



## LABORATORY MANUAL

**B.Tech. Semester- IV**

**OBJECT ORIENTED PROGRAMMING LAB**

**Subject code: LC-CSE-256G**

**Prepared by:**

Dr Manoj Kumar

**Checked by:**

Dr Ritu Pahwa

**Approved by:**

Name : Prof. (Dr.) Isha Malhotra

**Sign.: .....**

**Sign.: .....**

**Sign.: .....**

**DEPARTMENT OF CSE(AI&ML)  
DRONACHARYA COLLEGE OF ENGINEERING  
KHENTAWAS, FARRUKH NAGAR, GURUGRAM (HARYANA)**

## **Table of Contents**

1. Vision and Mission of the Institute
2. Vision and Mission of the Department
3. Programme Educational Objectives (PEOs)
4. Programme Outcomes (POs)
5. Programme Specific Outcomes (PSOs)
6. University Syllabus
7. Course Outcomes (COs)
8. CO- PO and CO-PSO mapping
9. Course Overview
10. List of Experiments
11. DOs and DON'Ts
12. General Safety Precautions
13. Guidelines for students for report preparation
14. Lab assessment criteria
15. Lab Experiments

## **Vision and Mission of the Institute**

**Vision:**

“To impart Quality Education, to give an enviable growth to seekers of learning, to groom them as World Class Engineers and managers competent to match the expending expectations of the Corporate World has been ever enlarging vision extending to new horizons of Dronacharya College of Engineering”

**Mission:**

- M1:** To prepare students for full and ethical participation in a diverse society and encourage lifelong learning by following the principle of ‘Shiksha evam Sahayata’ i.e., Education & Help.
- M2:** To impart high-quality education, knowledge and technology through rigorous academic programs, cutting-edge research, & Industry collaborations, with a focus on producing engineers& managers who are socially responsible, globally aware, & equipped to address complex challenges.
- M3:** Educate students in the best practices of the field as well as integrate the latest research into the academics.
- M4:** Provide quality learning experiences through effective classroom practices, innovative teaching practices and opportunities for meaningful interactions between students and faculty.
- M5:** To devise and implement programmes of education in technology that are relevant to the changing needs of society, in terms of breadth of diversity and depth of specialization.

## **Vision and Mission of the Department**

**Vision:**

To cultivate skills and make proficient engineers cum trainers in the domain of Artificial Intelligence & Machine Learning for exceptional contributions to the society.

**Mission:**

- M1:** To impart intense training and learning to generate knowledge through the state-of-the-art concepts and technologies in Artificial Intelligence and Machine Learning.
- M2:** To establish centres of excellence by collaborating with the leading industries to exhilarate innovative research and development in AIML and its allied technology.
- M3:** To inculcate regenerative self-learning abilities, team spirit, and professional ethics among the students for noble cause.

## **Programme Educational Objectives (PEOs)**

**PEO1- ANALYTICAL SKILLS:**

Using a solid foundation in mathematical, scientific, engineering, and current computing principles, formulate, analyse, and resolve engineering issues in real-world domain.

**PEO2- TECHNICAL SKILLS:**

Apply artificial intelligence theory and concepts to analyse the requirements, realise technical specifications, and design engineering solutions.

**PEO3- SOFT SKILLS:**

Through inter-disciplinary projects and a variety of professional activities, demonstrate technical proficiency, AI competency, and foster collaborative learning and a sense of teamwork.

**PEO4- PROFESSIONAL ETHICS:**

Excel as socially responsible engineers or entrepreneurs with high moral and ethical standards, competence, and soft skills that will enable them to contribute to societal demands and achieve sustainable advancement in emerging computer technologies.

### PROGRAM OUTCOMES (POs)

- PO1: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- PO2: Problem analysis:** Identify, formulate, review research literature, and analyze 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 the public health and safety, and the 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 teamwork:** 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 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)

- PSO1: Fundamentals and critical knowledge of the Computer System:**  
Apply the knowledge gained pertaining to build, assess, and analyze the software and hardware aspects of the program to solve real world business problems.
- PSO2: Comprehensive and applicative knowledge of Software Development:**  
Ability to evaluate and apply knowledge of data engineering, methodologies, and able to plan, develop, test, analyze, and manage required aspects in heterogeneous platforms individually or in team work.
- PSO3: Applications in Computing Domain:**  
Ability to acquire computational knowledge and project development abilities using novel tools and methodologies to tackle challenges in the fields related to Deep Learning, Machine learning, Artificial Intelligence.
- PSO4: Applications in Innovations and Research:**  
Capacity to direct a team or firm that develops products and to use the knowledge learned to recognize actual research issues

## OOPS Lab (LC-CSE-256G)

---

### University Syllabus

Course code	LC-CSE-256G				
Category	Engineering Science courses				
Course title	OBJECT ORIENTED PROGRAMMING LAB				
Scheme and Credits	L	T	P	Credits	Semester = 4
	0	0	4	2	
Classwork	25 Marks				
Exam	25 Marks				
Total	50 Marks				
Duration of Exam	03 Hours				

1. Program to define a structure of a basic JAVA program
2. Program to define the data types, variable, operators, arrays and control structures.
3. Program to define class and constructors. Demonstrate constructors.
4. Program to define class, methods and objects. Demonstrate method overloading.
5. Program to define inheritance and show method overriding.
6. Program to demonstrate Packages.
7. Program to demonstrate Exception Handling.
8. Program to demonstrate Multithreading.
9. Program to demonstrate I/O operations.
10. Program to demonstrate Network Programming.
11. Program to demonstrate Applet structure and event handling.
12. Program to demonstrate Layout managers.



# OOPS Lab (LC-CSE-256G)

---

## Course Outcomes (COs)

Upon successful completion of the course, the students will be able to:

**CO1:** Understand the basics of object-oriented programming using JAVA.

**CO2:** Apply the concept of classes, Java, JDK Components and develop Simple Java Programs.

**CO3:** Develop Simple Java Programs using inheritance and Exception handling.

**CO4:** Develop Multi-threading Programming and Interfaces.

**CO5:** Develop GUI applications using Applet classes, Swing components and Event handling programs.

## CO-PO Mapping

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	1	1	1						2	2		
CO2	1	1	1						2	2		
CO3	2	2	2	2					2	2		
CO4	2	2	3	3					2	3	2	
CO5	3	3	3	3	3				3	3	3	3

## CO-PSO Mapping

CO	PSO1	PSO2	PSO3	PSO4
CO1	1	2	1	
CO2	1	2	1	
CO3	2	2	2	
CO4	2	2	2	2
CO5	3	3	3	3

\*3-HIGH  
\*2-MEDIUM  
\*1-LOW

## Course Overview

Object-Oriented Programming (OOP) is a programming paradigm that treats data and the functions that operate on that data as a single unit. In Java, objects are the basic building blocks of all programs.

An object-oriented programming course in Java typically covers the following topics:

**Objects and classes:** Objects are instances of classes, which are templates for creating objects. Classes contain data members (variables) and methods (functions) that define the behavior of objects. **Inheritance:** Inheritance is the ability of one class to inherit the properties of another class. This allows for code reuse and makes it easier to create complex programs.

**Polymorphism:** Polymorphism is the ability of an object to take on different forms. This is achieved through the use of abstract classes and interfaces.

**Encapsulation:** Encapsulation is the hiding of data members from other parts of the program. This helps to protect data and make programs more secure.

**Exception handling:** Exception handling is a way of dealing with errors that occur during program execution. This helps to prevent programs from crashing and makes them more robust.

In addition to these core concepts, an object-oriented programming course in Java may also cover topics such as:

**GUI programming:** GUI programming is the development of graphical user interfaces. This is a key skill for Java developers, as many Java applications use GUIs.

**Collections:** Collections are data structures that store and organize data. Java provides a number of different collection classes, which can be used to store and manipulate data in a variety of ways.

**Algorithms:** Algorithms are step-by-step procedures for solving problems. Java developers need to be familiar with a variety of algorithms in order to write efficient and effective programs.

Object-oriented programming is a powerful programming paradigm that can be used to create complex and sophisticated programs. A course in object-oriented programming in Java can provide you with the skills and knowledge you need to become a successful Java developer.

## OOPS Lab (LC-CSE-256G)

---

### List of Experiments mapped with COs

S No	Program Name	CO1, CO2
1.	WAP to demonstrate data types available in java	CO1, CO2
2.	WAP to design a simple calculator using switch case statement	CO1, CO2
3.	WAP to print all prime numbers between 1 to 1000	CO1, CO2
4.	WAP to implement linear search in 1D array	CO1, CO2
5.	WAP to implement bubble sort in 1 D array	CO1, CO2
6.	WAP to multiply 2 matrices in java	CO1, CO2
7.	WAP to implement recursion function in java	CO1, CO2
8.	WAP to demonstrate some in-built functions on Strings	CO1, CO2
9.	WAP to demonstrate concept of Class, Object, and methods in java.	CO1, CO2
10.	WAP to demonstrate method overloading in java	CO1, CO2
11.	WAP to demonstrate inheritance in java	CO1, CO3
12.	WAP to demonstrate multiple inheritance using interface	CO1, CO3
13.	WAP to demonstrate method over riding in java	CO1, CO3
14.	WAP to demonstrate exception handling in java	CO1, CO3
15.	WAP to demonstrate multi-threading in java	CO1, CO4
16.	WAP to read, write, append data in files	CO1
17.	WAP to demonstrate database connectivity using JDBC	CO4
18.	WAP to create a Swing Application with JDBC	CO4
19.	WAP to design a Menu using Swing in Java	CO4

### **DOs and DON'Ts**

#### **DOs**

1. Login-on with your username and password.
2. Log off the Computer every time when you leave the Lab.
3. Arrange your chair properly when you are leaving the lab.
4. Put your bags in the designated area.
5. Ask permission to print.

