JAVA ABSTRACTION

🔶 What is Abstraction in Java?

Abstraction is the process of hiding the internal details and showing only the essential features of an object.

🔸 Real-Life Example:

When you drive a car, you only use the steering, brakes, and accelerator. You don't need to know how the engine works internally.

🔷 Why Use Abstraction?

- To reduce complexity.
- To increase security (by hiding data).
- To focus only on what an object does, not **how** it does it.

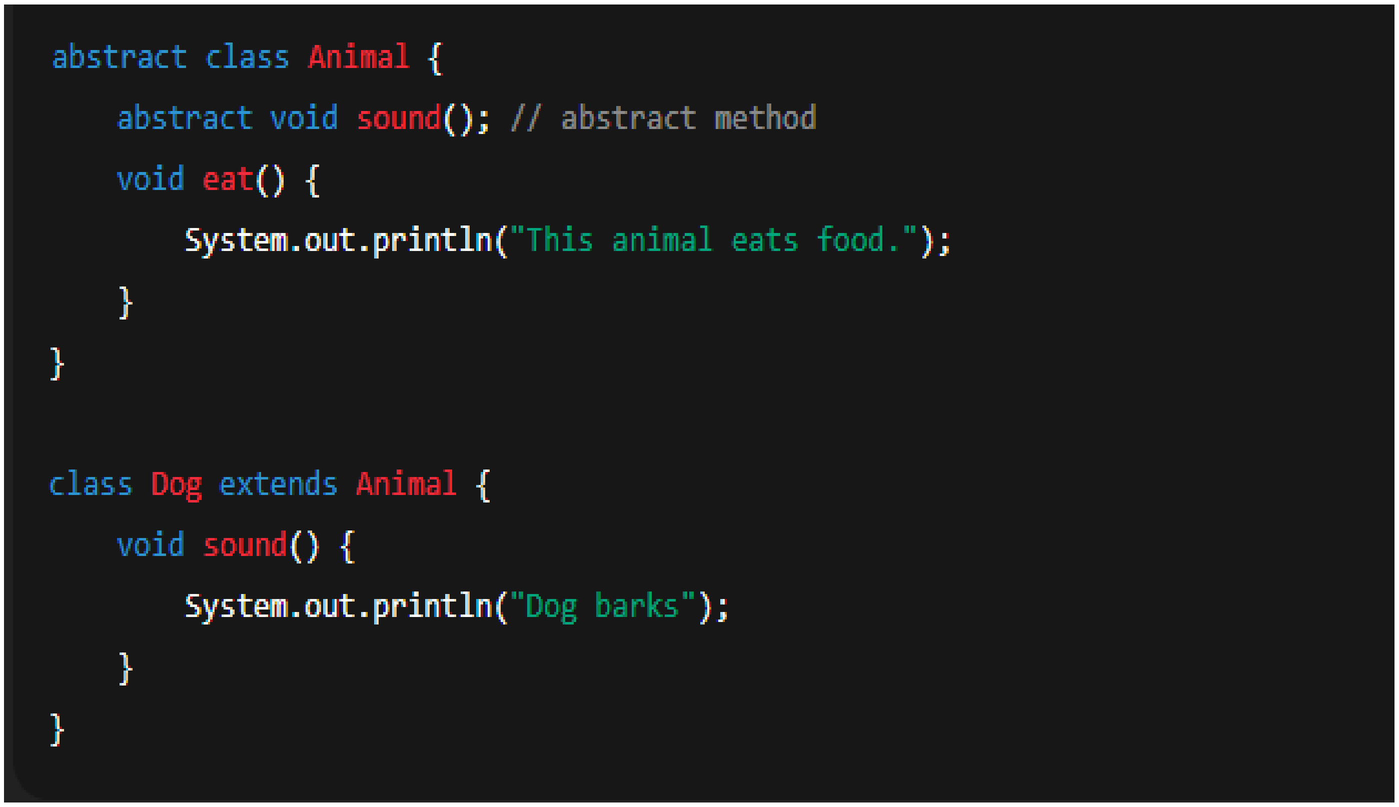
🔷 How to Achieve Abstraction in Java?

