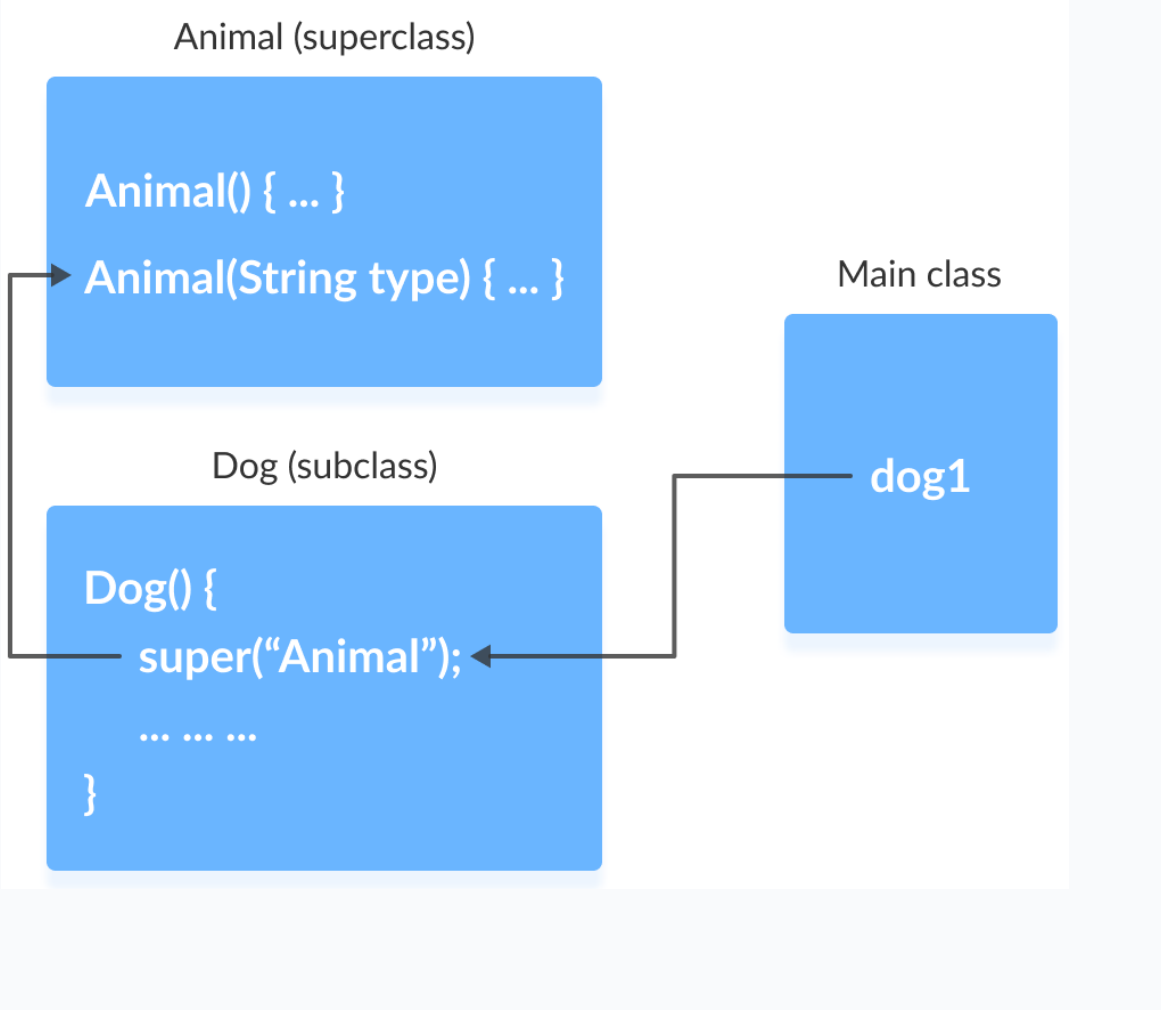
The compiler can automatically call the no-arg constructor. However, it cannot call parameterized constructors.

