

RV College of Engineering®
(Autonomous Institution Affiliated to VTU, Belagavi)
BENGALURU – 560 059

**DEPARTMENT OF
INFORMATION SCIENCE & ENGINEERING**



**OBJECT ORIENTED PROGRAMMING
USING JAVA LAB**

(18CS45)
Common to IS & CS
IV SEMESTER - B.E.
LABORATORY RECORD
2021

USN: 1RV19IS046

NAME: Sahil Sharma

Academic year: 2020– 21

RV College of Engineering®

**(Autonomous Institution Affiliated to VTU, Belagavi)
Bengaluru – 560 059**

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING



Laboratory Certificate

This is to certify that **Mr. Sahil Sharma** with USN **1RV19IS046** of 4th Semester has satisfactorily completed the course of experiments in **Object Oriented Programming using Java Laboratory [18CS45]** prescribed by the Department during the year 2020-21.

Marks	
Maximum	Obtained
50	

Signature of the Student

Signature of the Staff in-charge

Head of the Department

Date:

RV College of Engineering[®], Bengaluru- 560

059

(Autonomous Institution Affiliated to VTU, Belagavi)

Department of Information Science and Engineering



Vision

To be the hub for innovation in Information Science & Engineering through Teaching, Research, Development and Consultancy; thus make the department a well-known resource centre in advanced, sustainable and inclusive technology.

Mission

ISE1: To enable students to become responsible professionals, strong in fundamentals of Information Science and engineering through experiential learning.

ISE2: To bring research and entrepreneurship into class rooms by continuous design of innovative solutions through research publications and dynamic development oriented curriculum.

ISE3: To facilitate continuous interaction with the outside world through student internship, faculty consultancy, workshops, faculty development programmes, industry collaboration and association with the professional societies.

ISE4: To create a new generation of entrepreneurial problem solvers for a sustainable future through green technology with an emphasis on ethical practices, inclusive societal concerns and environment.

ISE5: To promote team work through inter-disciplinary projects, co-curricular and social activities.

Programme Educational Objectives (PEO's)

PEO1: To provide adaptive and agile skills in Information Science and Engineering needed for professional excellence / higher studies /Employment, in rapidly changing scenarios.

PEO2: To provide students a strong foundation in basic sciences and its applications to technology.

PEO3: To train students in core areas of Information science and Engineering, enabling them to analyse, design and create products and solutions for the real world problems, in the context of changing technical, financial, managerial and legal issues.

PEO4: To inculcate leadership, professional ethics, effective communication, team spirit, multi- disciplinary approach in students and an ability to relate Information Engineering issues to social and environmental context.

PEO5: To motivate students to develop passion for lifelong learning, innovation, career growth and professional achievement.

Programme Outcomes(PO's)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems

PO2: Problem analysis: Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modeling to complex engineering activities, with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess Societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes(PSOs)

PSO	Description
PSO1	Recognize and appreciate the principles of theoretical foundations, data organization, data communication, security and data analytical methods in the evolving technology
PSO2	Learn the applicability of various system software for the development of quality products in solving real-world problems with a focus on performance optimization
PSO3	Demonstrate the ability of team work, professional ethics, communication and documentation skills in designing and implementation of software products using the SDLC principles

Course Outcomes (COs)

At the end of the course students should be able to:

CO1: Explore the fundamentals of Object-oriented concepts and apply features of object-oriented programming of Java to solve real world problems.

CO2: Design Classes and establish relationship among Classes for various applications from problem definition.

CO3: Analyze and implement reliable object-oriented applications using Java features such as Exception Handling, Multithreaded Programming, Lambda Expressions, Collection framework, Strings, JavaFX GUI Programming.

CO4: Design and develop real world applications using Object Oriented concepts and Java programming.

Mapping of Course Outcomes with Program Outcomes

Course Outcomes	Program Outcomes													
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	M	L	L	L	-	-	-	-	-	-	-	L	-	L
CO2	M	M	H	L	M	L	L	L	L	L	-	L	M	-
CO3	L	H	M	M	H	-	-	L	M	M	-	L	M	H
CO4	L	M	H	M	H	L	L	L	H	M	L	M	M	H

Do's and Don'ts in the Laboratory

Do's.....

- Come Prepared to the lab with the programs.
- Use the computers for academic purposes only.
- Follow the lab exercise cycles as per the instructions given by the department.
- Keep the chairs back to their position before you leave.
- Handle the computer with care.
- Keep your lab clean.

Don'ts.....

- Coming late to the lab and leaving the lab early.
- Move around in the lab during the lab session.
- Download or install any software onto the computers.
- Tamper system files or Try to access the server.
- Write Records in lab.
- Change the system assigned to you without the notice of lab staff.
- Carrying CD's, Pen Drives and other storage devices into lab.
- Using others login id's.

Rubrics for OOP using Java Lab

Each program (Part-A) is evaluated for 10 marks.

Lab Write-up and Execution rubrics (Max: 6 marks)						
Sl no	Criteria	Measuring methods	Excellent	Good	Poor	CO
1	Understanding of problem and requirements (2 Marks)	Observations	Student exhibits thorough understanding of program requirements and applies object oriented programming concepts with Java. (1.5M - 2M)	Student has sufficient understanding of program requirements and applies object oriented programming concepts with Java. (0.5M - 1M)	Student does not have clear understanding of program requirements and is unable to apply object oriented programming concepts with Java. (0M)	CO1
2	Execution (2Marks)	Observations	Student demonstrates the execution of the program with optimized code with all the necessary conditions and test cases handled. (1.5M - 2M)	Student demonstrates the execution of the program without optimization of the code and handles only few test cases. (0.5M - 1M)	Student has not executed the program. (0 M)	CO3,CO4
3	Results and Documentation (2Marks)	Observations	Documentation with appropriate comments and output is covered in data sheets and manual. (1.5M - 2M)	Documentation with only few comments and only few output cases is covered in data sheets and manual. (0.5M - 1M)	Documentation with no comments and no output cases is covered in data sheets and manual. (0 M)	CO4
Viva Voce rubrics (Max: 4 marks)						
1	Conceptual Understanding (2 Marks)	Viva Voce	Explains object oriented programming concepts with Java and related concepts involved. (1.5 M-2M)	Adequately explains the object oriented programming concepts with Java and related concepts involved. (0.5M-1M)	Unable to explain the concepts. (0M)	CO1
2	Use of appropriate Design Techniques (1 Mark)	Viva Voce	Insightful explanation of appropriate design techniques for the given problem to derive solution. (1 M)	Sufficiently explains the use of appropriate design techniques for the given problem to derive solution. (0.5 M)	Unable to explain the design techniques for the given problem. (0 M)	CO2,CO3
3	Communication of Concepts (1 Mark)	Viva Voce	Communicates the concept used in problem solving well.	Sufficiently communicates the concepts used in problem solving.	Unable to communicate the concepts used in problem.	CO1,CO2

			(1 M)	(0.5 M)	(0 M)	
--	--	--	-------	---------	-------	--

Particulars of the Experiments Performed

CONTENTS

Sl. No	Date	Experiments	Page No.	Marks obtained	Signature of Staff
		PART A			
		Classes and Objects			
1		Create a Java class called Complex with the following details as member variables within it. (i) Real (ii) Imaginary Develop a Java program to perform addition and subtraction of two complex numbers by using the method add() and subtract() respectively, by passing object as parameter and display result using method display(). Initialize the real and imaginary values of the complex number using parameterized constructor. Also demonstrate overloading constructors and methods.			
2		Design an Address class with member variables Street num, city, state and country and appropriate constructor. Design a Student class with constructor (Student (String USN, String Name, Address addr)), College class with constructor (College (String Name, Address addr)) and Employee class with constructor (Employee (String EmpID, String Name, Address addr)). Write a Java program to create 'n' Student objects, College Objects and Employee objects and print the student, college and employee addresses respectively and demonstrate passing of object as a parameter to the constructor.			
		Inheritance and polymorphism			
3		Design a base class Circle with member variables (radius and color) of type double, methods (getRadius(), getArea()) and constructors (Circle(radius), Circle(radius, color)). Derive subclass called Cylinder from the superclass Circle with member variable (height) of type double, public methods (getHeight(), getVolume(), getArea()) and its constructors (Cylinder(height, radius), Cylinder(height, radius,color)). Create the two instances of cylinder and print similar cylinders if the area, volume and color of cylinders are same. Demonstrate the code reuse and polymorphism properties of Object oriented programming by inheriting the constructors and methods of the base class.			
		Package and Interface			
4		Create a class Thirdsem. Put this class into a package called CSE. Define a method Welcomemsg which prints a line "Welcome to CSE dept- 3rd sem young budding Engineers". Create a class Csedept. Put this class into a package called RVCE. Inherit the class Thirdsem in CSE package to Csedept class in RVCE package and call Welcomemsg method to display welcome message and also verify Public method Overriding, Private method overriding and default method overriding from different packages in java with the same program			

