**DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT**



**DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS**

**LAB MANUAL**

**Academic year 2024-2026 (Odd semester)**

**Semester –I**

**Course code** MMC104

**Object-Oriented Programming Using Java**

**Prepared by**

**Dr.Chitra K**

**Mr.Diwakara Vasuman**

**HOD-MCA Principal**

**Dr. Manjula Sanjay Dr. Ravi Shankar M**

**Vision of the Institute**

To be a center of excellence in education, research & training and to produce citizens with exceptional leadership qualities to serve national and global needs

**Mission of the Institute**

To achieve our objectives in an environment that enhances creativity, innovation and scholarly pursuits while adhering to our vision.

**Computer Lab Rules and Regulations**

**DO’s**

* Come prepared to the Lab.
* Submit your Records to the faculty and sign in the Log Book on entering the Lab
* Observation books have to be brought for all the labs.
* Backlog exercises to be executed after completing regular exercises.
* Regularly attend all the labs
* Put the chairs back to its position before you leave.
* Treat all the devices with care and consideration.
* Behave in a responsible manner at all the times and maintain silence.
* Before leaving the lab shut down the system and rearrange the chairs
* Keep your premises clean

**DON’T**

* Using Mobile phones and pen drives
* Move around in the lab during the lab session.
* Tamper System Files or Try to access the Server.
* Write Records in the Lab
* Change the system assigned to you without the notice of the Lab Staff.
* Write on the table or mouse pads.
* Do not install or download any software or modify or delete any system files on any lab

**MCA Department**

**Vision of the Department**

Nurture Continuous Learning through research and innovations in the field of Computer Science, Technology and Applications, to build competent professionals.

**Mission of the Department**

* Create a learning environment to motivate students to build strong technology skills.
* Promote value based ethical practices in all facets of learning
* Instill Entrepreneurial collaborative thinking through structured interventions and industry participation.

**Program Education Outcome (PEO’s):**

**PEO1:** Analyze real life problems, design computing systems appropriate to its solutions that are technically sound, economically feasible and socially acceptable.

**PEO2:** Exhibit professionalism, ethical attitude, communication skills, team work in their profession and adapt to current trends by engaging in lifelong learning.

**PEO3:** Demonstrate Leadership and Entrepreneurship Skills by incorporating organizational goals.

**Program Outcome (PO’s):**

**PO1.** Computational Knowledge: Understand and apply mathematical foundation, computing and domain knowledge for the conceptualization of computing models from defined problems.

**PO2.** Problem Analysis: Ability to identify, critically analyze and formulate complex computing problems using fundamentals of computer science and application domains.

**PO3.** Design / Development of Solutions: Ability to transform complex business scenarios and contemporary issues into problems, investigate, understand and propose integrated solutions using emerging technologies

**PO4.** Conduct Investigations of Complex Computing Problems: Ability to devise and conduct experiments, interpret data and provide well informed conclusions.

**PO5.** Modern Tool Usage: Ability to select modern computing tools, skills and techniques necessary for innovative software solutions

**PO6.** Professional Ethics: Ability to apply and commit professional ethics and cyber regulations in a global economic environment.

**PO7.** Life-long Learning: Recognize the need for and develop the ability to engage in continuous learning as a Computing professional.

**PO8.** Project Management and Finance: Ability to understand, management and computing principles with computing knowledge to manage projects in multidisciplinary environments.

**Program Specific Outcomes (PSO’s):**

**PSO1:** The graduates of the Program will have skills to develop, deploy and maintain applications for desktop, web, mobile, cloud and cross platforms using modern tools and technologies.

**PSO2:** The graduates of the program analyze the societal needs to provide novel solutions through technological based research.

**Internal evaluation procedure**

**Marking Scheme for Internal Marks:**

Total Marks: 50

Allocation of 50 marks is as follows:

Continuous Evaluation – 30 Marks

CIE Marks - 20 Marks

**Continuous Evaluation:**

Observation-10, Record-10, execution-20, viva-10 - Total 50 Marks

Scale down to 30 Marks

CIE 1 – 100 Marks: Write up -20, Execution – 60, Viva -20

CIE 2 – 100 Marks: Write up -20, Execution – 60, Viva -20

Total Marks are scaled down to 20 Marks

**External evaluation procedure**

**Marking Scheme for External Marks:**

Total Marks: 100

**Division of 100 marks is as follows:**

Procedure: 15

Write up & Execution: 70

Viva: 15

**Course Outcomes**

|  |  |  |
| --- | --- | --- |
| **CO’s** | **Description** | **Revised Blooms Level** |
| **CO1** | Understand the Programming Concepts of java Programming. | **L1, L2** |
| **CO2** | Apply the proficiency in writing and executing Java programs using an integrated development environment (IDE). | **L3** |
| **CO3** | Analyze object-oriented programming principles such as classes, objects, inheritance, and polymorphism to create reusable and modular code. | **L4** |
| **CO4** | Develop an understanding of basic concepts of File handling | **L5** |
| **CO5** | Create a program using problem-solving skills by applying Java programming techniques to solve computational problems. | **L6** |

**Mapping of Course Outcomes with Program Outcomes and PSO**

Correlation levels: 1-Slight (Low) 2-Moderate (Medium) 3-Substantial (High)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CO’s** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** |
| **1** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| **2** | **2** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| **3** | **-** | **2** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **2** | **-** |
| **4** | **-** | **-** | **2** |  | **-** | **-** | **-** | **2** | **-** | **-** | **-** | **-** |
| **AVERAGE** | **2** | **2** | **2** | **2** | **-** | **-** | **-** | **2** | **-** | **-** | **2** | **-** |

**Mapping of CO v/s PSO:**

|  |  |  |
| --- | --- | --- |
| **CO’s** | **PSO1** | **PSO2** |
| **1** | **2** | **-** |
| **2** | **2** | **-** |
| **3** | **2** | **-** |
| **4** |  | **2** |
| **AVERAGE** | **2** | **2** |

**Justifications for CO-PO/PSO Mapping**

|  |  |  |
| --- | --- | --- |
| **Mapping** | **Low/Medium/ High** | **Justification** |
| **CO2-PO1** | **2** | Moderately mapped, students be able to understand and apply the Java Programming concepts |
| **CO3-PO2** | **2** | Moderately mapped, students be able to analyse the Java Programming concepts |
| **CO3-PO11** | **2** | Moderately mapped, students be able to analyse the Java Programming concepts |
| **CO4-PO3** | **2** | Moderately mapped, students be able to implement the Java Programming concepts |
| **CO4-PO8** | **2** | Moderately mapped, students be able to implement the Java Programming concepts |
| **CO1-PSO1** | **2** | Moderately mapped, students be able to understand the Java Programming concepts |
| **CO2-PSO1** | **2** | Moderately mapped, students be able to develop applications using the Java Programming concepts |
| **CO3-PSO1** | **2** | Moderately mapped, students be able to implement the Java Programming concepts |
| **CO4-PSO2** | **2** | Moderately mapped, students be able to implement the applications using Java Programming concepts |

COURSE PLAN

List of programs

Java Programming: MMC104

|  |  |  |
| --- | --- | --- |
| S.no | Programs/Experiment | Page no |
| 1 | Write a Java program to check whether the given number is Palindrome or not | 11 |
| 2 | Write a Java program  To find the area and circumference of the circle by accepting the radius from the user.  To accept a number and find whether the number is Prime or not | 12-13 |
| 3 | Write a Java program to demonstrate a division by zero exception | 14 |
| 4 | Write a Java program to implement Inner class and demonstrate its Access protection. | 15 |
| 5 | Write a Java program to demonstrate Constructor Overloading and Method Overloading. | 16-17 |
| 6 | Write a JAVA program to demonstrate Inheritance. Simple Program on Java for the implementation of Multiple inheritance using interfaces to calculate the area of a rectangle and triangle. | 18-19 |
| 7 | Write a JAVA program to create an enumeration Day of Week with seven values SUNDAY through SATURDAY. Add a method isWorkday() to the DayofWeek class that returns true if the value on which it is called is MONDAY through FRIDAY. For example, the call DayOfWeek. SUNDAY.isWorkDay () returns false. | 20 |
| 8 | Write a Java program that displays the number of characters, lines and words in a text file. | 21-22 |
| 9 | Create a package named shape and Create some classes in the package representing some common shapes like Square, Triangle, and Circle. Import and compile these classes in other program. | 23-28 |
| 10 | Write a Java applet program, which handles keyboard event. | 29-30 |

Note 1: In the practical Examination each student has to pick one question from a lot of all 8 questions.

Note 2: Change of program is not permitted in the Practical Examination.