If a parameterized constructor has to be called, we need to explicitly define it in the subclass constructor.

```
class Employee{
 float salary=40000;
}
class Programmer extends Employee{
 int bonus=10000;
 public static void main(String args[]){
   Programmer p=new Programmer();
   System.out.println("Programmer salary is:"+p.salary);
   System.out.println("Bonus of Programmer is:"+p.bonus);
}
}
```

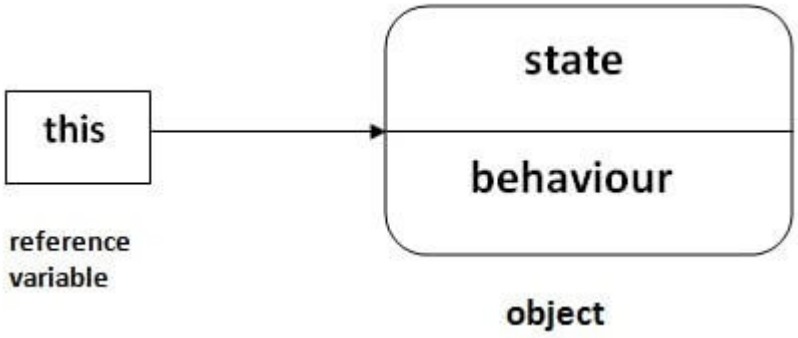
Test it Now

```
Programmer salary is:40000.0
Bonus of programmer is:10000
```



this keyword in java

There can be a lot of usage of **java this keyword**. In java, this is a **reference variable** that refers to the current object.



Usage of java this keyword

Here is given the 6 usage of java this keyword.

1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly)
3. this() can be used to invoke current class constructor.
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.
6. this can be used to return the current class instance from the method.

2.5 POLYMORPHISM

Polymorphism in Java is a concept by which we can perform a single action in many different ways. Poly means "many" and Morph means "forms", therefore polymorphism means "many forms". We can perform polymorphism in Java by using method overload or constructer overload or operator overload.

1. Method Overload: In method overload, a particular class has more than one method with the same name but with different set of parameters.
2. Constructer Overload: In constructer overload, a particular class has more than one constructer with the same name but with different set of parameters.
3. Operator Overload: In operator overload, user cannot overload any operators in order to avoid confusion, but Java provides the "+" operator which can be used to concatenate two strings as well as to perform addition.

```java
class Bike{
  void run(){System.out.println("running");}
}
class Splendor extends Bike{
  void run(){System.out.println("running safely with 60km");}

  public static void main(String args[]){
    Bike b = new Splendor();//upcasting
    b.run();
  }
}
```

Test it Now

Output:

```
running safely with 60km.
```

2.6 ABSTRACT CLASS

An abstract class in Java is a restricted class that cannot be used to create objects (to access it, it must be inherited from other class). Abstract methods can only be used in am abstract class and it does not have a body. The body is provided by the sub class. It is declared using "abstract" keyword. An abstract class can have both abstract as well as concrete methods.