#### **DON'Ts**

1. Do not share your username and password.
2. Do not remove or disconnect cables or hardware parts.
3. Do not personalize the computer setting.
4. Do not run programs that continue to execute after you log off.
5. Do not download or install any programs, games or music on computer in Lab.
6. Personal Internet use chat room for Instant Messaging (IM) and Sites is strictly prohibited.
7. No Internet gaming activities allowed.
8. Tea, Coffee, Water & Eatables are not allowed in the Computer Lab.

## **General Safety Precautions**

### **Precautions (In case of Injury or Electric Shock)**

1. To break the victim with live electric source, use an insulator such as fire wood or plastic to break the contact. Do not touch the victim with bare hands to avoid the risk of electrifying yourself.
2. Unplug the risk of faulty equipment. If main circuit breaker is accessible, turn the circuit off.
3. If the victim is unconscious, start resuscitation immediately, use your hands to press the chest in and out to continue breathing function. Use mouth-to-mouth resuscitation if necessary.
4. Immediately call medical emergency and security. Remember! Time is critical; be best.

### **Precautions (In case of Fire)**

1. Turn the equipment off. If power switch is not immediately accessible, take plug off.
2. If fire continues, try to curb the fire, if possible, by using the fire extinguisher or by covering it with a heavy cloth if possible, isolate the burning equipment from the other surrounding equipment.
3. Sound the fire alarm by activating the nearest alarm switch located in the hallway.
4. Call security and emergency department immediately:

**Emergency : 201 (Reception)**

**Security: 231 (Gate No.1)**

### Guidelines to students for report preparation

All students are required to maintain a record of the experiments conducted by them. Guidelines for its preparation are as follows: -

- 1) All files must contain a title page followed by an index page. *The files will not be signed by the faculty without an entry in the index page.*
- 2) Student's Name, roll number and date of conduction of experiment must be written on all pages.
- 3) For each experiment, the record must contain the following
  - (i) Aim/Objective of the experiment
  - (ii) Pre-experiment work (as given by the faculty)
  - (iii) Lab assignment questions and their solutions
  - (iv) Test Cases (if applicable to the course)
  - (v) Results/ output

**Note:**

1. Students must bring their lab record along with them whenever they come for the lab.
2. Students must ensure that their lab record is regularly evaluated.

## OOPS Lab (LC-CSE-256G)

### Lab Assessment Criteria

An estimated 10 lab classes are conducted in a semester for each lab course. These lab classes are assessed continuously. Each lab experiment is evaluated based on 5 assessment criteria as shown in following table. Assessed performance in each experiment is used to compute CO attainment as well as internal marks in the lab course.

<b>Grading Criteria</b>	<b>Exemplary (4)</b>	<b>Competent (3)</b>	<b>Needs Improvement (2)</b>	<b>Poor (1)</b>
<b>AC1: Pre-Lab written work (this may be assessed through viva)</b>	Complete procedure with underlined concept is properly written	Underlined concept is written but procedure is incomplete	Not able to write concept and procedure	Underlined concept is not clearly understood
<b>AC2: Program Writing/ Modeling</b>	Unable to understand the reason for errors/ bugs even after they are explicitly pointed out	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied	Assigned problem is properly analyzed & correct solution designed	Assigned problem is properly analyzed
<b>AC3: Identification &amp; Removal of errors/ bugs</b>	Able to identify errors/ bugs and remove them	Able to identify errors/ bugs and remove them with little bit of guidance	Is dependent totally on someone for identification of errors/ bugs and their removal	Unable to understand the reason for errors/ bugs even after they are explicitly pointed out
<b>AC4: Execution &amp; Demonstration</b>	All variants of input /output are tested, Solution is well demonstrated and implemented concept is clearly explained	All variants of input /output are not tested, However, solution is well demonstrated and implemented concept is clearly explained	Only few variants of input /output are tested, Solution is well demonstrated but implemented concept is not clearly explained	Solution is not well demonstrated and implemented concept is not clearly explained
<b>AC5: Lab Record Assessment</b>	All assigned problems are well recorded with objective, design constructs and solution along with Performance analysis using all variants of input and output	More than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	Less than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	

# LAB EXPERIMENTS



## OOPS Lab (LC-CSE-256G)

---

### Program No. 1

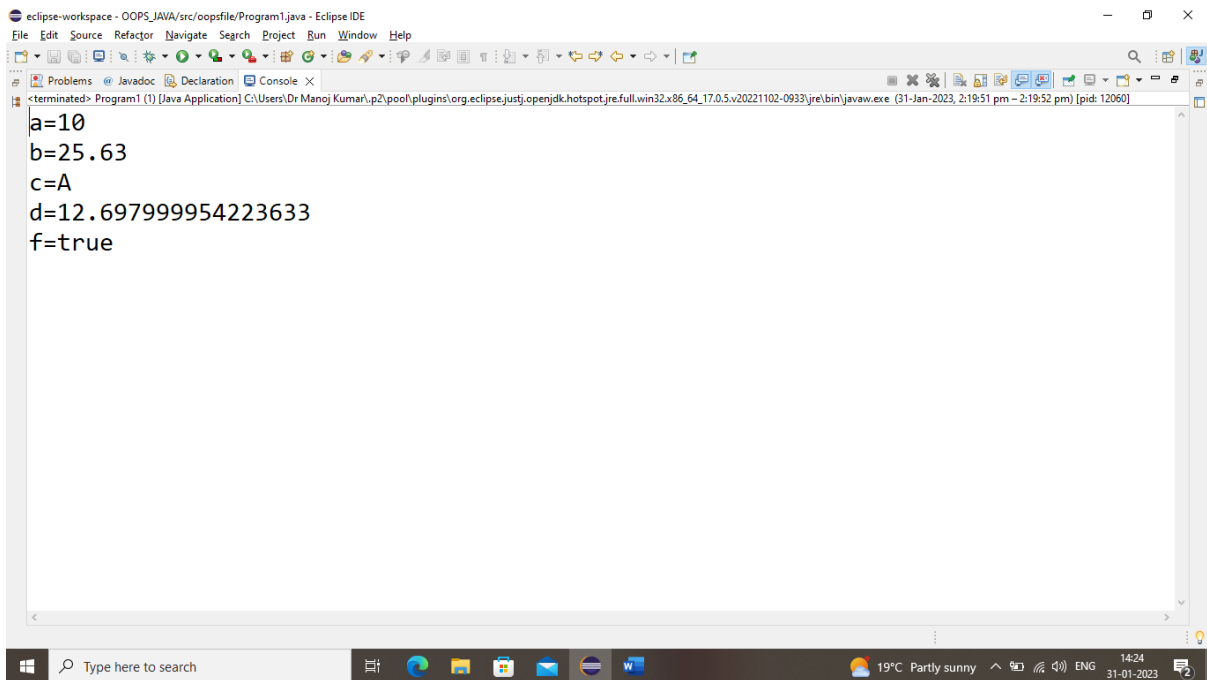
*WAP to demonstrate data types available in Java*

Source Code:

```
package oopsfile;
public class Program1
{
public static void main(String args[])
{
int a;
char c;
float b;
double d;
boolean f;
a=10;
c='A';
b=25.63f;
d=12.698f;
f=true;
System.out.println("a="+a);
System.out.println("b="+b);
System.out.println("c="+c);
System.out.println("d="+d);
System.out.println("f="+f);
}
}
```

# OOPS Lab (LC-CSE-256G)

## Output

A screenshot of the Eclipse IDE's console window. The title bar reads "eclipse-workspace - OOPS\_JAVA/src/oopsfile/Program1.java - Eclipse IDE". The console output shows the following text:

```
a=10  
b=25.63  
c=A  
d=12.697999954223633  
f=true
```

The console window also shows a status bar at the bottom with the text "<terminated> Program1 (1) [Java Application] C:\Users\Dr Manoj Kumar\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.5.v20221102-0933\jre\bin\javaw.exe (31-Jan-2023, 2:19:51 pm - 2:19:52 pm) [pid: 12060]".

```
a=10  
b=25.63  
c=A  
d=12.697999954223633  
f=true
```

## OOPS Lab (LC-CSE-256G)

---

### Program No. 2

*WAP to design a simple calculator using switch case statement*

Source Code:

```
package oopsfile;
import java.util.Scanner;
public class Program2 {
public static void main(String[] args) {
// TODO Auto-generated method stub
System.out.println("Press 1: Addition");
System.out.println("Press 2: Subtraction");
System.out.println("Press 3: Multiply");
System.out.println("Press 4: Division");
int a,b,c;
int choice;
System.out.println("Enter your choice");
Scanner sc=new Scanner(System.in);
choice=sc.nextInt();
System.out.println("Enter First Number");
a=sc.nextInt();
System.out.println("Enter Second Number");
b=sc.nextInt();
switch (choice)
{
case 1:
c=a+b;
System.out.print("Addition is "+c);
break;
case 2:
c=a-b;
System.out.print("Subtraction is "+c);
break;
case 3:
```