**Software and hardware requirements**

* **Software Requirements:**  Eclipse, NetBeans
* **Hardware** **Requirements**

|  |  |
| --- | --- |
| **Operating System** | Windows 7 |
| **RAM** | 64 MB |
| **Free disk space** | 100 MB free disk space |
| **Intel based desktop** | 166 MHz |

|  |  |
| --- | --- |
| **Java Programming Lab**  **Choice Based Credit System** | |
| **Semester:** | I |
| **CIE Marks:** | 50M |
| **Course Code:** | 24MCAI14 |
| **Course Outcomes: At the end of the course students will be able to**  **CO1:** Understand the Programming Concepts of java Programming.  **CO2:** Apply the proficiency in writing and executing Java programs using an integrated development environment (IDE).  **CO3:** Analyse object-oriented programming principles such as classes, objects, inheritance, and polymorphism to create reusable and modular code.  **CO4:** Develop an understanding of basic concepts of File handling**.**  **CO5:** Create a program using problem-solving skills by applying Java programming techniques to solve computational problems. | |

**1.Write a Java program to check whether the given number is Palindrome or not**

import java.util.Scanner;

public class PalindromeCheck {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input the number

System.out.print("Enter a number: ");

int number = scanner.nextInt();

// Store the original number for comparison

int originalNumber = number;

int reversedNumber = 0;

// Reverse the number

while (number != 0) {

int digit = number % 10;

reversedNumber = reversedNumber \* 10 + digit;

number /= 10;

}

// Check if the original number is equal to the reversed number

if (originalNumber == reversedNumber) {

System.out.println(originalNumber + " is a Palindrome.");

} else {

System.out.println(originalNumber + " is not a Palindrome.");

}

scanner.close();

}

}

**Output:**

Enter a number : 121

121 is a Palidrome

**2. Write a Java program**

* **To find the area and circumference of the circle by accepting the radius from the user.**
* **To accept a number and find whether the number is Prime or not**

import java.util.Scanner;

public class CircleAreaAndPrimeCheck {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Calculate the area and circumference of a circle

System.out.print("Enter the radius of the circle: ");

double radius = scanner.nextDouble();

double area = Math.PI \* radius \* radius;

double circumference = 2 \* Math.PI \* radius;

System.out.println("Area of the circle: " + area);

System.out.println("Circumference of the circle: " + circumference);

// Check if a number is prime or not

System.out.print("Enter a number: ");

int number = scanner.nextInt();

boolean isPrime = true;

if (number <= 1) {

isPrime = false;

}

else

{

for (int i = 2; i <= Math.sqrt(number); i++)

{

if (number % i == 0)

{

isPrime = false;

break;

}

}

}

if (isPrime)

System.out.println(number + " is a prime number.");

else

System.out.println(number + " is not a prime number.");

scanner.close();

}

}

**Output**

Enter the radius of the circle: 5

Area of the circle: 78.53981633974483

Circumference of the circle: 31.41592653589793

Enter a number: 17

17 is a prime number.

**3. Write a Java program to demonstrate a division by zero exception.**

public class DivisionByZeroExceptionDemo {

public static void main(String[] args) {

try {

int numerator = 10;

int denominator = 0;

int result = numerator / denominator;

System.out.println("Result: " + result);

} catch (ArithmeticException e) {

System.out.println("Exception caught: Division by zero is not allowed!");

}

}

}

**Output**

Exception caught: Division by zero is not allowed!

**4. Write a Java program to implement Inner class and demonstrate its Access protection.**

class Outer {

int a = 10;

public int b = 20;

private int c = 30;

protected int d = 40;

class inner {

int p = 5;

public int q = 15;

private int r = 25;

protected int s = 35;

void display() {

System.out.println("Inner class");

System.out.println("Value of a=" + a);

System.out.println("Value of public variable" + b);

System.out.println("Value of private variable" + c);

System.out.println("Value of protected variable" + d);

}

}

void outermet() {

inner inn = new inner();

inn.display();

System.out.println("Outer class");

System.out.println("Value of p=" + inn.p);

System.out.println("Value of public variable" + inn.q);

System.out.println("Value of private variable" + inn.r);

System.out.println("Value of protected variable" + inn.s);

}

}

public class lab02 {

public static void main(String[] args) {

{

Outer ot = new Outer();

ot.outermet();

}

}

}

**Output:**

Inner class Value of a=10

Value of public variable 20 Value of private variable 30 Value of protected variable 40 Outer class

Value of p=5

Value of public variable 15 Value of private variable 25 Value of protected variable 35