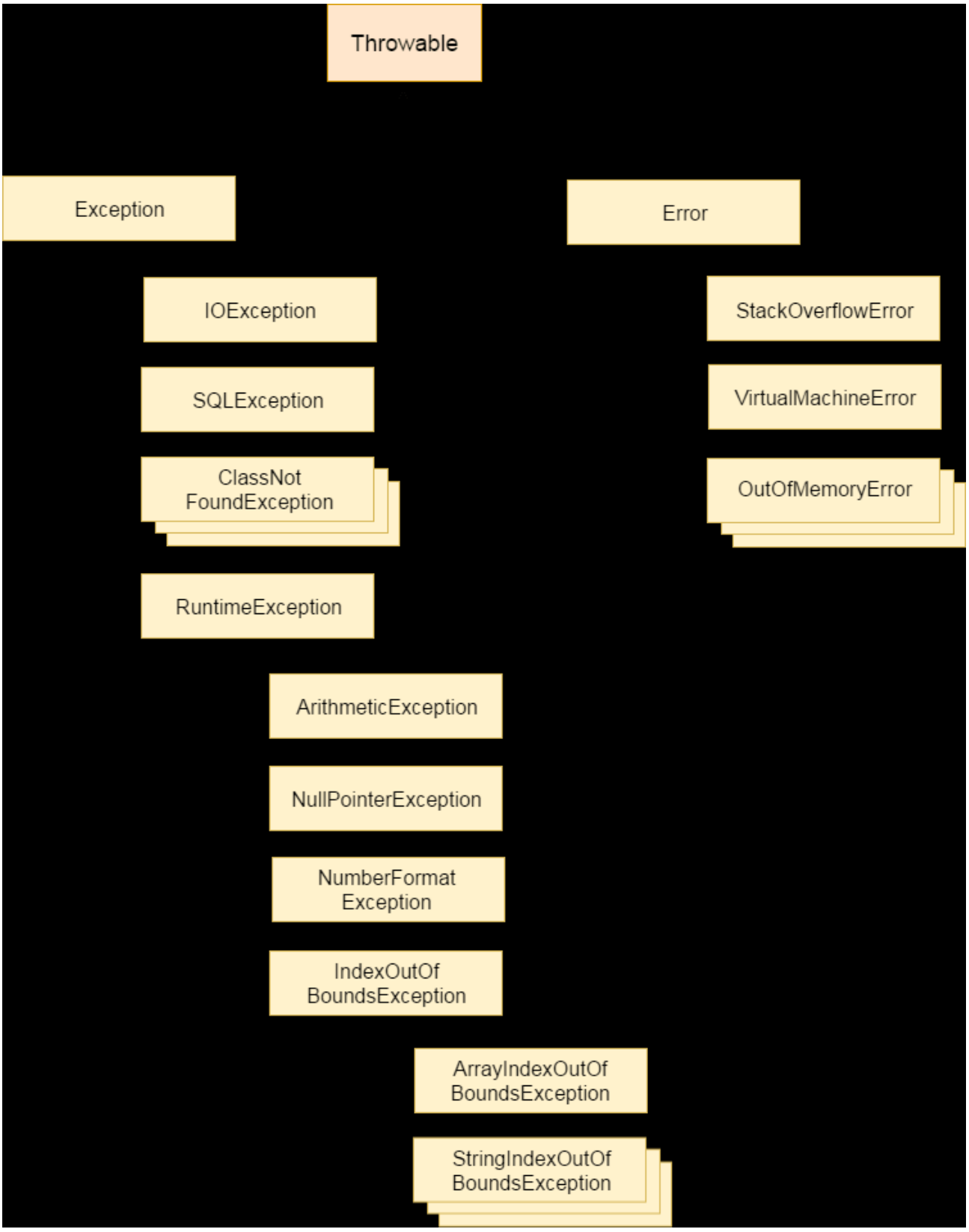
2.7 JAVA PACKAGES

Package in Java is a mechanism to encapsulate a group of classes, subclasses, sub packages and interfaces. They are used for preventing naming conflicts such as two classes with same name but of different package is possible.

2.8 EXCEPTION HANDLING

In Java, exceptions are handled by keywords such as try, catch, throws etc. Whenever the programmer thinks that a certain block of code can cause an exception (ex. FileNotFoundException), the programmer can use try block and catch block and also use throws keyword.

```java
package file;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;

/**
 *
 * @author DELL
 */
public class lab_prog_15 {

    public static void main(String[] args) {
        try{
            FileWriter writer = new FileWriter("indoc");
            writer.write("hello world");
            writer.close();

        }catch(IOException e){
            e.printStackTrace();
        }
        FileInputStream instream = null;
        FileOutputStream outstream = null;
        try{
            File infile = new File("indoc");
            File outfile  = new File("outdoc");
            instream = new FileInputStream(infile);
            outstream = new FileOutputStream(outfile);
            byte[] buffer = new byte[1024];
            int length;
            while((length = instream.read(buffer))>0){
                outstream.write(buffer,0,length);
            }
            instream.close();
            outstream.close();
            System.out.println("File copied successfully");
        }
        catch(IOException ioe){
            ioe.printStackTrace();
        }
    }

}
```


Throwable
Exception
Error
IOException
SQLException
ClassNot FoundException
RuntimeException
ArithmeticException
NullPointerException
NumberFormat Exception
IndexOutOf BoundsException
ArrayIndexOutOf BoundsException
StringIndexOutOf BoundsException
StackOverflowError
VirtualMachineError
OutOfMemoryError

2.9 THREADS

Threads in java is the path followed when executing the program. In Java creating a thread is accomplished by creating interfaces and by extending classes. Every thread in java is controlled by the java.lang.Thread class. Threads can be used to perform complicated tasks in the background without interrupting the main program.

2.10 I/O BASICS

In Java, there is an in-built package called java.io which provides a set of input streams and a set of output streams used to read and write to files or other input and output sources. The most used Scanner class and also the println statements are defined in this package.

