**Day 2:**

**Encapsulation :**binding or wrapping data and code in single unit is known as encapsulation. Example : Class.

class Employee {

private int id;

private String name;

private float salary;

void setValue(int id, String name, float salary){

this.id =id;

this.name=name;

if(this.salary<0){

this.salary=8000;

}else {

this.salary =salary

}

}

void display() {

System.out.println

}

}

Inheritance :

Class OldClass { super or base class or parent class

Property

Behaviour

}

Class NewClass extends OldClass{ sub class or child class or derived class.

Property

Behaviour

}

Types of inheritance

1. Single inheritance : one super class and one sub class

Class A {}

Class B extends A{}

1. Multilevel inheritance : one super class and n number of sub classes

Class A {}

Class B extends A{}

Class C extends B{}

Class D extends C{}

1. **Multiple inheritance** : more than one super class and one sub class

Class A {}

Class B {}

Class C extends A,B{} not possible in java. It doesn’t using class it support using interface.

1. Hierarchical inheritance. : one super class and n number of sub classes connected directly to super class.

Class A {}

Class B extends A{}

Class C extends A {}

Class D extends A{}

**Oops relationship**

Manager/ Developer Is a Employee

Employee Has a Address

Class Employee {

Id,name,salary

}

Class Manager extends Employee{

numberOfEmp it hold only one value

Address add = new Address(); it can hold more than one value

}

Class Developer extends Employee {

projectName;

}

Class ProjectManager extends Manager{

clientInfo;

}

Class Address {

City,state

}

Has relationship

Association

Aggregation

Composition

Class A {

B obj = new B();

}

Class B {

A obj1 = new A();

}

Association

Class Developer {

Address add = new Address();

}

Class Address {

}

Class Office {

Address add = new Address();

}

It is a type of association but it is known as aggregation. Aggregation means weak association.

Class Student {

StudentHistory sh = new StudentHistory();

}

Class StudentHistory {

}

It is a type of association but it is known as composition. Composition means strong association.

**Polymorphism**

One name many forms or many implementation

Compile time or static binding or early binding:

Method overloading : the method have same name but different parameter list(type of parameter list or number of parameter list must be different).

Run time or dynamic binding or late binding

Method overriding : the method have same name and same method signature (number of parameter list, type of parameter list and return type must be same). Method overriding we can’t achieve in same class we need inheritance concept.

**Non access specifiers**

Add extra behaviour for class, method and variable

abstract : we can use with method and class not with variable.

final : final we can with variable, method and class.

static: we can use with method and variable but not with class.

**access specifiers**

they provide visibility or accessibility of variable, method and class.

private

default

protected

public

**package**