5		<p>Create two classes called Lion and Snake that implements all the methods defined in an interface Animal. Declare eat() method in Animal interface and display eating habits of that particular animal .Create an interface called Tired Animal. In Tired Animal interface add method definition to an existing interface by extending Animal interface to verify Extending Interface concept in java.</p> <p>Note: Lion and Snake implement the required eat() method and has some of its own methods and instance variables</p>			
		Exception Handling			
6		<p>Design and implement a Java program for the following requirements:</p> <p>An Exception class called Demonetization Exception which returns the statement that says “Deposit of Old currency of (Rs____) crosses Rs. 5,000 and cannot be Deposited”.</p> <p>A class called ‘Account’ that creates account with 500 Rs minimum balance with following methods.</p> <p>deposit(amount, currencyType) method to deposit amount. This class should handle “Demonetization Exception” and print the message defined in this Exception class. If a currency type is “OLD” and the amount is greater than 5,000 then throw the Demonetization Exception, otherwise update the balance.</p> <p>currBalance() method that displays balance amount in the account.</p> <p>withdraw(amount) method to withdraw amount and update the balance. Use proper control structure to check Balance should not go less than 500.</p> <p>A ‘Customer’ class that creates Account object and call the methods deposit(), withdraw() and currBalance() based on the user choice.</p>			
		Multithreading			
7		<p>Design and develop a Java program for the fruit market problem. The farmer will be able to produce different types of fruits (apple, orange, grape, and watermelon), and put them in the market to sell. The market has limited capacity and farmers have to stand in a queue if the capacity is exceeded to sell their fruits. Consumers can come to the market any time and purchase their desired fruits; and if the fruits they want to buy runs out, they are willing to wait until the supply of that kind is ready. Examine and formulate an approach to address this problem and implement the same using Java constructs for programming.</p>			
		Lambda Expression			
8		<p>Write the following methods that return a lambda expression performing a specified action:</p> <p>(i) PerformOperation isOdd(): The lambda expression must return true if a number is odd or false if it is even.</p> <p>(ii) PerformOperation isPrime(): The lambda expression must return true if a number is prime or false if it is composite.</p> <p>(iii) PerformOperation isPalindrome(): The lambda expression must return true if a number is a palindrome or false if it is not.</p>			

		Write a JAVA program using above lambda expressions to take 2 integers as input where the first integer specifies the condition to check for (case 1 for Odd/Even, case 2 for Prime/Composite, or case 3 for Palindrome). The second integer denotes the number to be checked.			
		Collections			
9		Write a Java program to create a new array list, add some colors (string) and perform the following operations: (i) Add elements of List to ArrayList (ii) Copy ArrayList to Array (iii) Reverse ArrayList content (iv) Get Sub list from an ArrayList. (v) To sort a given ArrayList (vi) Clone an ArrayList to another ArrayList			
		String Handling			
10		i) Write a Java program to find the penultimate (next to last) word of a sentence. ii) Write program to replace a string "python" with "java" and "java" with "python" in a given string. iii) Write a program that splits a string into a number of substrings with the help of string split() method and then prints the substrings			
		PART B			
		Develop standalone Java application with neat UI using JavaFX framework to demonstrate the important features of Object Oriented approach (Abstraction/Encapsulation/Data Hiding, Inheritance and Polymorphism) and also the important features of Java such as Interfaces, Packages, Inheritance, Exception Handling, Multithreaded Programming, Collection Framework, Lambda Expressions, Regular Expressions			

REDUCED RECORD MARKS = (TOTAL/100) * _____ = _____

LAB INTERNALS			
RECORD	Max - 40	Part A (Max - 30)	
		Part B (Max - 10)	
TEST	Max - 10		
TOTAL	Max – 50		
<i>Signature of the faculty</i>			

S.No.	Program Rubrics	Understanding of problem and requirements	Execution	Results and Documentation	Conceptual Understanding	Use of appropriate Design Techniques	Communication of Concepts	Total
	Marks	2 Marks	2 Marks	2 Marks	2 Marks	1 Mark	1 Mark	10 Marks
	CO Mapping	CO1	CO3, CO4	CO4	CO1	CO2, CO3	CO1, CO2	
1.	Program 1							
2.	Program 2							
3.	Program 3							
4.	Program 4							
5.	Program 5							
6.	Program 6							
7.	Program 7							
8.	Program 8							
9.	Program 9							
10.	Program 10							

PART A

Program 1

Classes and Objects

Create a Java class called **Complex** with the following details as member variables within it.

(i) Real (ii) Imaginary

Develop a Java program to perform addition and subtraction of two complex numbers by using the method `add()` and `subtract()` respectively, by passing object as parameter and display result using method `display()`. Initialize the real and imaginary values of the complex number using parameterized constructor. Also demonstrate overloading constructors and methods.

Implementation Steps

Step 1:

Create a class named **Complex** with two member variables: **Real** and **Imaginary**.

Step 2:

Define a parameterized constructor to initialize the real and imaginary values of the complex number. Also a default constructor (empty) to be defined. (Overloading of constructors)

Step 3:

Define method `add()` in class **Complex** to add two complex numbers, which takes two complex numbers as parameters and returns the result as a complex number.

Define method `subtract()` in class **Complex** to subtract two complex numbers, which takes two complex numbers as parameters and returns the result as a complex number.

Step 4:

Define method `display()` in **Complex** class to display the result of addition and subtraction in appropriate format.

Step 5:

Create a main class in which **Complex** class objects are instantiated and call the methods for complex numbers addition and subtraction. Demonstrate method overloading and constructor overloading.

CODE

```
class Complex{
    int real;
    int img;

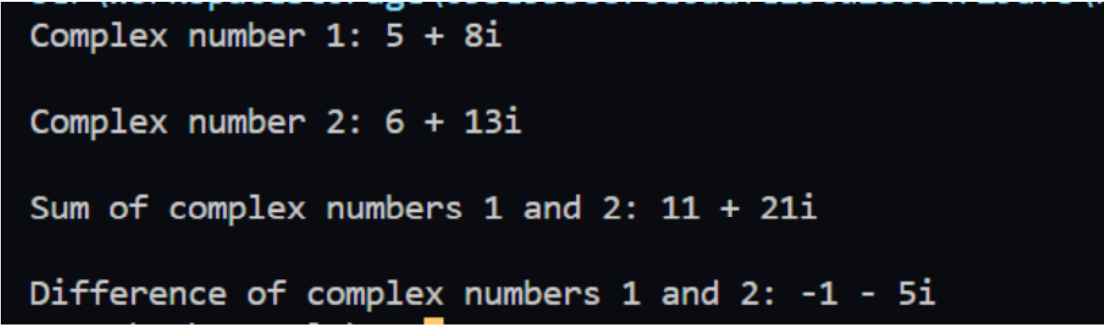
    Complex()
    {
        //
    }
    Complex(int r,int i)
    {
        real=r;
        img=i;
    }

    void add(Complex c1,Complex c2)
    {
        real=c1.real+c2.real;
        img=c1.img+c2.img;
    }
    void sub(Complex c1,Complex c2)
    {
        real=c1.real-c2.real;
        img=c1.img-c2.img;
    }
    void Display()
    {
        if(img>=0)
            System.out.println(real+" + "+img+"i");
        else{
            System.out.println(real + " - " + -1*img + "i");
        }
    }
};

public class prog1{
    public static void main(String args[])
    {
        Complex c1=new Complex(-3,2);
        Complex c2=new Complex(9,20);
        System.out.print("Complex number 1: ");
        c1.Display();
        System.out.print("\nComplex number 2: ");
        c2.Display();
        Complex c3=new Complex();
        c3.add(c1,c2);
        System.out.print("\nSum of complex numbers 1 and 2: ");
        c3.Display();
    }
}
```

```
        System.out.print("\nDifference of complex numbers 1 and 2: ");
        Complex c4=new Complex();
        c4.sub(c1, c2);
        c4.Display();
    }
}
```

OUTPUT:

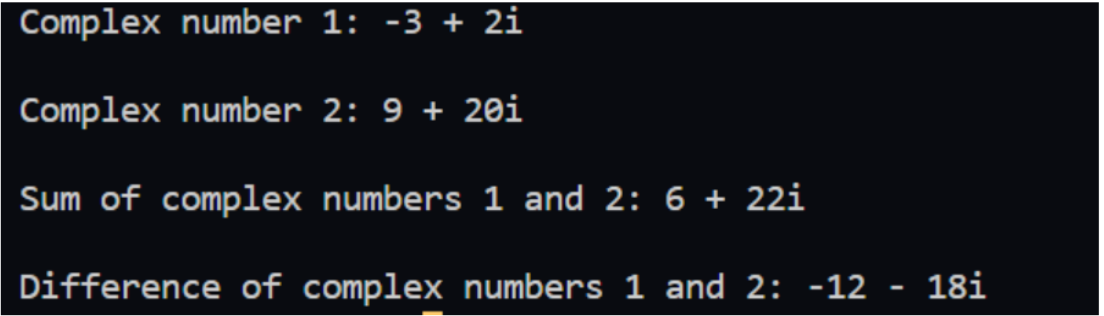


```
Complex number 1: 5 + 8i
```

```
Complex number 2: 6 + 13i
```

```
Sum of complex numbers 1 and 2: 11 + 21i
```

```
Difference of complex numbers 1 and 2: -1 - 5i
```



```
Complex number 1: -3 + 2i
```

```
Complex number 2: 9 + 20i
```

```
Sum of complex numbers 1 and 2: 6 + 22i
```

```
Difference of complex numbers 1 and 2: -12 - 18i
```


Program 2

Classes and Objects

Design an Address class with member variables Street num, city, state and country and appropriate constructor. Design a Student class with constructor (Student (String USN, String Name, Address addr)), College class with constructor (College (String Name, Address addr)) and Employee class with constructor (Employee (String EmpID, String Name, Address addr)). Write a Java program to create 'n' Student objects, College Objects and Employee objects and print the student, college and employee addresses respectively and demonstrate passing of object as a parameter to the constructor.

Implementation Steps

Step 1:

Create a class named Address with member variables: Street num, City, State and Country.

Define appropriate constructor to initialize the data members of class Address.

Step 2:

Create a class named Student with constructor (Student (String USN, String Name, Address addr)).

Step 3:

Create a class named College with constructor (College (String Name, Address addr))

Step 4:

Create a class named Employee with constructor (Employee (String EmpID, String Name, Address addr)).

Step 5:

Create a main class in which 'n' Student objects, College Objects and Employee objects are instantiated based on user input.