**5. Write a Java program to demonstrate Constructor Overloading and Method Overloading.**

class cube {

double l, b, h;

//constructors

cube() {

System.out.println("Constructor with no arguments\n");

l = 0;

b = 0;

h = 0;

}

cube(double i) {

System.out.println("Constructor with one arguments\n");

l = b = h = i;

}

cube(double x, double y, double z)

{

System.out.println("Constructor with three arguments\n");

l = x;

b = y;

h = z;

}

//methods

void area() {

System.out.println("Method with no arguments");

double a = l \* b \* h;

System.out.println("Area of Cube is " + a + "\n");

}

void area(double t) {

System.out.println("Method with one arguments");

double a = t \* t \* t;

System.out.println("Area of Cube is " + a + "\n");

}

void area(double p, double q, double r)

{

System.out.println("Method with three arguments");

double a = p \* q \* r;

System.out.println("Area of Cube is " + a + "\n");

}

}

public class lab01 {

public static void main(String[] args)

{

cube A = new cube(5, 6, 7);

A.area();

cube B = new cube(5);

B.area();

cube C = new cube();

C.area();

C.area(6);

C.area(6, 7, 8);

}

}

**Output:**

Constructor with three arguments

Method with no arguments

Area of Cube is 210.0

Constructor with one arguments

Method with no arguments

Area of Cube is 125.0

Constructor with no arguments

Method with no arguments

Area of Cube is 0.0

Method with one argument

Area of Cube is 216.0

Method with three arguments

Area of Cube is 336.0

**6. Write a JAVA program to demonstrate Inheritance. Simple Program on Java for the implementation of Multiple inheritance using interfaces to calculate the area of a rectangle and triangle.**

// Parent class

abstract class Shape {

String name;

// Constructor for the Shape class

Shape(String name) {

this.name = name;

}

// Abstract method to calculate area

abstract double calculateArea();

// Method to display shape details

void displayInfo() {

System.out.println("Shape: " + name);

}

}

// Child class for Rectangle

class Rectangle extends Shape {

double length, width;

// Constructor for Rectangle class

Rectangle(double length, double width) {

super("Rectangle");

this.length = length;

this.width = width;

}

// Overriding the calculateArea method

@Override

double calculateArea() {

return length \* width;

}

}

// Child class for Circle

class Circle extends Shape {

double radius;

// Constructor for Circle class

Circle(double radius) {

super("Circle");

this.radius = radius;

}

// Overriding the calculateArea method

@Override

double calculateArea() {

return Math.PI \* radius \* radius;

}

}

// Main class

public class ShapeInheritanceDemo {

public static void main(String[] args) {

// Create an object of the Rectangle class

Rectangle rectangle = new Rectangle(5, 3);

rectangle.displayInfo();

System.out.println("Area: " + rectangle.calculateArea());

System.out.println();

// Create an object of the Circle class

Circle circle = new Circle(4);

circle.displayInfo();

System.out.println("Area: " + circle.calculateArea());

}

}

**Output:**

Shape: Rectangle

Area: 15.0

Shape: Circle

Area: 50.26548245743669

// Interface 1

interface Engine {

void startEngine(); // Abstract method for starting the engine

}

// Interface 2

interface Transmission {

void changeGear(int gear); // Abstract method for changing the gear

}

// Class implementing multiple interfaces

class Car implements Engine, Transmission {

private int currentGear;

// Implementing the startEngine method from Engine interface

@Override

public void startEngine() {

System.out.println("Engine started.");

}

// Implementing the changeGear method from Transmission interface

@Override

public void changeGear(int gear) {

currentGear = gear;

System.out.println("Gear changed to: " + currentGear);

}

// Additional method in the Car class

void drive() {

System.out.println("Car is driving.");

}

}

// Main class

public class MultipleInheritanceDemo {

public static void main(String[] args) {

// Create an object of the Car class

Car myCar = new Car();

// Using methods from both interfaces

myCar.startEngine(); // From Engine interface

myCar.changeGear(2); // From Transmission interface

myCar.drive(); // Class-specific method

}

}

Output :-

Engine started.

Gear changed to: 2

Car is driving.