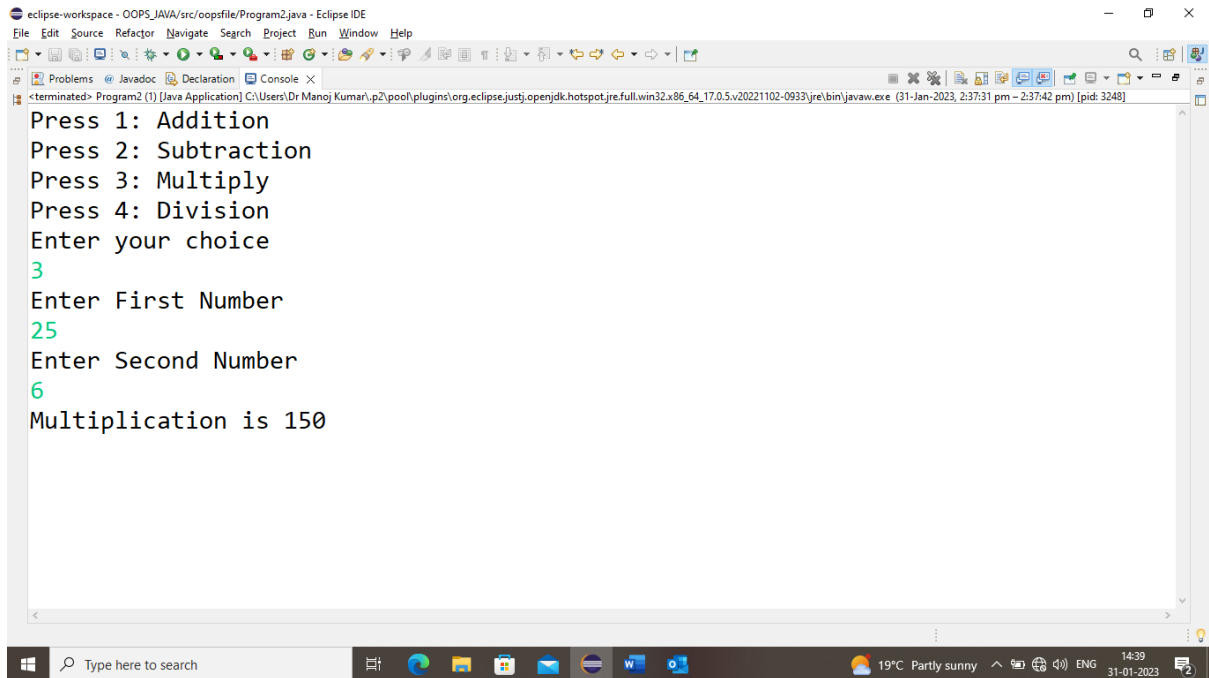
## OOPS Lab (LC-CSE-256G)

---

```
c=a*b;
System.out.print("Multiplication is "+c);
break;
case 4:
c=a/b;
System.out.print("Division is "+c);
break;
default:
System.out.println("Bad Choice");
break;
}
}
}
```

# OOPS Lab (LC-CSE-256G)

## Output



The screenshot shows the Eclipse IDE interface with a console window displaying the output of a Java program. The program prompts the user to choose an operation (Addition, Subtraction, Multiply, or Division) and then asks for two numbers. The user has entered '3' for the operation, '25' for the first number, and '6' for the second number. The output shows 'Multiplication is 150'.

```
eclipse-workspace - OOPS_JAVA/src/oopfile/Program2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console X
terminated> Program2 (1) [Java Application] C:\Users\Dr Manoj Kumar\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0932\jre\bin\javaw.exe (31-Jan-2023, 2:37:31 pm - 2:37:42 pm) [pid: 3248]
Press 1: Addition
Press 2: Subtraction
Press 3: Multiply
Press 4: Division
Enter your choice
3
Enter First Number
25
Enter Second Number
6
Multiplication is 150
```

Press 1: Addition  
Press 2: Subtraction  
Press 3: Multiply  
Press 4: Division  
Enter your choice  
3  
Enter First Number  
25  
Enter Second Number  
6  
Multiplication is 150

Press 1: Addition  
Press 2: Subtraction  
Press 3: Multiply  
Press 4: Division  
Enter your choice  
1  
Enter First Number  
250  
Enter Second Number  
300  
Addition is 550

## OOPS Lab (LC-CSE-256G)

---

### Program No. 3

*WAP to print all prime numbers between 1 to 1000*

```
package oopsfile;
import java.util.Scanner;

public class Program3
{
public static void main(String ar[])
{
int num=2;
//Scanner sc=new Scanner(System.in);
//System.out.print("Enter any number");
//num=sc.nextInt();
for(num=2;num<100;num++)
{
int i=2;
while(i<num)
{
if(num%i==0)
{
//System.out.println("NUmber is not Prime");
if(num==2)
System.out.println(num);
break;
}
i++;
if(i==num)
System.out.println(num);
}
}}
}
```

## OOPS Lab (LC-CSE-256G)

---

### Output

3  
5  
7  
11  
13  
17  
19  
23  
29  
31  
37  
41  
43  
47  
53  
59  
61  
67  
71  
73  
79  
83  
89  
97

## OOPS Lab (LC-CSE-256G)

---

### Program No. 4

*WAP to implement linear search in 1D array*

```
package oopsfile;

import java.util.Scanner;

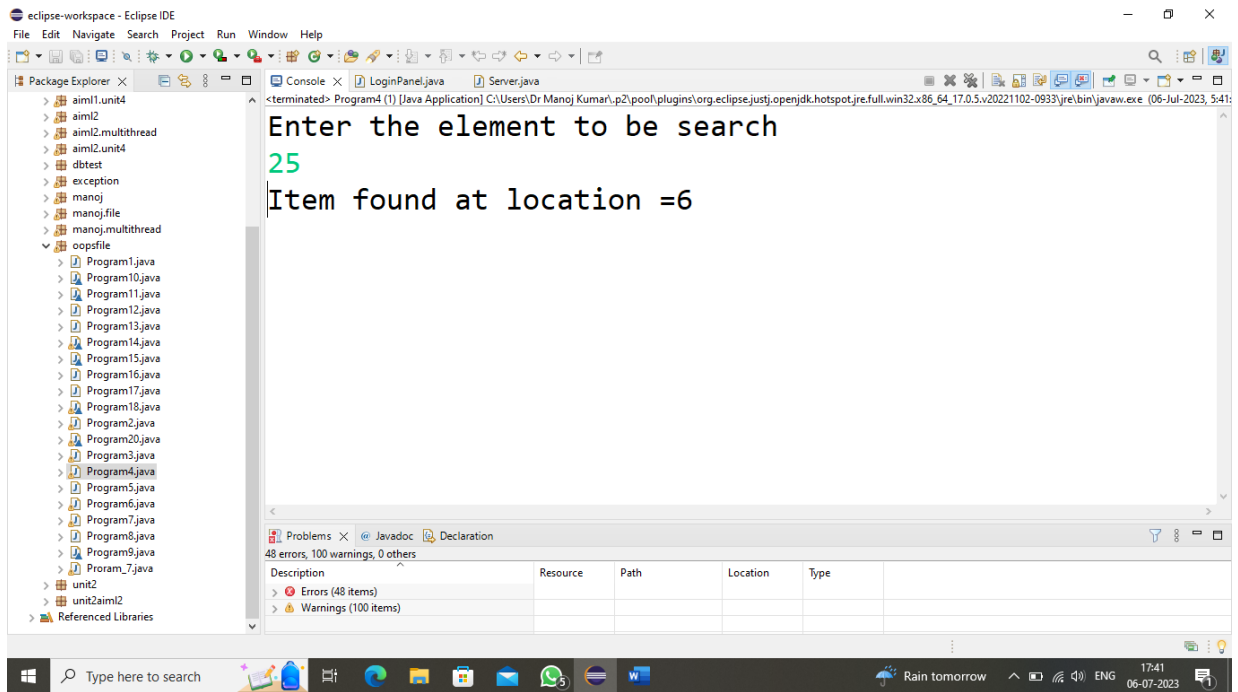
public class Program4 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a[]={10,20,30,50,5,15,25,35,45,55,76,102,7};
        int item=0;
        int loc=-1;
        System.out.println("Enter the element to be search");
        Scanner sc=new Scanner(System.in);
        item=sc.nextInt();
        for(int i=0;i<a.length;i++)
        {
            if(item==a[i])
            {
                loc=i;
                break;
            }
        }
        if(loc==-1)
            System.out.println("Item not Found");
        else
            System.out.println("Item found at location =" +loc);
    }
}
```



# OOPS Lab (LC-CSE-256G)

## Output



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project structure with the following packages and files:

- aim1.unit4
- aim12
- aim12.multithread
- aim12.unit4
- dbtest
- exception
- manoj
- manoj.file
- manoj.multithread
- oopsfile
  - Program1.java
  - Program10.java
  - Program11.java
  - Program12.java
  - Program13.java
  - Program14.java
  - Program15.java
  - Program16.java
  - Program17.java
  - Program18.java
  - Program2.java
  - Program20.java
  - Program3.java
  - Program4.java
  - Program5.java
  - Program6.java
  - Program7.java
  - Program8.java
  - Program9.java
  - Proram\_7.java
- unit2
- unit2aim12
- Referenced Libraries

The Console window shows the following output:

```
<terminated> Program4 (1) [Java Application] C:\Users\Dr Manoj Kumar\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (06-Jul-2023, 5:41:11)  
Enter the element to be search  
25  
Item found at location =6
```

The Problems window at the bottom shows 48 errors, 100 warnings, and 0 others.