2.11 INTERFACE

An interface in Java is an abstract data type that is used to specify a behavior that classes must implement. Interfaces are declared using the keyword "interface" and may only contain method signature and constant declaration.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 HARDWARE REQUIREMENTS

The most common set of the requirements defined by the operating system or software application is the physical computer resources also known as the hardware. A hard ware requirements is often accompanied by a hardware compatibility list (HDL), especially in case of operating systems.

- Hardware requirements for present project:
- Ram: 8GB
- Hard disk :1TB
- Processor: intel core, i5

3.2 SOFTWARE REQUIREMETS

Software requirements deal with the with the defining software resources requirements and pre-requisite that need to be installed on computer to provide optimal functioning of an application. These requirements or pre-requisite are generally not included in the software installation package and need to be installed separately before the software is installed.

Software requirements for present project:

Operating system:
windows 10

Compiler: Netbeans
8.1 installed

CHAPTER 4

DESIGN

4.1 DESIGN GOALS

In order to get the result as per the real time efficiently We should declare the all the variables without any mistake and declaring several functions.

It comprises of the two things

1. User :where he can login and can view whether he has already have an id or not
2. Admin: wherein he can login and can manage the data related to the users

4.2 ALGORITHM

ALGORITHM FOR THE COVID PATIENT PREREGISTRATION

Step 1: create the class incident with the instance varibles which are accessible with the help of this keyword and creating the methods and return the values of the method back.

Step 2: in the patient class with the help of the vector concept we are entering the data of long one and get back displayed .vector which is dynamic in nature

STEP3: we iterate the loop of entering the incident name as per the count asked and we are checking the Boolean value of the check and continues the iteration

STEP 4: creating the inheritance and inherit the properties of the Exception method and displaying the base class method with the help of super keyword.

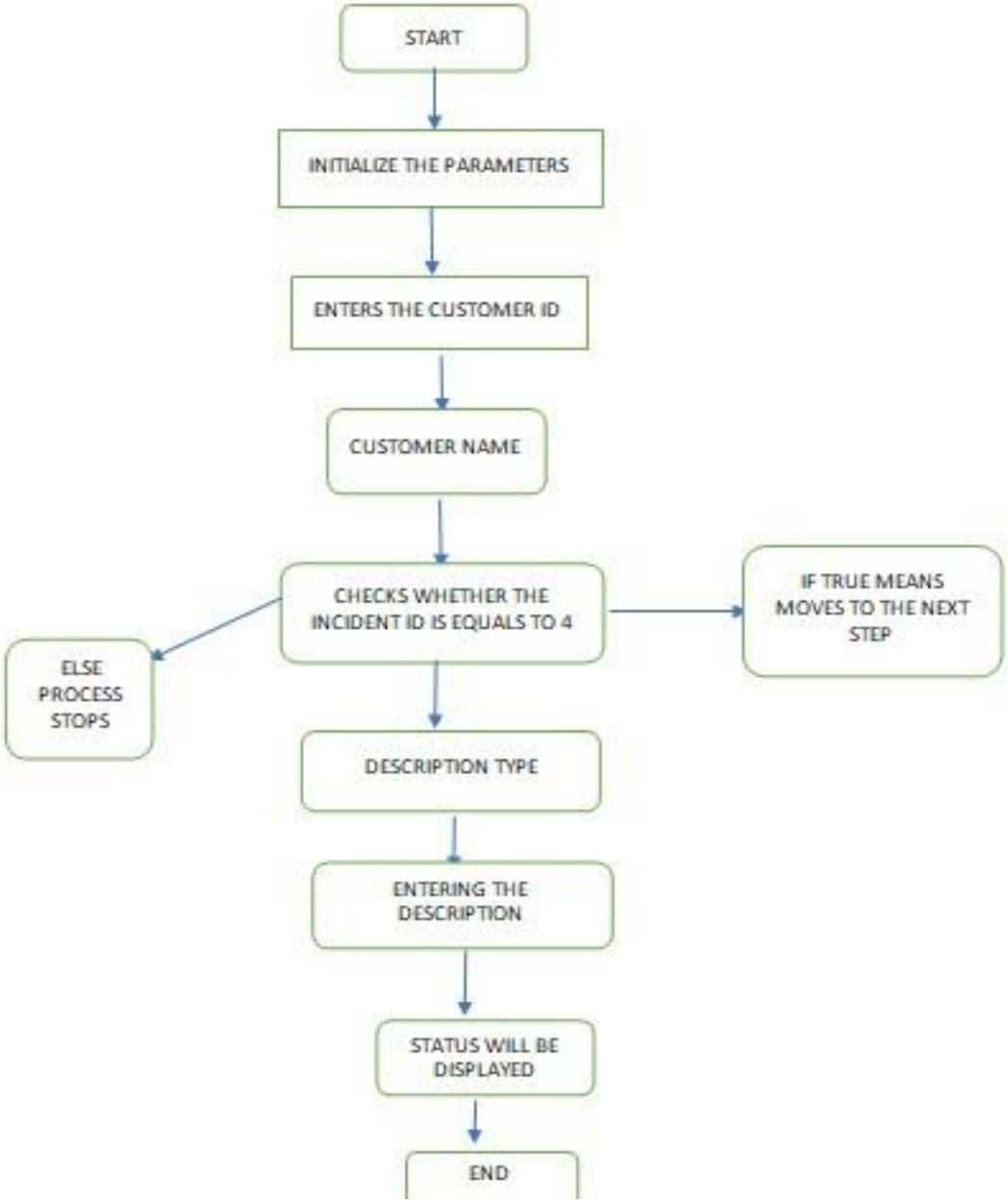
STEP 5:here we are creating the buffer in the name of br and access the data from that