**7. Write a JAVA program to create an enumeration Day of Week with seven values SUNDAY through SATURDAY. Add a method isWorkday() to the DayofWeek class that returns true if the value on which it is called is MONDAY through FRIDAY. For example, the call DayOfWeek. SUNDAY.isWorkDay () returns false.**

public class DayOfWeekTest {

// Enumeration for Day of Week

public enum DayOfWeek {

SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY;

// Method to check if it's a workday (Monday to Friday)

public boolean isWorkday() {

return this != SATURDAY && this != SUNDAY;

}

}

public static void main(String[] args) {

// Test the isWorkday() method

System.out.println(DayOfWeek.SUNDAY + " is workday: " + DayOfWeek.SUNDAY.isWorkday()); // false

System.out.println(DayOfWeek.MONDAY + " is workday: " + DayOfWeek.MONDAY.isWorkday()); // true

System.out.println(DayOfWeek.FRIDAY + " is workday: " + DayOfWeek.FRIDAY.isWorkday()); // true

System.out.println(DayOfWeek.SATURDAY + " is workday: " + DayOfWeek.SATURDAY.isWorkday()); // false

}

}

Output:

SUNDAY is workday: false

MONDAY is workday: true

FRIDAY is workday: true

SATURDAY is workday: false

8. Write a Java program that displays the number of characters, lines and words in a text file

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class TextFileAnalyzer {

public static void main(String[] args) {

String fileName = "C:\\Users\\diwak\\OneDrive\\Desktop\\Files\\College Data\\DSATM\\NBA MCA DSATM\\New Text Document.txt"; // Change this to your file name

try {

BufferedReader reader = new BufferedReader(new FileReader(fileName));

int charCount = 0;

int wordCount = 0;

int lineCount = 0;

String line;

while ((line = reader.readLine()) != null) {

lineCount++;

String[] words = line.split("\\s+"); // Split by whitespace

wordCount += words.length;

for (String word : words) {

charCount += word.length();

}

}

reader.close();

System.out.println("Number of characters: " + charCount);

System.out.println("Number of words: " + wordCount);

System.out.println("Number of lines: " + lineCount);

} catch (IOException e) {

System.err.println("Error reading the file: " + e.getMessage());

}

}

}

Output:

Number of characters: 84

Number of words: 16

Number of lines: 4

**9.** Create a package named shape and Create some classes in the package representing some common shapes like Square, Triangle, and Circle. Import and compile these classes in other program.

Program (Rectangle1.java)

package Shape;

public class Rectangle1 {

private double length, breadth;

public void setRectangle(double len, double br) {

length = len;

breadth = br;

}

public void area() {

double area = length \* breadth;

System.out.println("Area of Rectangle =" + area);

}

}

Program (Square.java)

package Shape;

public class Square {

private double side;

public void setSquare(double val) {

side = val;

}

public void area() {

System.out.println("Area of Square=" + (side \* side));

}

}

Program(Circle1.java)

package Shape;

public class Circle1 {

private double rad;

public void setCircle(double radius) {

rad = radius;

}

public void area() {

double area = 3.14 \* rad \* rad;

System.out.println("Area of Rectangle =" + area);

}

}

Program(Program10.java)

import Shape.Rectangle1;

import Shape.Square;

import Shape.Circle1;

public class Program10 {

public static void main(String args[]) {

Rectangle1 rect = new Rectangle1();

rect.setRectangle(5.6, 6.4);

rect.area();

Square sq = new Square();

sq.setSquare(10.5);

sq.area();

Circle1 round = new Circle1();

round.setCircle(5.6);

round.area();

}

}

Output

javac -d . Rectangle1.java

javac -d . Circle1.java

javac -d . Square.java

javac Program10.java

java Program10

Area of Rectangle =35.839999999999996

Area of Square=110.25

Area of Circle=98.4704

**10. Write a Java applet program, which handles keyboard event.**

import java.awt.\*; //Contains classes for GUI components like Frame, Label, Font

import java.awt.event.\*; //Provides event handling classes like KeyListener, KeyEvent

public class Program9 extends Frame implements KeyListener {

Label lbl;

Program9() {

addKeyListener(this); // Registers the frame to listen for keyboard events.

requestFocus(); // Ensures the frame receives keyboard input

lbl = new Label(); // Creates label

lbl.setBounds(100, 100, 200, 40); // Position and size

lbl.setFont(new Font("Calibri", Font.BOLD, 16)); // Font settings

add(lbl); // Adds label to frame

setSize(400, 300); // Frame size

setLayout(null); // No layout manager

setVisible(true); // Makes frame visible

}

public void keyPressed(KeyEvent e) {

if (e.getKeyChar() == 'M' || e.getKeyChar() == 'm')

lbl.setText("Good morning");

else if (e.getKeyChar() == 'A' || e.getKeyChar() == 'a')

lbl.setText("Good afternoon");

else if (e.getKeyChar() == 'E' || e.getKeyChar() == 'e')

lbl.setText("Good evening");

else if (e.getKeyChar() == 'N' || e.getKeyChar() == 'n')

lbl.setText("Good night");