Description	Resource	Path	Location	Type
> Errors (48 items)				
> Warnings (100 items)				

The Windows taskbar at the bottom shows the time as 17:41 on 06-07-2023.

## OOPS Lab (LC-CSE-256G)

---

### Program No. 5

*WAP to implement bubble sort in 1 D array*

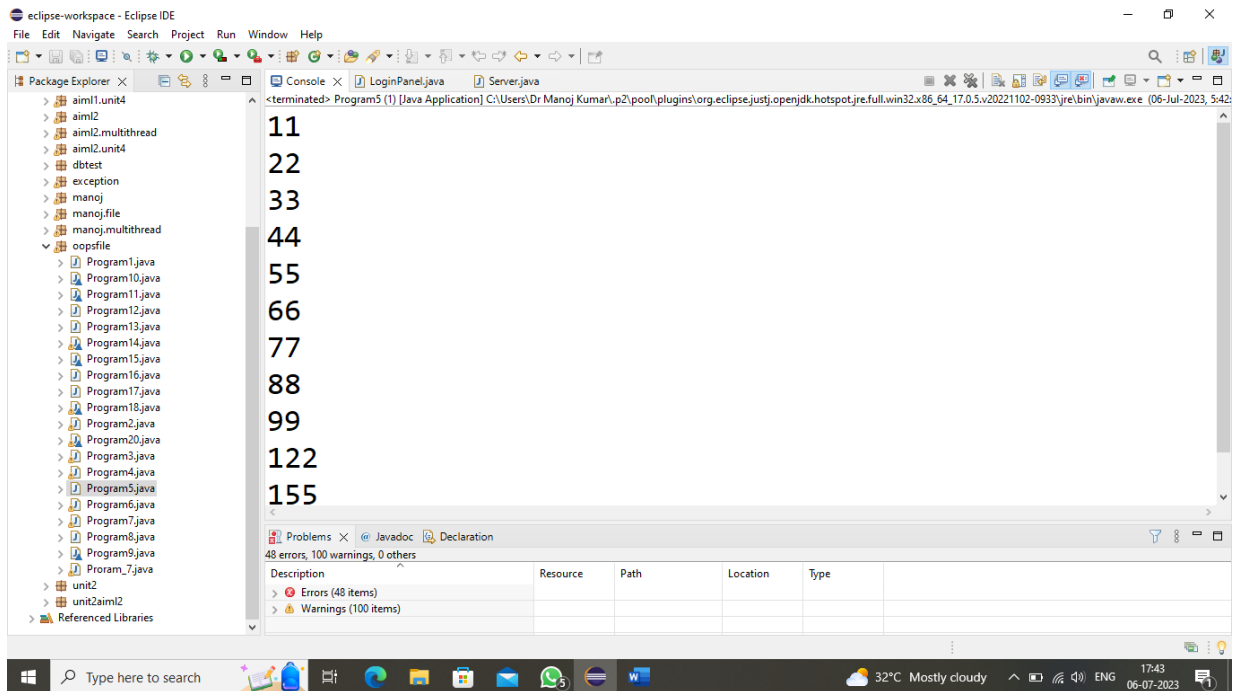
```
package oopsfile;

public class Program5 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a[]= {22,55,99,88,44,33,66,77,11,155,122};
        int l=a.length;
        for (int i = 0; i < l - 1; i++)
            for (int j = 0; j < l - i - 1; j++)
                if (a[j] > a[j + 1])
                {
                    int temp = a[j];
                    a[j] = a[j + 1];
                    a[j + 1] = temp;
                }
        for(int i=0;i<l;i++)
            System.out.println(a[i]);
    }
}
```

# OOPS Lab (LC-CSE-256G)

## Output



## OOPS Lab (LC-CSE-256G)

---

### Program No. 6

*WAP to multiply 2 matrices in java*

```
package oopsfile;

import java.util.Scanner;

public class Program6 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a[][]=new int[3][3];
        int b[][]=new int[3][3];
        int c[][]=new int[3][3];
// Data for Matrix A
        System.out.println("Enter the matrix A");
        Scanner sc=new Scanner(System.in);
        for(int i=0;i<3;i++)
        {
            for(int j=0;j<3;j++)
            {
                a[i][j]=sc.nextInt();
            }
        }
// Data for Matrix A
        System.out.println("Enter the matrix A");
        //Scanner sc=new Scanner(System.in);
        for(int i=0;i<3;i++)
        {
            for(int j=0;j<3;j++)
            {
                b[i][j]=sc.nextInt();
            }
        }
        for(int i=0;i<3;i++)
        {
            for(int j=0;j<3;j++)
            {
                c[i][j]=0;
                for(int k=0;k<3;k++)
                {
                    c[i][j]=c[i][j]+a[i][k]*b[k][j];
                }
            }
        }
        for(int i=0;i<3;i++)
        {
```

## OOPS Lab (LC-CSE-256G)

---

```
        for(int j=0;j<3;j++)
            {
                System.out.print(c[i][j]+" ");
            }
        System.out.println();
    }
}
```

## OOPS Lab (LC-CSE-256G)

---

### Output

Enter the matrix A

10 21 47

52 5 7

3 9 41

Enter the matrix B

1 2 3

4 7 8

6 0 1

Product of Matrix A and B

376 167 245

114 139 203

285 69 122

## OOPS Lab (LC-CSE-256G)

---

### Program No. 7

*WAP to implement recursion function in java*

```
package oopsfile;

import java.util.Scanner;

public class Program7 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int num;
        Scanner sc=new Scanner(System.in);
        num=sc.nextInt();
        int factorial;
        factorial=fact(num);
        System.out.println(factorial);
    }
    static int fact(int num)
    {
        if(num==1)
            return 1;
        else
            return num*fact(num-1);
    }
}
```

# OOPS Lab (LC-CSE-256G)

## Output

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left lists a project named 'oopsfile' containing several Java programs. The Console window displays the output of a Java application, which prompts the user to 'Enter any Number'. The user has entered '5', and the application has printed '120'. The Problems window at the bottom shows 48 errors and 100 warnings.

```
<terminated> Program7 (1) [Java Application] C:\Users\Dr.Manoj Kumar\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v20221102-0933\jre\bin\javaw.exe (06-Jul-2023, 5:48)
Enter any Number
5
120
```

Description	Resource	Path	Location	Type
48 errors, 100 warnings, 0 others				
Errors (48 items)				
Warnings (100 items)				



## OOPS Lab (LC-CSE-256G)

---

### Program No. 8

*WAP to demonstrate some in-built functions on Strings*

```
package oopsfile;

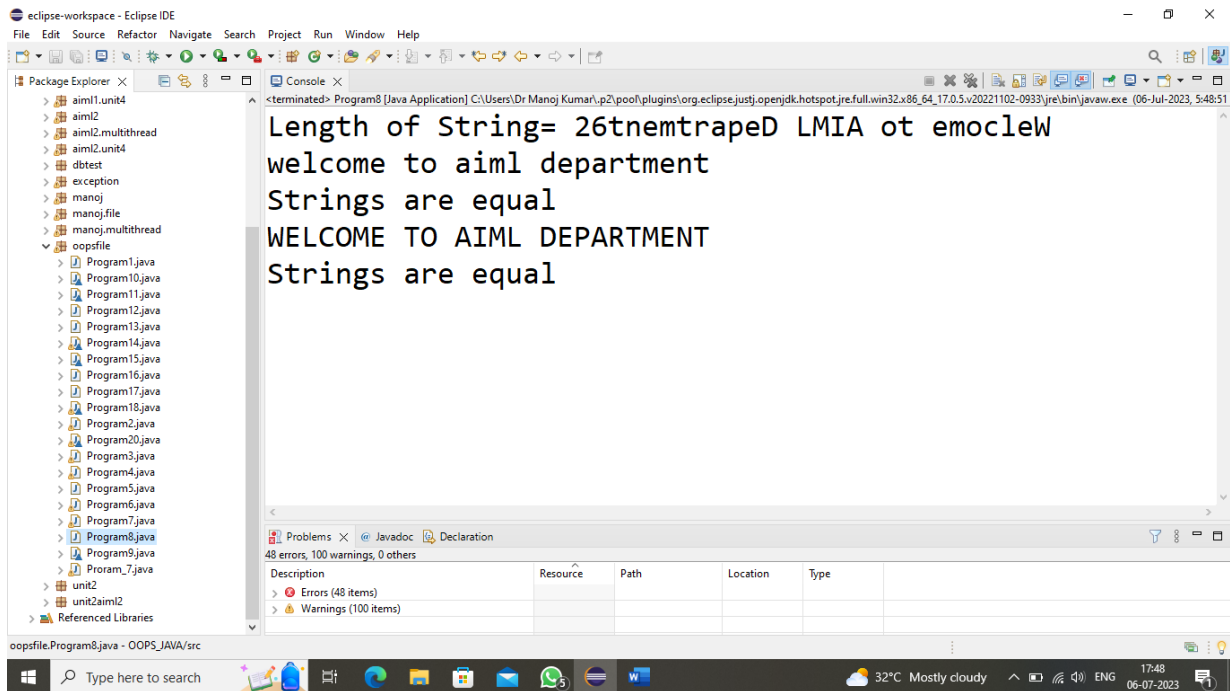
public class Program8
{
    public static void main(String arg[])
    {
        String str=new String("Welcome to AIML Department");
        //length of string
        System.out.print("Length of String= "+str.length());
        StringBuilder sb=new StringBuilder(str);

        String str2=new String(sb.reverse());
        System.out.println(str2);
        str2=new String(sb.reverse());
        System.out.println(str.toLowerCase());

        if(str.equals(str2))
        {
            System.out.println("Strings are equal");
        }
        else
        {
            System.out.println("Strings are not equal");
        }
        System.out.println(str.toUpperCase());
        if(str.equalsIgnoreCase(str2))
        {
            System.out.println("Strings are equal");
        }
        else
        {
            System.out.println("Strings are not equal");
        }
    }
}
```

