Java oops concepts

Here's a **clear + interview-ready breakdown of Java OOP concepts** 👇
(think of this as your mental checklist for exams, interviews, or teaching)

🔹 What is OOP in Java?

Object-Oriented Programming (OOP) is a way of designing software using **objects** that contain **data (fields)** and **behavior (methods)**.

Java is **purely object-oriented** (except primitives).

🧱 1. Class

A **class** is a blueprint/template for creating objects.

java

```
class Car {
    String color;
    void drive() {
        System.out.println("Car is driving");
    }
}
```

📦 2. Object

Copy code

An **object** is an instance of a class.

java

```
Car c = new Car();
c.drive();
```

Copy code

🔐 3. Encapsulation

Wrapping data + methods together and controlling access using **access modifiers**.

✔️ Why?

- Data security
- Controlled access
- Cleaner code

java

```java
class Account {
    private double balance;

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

Copy code

👉 Achieved using:

- `private`
- `public`
- `protected`
- `default`

🧬 4. Inheritance

One class acquires properties of another class

java

```java
class Vehicle {
    void start() {}
}

class Bike extends Vehicle {
    void ride() {}
}
```

Copy code

✔️ Benefits

- Code reuse

- Hierarchy

❌ Java does NOT support multiple inheritance with classes

✔ Supports it via **interfaces**

🔄 5. Polymorphism

One thing, many forms

a) Compile-time (Method Overloading)

Same method name, different parameters

java

```java
int add(int a, int b) {}
double add(double a, double b) {}
```

Copy code

---

b) Runtime (Method Overriding)

Child class provides its own implementation

java

```java
class Parent {
    void show() {}
}

class Child extends Parent {
    @Override
    void show() {}
}
```

👉 Decided at **runtime** using **dynamic binding**

Copy code

🧠 6. Abstraction

Hiding implementation details and showing only essentials

a) Abstract Class

java

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

b) Interface

Copy code

java

```java
interface Flyable {
    void fly();
}
```

Key differences:

Copy code

Abstract Class	Interface
Can have constructors	No constructors
Can have variables	Variables are `public static final`
Supports inheritance	Supports multiple inheritance

🧩 7. Interface

Defines **what a class should do**, not **how**

java

```java
interface Payment {
    void pay();
}
```

java

```java
class UpiPayment implements Payment {
    public void pay() {}
}
```

Copy code

🔒 8. Access Modifiers

Modifier	Same Class	Same Package	Subclass	Everywhere
private	✔	✖	✖	✖
default	✔	✔	✖	✖
protected	✔	✔	✔	✖
public	✔	✔	✔	✔

🧪 9. Constructor

Used to initialize objects

java Copy code

```java
class Person {
    Person() {
        System.out.println("Created");
    }
}
```

Types:

- Default
- Parameterized
- Copy (manual)

🔁 10. Method Overloading vs Overriding

Overloading	Overriding
Same class	Parent-child
Compile-time	Runtime
Different parameters	Same signature
Inheritance not needed	Inheritance required

🧹 11. `this` and `super`

- `this` → current object
- `super` → parent class

java Copy code

```java
this.name = name;
super.show();
```

🧨 12. Final Keyword

- `final variable` → constant
- `final method` → cannot override
- `final class` → cannot extend

🗒️ 13. Object Class (Very Important)

Every class inherits from `Object`

Common methods:

- `toString()`
- `equals()`
- `hashCode()`
- `clone()`

🎯 Interview Tip (1-liner answers)

- **Encapsulation** → Data hiding
- **Inheritance** → Code reuse
- **Polymorphism** → One interface, many implementations
- **Abstraction** → Hide complexity

If you want, I can:

- 🔥 Give **real-world examples**
- 🧠 Share **interview questions & answers**

- 🧪 Provide **coding problems**
- 📚 Create **teaching notes / slides** (since you teach Java)

Just tell me how deep you want to go 😄