Inheritance in Java —

Inheritance means a child class can use the properties and methods of a parent class.

Think of it like:

Parent
↓
Child

For example, a **Dog is an Animal**. So Dog can inherit things from Animal.

Simple Example

```
class Animal {

    void eat() {
        System.out.println("Animal eats");
    }
}

class Dog extends Animal {

    void bark() {
        System.out.println("Dog barks");
    }
}

public class Main {

    public static void main(String[] args) {

        Dog d = new Dog();

        d.eat();   // inherited from Animal
        d.bark();  // Dog's own method
    }
}
```

Output

```
Animal eats
Dog barks
```

How does it work?

The important word is:

extends

```
class Dog extends Animal
```

This means:

Dog is a child of Animal and can use Animal's accessible methods and variables.

So Dog gets:

```
eat()
```

from Animal, and has its own:

```
bark()
```

Real-Life Example

Think about a company:

Employee
↓
Manager

An Employee might have:

void work()

A Manager can inherit work() and add:

void manageTeam()

Similarly:

Animal
↓
Dog
↓
Puppy

Inheritance allows us to **reuse existing code** instead of writing the same code again.

Types of Inheritance in Java

You should know these:

1. **Single inheritance**

```
Animal
  ↓
 Dog
```

5. **Multilevel inheritance**

```
Animal
  ↓
 Dog
  ↓
Puppy
```

11. **Hierarchical inheritance**

```
   Animal
   /   \
 Dog   Cat
```

Java **does not support multiple inheritance with classes**:

```
Animal   Bird
  \      /
    Dog
```

But multiple inheritance of type can be achieved using **interfaces**.

Easy memory trick

Inheritance = "Child gets features from Parent."

class Child extends Parent

So remember:

extends **= inheritance**.

Types of Inheritance in Java

Java mainly has these inheritance types:

15. Single inheritance
16. Multilevel inheritance
17. Hierarchical inheritance
18. Multiple inheritance — **not supported with classes**
19. Hybrid inheritance — **not directly supported with classes**

Let's see each with a simple example and output.

---

1. Single Inheritance

One parent → One child

Animal
↓
Dog

Example

```
class Animal {

    void eat() {
        System.out.println("Animal eats");
    }
}

class Dog extends Animal {

    void bark() {
        System.out.println("Dog barks");
    }
}

public class Main {

    public static void main(String[] args) {

        Dog d = new Dog();

        d.eat();
        d.bark();
    }
}
```

Output

Animal eats
Dog barks

Explanation

Dog inherits eat() from Animal.
class Dog extends Animal
So the Dog object can use both:
eat() → Animal
bark() → Dog

---

2. Multilevel Inheritance

Grandparent → Parent → Child

Animal
↓
Dog
↓
Puppy

Example

```
class Animal {

    void eat() {
        System.out.println("Animal eats");
    }
}

class Dog extends Animal {

    void bark() {
        System.out.println("Dog barks");
    }
}

class Puppy extends Dog {

    void play() {
        System.out.println("Puppy plays");
    }
}

public class Main {

    public static void main(String[] args) {

        Puppy p = new Puppy();

        p.eat();
        p.bark();
        p.play();
    }
}
```

Output

Animal eats

Dog barks
Puppy plays

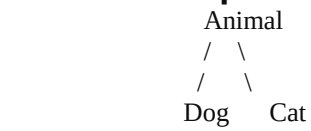
Explanation

Puppy gets:
eat() ← Animal
bark() ← Dog
play() ← Puppy
So inheritance happens through multiple levels.

---

3. Hierarchical Inheritance

One parent → Multiple children



Example

```
class Animal {

   void eat() {
      System.out.println("Animal eats");
   }
}

class Dog extends Animal {

   void bark() {
      System.out.println("Dog barks");
   }
}

class Cat extends Animal {

   void meow() {
      System.out.println("Cat meows");
   }
}

public class Main {

   public static void main(String[] args) {

      Dog d = new Dog();
      Cat c = new Cat();
```

```
        d.eat();
        d.bark();

        c.eat();
        c.meow();
    }
}
```

Output

```
Animal eats
Dog barks
Animal eats
Cat meows
```

Explanation

Both `Dog` and `Cat` inherit from `Animal`.

```
Dog → eat()
Cat → eat()
```

But they also have their own methods:

```
Dog → bark()
Cat → meow()
```

---

4. Multiple Inheritance

Multiple parents → One child

The structure would be:

```
Animal      Bird
  \        /
   \      /
    FlyingDog
```

Java **does not allow multiple inheritance using classes**.

For example, this is **not allowed**:

```
class Animal {
    void eat() {
        System.out.println("Animal eats");
    }
}

class Bird {
    void fly() {
        System.out.println("Bird flies");
    }
}
```

```
// □ Not allowed in Java
class FlyingDog extends Animal, Bird {
}
```

You will get a compilation error.

Why?

Java avoids problems where two parent classes could contain methods with the same name and Java wouldn't know which one to use.

But Java can achieve multiple inheritance using interfaces.

```
interface Animal {
   void eat();
}

interface Bird {
   void fly();
}

class FlyingDog implements Animal, Bird {

   public void eat() {
      System.out.println("Dog eats");
   }

   public void fly() {
      System.out.println("Dog flies");
   }
}

public class Main {

   public static void main(String[] args) {

      FlyingDog d = new FlyingDog();

      d.eat();
      d.fly();
   }
}
```

Output

```
Dog eats
Dog flies
```

Here:

```
class FlyingDog implements Animal, Bird
```

means one class implements multiple interfaces.

Quick Revision

Type	Structure	Supported with Classes?
Single	A → B	□ Yes
Multilevel	A → B → C	□ Yes
Hierarchical	A → B, C	□ Yes
Multiple	A, B → C	□ No
Hybrid	Combination	□ Not directly

□ Easy way to remember

Single → One parent, one child

Multilevel → Grandparent → Parent → Child

Hierarchical → One parent → Many children

Multiple → Many parents → One child □ classes