“OOPS IN JAVA”

✓ OOPs kya hota hai?

OOPs (Object Orineted Programming System) ek programming approach hai jisme program objects ke around design hota hai, na ki sirf functions ke.

👉 Java purely object-orineted nahi hai (kyuki primitive data types hain), but mostly OOP follow karta hai.

- ### Class

  Class ek blueprint/template hoti hai jisse objects bante hain.

```
class Student {
    int id;
    String name;
}
```

- ### Object

  Object class ka real-world instance hota hai.

```
Student s1 = new Student();
s1.id = 1;
s1.name = "Muskan";
```

- ### 4 Pillars of OOPs

  #### ➢ Encapsulation

  👉 Data+Methods ko ek unit me badhana= Encapsulation

  👉 Data ko Direct access se bachata hai (data hiding).

```
class Account {
    private int balance = 1000;

    public int getBalance() {
        return balance;
    }
}
```

- ✓ Private data secure
- ✓ Getter/Setter se access

👉 Ek class dusari class ki properties inherit karti hai

👉 extends keyword use hota hai

```
class Parent {
    void show() {
        System.out.println("Parent class");
    }
}

class Child extends Parent {
}
```

- ✓ Code reusability
- ✓ IS-A relationship

➢ Polymorphism

👉 Ek kaam, multiple forms

(a) Method Overloading (Compile Time)

Same method name, different parameters

```
class Demo {
    void add(int a, int b) {}
    void add(int a, int b, int c) {}
}
```

(b) Method Overriding (Run Time)

Parent method ko child class me change karna

```
class Parent {
    void show() {
        System.out.println("Parent");
    }
}

class Child extends Parent {
    void show() {
        System.out.println("Child");
    }
}
```

➢ Abstraction

👉 Implementation details hide karna

👉 Sirf what to do batana, how to do nahi

Abstract Class :

```
abstract class Shape {
    abstract void draw();
}
```

Interface :

```
interface Vehicle {
    void run();
}
```

✓ Interface = 100% abstraction
✓ Multiple inheritance possible

- Interface vs Abstract Class

Interface	Abstract Class
100% abstraction	Partial abstraction
implements	extends
Multiple inheritance	Single inheritance
No constructor	Constructor allowed

❖ `this` keyword

👉 Current object ko refer karta hai

```
class Test {
    int x;
    Test(int x) {
        this.x = x;
    }
}
```

❖ `super` keyword

👉 Parent class ke variable/ method ko access karta hai

```
class Parent {
    int a = 10;
}

class Child extends Parent {
    void show() {
        System.out.println(super.a);
    }
}
```

❖ Constructor

👉Object banate time automatically call hota hai

👉Same name as class

```
class Demo {
    Demo() {
        System.out.println("Constructor called");
    }
}
```

❖ Benefits of OOPs

- ✓ Code reusable
- ✓ Easy maintenance
- ✓ Secure
- ✓ Real-world modeling