STEP 6: even though theres no data present in the buffer and we are trying to acces the data there it throws the exception which makes the program to run complete else it would have terminate the program.

STEP 7: reading the string value and the long values with the help of the readline method

STEP 8:checking whether the entered incident id equal to the values which we are given else it throws the exception.

4.3 FLOWCHART



CHAPTER 5

IMPLEMENTATION

5.1 MODULE 1 FUNCTIONALITY

Module contains the class patient and the arraylist ,creating the reference variable in the name of vector .for display the incidents we are using the iterator concept ,we are using the some of the methods such as addNewIncident to add the new incident one,creating the instance variable and the accessing them with the help of this keyword.

```
package com.mini.project;
import java.util.*;
//import com.tech.exception.InvalidStatusException;
public class patient {
        private int patientId;
        private String patientName;
        private Vector<Incident> vector = new Vector<Incident>();
        public patient(int patientId, String patientName) {
                this.patientId = patientId;
                this.patientName = patientName;
        }
        public void addNewIncident(Incident incident) {
                vector.add(incident);
        }
        public Vector<Incident> getIncidentList() {
                return vector;
        }
        public void changeIncidentStatus(int incidentId, String status)
                        throws InvalidStatusException {
                Iterator<Incident> itr = vector.iterator();
                boolean check = true;
                while (itr.hasNext()) {
                        Incident incident = itr.next();
                        if (incident.getIncidentId() == incidentId) {
                                incident.setIncidentStatus(status);
                                check = false;
                        }
                }
                if (check == false) {
                        System.out.println("Incident not found");
                }
        }
        public void displayIncidents() {
                Vector<Incident> incidentList = getIncidentList();
                Iterator<Incident> itr = incidentList.iterator();
                while (itr.hasNext()) {
                        Incident incident = itr.next();
                        System.out.println(incident);
                }
        }

}
```

5.2 MODULE 2 FUNCTIONALITY

Here it contains the incident class in this we are using all the private variables which are accessible only in this class, here it throws the exception if the data entered in case not present in the code then it throws the exception.we are overriding the String tostring class

.

```
package com.mini.project;
//import com.tech.exception.InvalidStatusException;
public class Incident {
	private int incidentId;
	private String incidentType;
	private String incidentDescription;
	private String incidentStatus = "getting soon fine ";
	public Incident(int incidentId, String incidentType,
			String incidentDescription)  {
			this.incidentId = incidentId;
			this.incidentType = incidentType;
			this.incidentDescription = incidentDescription;
		}
	public int getIncidentId() {
		return incidentId;
	}
	public String getIncidentType() {
		return incidentType;
	}
	public String getIncidentDescription() {
		return incidentDescription;
	}
	public String getIncidentStatus() {
		return incidentStatus;
	}
	public void setIncidentStatus(String incidentStatus) throws InvalidStatusException {

		if(incidentStatus.equals("InProgress")||incidentStatus.equals("Closed"))
		this.incidentStatus = incidentStatus;
		else
		{
			throw new InvalidStatusException("Invalid status Exception. Should be either Inprogress or Closed ");
		}
	}
	@Override
	public String toString() {
		// TODO Auto-generated method stub
		return "incident id:"+incidentId+"\nincident type:"+incidentType+"\nIncidentDescription:"+incidentDescription+"\nIncident Status:"+incidentStatus+"\n";
	}
}
```

