**OOPS Concepts**

**Inheritance**- Inheritance is a concept in object-oriented programming (OOP) where a new class is based on an existing class .

// Parent class (superclass)

class Animal {

// Method to be overridden by subclasses

public void speak() {

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

}

}

// Child class (subclass) inheriting from Animal

class Dog extends Animal {

// Override speak method

@Override

public void speak() {

System.out.println("Woof!");

}

}

// Child class (subclass) inheriting from Animal

class Cat extends Animal {

// Override speak method

@Override

public void speak() {

System.out.println("Meow!");

}

}

// Main class to test inheritance

public class Main {

public static void main(String[] args) {

// Creating instances of the subclasses

Dog dog = new Dog();

Cat cat = new Cat();

// Calling the speak method of each instance

dog.speak(); // Output: Woof!

cat.speak(); // Output: Meow!

}

}

**Encapsulation**- Warping up whole program in one unit is known as encapsulation.

// Example of encapsulation in Java

// Class representing a Bank Account

class BankAccount {

private String accountNumber;

private double balance;

// Constructor

public BankAccount(String accountNumber, double balance) {

this.accountNumber = accountNumber;

this.balance = balance;

}

// Getter for accountNumber

public String getAccountNumber() {

return accountNumber;

}

// Getter for balance

public double getBalance() {

return balance;

}

// Method to deposit money into the account

public void deposit(double amount) {

balance += amount;

System.out.println(amount + " deposited. New balance: " + balance

**Abstraction**-Showing as much as information is needed to use the program.

**Polymophism**-It helps program to differentiate entities with similar names

1. Compile time Polymorphism-

When decision is made which method is call on compile time.

**Method overloading**-When a class have a two methods with name but different parameters attached to it.

class Calculator {

int add(int a, int b) {

return a + b;

}

double add(double a, double b) {

return a + b;

}

}

2) Runtime Polymorphism-

When the Decision is made at run time not on compile time.

Eg Method Overridding: When a Sub class defines a method which is also defined in super class this type is known as method over ridding

// Superclass

class Animal {

void sound() {

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

}

}

// Subclass

class Dog extends Animal {

void sound() {

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

}

}

**Access Modifers in oops**-

Access specifiers are special types of keywords that are used to specify or control the accessibility of entities like classes, methods, and so on. Private, Public, and Protected are examples of access specifiers or access modifiers.

The key components of OOPs, encapsulation and data hiding, are largely achieved because of these access specifiers.