Print the student, college and employee addresses respectively and demonstrate passing of object as a parameter to the constructor.

```
{
    //
}

College(String n,Address a)
{
    name=n;
    addr=a;
}

void Display() {
    System.out.printf("%-15s%-15s%-15s%-15s%-
5d",name,addr.city,addr.state,addr.country,addr.street);
}
};

class Employee{
    String empId;
    String name;
    Address addr;

    Employee()
    {
        //
    }

    Employee(String e,String n,Address a)
    {
        empId=e;
        name=n;
        addr=a;
    }

    void Display() {
        System.out.printf("%-15s%-15s%-15s%-15s%-15s%-
5d",name,empId,addr.city,addr.state,addr.country,addr.street);
    }
};

public class prog2 {
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the number of objects: ");
        int n=sc.nextInt();
        int i=0;
        Student stu[]=new Student[n];
    }
}
```

```

College cg[]=new College[n];
Employee em[]=new Employee[n];
for(i=0;i<n;i++)
{
    System.out.println("\nEnter the Data for Student "+(i+1)+"\n");
;
    System.out.print("Student Name: ");
    sc.nextLine();
    String stname=sc.nextLine();
    System.out.print("Student USN: ");
    String stusn=sc.nextLine();
    String city,state,country;
    int street;
    System.out.print("Student city: ");
    city=sc.nextLine();
    System.out.print("Student state: ");
    state=sc.nextLine();
    System.out.print("Student country: ");
    country=sc.nextLine();
    System.out.print("Student street number: ");
    street=sc.nextInt();
    Address addr=new Address(street,city,state,country);
    stu[i]=new Student(stusn,stname,addr);
//
    System.out.println("\nEnter the Data for College " + (i + 1) +
"\n");
    System.out.print("College Name: ");
    sc.nextLine();
    String cname=sc.nextLine();
    System.out.print("College city: ");
    city = sc.nextLine();
    System.out.print("College state: ");
    state = sc.nextLine();
    System.out.print("College country: ");
    country = sc.nextLine();
    System.out.print("College street number: ");
    street = sc.nextInt();
    addr = new Address(street, city, state, country);
    cg[i] = new College(cname,addr);

    System.out.println("\nEnter the Data for Employee " + (i + 1)
+ "\n");
    System.out.print("Employee Name: ");
    sc.nextLine();
    String ename=sc.nextLine();
    System.out.print("Employee ID: ");
    String id=sc.nextLine();
    System.out.print("Employee city: ");

```

```

        city = sc.nextLine();
        System.out.print("Employee state: ");
        state = sc.nextLine();
        System.out.print("Employee country: ");
        country = sc.nextLine();
        System.out.print("Employee street number: ");
        street = sc.nextInt();
        addr = new Address(street, city, state, country);
        em[i] = new Employee(id,ename,addr);

    }
    System.out.println("\nData for students:");
    System.out.println("-----");
    System.out.println("-----");
    System.out.printf("%-15s%-15s%-15s%-15s%-15s%-15s", "NAME", "USN", "CITY", "STATE", "COUNTRY", "STREET NO.");
    System.out.println("\n-----");
    System.out.println("-----");
    for(i=0;i<n;i++)
    {
        stu[i].Display();
        System.out.println();
    }
    System.out.println("-----");
    System.out.println("-----");
    System.out.println("\nData for colleges:");
    System.out.println("-----");
    System.out.println("-----");
    System.out.printf("%-15s%-15s%-15s%-15s%-15s", "NAME", "CITY", "STATE", "COUNTRY", "STREET NO.");
    System.out.println("\n-----");
    System.out.println("-----");
    for (i = 0; i < n; i++) {
        cg[i].Display();
        System.out.println();
    }
    System.out.println("-----");
    System.out.println("-----");
    System.out.println("\nData for employees:");
    System.out.println("-----");
    System.out.println("-----");
    System.out.printf("%-15s%-15s%-15s%-15s%-15s", "NAME", "EMPID", "CITY", "STATE", "COUNTRY", "STREET NO.");
    System.out.println("\n-----");
    System.out.println("-----");
    for(i=0;i<n;i++)
    {
        em[i].Display();

```

```
        System.out.println();
    }
    System.out.println("-----");
    -----");
}
}
```

OUTPUT:

```
Enter the number of objects: 2

Enter the Data for Student 1

Student Name: Sahil Sharma
Student USN: 1RV19IS046
Student city: Jammu
Student state: J&K
Student country: INDIA
Student street number: 1

Enter the Data for College 1

College Name: RVCE
College city: Bangalore
College state: Karnataka
College country: INDIA
College street number: 1

Enter the Data for Employee 1

Employee Name: John
Employee ID: 1Rt23
Employee city: Delhi
Employee state: Delhi
Employee country: INDIA
Employee street number: 1

Enter the Data for Student 2
```

Enter the Data for Student 2

Student Name: Vivek Kumar
 Student USN: 1VK34
 Student city: Jammu
 Student state: J&K
 Student country: INDIA
 Student street number: 43

Enter the Data for College 2

College Name: SBICA
 College city: Delhi
 College state: Delhi
 College country: INDIA
 College street number: 343

Enter the Data for Employee 2

Employee Name: Martin
 Employee ID: 1yt56
 Employee city: Kolkata
 Employee state: West Bengal
 Employee country: INDIA
 Employee street number: 45

Data for students:

NAME	USN	CITY	STATE	COUNTRY	STREET NO.
Sahil Sharma	1RV19IS046	Jammu	J&K	INDIA	1
Vivek Kumar	1VK34	Jammu	J&K	INDIA	43

Data for colleges:

NAME	CITY	STATE	COUNTRY	STREET NO.
RVCE	Bangalore	Karnataka	INDIA	1
SBICA	Delhi	Delhi	INDIA	343

Data for employees:

NAME	EMPID	CITY	STATE	COUNTRY	STREET NO.
John	1Rt23	Delhi	Delhi	INDIA	1
Martin	1yt56	Kolkata	West Bengal	INDIA	45

Program 3

Inheritance and Polymorphism

Design a base class Circle with member variables (radius and color) of type double, methods (getRadius(), getArea()) and constructors (Circle(radius), Circle(radius, color)). Derive subclass called Cylinder from the superclass Circle with member variable (height) of type double, public methods (getHeight(), getVolume(), getArea()) and its constructors(Cylinder(height, radius), Cylinder(height, radius,color)). Create the two instances of cylinder and print similar cylinders if the area, volume and color of cylinders are same. Demonstrate the code reuse and polymorphism properties of Object oriented programming by inheriting the constructors and methods of the base class.

Implementation Steps

Step 1: Create a base class Circle with member variables: radius and color of type double.

Define member methods: getRadius(), getArea()

Define constructors: Circle(radius), Circle (radius, color)

Step 2:

Derive subclass called Cylinder from the superclass Circle with member variable: height of type double

Define member methods: getHeight(), getVolume(), getArea()

Define constructors: Cylinder (height, radius), Cylinder (height, radius,color).

Demonstrate the code reuse and polymorphism properties of Object oriented programming by inheriting the constructors and methods of the base class.

Step 3:

In the main class create the two instances of cylinder class and print similar cylinders if the area, volume and color of cylinders are same.

CODE

```
import java.util.*;

class Circle{
    double radius;
    String color;

    Circle()
    {
        //
    }
    Circle(double r)
    {
        radius=r;
    }

    Circle(double r,String c)
    {
        radius=r;
        color=c;
    }

    double getRadius()
    {
        return radius;
    }

    double getArea()
    {
        return 3.14*radius*radius;
    }
};

class Cylinder extends Circle{
    double height;

    Cylinder()
    {
        //
    }
    Cylinder(double h,double r)
    {
        super(r);
        height=h;
    }

    Cylinder(double h,double r,String c)
```

```
{
    super(r,c);
    height=h;
}

double getHeight()
{
    return height;
}

double getVolume()
{
    return super.getArea()*height;
}

double getArea()
{
    return 2*3.14*radius*(radius+height);
}
}

public class prog3 {
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter radius, height and color of cylinder 1: ");
;
        Cylinder c1=new Cylinder();
        c1.radius=sc.nextDouble();
        c1.height=sc.nextDouble();
        c1.color=sc.nextLine();
        System.out.print("\nEnter radius, height and color of cylinder 2: ");
;
        Cylinder c2 = new Cylinder();
        c2.radius = sc.nextDouble();
        c2.height = sc.nextDouble();
        c2.color = sc.nextLine();

        System.out.println("\nTotal surface area of cylinder 1= "+c1.getArea()+"\nVolume of cylinder 1= "+ c1.getVolume()+"\nColor of cylinder 1="+c1.color);
        System.out.println("\nTotal surface area of cylinder 2= "+c2.getArea()+"\nVolume of cylinder 2= "+ c2.getVolume()+"\nColor of cylinder 2="+c2.color);
        if(c1.getArea()==c2.getArea()&& c1.getVolume()==c2.getVolume()&& c1.color.equals(c2.color))
        {
            System.out.println("\nBoth the cylinders are similar");
        }
    }
}
```

```
    }  
    else{  
        System.out.println("\nBoth the cylinders are different");  
    }  
}  
}
```

OUTPUT:

```
Enter radius, height and color of cylinder 1: 2 3 red  
  
Enter radius, height and color of cylinder 2: 2 3 red  
  
Total surface area of cylinder 1= 62.800000000000004  
Volume of cylinder 1= 37.68  
Color of cylinder 1= red  
  
Total surface area of cylinder 2= 62.800000000000004  
Volume of cylinder 2= 37.68  
Color of cylinder 2= red  
  
Both the cylinders are similar
```

```
Enter radius, height and color of cylinder 1: 6.4 7.2 Black  
  
Enter radius, height and color of cylinder 2: 2.3 9.2 Black  
  
Total surface area of cylinder 1= 546.6112000000002  
Volume of cylinder 1= 926.0236800000001  
Color of cylinder 1= Black  
  
Total surface area of cylinder 2= 546.6112000000002  
Volume of cylinder 2= 926.0236800000001  
Color of cylinder 2= Black  
  
Both the cylinders are different
```

Program 4

Package and Interfaces

Create a class **Thirdsem**. Put this class into a package called **CSE**. Define a method **Welcomemsg** which prints a line “Welcome to CSE dept- 3rd sem young budding Engineers”.