5.3 MODULE 3 FUNCTIONALITY

The main thing here is that the all the exceptions are defined in the exception class which is treating as the super class and we are inheriting the properties of that class,here also we are overriding the getMessage() method here

```java
package com.mini.project;

public class InvalidIncidentIdException extends Exception {

	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
	public InvalidIncidentIdException() {
		// TODO Auto-generated constructor stub
	}
	public InvalidIncidentIdException(String msg) {
		// TODO Auto-generated constructor stub
		super(msg);
	}
	@Override
	public String getMessage() {
		// TODO Auto-generated method stub
		return super.getMessage();
	}
```

5.4 MODULE 4 FUNCTIONALITY

Here we are overriding the the method named as getmessage from the given statement

```java
package com.mini.project;

public class InvalidStatusException extends Exception {

	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
	public InvalidStatusException() {
		// TODO Auto-generated constructor stub
	}

	public InvalidStatusException(String msg) {
		// TODO Auto-generated constructor stub
		super(msg);
	}
	@Override
	public String getMessage() {
		// TODO Auto-generated method stub
		return super.getMessage();
	}

}
```

5.5 MODULE 5 FUNCTIONALITY

In the main function we are putting all the test cases here .taking the different type of the input functions which are the bufferreader,readline(),and the Exceptions ,we are here using

the switch class to select whether we want to change the status or want to exit .using the try catch we are handing the exception which helps to abnormal termination from the program.

```
package com.mini.project;
import java.io.*;
public class ServiceDesk {
        public static void main(String[] args) throws IOException{
                boolean check = true ;
                int incidentId = 0;
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                System.out.println("enter the id:");
                int patientId = Integer.parseInt(br.readLine());
                System.out.println("enter the name :");
                String patientName = br.readLine();
                patient patient = new patient(patientId, patientName);
                System.out.println("enter the number of the patients want to enter :");
                int n = Integer.parseInt(br.readLine());
                for (int i = 1; i <= n; i++) {
                        System.out.println("Enter incident " + i + " details");
                        System.out.println("incident id:");
                        String sincidentId = br.readLine();
                        System.out.println("incident type:");
                        String incidentType = br.readLine();
                        System.out.println("incident description:");
                        String incidentDescription = br.readLine();
                        if (sincidentId.length() != 4 || Integer.parseInt(sincidentId) <= 0) {
                                try {
                                        System.out.println(sincidentId.length() + " "
                                                        + Integer.parseInt(sincidentId));
                                        throw new InvalidIncidentIdException(
                                                        "Invalid incident Id. Id must be equal to 4 digits and greater than zero");
                                } catch (InvalidIncidentIdException e) {
                                        // TODO Auto-generated catch block
                                        System.err.println(e.getMessage());
                                        i = i - 1;
                                        System.out.println("Enter details again");
                                        continue;
                                }
                        } else {
                                incidentId = Integer.parseInt(sincidentId);
                        }
                        Incident incident = new Incident(incidentId, incidentType,
                                        incidentDescription);
                        patient.addNewIncident(incident);
                }
                patient.displayIncidents();
                while (check) {
                        System.out.println("1.Change incident status\n2.Not necessary\nEnter choice:");
                        int ch = Integer.parseInt(br.readLine());
                        switch (ch) {
                        case 1:
```

CHAPTER 6

6.1 RESULTS

Getting enter the main function by entering the id,no of patients to register etc.