# OOPS Lab (LC-CSE-256G)

## Output



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left lists several packages, including 'oopsfile' which contains 'Program8.java'. The Console window displays the following output:

```
<terminated> Program8 [Java Application] C:\Users\Dr Manoj Kumar\p2\poo\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (06-Jul-2023, 5:48:51)
Length of String= 26tnemtrapeD LMIA ot emoclew
welcome to aiml department
Strings are equal
WELCOME TO AIML DEPARTMENT
Strings are equal
```

Below the console, the Problems view shows 48 errors and 100 warnings. The Windows taskbar at the bottom indicates the system time is 17:48 on 06-07-2023.

## OOPS Lab (LC-CSE-256G)

---

### Program No. 9

*WAP to demonstrate concept of Class, Object, and methods in java.*

```
package oopsfile;
```

```
class Add
```

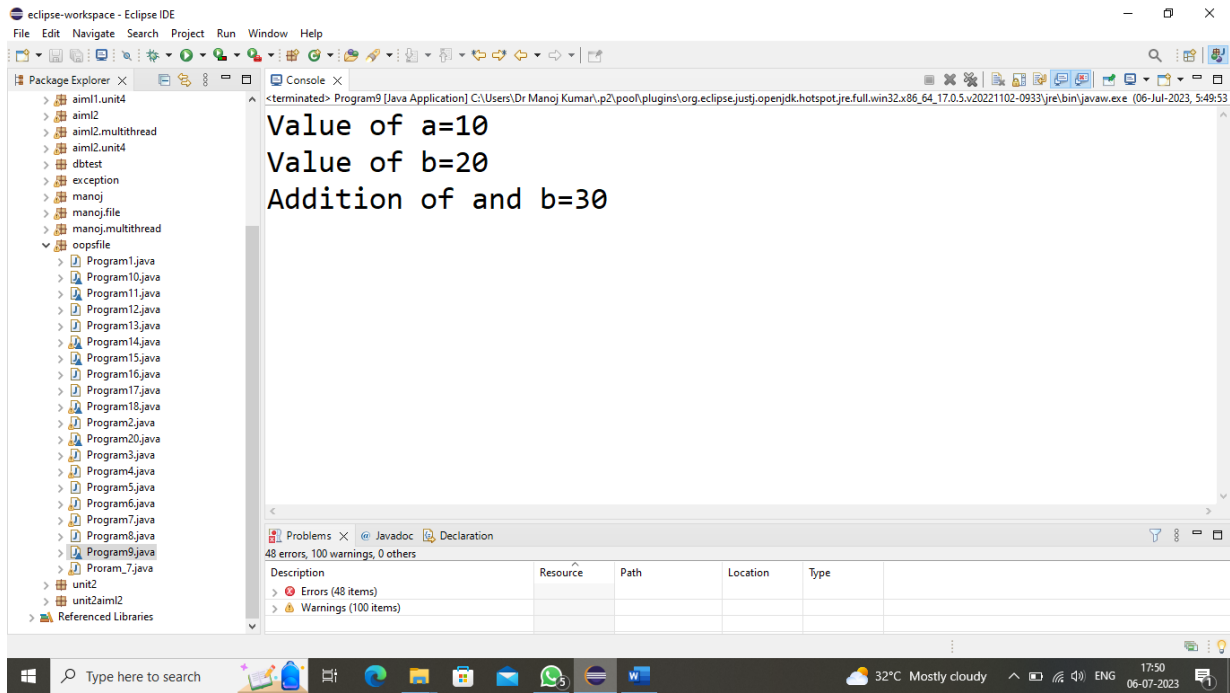
```
{
    int a, b;
    void getData(int x,int y)
    {
        a=x;
        b=y;
    }
    int add()
    {
        return a+b;
    }
}
```

```
class Program9
```

```
{
    public static void main(String[] args)
    {
        Add add=new Add();
        add.getData(10, 20);
        System.out.println("Value of a="+add.a);
        System.out.println("Value of b="+add.b);
        System.out.println("Addition of a and b="+add.add());
    }
}
```

# OOPS Lab (LC-CSE-256G)

## Output



## OOPS Lab (LC-CSE-256G)

---

### Program No. 10

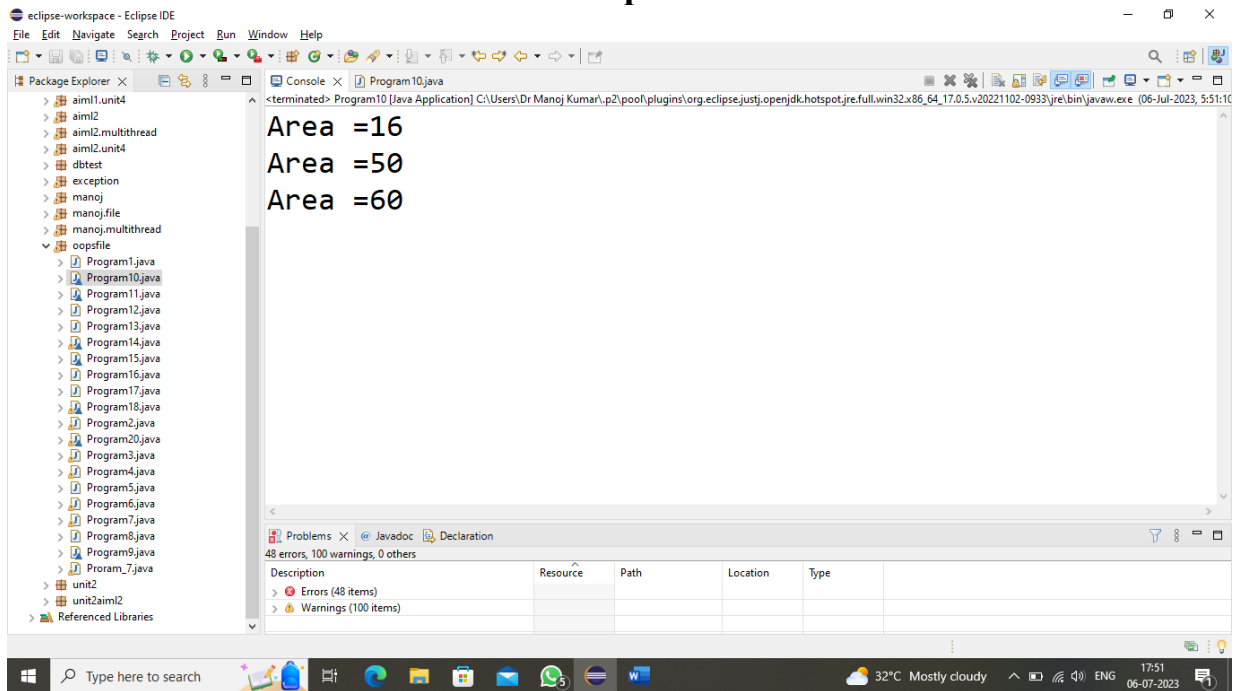
*WAP to demonstrate method overloading in java*

package oopsfile;

```
class Program10
{
    public static void main(String args[])
    {
        CalArea ca=new CalArea();
        int area;
        area=ca.area(4);// one parameter
        System.out.println("Area =" +area);
        area=ca.area(10,5);// two parameter
        System.out.println("Area =" +area);
        area=ca.area(4,5,3);// three parameter
        System.out.println("Area =" +area);
    }
}
class CalArea
{
    int area(int x)
    {
        return x*x;
    }
    int area(int x,int y)
    {
        return x*y;
    }
    int area(int x,int y,int z)
    {
        return x*y*z;
    }
}
```