Create a class **Cse_dept**. Put this class into a package called **RVCE**. Inherit the class **Thirdsem** in **CSE** package to **Cse_dept** class in **RVCE** package and call **Welcomemsg** method to display welcome message and also verify **Public method Overriding**, **Private method overriding** and **default method overriding** from different packages in java with the same program

Implementation Steps

Step 1:

Create a class named **Thirdsem** inside a package called **CSE**.

Define a member method **Welcomemsg** in class **Thirdsem** to print “Welcome to CSE dept- 3rd sem young budding Engineers”.

Step 2:

Create a class named **Csdept** inside a package called **RVCE**.

Step 3:

Inherit the class **Thirdsem** of **CSE** package in class **Csdept** of **RVCE** package and call **Welcomemsg** method to display welcome message.

Step 4:

Demonstrate and verify **Public method Overriding**, **Private method overriding** and **default method overriding** from different packages in java

CODE

- **Package ise**

```
package prog4.ise;

public class Thirdsem {
    public static void welcomemsg()
    {
        System.out.println("Welcome to ISE dept -
4th sem young budding Engineers.\n");
    }
    private void fun1()
    {
        System.out.println("This is a private method fun1() of class Third
sem in package ISE.");
    }

    public void fun2()
    {
        System.out.println("This is a public method fun2() of class Thirds
em in package ISE.");
    }
    protected void fun3()
    {
        System.out.println("This is a protected method fun3() of class Thi
rdsem in package ISE.");
    }
    void fun4()
    {
        System.out.println("This is a default method fun4() of class Third
sem in package ISE.");
    }
}
```

- **Package rvce**

```
package prog4.rvce;
import prog4.ise.*;

public class Iseddept extends Thirdsem {
    private void fun1()
    {
        System.out.println("This is a private method fun1() of class Isede
pt in package RVCE.");
    }
}
```

```
public void fun2()
{
    System.out.println("This is a public method fun2() of class Isedep
t in package RVCE.");
}
protected void fun3()
{
    System.out.println("This is a protected method fun3() of class Ise
dept in package RVCE.");
}
void fun4()
{
    System.out.println("This is a default method fun4() of class Isede
pt in package ISE.");
}

public static void main(String args[]) {
    welcomemsg();
    Iseddept obj1 = new Iseddept();
    Thirdsem obj2 = new Thirdsem();
    obj1.fun1();
    obj1.fun2();
    obj1.fun3();
    obj1.fun4();
    //    obj2.fun1();
    obj2.fun2();
    //    obj2.fun3();
    //    obj2.fun4();
}
}
```

OUTPUT:

```
Welcome to ISE dept - 4th sem young budding Engineers.

This is a private method fun1() of class Iseddept in package RVCE.
This is a public method fun2() of class Iseddept in package RVCE.
This is a protected method fun3() of class Iseddept in package RVCE.
This is a default method fun4() of class Iseddept in package ISE.
This is a public method fun2() of class Thirdsem in package ISE.
```

Program 5

Package and Interfaces

Create two classes called Lion and Snake that implements all the methods defined in an interface Animal. Declare eat() and sound() methods in Animal interface and display eating habits and sound made by that particular animal respectively. Create an interface called Tired Animal. In Tired Animal interface add method definition to an existing interface by extending Animal interface to verify Extending Interface concept in java.

Note: Lion and Snake implement the required eat() method and has some of its own methods and instance variables

Implementation Steps

Step 1:

Create an interface called Animal which specifies what kind of methods a class should have if it "implements" this interface.

Declare a method eat() – to display eating habit of a particular animal in the class which implements this interface.

Declare a method sound() – to display the sound made by the particular animal in the class which implements this interface.

Step 2:

Create two classes called Lion and Snake that implements Animal interface.

Lion and Snake classes should implement the required eat() and sound() method and can have some of its own methods and instance variables

Step 3:

Create an interface called Tired Animal by extending Animal interface.

Add necessary methods.

Demonstrate and verify extending interface concept in java.

CODE

```
interface Animal{
    void eat();
    void sound();
    void howManyLegs();
}

interface TiredAnimal extends Animal{
    void whenTired();
}

class Lion implements TiredAnimal{
    public void eat()
    {
        System.out.println("I am a Lion and i eat every animal that is scared of me.");
    }
    public void sound()
    {
        System.out.println("Lion: I make the roaring sound.");
    }
    public void howManyLegs()
    {
        System.out.println("Lion: I have four legs.");
    }
    public void whenTired()
    {
        System.out.println("Lion: When I am tired, I sleep in my den.");
    }
    public void looks()
    {
        System.out.println("Lion: I have a lot of hair around my neck.");
    }
}

class Snake implements TiredAnimal{
    public void eat()
    {
        System.out.println("I am a Snake and my favourite food is rats.");
    }
    public void sound()
    {
        System.out.println("Snake: I make the hissing sound.");
    }
    public void howManyLegs()
    {
        System.out.println("Snake: I have no legs, I crawl.");
    }
}
```



```
    }
    public void whenTired()
    {
        System.out.println("Snake: When I am tired, I sleep in a burrow.")
;
    }
    public void looks()
    {
        System.out.println("Snake: I have cool looking shapes on my body."
);
    }
}

public class prog5 {
    public static void main(String[] args)
    {
        Lion lion=new Lion();
        Snake snake=new Snake();
        System.out.println("\n\t\tMETHODS OF INTERFACE IMPLEMENTED BY LION
CLASS\n\t\t-----");

        lion.eat();
        lion.howManyLegs();
        lion.sound();
        lion.whenTired();
        lion.looks();

        System.out.println("\n\t\tMETHODS OF INTERFACE IMPLEMENTED BY SNAKE
CLASS\n\t\t-----");
        snake.eat();
        snake.howManyLegs();
        snake.sound();
        snake.whenTired();
        snake.looks();

    }
}
```

OUTPUT:

METHODS OF INTERFACE IMPLEMENTED BY LION CLASS

I am a Lion and i eat every animal that is scared of me.
Lion: I have four legs.
Lion: I make the roaring sound.
Lion: When I am tired, I sleep in my den.
Lion: I have a lot of hair around my neck.

METHODS OF INTERFACE IMPLEMENTED BY SNAKE CLASS

I am a Snake and my favourite food is rats.
Snake: I have no legs, I crawl.
Snake: I make the hissing sound.
Snake: When I am tired, I sleep in a burrow.
Snake: I have cool looking shapes on my body.

Program 6

Exception Handling

Design and implement a Java program for the following requirements:

- a) An Exception class called **Demonetization Exception** which returns the statement that says “Deposit of Old currency of (Rs____) crosses Rs. 5,000 and cannot be Deposited”.
- b) A class called ‘Account’ that creates account with 500 Rs minimum balance with following methods.
 - i. **deposit(amount, currencyType)** method to deposit amount. This class should handle “Demonetization Exception” and print the message defined in this Exception class. If a currency type is “OLD” and the amount is greater than 5,000 then throw the Demonetization Exception, otherwise update the balance.
 - ii. **currBalance()** method that displays balance amount in the account.
 - iii. **withdraw(amount)** method to withdraw amount and update the balance. Use proper control structure to check Balance should not go less than 500.
- c) A ‘Customer’ class that creates Account object and call the methods **deposit()**, **withdraw()** and **currBalance()** based on the user choice.

Implementation Steps

Step 1:

Create a user defined exception class named **DemonetizationException**

Step 2:

Create a class named **Account** with default constructor with balance amount = 500

Step 3: Provide three methods in Account class

- **deposit(amount, currencyType)** method to deposit amount. This class should handle “Demonetization Exception” and print the message defined in this Exception class. If a currency type is “OLD” and the amount is greater than 5,000 then throw the Demonetization Exception, otherwise update the balance.
- **withdraw(amount)** method to withdraw amount and update the balance. Use proper control structure to check Balance should not go less than 500.
- **currBalance()** method that displays balance amount in the account.

Step 4:

Create a ‘Customer’ class that creates Account object and call the methods **deposit()**, **withdraw()** and **currBalance()** based on the user choice.

CODE

```
import java.lang.*;
import java.util.*;

class DemonetizationException extends Exception{
    public DemonetizationException(String msg){
        super(msg);
    }
}

class Account{
    int balance;
    Account()
    {
        balance=500;
    }

    void deposit(int amount,String currencyType) throws DemonetizationException
    {
        if(currencyType.equals("OLD")&&amount>5000)
        {
            String msg="Deposit of old currency of (Rs. "+amount+" ) crosses Rs. 5000 and cannot be deposited.";
            throw new DemonetizationException(msg);
        }
        else{
            balance+=amount;
            System.out.println("Money successfully deposited in the bank");
        }
        currBalance();
    }

    void withdraw(int amount)
    {
        if(balance-amount<0){
            System.out.println("!!Not enough money!!");
        }
        else if(balance-amount<500)
        {
            System.out.println("The balance in the account goes below 500 on withdrawal of given amount.");
            currBalance();
        }
        else{
            balance-=amount;
            System.out.println("Transaction is successful.");
        }
    }
}
```

```
        currBalance();
    }
}
void currBalance()
{
    System.out.println("The current balance = "+balance);
}
}

public class prog6_Customer {
    public static void main(String args[]) throws DemonetizationException
    {

        Account member = new Account();
        Scanner sc=new Scanner(System.in);
        int ch,amt;
        String type;
        while(true)
        {
            System.out.println("\nThe basic operations available to a customer:");
            System.out.print("1. Deposit\n2. Withdrawal\n3. Current Balance\n4. Exit\n\nEnter your choice: ");
            ch=sc.nextInt();
            switch(ch)
            {
                case 1:{
                    System.out.println("Enter the currency type and the amount");
                    sc.nextLine();
                    type=sc.nextLine();
                    amt=sc.nextInt();
                    try{
                        member.deposit(amt,type);
                    }
                    catch(DemonetizationException e)
                    {
                        System.out.println(e);
                    }
                    break;
                }
                case 2:{
                    System.out.println("Enter the amount to be withdrawn:");
                    amt=sc.nextInt();
                    member.withdraw(amt);
                    break;
                }
            }
        }
    }
}
```

```
        case 3:{
            member.currBalance();
            break;
        }
        case 4:{
            System.exit(0);
        }
    }
}
}
```