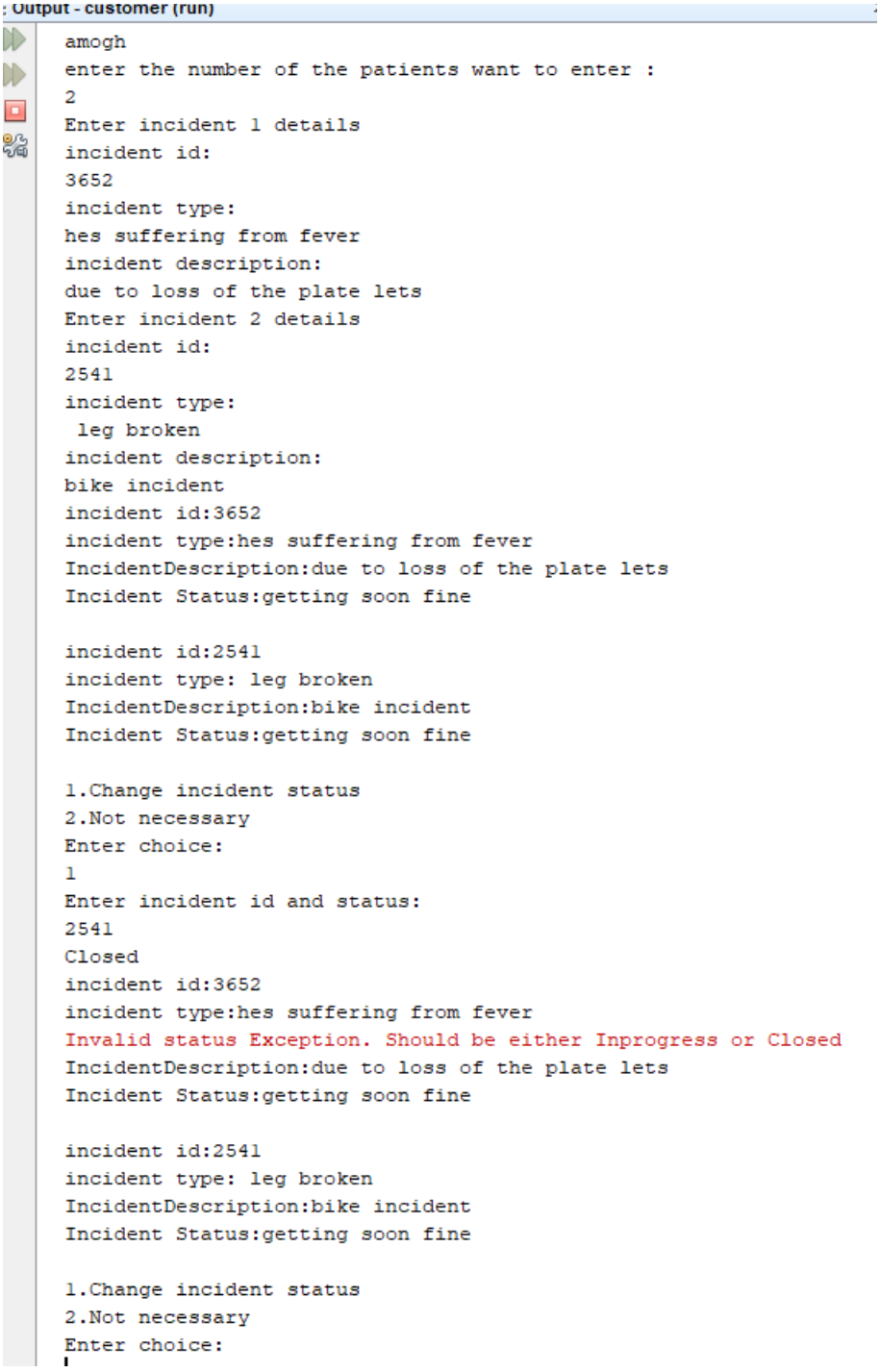


Fig 6.1 patient details

```
1.Change incident status
2.Not necessary
Enter choice:
1
Enter incident id and status:
3652
Closed
Incident not found
incident id:3652
incident type:hes suffering from fever
IncidentDescription:due to loss of the plate lets
Incident Status:Closed

incident id:2541
incident type: leg broken
IncidentDescription:bike incident
Incident Status:getting soon fine

1.Change incident status
2.Not necessary
Enter choice:
```

Fig 6.2 changing the status

If we want to change the status we can take a 1st option ,else can exit with the help of 2nd option

