TASK 05

1.What are the four main principles of Object-Oriented Programming? Explain each with an example.

1. INHERITANCE

class Animal {

void sound() {

System.out.println("Animal makes a sound");

}

}

// Child class inherits Animal

class Dog extends Animal {

void bark() {

System.out.println("Dog barks");

}

}

1. DATA ABSTRACTION

bstract class Shape {

abstract void draw(); // Abstract method

}

class Circle extends Shape {

void draw() {

System.out.println("Drawing Circle");

}

}

public class Main {

public static void main(String[] args) {

Shape s = new Circle(); // Polymorphic behavior

s.draw(); // Output: Drawing Circle

}

}

1. POLYMORPHISM

class Animal {

void makeSound() {

System.out.println("Animal sound");

}

}

class Cat extends Animal {

void makeSound() {

System.out.println("Cat meows");

}

}

1. ENCAPSULATION

class Employee {

private String name;

private double salary;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

if (salary > 0)

this.salary = salary;

}

}

2. How does inheritance work in OOP? Write a sample program to demonstrate single and multilevel inheritance.

**Inheritance** allows a class (called the **child** or **subclass**) to inherit properties and behaviors (fields and methods) from another class (called the **parent** or **superclass**).

**✅ Benefits of Inheritance:**

* Promotes **code reuse**.
* Supports **hierarchical classification**.
* Enables **polymorphism** (dynamic method binding).

3. What is the difference between method overloading and method overriding? Provide code examples.

Method overloading :

* same method name but different parameters
* occurs in same class
* parameters must be different

class Calculator {

int add(int a, int b) {

return a + b;

}

double add(double a, double b) {

return a + b;

}

int add(int a, int b, int c) {

return a + b + c;

}

}

public class Main {

public static void main(String[] args) {

Calculator calc = new Calculator();

System.out.println(calc.add(5, 10)); // 15

System.out.println(calc.add(2.5, 3.5)); // 6.0

System.out.println(calc.add(1, 2, 3)); // 6

}

}

|  |
| --- |
|  |

Method Overriding :

\* same method name and signatures

\* occurs in subclass

\* parameters must be same

class Animal {

void makeSound() {

System.out.println("Animal makes a sound");

}

}

class Dog extends Animal {

@Override

void makeSound() {

System.out.println("Dog barks");

}

}

public class Main {

public static void main(String[] args) {

Animal a = new Dog(); // Upcasting

a.makeSound(); // Output: Dog barks (runtime decision)

}

}

4. What is encapsulation and how does it help in software development? Show how it is implemented in code.

Wrapping of data and methods in a single unit is called encapsulation. It is usually achieved by access modifiers.

It has several benefits in software development

1. Increased security
2. Increased maintainability
3. Data hiding
4. Control

public class Person {

// Private variables - data hiding

private String name;

private int age;

// Public getter method for 'name'

public String getName() {

return name;

}

// Public setter method for 'name'

public void setName(String name) {

this.name = name;

}

// Public getter method for 'age'

public int getAge() {

return age;

}

// Public setter method for 'age' with validation

public void setAge(int age) {

if (age > 0) {

this.age = age;

} else {

System.out.println("Age must be positive.");

}

}

}