# OOPS Lab (LC-CSE-256G)

## Output



## OOPS Lab (LC-CSE-256G)

---

### Program No. 11

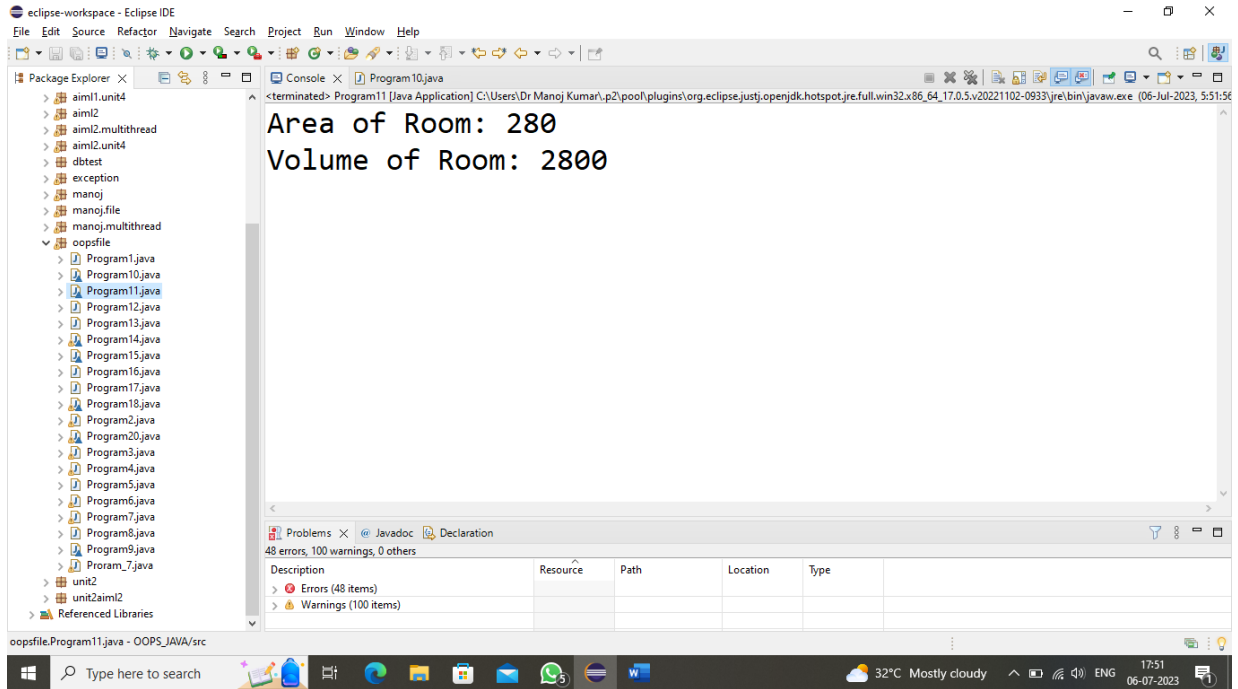
*WAP to demonstrate inheritance in java*

```
package oopsfile;

class Room
{
    int length,breadth;
    Room(int x,int y)
    {
        length=x;
        breadth=y;
    }
    int area()
    {
        return length*breadth;
    }
}
class Bedroom extends Room
{
    int height;
    Bedroom(int x,int y,int z)
    {
        super(x,y);
        height=z;
    }
    int volume()
    {
        return length*breadth*height;
    }
}
class Program11
{
    public static void main(String args[])
    {
        Bedroom b=new Bedroom(14,20,10);
        System.out.println("Area of Room: "+b.area());
        System.out.println("Volume of Room: "+b.volume());
    }
}
```

# OOPS Lab (LC-CSE-256G)

## Output





## OOPS Lab (LC-CSE-256G)

---

### Program No. 12

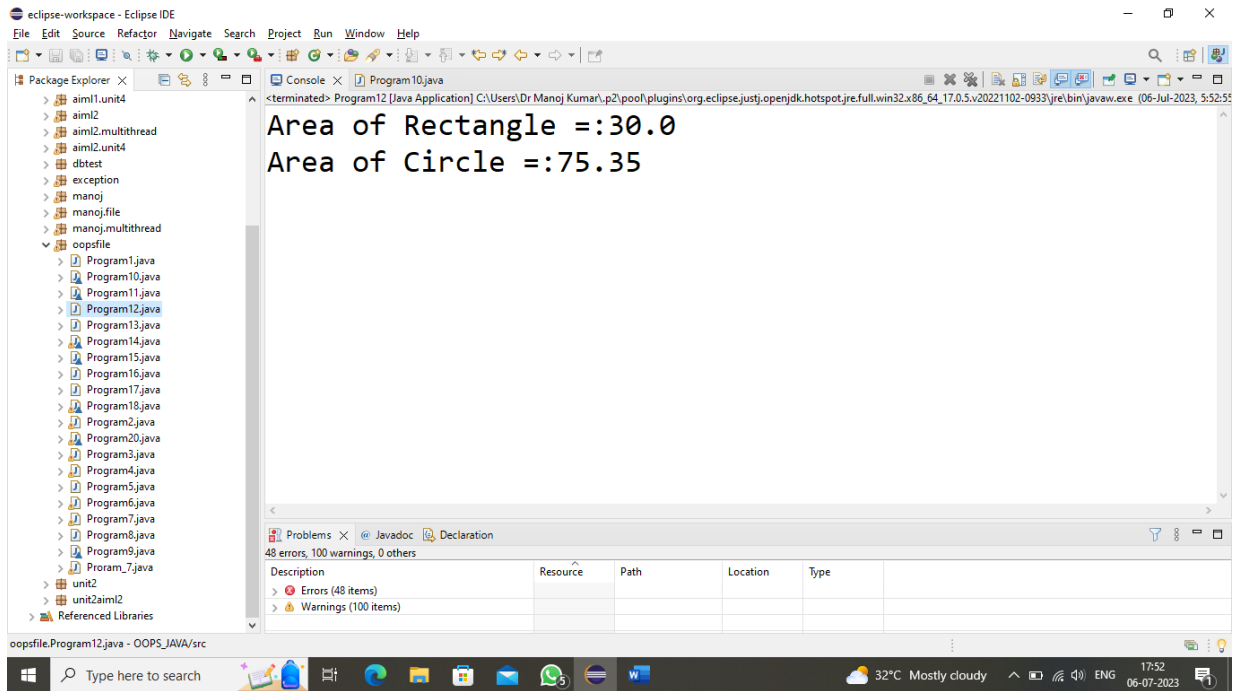
*WAP to demonstrate multiple inheritance using interface*

```
package oopsfile;

interface Area
{
    final static float pi=3.014F;
    float compute(float x, float y);
}
class Rectangle implements Area
{
    public float compute(float x, float y)
    {
        return x*y;
    }
}
class Circle implements Area
{
    public float compute(float x, float y)
    {
        return pi*x*x;
    }
}
public class Program12
{
    public static void main(String args[])
    {
        Rectangle r=new Rectangle();
        Circle c=new Circle();
        System.out.println("Area of Rectangle ="+r.compute(5, 6));
        System.out.println("Area of Circle ="+c.compute(5, 6));
    }
}
```

# OOPS Lab (LC-CSE-256G)

## Output



## OOPS Lab (LC-CSE-256G)

---

### Program No. 13

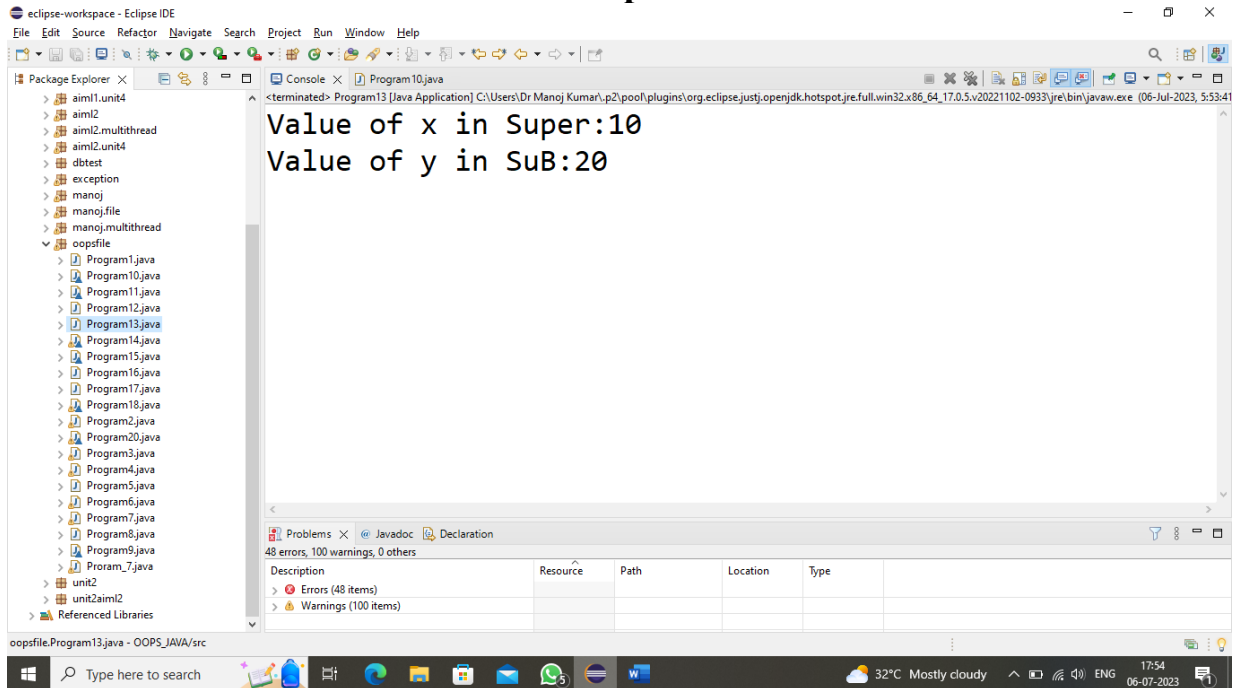
*WAP to demonstrate method over riding in java*

```
package oopsfile;

class Super
{
    int x;
    Super(int x)
    {
        this.x=x;
    }
    void display()
    {
        System.out.println("Value of x in Super:"+x);
    }
}
class Sub extends Super
{
    int y;
    Sub(int x,int y)
    {
        super(x);
        this.y=y;
    }
    void display()
    {
        System.out.println("Value of x in Super:"+x);
        System.out.println("Value of y in SuB:"+y);
    }
}
public class Program13
{
    public static void main(String args[])
    {
        Sub s=new Sub(10,20);
        s.display();
    }
}
```