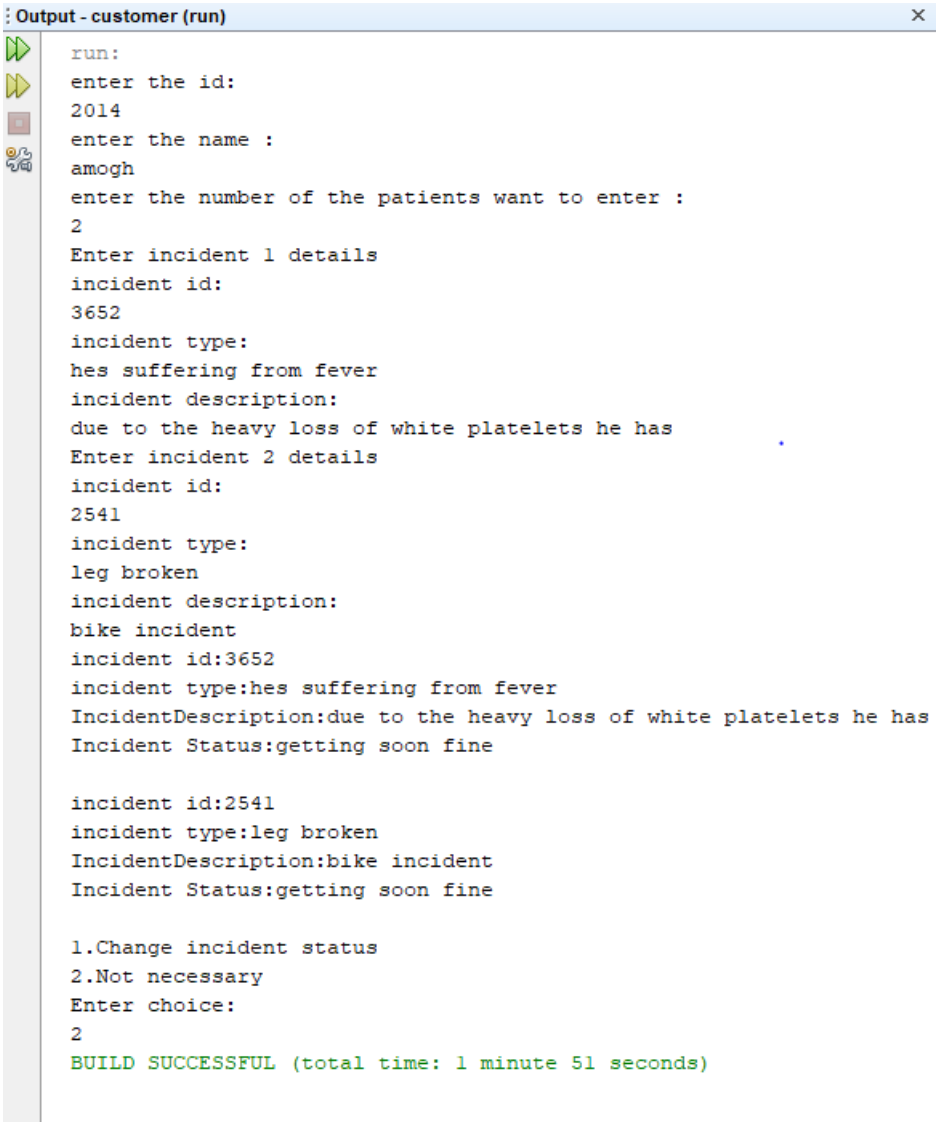


Fig 6.3 Complete details of the two patients

```
run:
enter the id:
2014
enter the name :
rajesh
enter the number of the patients want to enter :
1
Enter incident 1 details
incident id:
3652
incident type:
hes
incident description:
hes fine
incident id:3652
incident type:hes
IncidentDescription:hes fine
Incident Status:getting soon fine

1.Change incident status
2.Not necessary
Enter choice:
2
BUILD SUCCESSFUL (total time: 24 seconds)
```

Fig 6.4 Here complete details of single patient .

CHAPTER 7

CONCLUSION

So we are trying to create the project in the java here rather than in the python which is the advanced one .here the patient can register with the help of the admin the data can be entered securely and can change the status ,and the status can be updated .so to help the people of the unknow person, to get much usage of getting technical knowledge of solving the problems can make this type of project to think of it and get the knowledge.

REFERENCES

[1] https://www.programiz.com

[2] https://www.geeksforgeeks.org

[3] https://www.javatpoint.com

[4] https://anzeljg.github.io