OUTPUT:

The basic operations available to a customer:

1. Deposit
2. Withdrawal
3. Current Balance
4. Exit

Enter your choice: 1

Enter the currency type and the amount

NEW

10000

Money successfully deposited in the bank

The current balance = 10500

The basic operations available to a customer:

1. Deposit
2. Withdrawal
3. Current Balance
4. Exit

Enter your choice: 1

Enter the currency type and the amount

OLD

4000

Money successfully deposited in the bank

The current balance = 14500

```
The basic operations available to a customer:
1. Deposit
2. Withdrawal
3. Current Balance
4. Exit

Enter your choice: 1
Enter the currency type and the amount
OLD
6000
DemonetizationException: Deposit of old currency of (Rs. 6000) crosses Rs. 5000 and cannot be deposited.

The basic operations available to a customer:
1. Deposit
2. Withdrawal
3. Current Balance
4. Exit

Enter your choice: 2
Enter the amount to be withdrawn:
5000
Transaction is successfull.
The current balance = 9500

The basic operations available to a customer:
1. Deposit
2. Withdrawal
3. Current Balance
4. Exit

Enter your choice: 2
Enter the amount to be withdrawn:
9200
The balance in the account goes below 500 on withdrawal of given amount.
The current balance = 9500

The basic operations available to a customer:
1. Deposit
2. Withdrawal
3. Current Balance
4. Exit

Enter your choice: 3
The current balance = 9500

The basic operations available to a customer:
1. Deposit
2. Withdrawal
3. Current Balance
4. Exit

Enter your choice: 4
```

Program 7

Multithreading

Design and develop a Java program for the fruit market problem. The farmer will be able to produce different types of fruits (apple, orange, grape, and watermelon), and put them in the market to sell. The market has limited capacity and farmers have to stand in a queue if the capacity is exceeded to sell their fruits. Consumers can come to the market any time and purchase their desired fruits; and if the fruits they want to buy runs out, they are willing to wait until the supply of that kind is ready. Examine and formulate an approach to address this problem and implement the same using Java constructs for programming.

Implementation Steps

Step 1:

Create a Market class with a parameterized constructor named **public Market (int fruitsNumber)** to initialize the number of fruits and define an ArrayList<String> named Fruits.

Step 2:

Provide four methods in Market class to formulate the Producer-Consumer implementation in Java

synchronized boolean isFull(): returns TRUE if fruit basket is full else returns FALSE

synchronized boolean isEmpty(): returns TRUE if Fruits are not available else returns FALSE

synchronized void farmer(String fruit): if isFull() is TRUE the farmer has to wait or add the fruit in the Arraylist

synchronized String consumer(): if isEmpty() is TRUE the consumer has to wait or remove the fruit from Arraylist and notify the farmer.

Step 3:

Print Appropriate Output

CODE

```
import java.util.*;
import java.lang.*;

class CustomException extends Exception{
    CustomException(String str)
    {
        super(str);
    }
}

class Market{
    int fruitsNumber;
    Scanner sc;
    List<String> fruits;
    public Market(int size)
    {
        if(size<0)
        {
            throw new IllegalArgumentException("The given argument is not
valid!!");
        }
        else{
            fruitsNumber=size;
            fruits=new ArrayList<>();
            sc=new Scanner(System.in);
        }
    }
    public synchronized boolean isFull()
    {
        if(fruits.size()==fruitsNumber)
        {
            return true;
        }
        else{
            return false;
        }
    }
    public synchronized boolean isEmpty()
    {
        if(fruits.size()==0)
        {
            return true;
        }
        else{
            return false;
        }
    }
}
```

```
}

public void display()
{
    System.out.print("Fruits Basket: [");
    int i=0;
    for(i=0;i<fruits.size()-1;i++)
    {
        System.out.print(fruits.get(i)+", ");
    }
    if(fruits.size()==1||i!=0)
        System.out.print(fruits.get(i)+"]\n");
    else
        System.out.print("]\n");
}

public synchronized void farmer() throws InterruptedException {
    while (isFull()) {

        try {
            System.out.println("\n!!!Market is full have to wait for c
onsumer to consume!!!");
            wait();
            System.out.println("\n!!!Wait is over, now we can add more
fruits!!!");
        } catch (InterruptedException e) {
            System.out.println("Interruption occured!!");
        }
    }
    display();
    System.out.print("Enter the name of fruit d(or press 1 to exit the
program) : ");
    String fruit = sc.nextLine();
    if(fruit.equals("1"))
    {
        System.exit(0);
    }
    fruits.add(fruit);
    System.out.println("Fruit " + fruit + " is successfully added to t
he basket\n");
    notifyAll();
    Thread.sleep(500);
}

public synchronized void consumer() throws InterruptedException {
    while (fruits.isEmpty()) {
        try {
            wait();
        }
    }
}
```

```
        } catch (InterruptedException e) {
            System.out.println("Interruption occurred!!");
        }
    }
    display();
    System.out.println("\nThe consumer consumed the fruit " + fruits.remove(0) + " successfully..\n");
    notifyAll();
    Thread.sleep(500);
}
}
```

```
class Producer extends Thread{
    Market obj;
    Producer(Market o)
    {
        obj=o;
    }
    public void run()
    {
        try {
            while(true)
            {
                obj.farmer();
            }

        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```
class Consumer extends Thread{
    Market obj;
    Consumer(Market o)
    {
        obj=o;
    }
    public void run()
    {
        try {
            while(true)
            {
                obj.consumer();
            }

        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```
    }  
  }  
}  
  
public class prog7 {  
    public static void main(String args[]) throws InterruptedException {  
        Scanner sc=new Scanner(System.in);  
        Market obj=new Market(3);  
        Producer p=new Producer(obj);  
        Consumer c=new Consumer(obj);  
        p.start();  
        c.start();  
    }  
}
```

OUTPUT:

```
Fruits Basket: []  
Enter the name of fruit d(or press 1 to exit the program) : apple  
Fruit apple is successfully added to the basket  
  
Fruits Basket: [apple]  
Enter the name of fruit d(or press 1 to exit the program) : orange  
Fruit orange is successfully added to the basket  
  
Fruits Basket: [apple, orange]  
Enter the name of fruit d(or press 1 to exit the program) : mango  
Fruit mango is successfully added to the basket  
  
!!!Market is full have to wait for consumer to consume!!!  
Fruits Basket: [apple, orange, mango]  
  
The consumer consumed the fruit apple successfully..  
  
Fruits Basket: [orange, mango]  
  
The consumer consumed the fruit orange successfully..  
  
Fruits Basket: [mango]  
  
The consumer consumed the fruit mango successfully..  
  
!!!Wait is over, now we can add more fruits!!!  
Fruits Basket: []  
Enter the name of fruit d(or press 1 to exit the program) : 1
```

Program 8

Lambda Expressions

Write the following methods that return a lambda expression performing a specified action:

- i. **Perform Operation isOdd():** The lambda expression must return true if a number is odd or false if it is even.
- ii. **Perform Operation isPrime():** The lambda expression must return true if a number is prime or false if it is composite.
- iii. **Perform Operation isPalindrome():** The lambda expression must return true if a number is a palindrome or false if it is not.

Write a JAVA program using above lambda expressions to take 2 integers as input where the first integer specifies the condition to check for (case 1 for Odd/Even, case 2 for Prime/Composite, or case 3 for Palindrome). The second integer denotes the number to be checked.

Implementation Steps

Step 1:

Create a functional interface performOperation

Step 2:

Provide the implementation of following methods that return a lambda expression performing a specified action:

- performOperation checkEvenOdd(): The lambda expression must return if a number is even or if it is odd. checkPrime()
- performOperation checkPrime(): The lambda expression must return if a number is prime or if it is composite.
- performOperation checkPalindrome(): The lambda expression must return if a number is a palindrome or if it is not.

Step 3:

Get 2 integers as input from user where first integer specifies one of the three above operations to be performed and the second integer specifies the number to be checked

CODE

```
import java.util.*;
import java.lang.*;

interface performOperation{
    boolean func(int n);
}

public class prog8 {
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        performOperation isOdd=(n)->{
            if(n%2==0)
            {
                return false;
            }
            else{
                return true;
            }
        };

        performOperation isPrime=(n)->{
            for(int i=2;i<=Math.sqrt(n);i++)
            {
                if(n%i==0)
                {
                    return false;
                }
            }
            return true;
        };

        performOperation isPalindrome=(n)->{
            String num=Integer.toString(n);
            int sz=num.length()-1;
            for(int i=0;i<sz/2;i++)
            {
                if(num.charAt(i)!=num.charAt(sz-i))
                {
                    return false;
                }
            }
            return true;
        };
        int ch,num;
```

```
        System.out.print("\n1. Check if number is even or odd.\n2. Check if number is a prime.\n3. Check if number is a palindrome.\n4. Exit\n\nEnter your choice: ");
        ch=sc.nextInt();
        while(true)
        {
            switch (ch)
            {
                case 1:
                {
                    System.out.print("Enter the number: ");
                    num=sc.nextInt();
                    if(isOdd.func(num))
                    {
                        System.out.println("\nThe entered number is ODD.")
;
                    }
                    else{
                        System.out.println("\nThe entered number is EVEN."
);
                    }
                    break;
                }
                case 2:
                {
                    System.out.print("Enter the number: ");
                    num=sc.nextInt();
                    if(isPrime.func(num))
                    {
                        System.out.println("\nThe entered number is PRIME."
);
                    }
                    else{
                        System.out.println("\nThe entered number is NOT PR
IME.");
                    }
                    break;
                }
                case 3:
                {
                    System.out.print("Enter the number: ");
                    num=sc.nextInt();
                    if(isPalindrome.func(num))
                    {
                        System.out.println("\nThe entered number is a PALI
NDROME.");
                    }
                    else{
```

```
        System.out.println("\nThe entered number is NOT PA  
LINDROME.");  
    }  
    break;  
}  
default:  
{  
    System.exit(0);  
}  
}  
System.out.print("\n\n1. Check if number is even or odd.\n2.  
Check if number is a prime.\n3. Check if number is a palindrome.\n4. Exit  
\n\nEnter your choice: ");  
ch=sc.nextInt();  
}  
}  
}
```

OUTPUT:

```
1. Check if number is even or odd.  
2. Check if number is a prime.  
3. Check if number is a palindrome.  
4. Exit
```

```
Enter your choice: 1  
Enter the number: 464
```

```
The entered number is EVEN.
```

```
1. Check if number is even or odd.  
2. Check if number is a prime.  
3. Check if number is a palindrome.  
4. Exit
```

```
Enter your choice: 1  
Enter the number: 555
```

```
The entered number is ODD.
```


1. Check if number is even or odd.
2. Check if number is a prime.
3. Check if number is a palindrome.
4. Exit

Enter your choice: 2

Enter the number: 7

The entered number is PRIME.