}

public void keyReleased(KeyEvent e) {

}

public void keyTyped(KeyEvent e) {

}

public static void main(String[] args) {

new Program9();

}

}

**Output**

1. Pressing 'M' or 'm': The window's label will display "Good Morning".
2. Pressing 'A' or 'a': The window's label will display "Good Afternoon".
3. Pressing 'E' or 'e': The window's label will display "Good Evening".
4. Pressing 'N' or 'n': The window's label will display "Good Night".

**VIVA QUESTIONS**

**JAVA PROGRAMMING**

1.Why is Java a platform independent language?

[**Java language**](https://www.interviewbit.com/blog/features-of-java/) was developed in such a way that it does not depend on any hardware or software due to the fact that the [**compiler**](https://www.interviewbit.com/online-java-compiler/) compiles the code and then converts it to platform-independent byte code which can be run on multiple systems.

The only condition to run that byte code is for the machine to have a runtime environment (JRE) installed in it

2. Why is Java not a pure object oriented language?

Java supports primitive data types - byte, boolean, char, short, int, float, long, and double and hence it is not a pure [**object oriented language**](https://www.interviewbit.com/oops-interview-questions/).

3. Difference between Heap and Stack Memory in java. And how java utilizes this.

Stack memory is the portion of memory that was assigned to every individual program. And it was fixed. On the other hand, Heap memory is the portion that was not allocated to the java program but it will be available for use by the java program when it is required, mostly during the runtime of the program.

4.How is Java different from C++?

C++ is only a  compiled language, whereas Java is compiled as well as an interpreted language.

Java programs are machine-independent whereas a c++ program can run only in the machine in which it is compiled.

C++ allows users to use pointers in the program. Whereas java doesn’t allow it. Java internally uses pointers.

C++ supports the concept of Multiple inheritances whereas Java doesn't support this. And it is due to avoiding the complexity of name ambiguity that causes the diamond proble

5. What do you mean by data encapsulation?

Data Encapsulation is an Object-Oriented Programming concept of hiding the data attributes and their behaviours in a single unit.

It helps developers to follow modularity while developing software by ensuring that each object is independent of other objects by having its own methods, attributes, and functionalities.

It is used for the security of the private properties of an object and hence serves the purpose of data hiding.

.6. Define Copy constructor in java.

Copy Constructor is the constructor used when we want to initialize the value to the new object from the old object of the same class.

7. Comment on method overloading and overriding.

In Java, **method overloading** is made possible by introducing different methods in the same class consisting of the same name. Still, all the functions differ in the number or type of parameters. It takes place inside a class and enhances program readability.

 8. Explain the use of final keyword in variable, method and class.

In Java, the final keyword is used as defining something as constant /final and represents the non-access modifier.

**final variable:**

When a variable is declared as final in Java, the value can’t be modified once it has been assigned.

If any value has not been assigned to that variable, then it can be assigned only by the constructor of the class.

**final method:**

A method declared as final cannot be overridden by its children's classes.

A constructor cannot be marked as final because whenever a class is inherited, the constructors are not inherited. Hence, marking it final doesn't make sense. Java throws compilation error saying - modifier final not allowed here

**final class:**

No classes can be inherited from the class declared as final. But that final class can extend other classes for its usage.

9.When can you use super keyword?

The super keyword is used to access hidden fields and overridden methods or attributes of the parent class.

Following are the cases when this keyword can be used:

Accessing data members of parent class when the member names of the class and its child subclasses are same.

To call the default and parameterized constructor of the parent class inside the child class.

Accessing the parent class methods when the child classes have overridden them.

The following example demonstrates all 3 cases when a super keyword is used.

10.What is the main objective of garbage collection?

The main objective of this process is to free up the memory space occupied by the unnecessary and unreachable objects during the Java program execution by deleting those unreachable objects.

11.What is a ClassLoader?

Java Classloader is the program that belongs to JRE (Java Runtime Environment). The task of ClassLoader is to load the required classes and interfaces to the JVM when required.

**Example-** To get input from the console, we require the scanner class. And the Scanner class is loaded by the ClassLoader.