# OOPS Lab (LC-CSE-256G)

## Output



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left lists a project named 'oopsfile' containing several Java files, including Program10.java. The Console window displays the output of the program:

```
<terminated> Program13 [Java Application] C:\Users\Dr Manoj Kumar\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64.17.0.5.v20221102-0933\jre\bin\javaw.exe (06-Jul-2023, 5:53:41)
Value of x in Super:10
Value of y in Sub:20
```

Below the console, the Problems window shows 48 errors and 100 warnings. The table below summarizes the error and warning counts:

Description	Resource	Path	Location	Type
> Errors (48 items)				
> Warnings (100 items)				

The Windows taskbar at the bottom shows the system tray with a search bar, task icons, and system information: 32°C Mostly cloudy, 17:54, 06-07-2023.

## OOPS Lab (LC-CSE-256G)

---

### Program No. 14

*WAP to demonstrate exception handling in java*

```
package oopsfile;

class MyException extends Exception
{
    public MyException(String message) {
        // TODO Auto-generated constructor stub
        super(message);
    }
}
class Program14
{
    public static void main(String args[])
    {
        int a=5;
        int y=1000;
        try {
            float z=(a)/(y);
            if(z<0.01)
            {
                throw new MyException("Number is too small");
            }
        }
        catch(MyException e)
        {
            System.out.println("Caught my Exception");
        }
    }
}
```

# OOPS Lab (LC-CSE-256G)

## Output

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left lists a project named 'oopsfile' containing several Java files, including 'Program14.java'. The Console window displays the output: 'Caught my Exception'. The Problems view at the bottom shows a summary of 48 errors and 100 warnings. The Windows taskbar at the bottom indicates the system time is 17:55 on 06-07-2023.

Description	Resource	Path	Location	Type
> Errors (48 items)				
> Warnings (100 items)				

## OOPS Lab (LC-CSE-256G)

---

### Program No. 15

*WAP to demonstrate multi-threading in java*

```
package oopsfile;

class A extends Thread
{
    public void run()
    {
        for(int i=1;i<=5;i++)
        {
            System.out.println("From Thread A="+i);
        }
        System.out.println("Exit from A");
    }
}

class B extends Thread
{
    public void run()
    {
        for(int j=1;j<=5;j++)
        {
            System.out.println("From Thread B="+j);
        }
        System.out.println("Exit from B");
    }
}

class C extends Thread
{
    public void run()
    {
        for(int k=1;k<=5;k++)
        {
            System.out.println("From Thread C="+k);
        }
        System.out.println("Exit from C");
    }
}

class Program15 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new A().start();
        new B().start();
        new C().start();
    }
}
```

## OOPS Lab (LC-CSE-256G)

---

}

### Output

From Thread A=1  
From Thread A=2  
From Thread A=3  
From Thread A=4  
From Thread A=5  
Exit from A  
From Thread B=1  
From Thread B=2  
From Thread B=3  
From Thread B=4  
From Thread B=5  
Exit from B  
From Thread C=1  
From Thread C=2  
From Thread C=3  
From Thread C=4  
From Thread C=5  
Exit from C



## OOPS Lab (LC-CSE-256G)

---

### Program No. 16

*WAP to read, write, append data in files*

```
package oopsfile;
import java.io.*;
public class Program16
{
    public static void main(String args[]) throws Exception
    {
        File old=new File("G:\\DRONACHARYA COLLEGE\\C Codes\\Java
Code\\File1.txt");
        File neww=new File("G:\\DRONACHARYA COLLEGE\\C Codes\\Java
Code\\File2.txt");
        FileInputStream fis =new FileInputStream(old);
        FileOutputStream fos=new FileOutputStream(neww);
        int i;
        while((i=fis.read())!=-1)
        {
            fos.write(i);
        }
        fis.close();
        fos.close();
        System.out.println("Data has been copied Successfully");
    }
}
```

# OOPS Lab (LC-CSE-256G)

## Output

The screenshot displays the Eclipse IDE interface. The Package Explorer on the left shows a project structure with a folder named 'oopsfile' containing several Java files, including 'Program16.java'. The Console window shows the output: 'Data has been copied Successfully'. The Problems window at the bottom indicates '48 errors, 100 warnings, 0 others'.

Console Output:

```
<terminated> Program16 [Java Application] C:\Users\Dr Manoj Kumar\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (06-Jul-2023, 5:58:00)  
Data has been copied Successfully
```

Problems Window:

Description	Resource	Path	Location	Type
Errors (48 items)				
Warnings (100 items)				

## OOPS Lab (LC-CSE-256G)

---

### Program No. 17

*WAP to demonstrate database connectivity using JDBC*

```
package oopsfile;
import java.sql.*;
public class Program17 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        try
        {
            Class.forName("com.mysql.cj.jdbc.Driver");

            Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/dbtest","root","Manoj@8043
");
            Statement stat=con.createStatement();

            ResultSet rs=stat.executeQuery("select * from student");

            while(rs.next())
            {
                System.out.print(rs.getInt(1)+"\t");
                System.out.print(rs.getString(2)+"\t");
                System.out.print(rs.getString(3)+"\t");
                System.out.println();
            }
        }

        catch(ClassNotFoundException e)
        {
        }
        catch(SQLException e)
        {
        }
    }
}
```

# OOPS Lab (LC-CSE-256G)

## Output

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project structure for 'College Data Management' and 'OOPS\_JAVA'. The Console window shows the output of a Java application, displaying two rows of data: '25201 MANOJ CSE' and '25441 NAVEEN AIML'. The Problems window at the bottom indicates 48 errors and 100 warnings.

```
<terminated> Program17 [Java Application] C:\Users\Dr Manoj Kumar\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (06-Jul-2023, 6:07:44)
25201 MANOJ CSE
25441 NAVEEN AIML
```

Description	Resource	Path	Location	Type
> Errors (48 items)				
> Warnings (100 items)				

## OOPS Lab (LC-CSE-256G)

---

### Program No. 18

*WAP to create a Swing Application with JDBC*

```
package oopsfile;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

class Program18
{
    public static void main(String[] args)
    {
        MyFram mf=new MyFram();
        mf.setVisible(true);
        mf.setTitle("Student");
        mf.setSize(400, 400);
        mf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mf.setLocation(200, 200);
    }
}

class MyFram extends JFrame implements ActionListener
{
    JLabel nameLabel,branchLabel,rollLabel;
    JTextField nameField,branchField,rollField;
    JButton submit;
    MyFram()
    {
        nameLabel=new JLabel("NAME");
        branchLabel=new JLabel("BRANCH");
        rollLabel=new JLabel("ROLL NO");

        nameField=new JTextField(10);
        branchField=new JTextField(10);
        rollField=new JTextField(10);
        submit=new JButton("SUBMIT");
        setLayout(null);
        add(nameLabel);
        nameLabel.setBounds(50, 40, 60, 50);
        add(nameField);
        nameField.setBounds(150, 40, 100, 35);
    }
}
```