1. Check if number is even or odd.
2. Check if number is a prime.
3. Check if number is a palindrome.
4. Exit

Enter your choice: 2

Enter the number: 10

The entered number is NOT PRIME.

1. Check if number is even or odd.
2. Check if number is a prime.
3. Check if number is a palindrome.
4. Exit

Enter your choice: 3
Enter the number: 1331

The entered number is a PALINDROME.

1. Check if number is even or odd.
2. Check if number is a prime.
3. Check if number is a palindrome.
4. Exit

Enter your choice: 3
Enter the number: 4543

The entered number is NOT PALINDROME.

1. Check if number is even or odd.
2. Check if number is a prime.
3. Check if number is a palindrome.
4. Exit

Enter your choice: 4_

Program 9

Collections

Write a Java program to create a new array list, add some colors (string) and perform the following operations:

- (i) Add elements of List to ArrayList**
- (ii) Copy ArrayList to Array**
- (iii) Reverse ArrayList content**
- (iv) Get Sub list from an ArrayList.**
- (v) To sort a given ArrayList**
- (vi) Clone an ArrayList to another ArrayList**

Implementation Steps

Step 1:

Create a class ArrayListOper class and implement methods to perform following ArrayList operations using the methods of ArrayList and Object class

- Use add() method of ArrayList to add the elements to the list
- Use toArray() method of ArrayList to convert the ArrayList to Array.
- Reverse the ArrayList by appending the elements of the list in reverse order and using get() method of the ArrayList
- Use subList(fromIndex, endIndex) method of the ArrayList to get the sub list from the ArrayList
- Use clone() method of ArrayList class to create the shallow copy of the existing ArrayList.

Step 2:

Print appropriate output

CODE

```

import java.lang.*;
import java.util.*;

public class prog9 {
    public static void main(String[] args) {
        ArrayList<String> str = new ArrayList<String>();
        System.out.println("\nArraylist before adding colors: " + str);
        str.add("Purple");
        str.add("Indigo");
        str.add("Pink");
        System.out.println("\nArraylist after adding colors: " + str);
        List<String> str1 = new ArrayList<String>();
        str1.add("orange");
        str1.add("violet");
        str1.add("yellow");
        str.addAll(str1);
        System.out.println("\nArray List after adding new list into it: "
+ str);
        System.out.print("\nCopying list " + str + " to an array ");
        String arr[] = new String[str.size()];
        arr = str.toArray(arr);
        System.out.print(" gives " );
        for(String s : arr) {
            System.out.print(s + " ");
        }
        System.out.println();
        System.out.print("\nReversing the array list gives ");
        Collections.reverse(str);
        System.out.println(str);
        List<String> str2 = str.subList(0,3);
        System.out.println("\nSublist of "+str+" from index 1 to 4 is "+str2);
        Collections.sort(str);
        System.out.println("\nSorted array list is "+str);
        List<String> str3 = new ArrayList<String>();
        str3 = (ArrayList)str.clone();
        System.out.println("\nCloned array list is "+str3);
    }
}

```

OUTPUT:

Arraylist before adding colors: []

Arraylist after adding colors: [Purple, Indigo, Pink]

Array List after adding new list into it: [Purple, Indigo, Pink, orange, violet, yellow]

Copying list [Purple, Indigo, Pink, orange, violet, yellow] to an array gives Purple Indigo Pink orange violet yellow

Reversing the array list gives [yellow, violet, orange, Pink, Indigo, Purple]

Sublist of [yellow, violet, orange, Pink, Indigo, Purple] from index 1 to 4 is [yellow, violet, orange]

Sorted array list is [Indigo, Pink, Purple, orange, violet, yellow]

Cloned array list is [Indigo, Pink, Purple, orange, violet, yellow]

Program 10

String Handling

Write a Java program to create a new array list, add some colors (string) and perform the following operations:

- i. Write a Java program to find the penultimate (next to last) word of a sentence.
- ii. Write program to replace a string "python" with "java" and "java" with "python" in a given string.
- iii. Write a program that splits a string into a number of substrings with the help of string split () method and then prints the substrings.

Implementation Steps

- i) Write a Java program to find the penultimate (next to last) word of a sentence

Step 1: Read a sentence from user using Scanner Class

Step 2: Use Split String method in Java to split the words separated by spaces and store the words in an array of Strings

Step 3: Print the last but one word in the array.

- ii) Write program to replace a string "python" with "java" and "java" with "python" in a given string.

Step 1: Use Replace String method in Java to replace all the occurrences of string "python" with "java" and "java" with "python" in a given string.

Step 2: Print Appropriate Output

- iii) Write a program that splits a string into a number of substrings with the help of string split () method and then prints the substrings.

Step 1: Use Split String method in Java to split the input string based on any special characters.

Step 2: Print Appropriate Output

CODE

```
import java.lang.*;
import java.util.*;

public class prog10 {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the sentence: ");
        String str = sc.nextLine();
        penultimate(str);
        replace(str);
        substr(str);
    }
    private static void penultimate(String str){
        String[] arr = str.split(" ");
        System.out.println("Words are: ");
        for(String a: arr)
            System.out.println(a+" ");

        if(arr.length>=2) {
            System.out.println("Penultimate word: "+arr[arr.length-2]);
        }
        else {
            System.out.println("Penultimate word can't be printed(There is only one word in string)");
        }
    }

    private static void replace(String str) {
        System.out.println("Replacing \"Java\" with \"Python\" if found any");
        String s,s1;
        s = str.replaceAll("Java","Python");
        s1 = s.replaceAll("java","python");
        System.out.println("New string : " + s1);
    }

    private static void substr(String str) {
        Scanner sc = new Scanner(System.in);
        String ch="";
        System.out.println("Enter character based on which substrings to be made: ");
        ch = sc.nextLine();
        System.out.println("Breaking the sentence into substrings ");
        String[] sub = str.split(ch,0);
        for(String a: sub) {
            System.out.println(a+" ");
        }
    }
}
```

```
}  
}
```

OUTPUT:

```
Enter the sentence: This is a java program  
Words are:  
This  
is  
a  
java  
program  
Penultimate word: java  
Replacing "Java" with "Python" if found any  
New string : This is a python program  
Enter character based on which substrings to to be made:  
i  
Breaking the sentence into substrings  
Th  
s  
s a java program  
-  
  
Enter the sentence: hello how are you  
Words are:  
hello  
how  
are  
you  
Penultimate word: are  
Replacing "Java" with "Python" if found any  
New string : hello how are you  
Enter character based on which substrings to to be made:  
o  
Breaking the sentence into substrings  
hell  
h  
w are y  
u  
-
```


PART B

Open Ended Experiment

Student will design, develop and implement an application using the appropriate OOP concepts using Java:

Develop standalone Java application with neat UI using JavaFX framework to demonstrate the important features of Object Oriented approach (Abstraction/Encapsulation/Data Hiding, Inheritance and Polymorphism) and also the important features of Java such as Interfaces, Packages, Inheritance, Exception Handling, Multithreaded Programming, Collection Framework, Lambda Expressions, Regular Expressions