Java provides two ways to achieve abstraction:

1. **Abstract Classes (0–100% abstraction)**
2. **Interfaces (100% abstraction before Java 8, partial after)**

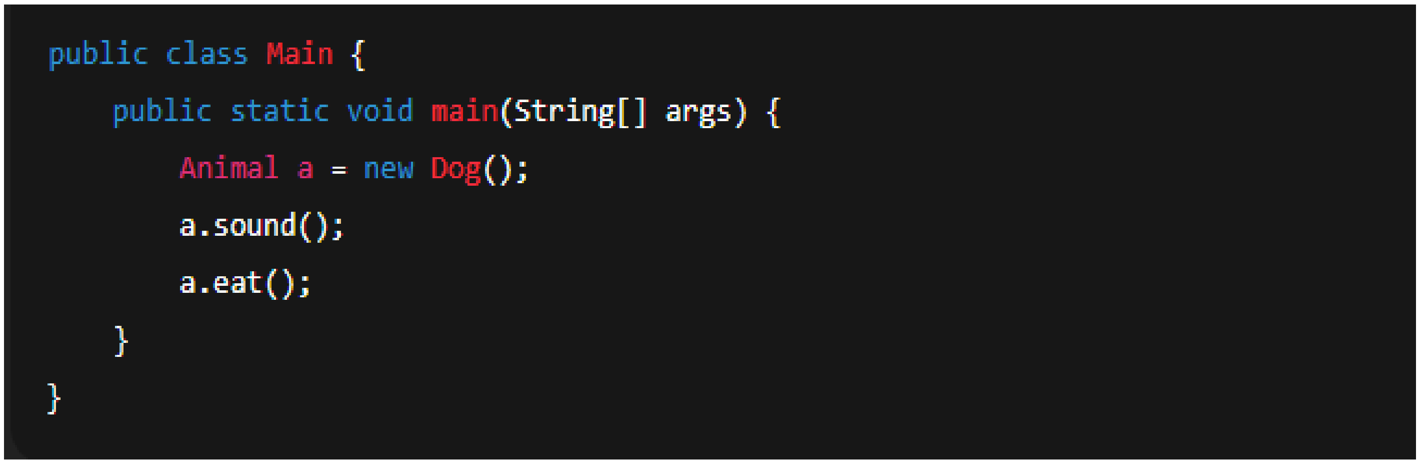
🔶 1. Abstract Class

- **A class declared with the abstract keyword.**
- **Can have abstract (no body) and non-abstract methods.**
- **Cannot be instantiated (no new object).**



```java
abstract class Animal {
    abstract void sound(); // abstract method
    void eat() {
        System.out.println("This animal eats food.");
    }
}

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



```java
public class Main {
    public static void main(String[] args) {
        Animal a = new Dog();
        a.sound();
        a.eat();
    }
}
```

🔶 2. Interface (From Java 8+)

- All methods are **abstract by default** (before Java 8).
- From Java 8, can have **default** and **static** methods.
- A class implements an interface using implements keyword.

```java
interface Animal {
    void sound();
}

class Cat implements Animal {
    public void sound() {
        System.out.println("Cat meows");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Animal a = new Cat();
        a.sound();
    }
}
```

🧠 Key Differences: Abstract Class vs Interface

Feature	Abstract Class	Interface
Keyword	abstract	interface
Constructors	Yes	No
Inheritance type	extends	implements
Multiple inheritance	Not supported	Supported (multiple interfaces)
Method types	Both abstract + normal	Only abstract (Java <8)

🔚 Summary

- **Abstraction = Hiding implementation, showing functionality**
- Use **abstract class** when:
    - You want some default behavior.
    - You need fields or constructors.
- Use **interface** when:
    - You want to support multiple inheritance.
    - You want full abstraction.