Task 05 - Object-Oriented Programming (OOP) Concepts

# 1. Four Main Principles of Object-Oriented Programming

The four main principles of OOP are:

1. 1. Encapsulation

Encapsulation is the concept of wrapping data and methods into a single unit called a class. It restricts direct access to some of the object's components, which can prevent the accidental modification of data.

Example:

class Student {  
 private String name;  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
}

1. 2. Inheritance

Inheritance allows a class to acquire the properties and methods of another class. It promotes code reusability and establishes a relationship between the parent and child classes.

Example:

class Animal {  
 void eat() {  
 System.out.println("This animal eats food.");  
 }  
}  
  
class Dog extends Animal {  
 void bark() {  
 System.out.println("Dog barks.");  
 }  
}

1. 3. Polymorphism

Polymorphism allows methods to do different things based on the object it is acting upon. It can be achieved through method overloading and method overriding.

Example:

// Overloading  
class MathOperation {  
 int add(int a, int b) {  
 return a + b;  
 }  
 double add(double a, double b) {  
 return a + b;  
 }  
}

1. 4. Abstraction

Abstraction hides complex implementation details and shows only the necessary features of an object.

Example:

abstract class Shape {  
 abstract void draw();  
}  
  
class Circle extends Shape {  
 void draw() {  
 System.out.println("Drawing Circle");  
 }  
}

# 2. Inheritance in OOP

Inheritance allows a class (child class) to inherit fields and methods from another class (parent class).

Single Inheritance Example:

class Animal {  
 void eat() {  
 System.out.println("This animal eats food.");  
 }  
}  
  
class Dog extends Animal {  
 void bark() {  
 System.out.println("Dog barks.");  
 }  
}

Multilevel Inheritance Example:

class Animal {  
 void eat() {  
 System.out.println("Animal eats food.");  
 }  
}  
  
class Dog extends Animal {  
 void bark() {  
 System.out.println("Dog barks.");  
 }  
}  
  
class Puppy extends Dog {  
 void weep() {  
 System.out.println("Puppy weeps.");  
 }  
}

# 3. Method Overloading vs Method Overriding

Method Overloading: Multiple methods with the same name but different parameters within the same class.  
Method Overriding: A subclass provides a specific implementation of a method already defined in its superclass.

Example of Method Overloading:

class Print {  
 void show(int a) {  
 System.out.println(a);  
 }  
  
 void show(String b) {  
 System.out.println(b);  
 }  
}

Example of Method Overriding:

class Animal {  
 void sound() {  
 System.out.println("Animal makes a sound");  
 }  
}  
  
class Dog extends Animal {  
 void sound() {  
 System.out.println("Dog barks");  
 }  
}

# 4. Encapsulation and Its Benefits

Encapsulation binds data and code together and keeps both safe from outside interference and misuse. It helps in achieving modularity and maintainability in software development.

Example of Encapsulation:

class Employee {  
 private int id;  
 private String name;  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
}