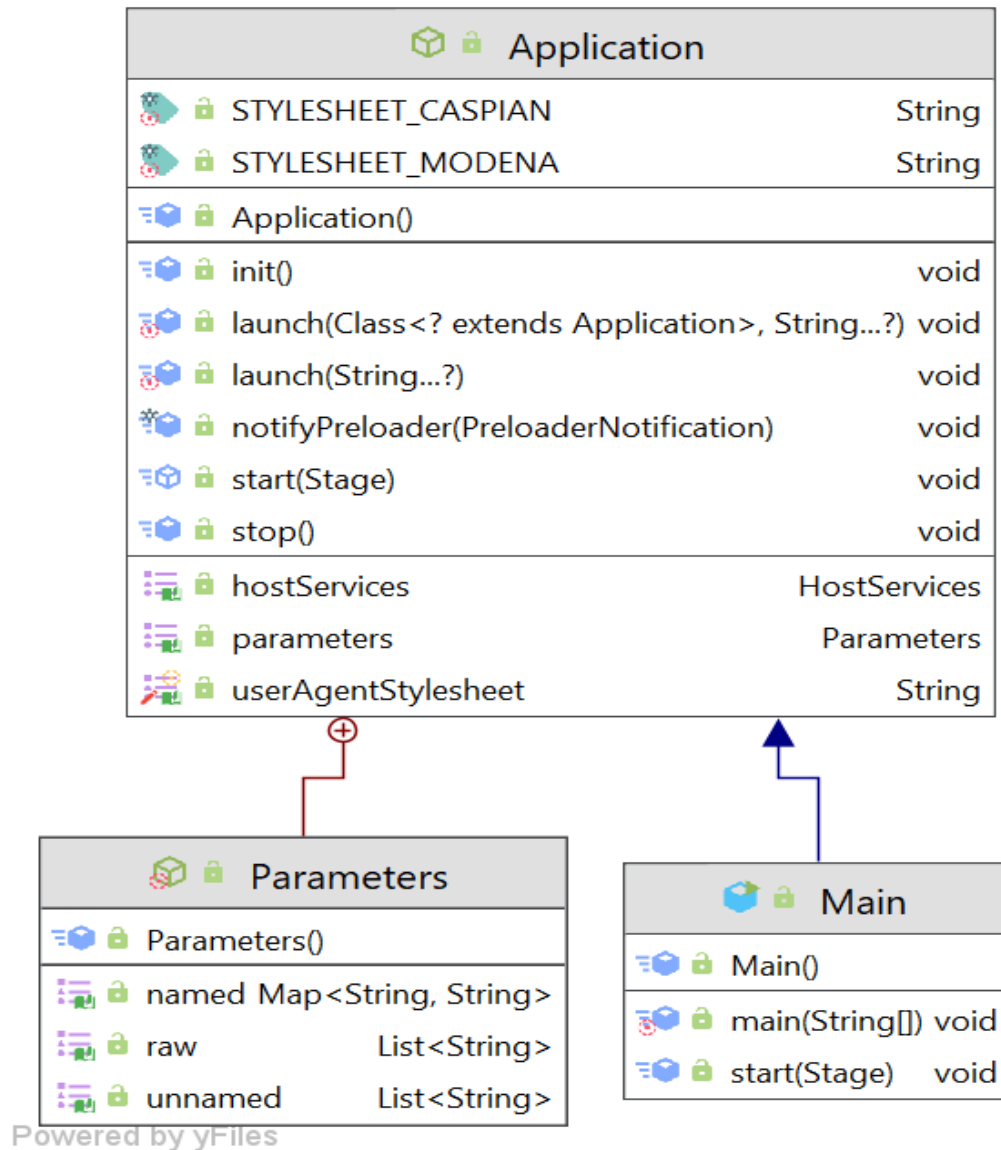
Students should be able to develop a Java application incorporating the lambda expression and collection framework with neat UI using JavaFX framework.

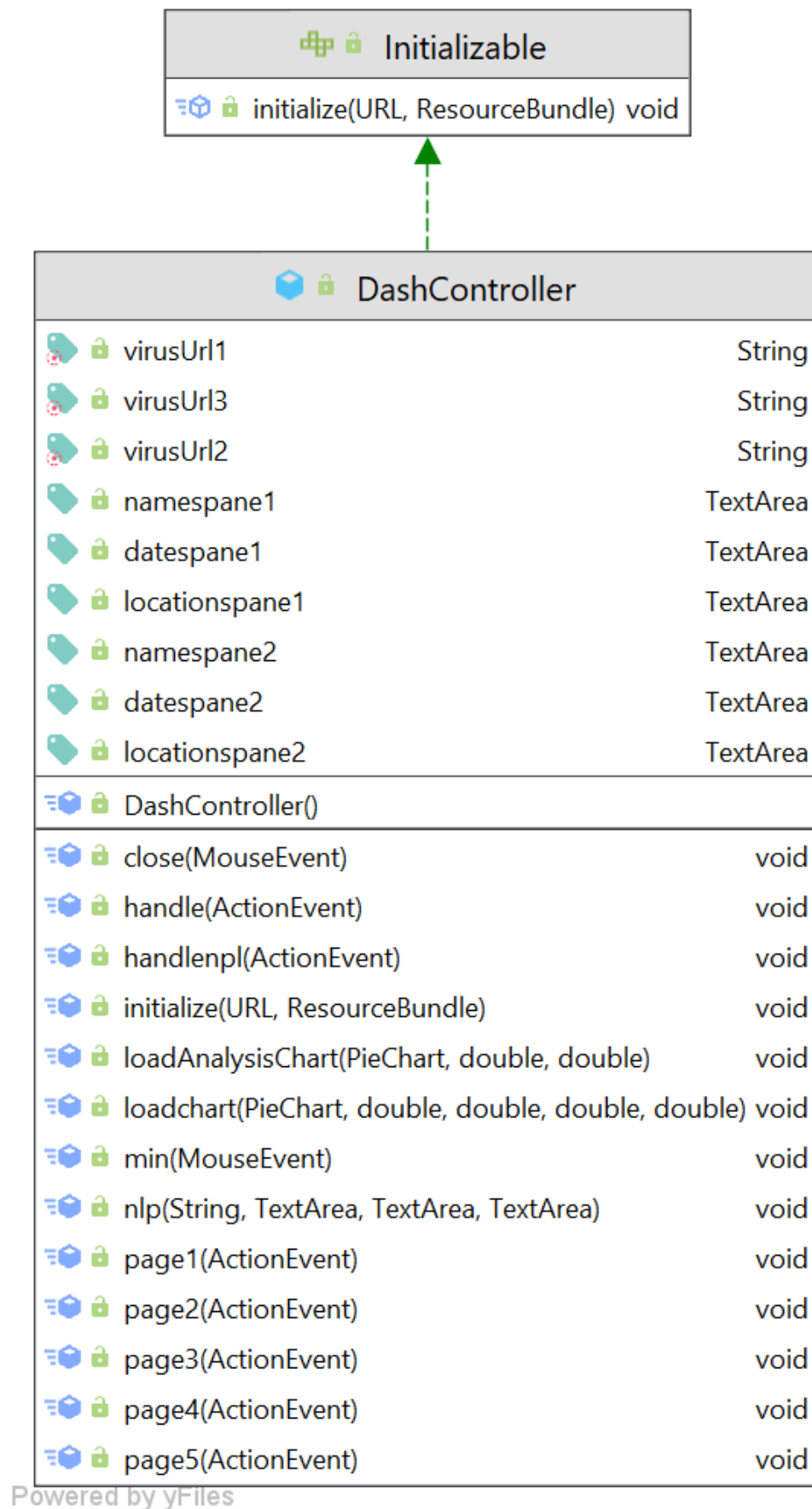
Example:

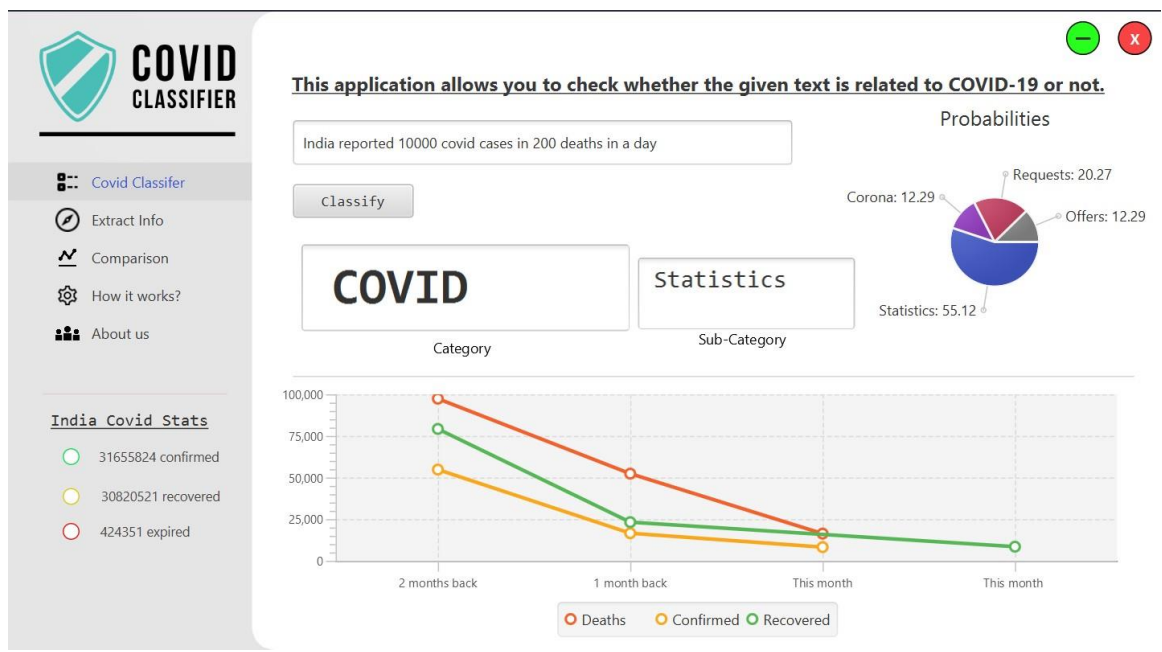
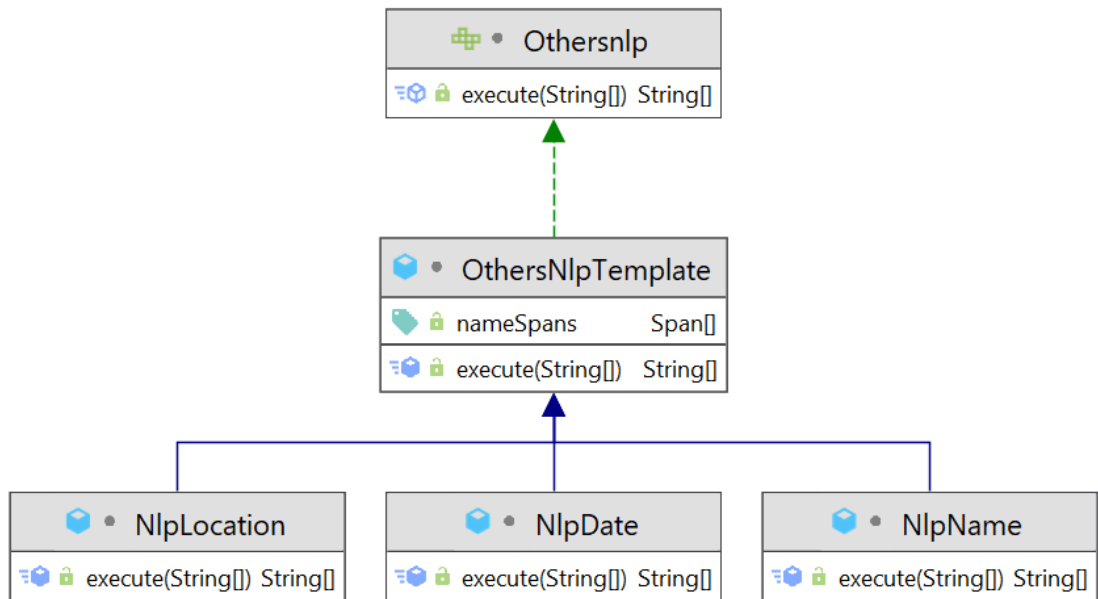
Student Information database using RFID tags


E-library Management with Face Recognition

Banking Database UI using Voice Recognition

OUTPUT:







- Covid Classifier
- Extract Info
- Comparison
- How it works?
- About us

India Covid Stats

- 31655824 confirmed
- 30820521 recovered
- 424351 expired

-
X

This application allows you to check whether the given text is related to COVID-19 or not.