## OOPS Lab (LC-CSE-256G)

---

```
        add(rollLabel);
        rollLabel.setBounds(50, 100, 60, 50);
        add(rollField);
        rollField.setBounds(150, 100, 100, 35);
        add(branchLabel);
        branchLabel.setBounds(50, 150, 60, 50);
        add(branchField);
        branchField.setBounds(150, 150, 100, 35);
        add(submit);
        submit.setBounds(100, 250, 100, 30);
        submit.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource()==submit)
        {
            try
            {
                Class.forName("com.mysql.cj.jdbc.Driver");
                Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/dbtest","root","Manoj@8043");
                PreparedStatement stat=con.prepareStatement("insert into student
values(?,?,?)");

                String name=nameField.getText();
                String branch=branchField.getText();
                String roll=rollField.getText();
                int rol=Integer.parseInt(roll);
                stat.setInt(1, rol);
                stat.setString(2, name);
                stat.setString(3, branch);
                int i= stat.executeUpdate();
                JOptionPane.showMessageDialog(this,"Data Saved Successfully");
                rollField.setText("");
                nameField.setText("");
                branchField.setText("");
            }
            catch(ClassNotFoundException ex)
            {
                JOptionPane.showMessageDialog(this,"Class Not Found");
            }
            catch(SQLException ed)
            {
                JOptionPane.showMessageDialog(this,"Data Base Error");
            }
        }
    }
}
```

# OOPS Lab (LC-CSE-256G)

## Output

The screenshot displays the Eclipse IDE interface. The Package Explorer on the left shows a project structure with a package named 'oopsfile' containing several Java programs. The main editor window shows a Java code snippet with the following lines highlighted in blue:

```
calhost:3306/dbtest", "root", "");  
into student values(?,?,?)");
```

Overlaid on the IDE is a 'Student' dialog box with three input fields: 'NAME' (containing 'SHIPRA'), 'ROLL NO' (containing '25445'), and 'BRANCH' (containing 'AIML'). A 'SUBMIT' button is located at the bottom of this dialog. A smaller 'Message' dialog box is also present, displaying the text 'Data Saved Successfully' with an 'OK' button.

The bottom of the IDE shows a 'Problems' view with a table containing error and warning information:

Description	Resource	Path	Location	Type
Errors (48 items)				
Warnings (100 items)				

The Windows taskbar at the bottom shows the system tray with the date '06-07-2023' and time '18:11'.

## OOPS Lab (LC-CSE-256G)

---

### Program No. 19

*WAP to design a Menu using Swing in Java*

```
package oopsfile;

import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;

class Program20 {
    public static void main(String[] args)
    {
        MenuTest2 mt=new MenuTest2();
        mt.setTitle("MENU TEST");
        mt.setSize(400, 400);
        mt.setVisible(true);
        mt.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

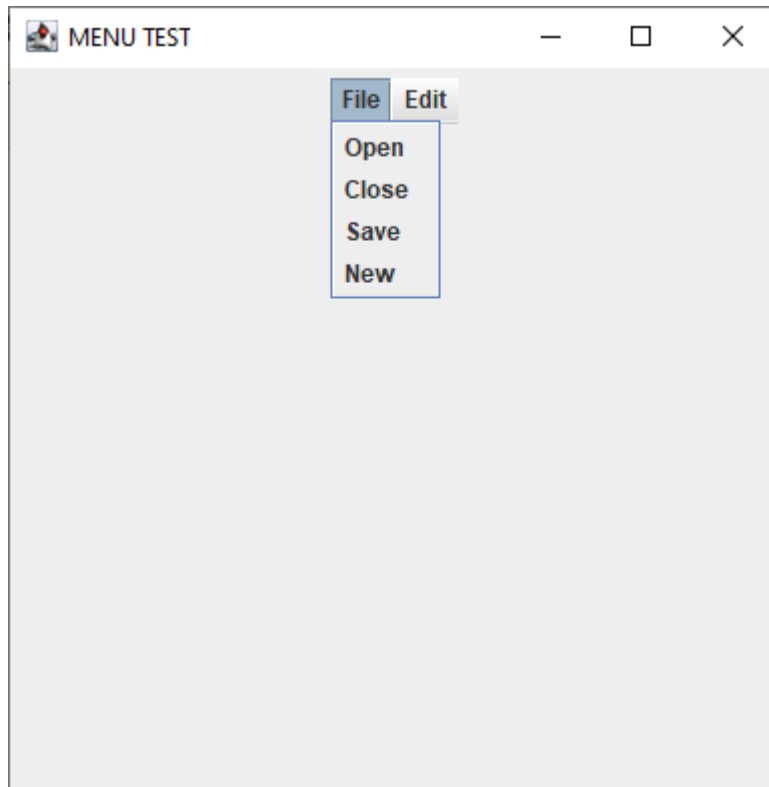
class MenuTest2 extends JFrame
{
    JPanel panel;
    MenuTest2()
    {
        JMenuBar jmb=new JMenuBar();
        JMenu file=new JMenu("File");
        JMenuItem open=new JMenuItem("Open");
        JMenuItem close=new JMenuItem("Close");
        JMenuItem save=new JMenuItem("Save");
        JMenuItem ne =new JMenuItem("New");
        file.add(open);
        file.add(close);
        file.add(save);
        file.add(ne);
        JMenu edit=new JMenu("Edit");
        JMenuItem cut=new JMenuItem("Cut");
        JMenuItem copy=new JMenuItem("Copy");
        JMenuItem paste=new JMenuItem("Paste");
        edit.add(cut);
        edit.add(copy);
        edit.add(paste);
        jmb.add(file);
        jmb.add(edit);
        panel=new JPanel();
        panel.add(jmb);
        add(panel);
    }
}}
```



# OOPS Lab (LC-CSE-256G)

---

## Output



# OOPS Lab (LC-CSE-256G)

---

## Viva Voce Questions with Answers

1. What is object-oriented programming (OOP)?

OOP is a programming paradigm that treats data as objects. Objects have state and behavior, and they can interact with each other.

2. What are the four pillars of OOP?

The four pillars of OOP are abstraction, encapsulation, inheritance, and polymorphism.

3. What is abstraction?

Abstraction is the process of hiding the implementation details of an object and only exposing the essential details to the user.

4. What is encapsulation?

Encapsulation is the bundling of data and methods together into a single unit. This helps to protect the data from unauthorized access.

5. What is inheritance?

Inheritance is the ability of a class to inherit the properties and methods of another class. This allows for code reuse and makes it easier to create complex objects.

6. What is polymorphism?

Polymorphism is the ability of an object to take on different forms. This is achieved through the use of virtual methods.

7. What is a class?

A class is a blueprint for creating objects. It defines the data and methods that objects of that class will have.

8. What is an object?

An object is an instance of a class. It has its own state and behavior, and it can interact with other objects.

9. What is a method?

A method is a block of code that is associated with a class or object. It is used to perform a specific task.

10. What is a constructor?

A constructor is a special method that is used to initialize an object.

11. What is a static method?

A static method is a method that is associated with a class, not with an object. It can be called without creating an instance of the class.

12. What is a final method?

A final method cannot be overridden by a subclass.

13. What is a final class?

A final class cannot be inherited from.

## OOPS Lab (LC-CSE-256G)

---

14. What is a public method?

A public method can be accessed from anywhere in the program.

15. What is a private method?

A private method can only be accessed from within the class that it is defined in.

16. What is a protected method?

A protected method can be accessed from within the class that it is defined in, and from subclasses of that class.

17. What is an abstract method?

An abstract method is a method that has no implementation. It must be overridden by subclasses.

18. What is an interface?

An interface is a collection of abstract methods. It cannot be instantiated, but it can be used to define the behavior of a class.

19. What is a package?

A package is a collection of classes and interfaces. It is used to organize code and to control access to classes and interfaces.

20. What are the advantages of object-oriented programming?

The advantages of object-oriented programming include:

- a) Reusability of code
- b) Increased modularity
- c) Improved code readability and maintainability
- d) Increased flexibility and extensibility

21. What is JDBC?

JDBC stands for Java Database Connectivity. It is a Java API that allows Java programs to connect to a database and perform SQL operations. JDBC is a standard API, so it can be used to connect to any database that supports JDBC.

22. What are the different types of JDBC drivers?

There are four types of JDBC drivers:

JDBC-ODBC bridge driver: This driver uses the ODBC driver to connect to the database. It is the simplest type of JDBC driver, but it is also the least efficient.

Native-API driver: This driver uses the client-side libraries of the database. It is more efficient than the JDBC-ODBC bridge driver, but it requires the native libraries to be installed on each client machine.

Network Protocol driver: This driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is the most efficient type of JDBC driver, but it requires an application server to be installed.

Thin driver: This driver converts JDBC calls directly into the vendor-specific database protocol. It is entirely written in Java, so it is platform-independent.

23. What are the steps involved in using JDBC?

The basic steps involved in using JDBC are:

- a) Import the JDBC packages.
- b) Register the JDBC driver.

## OOPS Lab (LC-CSE-256G)

---

- c) Create a connection to the database.
- d) Create a statement.
- e) Execute the statement.
- f) Process the results.
- g) Close the connection.

24. What are the different types of JDBC statements?

- a) There are three types of JDBC statements:
- b) Statement: This is the simplest type of JDBC statement. It can be used to execute any SQL statement.
- c) PreparedStatement: This type of statement is pre-compiled, which makes it more efficient than a regular statement.
- d) CallableStatement: This type of statement is used to execute stored procedures.

25. What are the benefits of using JDBC?

- a) There are many benefits to using JDBC, including:
- b) Platform independence: JDBC is a standard API, so it can be used to connect to any database that supports JDBC.
- c) Performance: JDBC is a well-optimized API, so it can be used to perform database operations efficiently.
- d) Ease of use: JDBC is a relatively easy API to use, even for beginners.