John and Peter lives in London and were born in 2002

Classify

NON-COVID

Category

NONE

Sub-Category

Names


[0..1) person John
 [2..3) person Peter

Dates

[10..11) date 2002

Locations

[5..6) location London







- Covid Classifier
- Extract Info
- Comparison
- How it works?
- About us

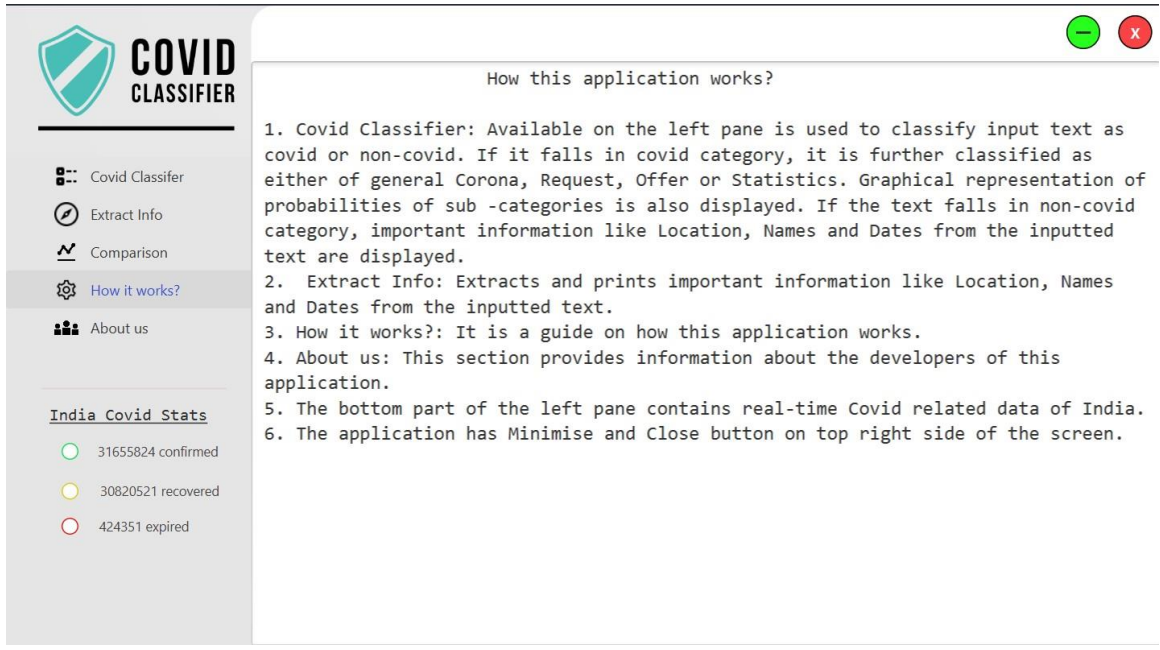
India Covid Stats

- 31655824 confirmed
- 30820521 recovered
- 424351 expired

-
X

-> Comparison of ML model using different algorithms

Max-Entropy	Perceptron
<p>Category</p>  <p>Corona: 50.0 Others: 49.99</p>	<p>Category</p>  <p>Corona: 88.07 Others: 11.92</p>
<p>Sub-Category</p>  <p>Corona: 25.38 Statistics: 24.87 Requests: 24.87 Offers: 24.87</p>	<p>Sub-Category</p>  <p>Corona: 48.58 Statistics: 21.11 Offers: 9.17 Requests: 21.11</p>



COVID CLASSIFIER

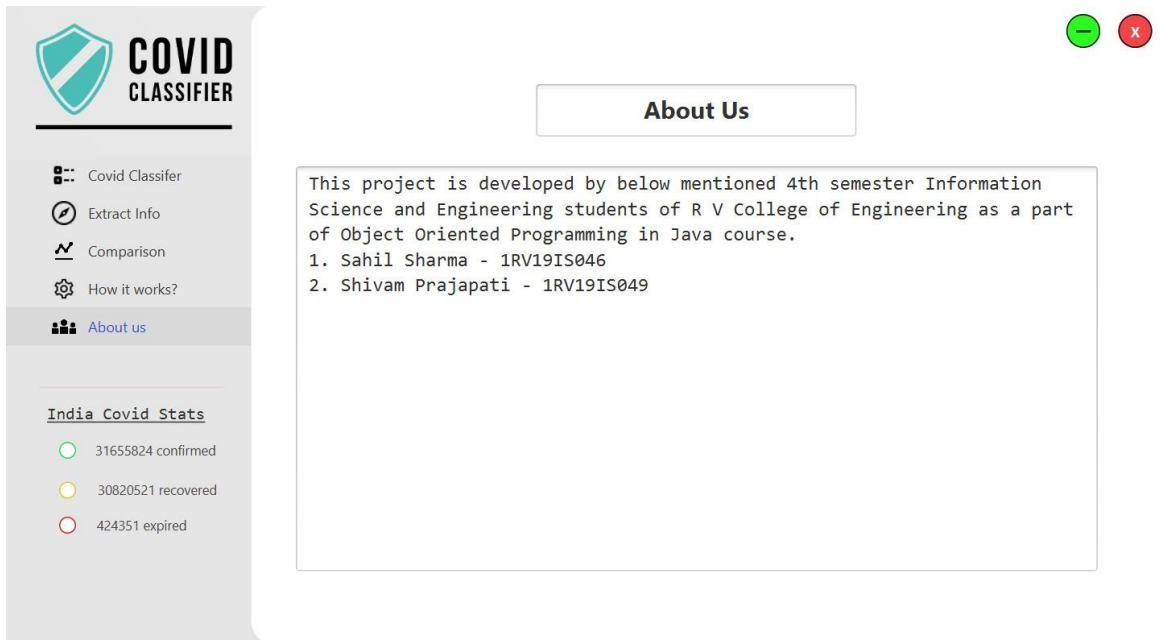
- Covid Classifier
- Extract Info
- Comparison
- How it works?**
- About us

India Covid Stats

- 31655824 confirmed
- 30820521 recovered
- 424351 expired

How this application works?

1. Covid Classifier: Available on the left pane is used to classify input text as covid or non-covid. If it falls in covid category, it is further classified as either of general Corona, Request, Offer or Statistics. Graphical representation of probabilities of sub -categories is also displayed. If the text falls in non-covid category, important information like Location, Names and Dates from the inputted text are displayed.
2. Extract Info: Extracts and prints important information like Location, Names and Dates from the inputted text.
3. How it works?: It is a guide on how this application works.
4. About us: This section provides information about the developers of this application.
5. The bottom part of the left pane contains real-time Covid related data of India.
6. The application has Minimise and Close button on top right side of the screen.



COVID CLASSIFIER

- Covid Classifier
- Extract Info
- Comparison
- How it works?
- About us**

India Covid Stats

- 31655824 confirmed
- 30820521 recovered
- 424351 expired

About Us

This project is developed by below mentioned 4th semester Information Science and Engineering students of R V College of Engineering as a part of Object Oriented Programming in Java course.

1. Sahil Sharma - 1RV19IS046
2. Shivam Prajapati - 1RV19IS049

Write-up and Execution (Max: 10 marks)						
Sl no	Criteria	Measuring methods	Excellent	Good	Poor	Score
1	Problem Formulation and Requirement Specification (2 Marks) CO1	Observations	Student exhibits thorough understanding of program requirements and applies object oriented programming concepts with Java. (1.5M - 2M)	Student has sufficient understanding of program requirements and applies object oriented programming concepts with Java. (0.5M - 1M)	Student does not have clear understanding of program requirements and is unable to apply object oriented programming concepts with Java. (0M)	
2	Execution/Demonstration (6 Marks) CO3,CO4	Observations Lambda expression : 2 Marks	Student demonstrates the execution of Lambda Expression in the program with optimized code with all the necessary conditions and test cases handled. (1.5M - 2M)	Student demonstrates the execution of Lambda Expression in the program without optimization of the code and handles only few test cases. (0.5M –1M)	Student has not demonstrated Lambda expression in the program. (0 M)	
		Observations Collection framework : 2 Marks	Student demonstrates the execution of Collection framework in the program with optimized code with all the necessary conditions and test cases handled. (1.5M - 2M)	Student demonstrates the execution of Collection framework in the program without optimization of the code and handles only few test cases. (0.5M - 1M)	Student has not demonstrated Collection framework in the program. (0 M)	
		Observations JavaFX: 2 Marks	Student demonstrates the UI developed using JavaFX in the program effectively. (1.5M - 2M)	Student demonstrates the UI developed using JavaFX in the program moderately. (0.5M –1M)	Student has not demonstrated UI using JavaFX. (0 M)	
3	Results and Documentation, Report (2 Marks) CO4	Observations	Documentation with appropriate comments and output is covered in data sheets and manual. (1.5M - 2M)	Documentation with only few comments and only few output cases is covered in data sheets and manual. (0.5M –1M)	Documentation with no comments and no output cases is covered in data sheets and manual. (0 M)	
Total Marks						
Faculty Signature						

