Java Inheritance

Table of contents

Introduced in 2026-03-05-object-oriented-programming-i[1] (Lecture, Week 2) — "things you have because your parents have them".

Subclasses and superclasses

Inheritance allows creating a new class from an existing class:

- The new class is the **subclass** (child/derived class).
- The existing class it's derived from is the **superclass** (parent/base class).

`extends` is the keyword used to implement inheritance in Java:

```
public class Employee {
    private String name;
    private String address;

    public Employee(String name, String address) {
        this.name = name;
        this.address = address;
    }
    public String getName() { return name; }
    public String getAddress() { return address; }
    public void setAddress(String address) { this.address = address; }
    public String printDetails() { return name + " " + address; }
}
```

[1] courses/csse2002/2026-03-05-object-oriented-programming-i.pdf

```java
public class ContractEmployee extends Employee {
    private double hourlyRate;
    private int hoursWorked;

    public ContractEmployee(String name, String address, double hourlyRate) {
        super(name, address);
        this.hourlyRate = hourlyRate;
        this.hoursWorked = 0;
    }
    public void logHours(int hours) { this.hoursWorked += hours; }
    public double calculateWeeklyPay() { return hourlyRate * hoursWorked; }
}
```

The `super` keyword

`super` is used in a subclass to access superclass members (attributes, constructors, and methods) — e.g. `super(name, address)` above calls `Employee`'s constructor to initialise the inherited fields before the subclass's own constructor body runs.

Method overriding

The subclass inherits the attributes and methods of its superclass. If the **same method signature** is defined in both, the subclass's version **overrides** the superclass's version:

```java
public class ContractEmployee extends Employee {
    @Override
    public String printDetails() {
        return "Role: Contract Employee, Hourly Rate: $" + hourlyRate + ", Hours
        ↪  Worked: " + hoursWorked;
    }
}

public class FullTimeEmployee extends Employee {
    @Override
    public String printDetails() {
        return "Role: Full-Time Employee, Monthly Salary: $" + monthlySalary;
    }
}
```

Both the superclass and subclass method MUST share the same method signature (see java-encapsulation[2]) for this to be overriding rather than a separate overload.

[2] courses/csse2002/java-encapsulation.pdf

`super.methodName(...)` can be used inside an override to still call the superclass's version of the method, e.g. to extend rather than replace its behaviour:

```
public class FullTimeEmployee extends Employee {
    @Override
    public String printDetails() {
        return super.printDetails() + ", Role: Full-Time Employee, Monthly Salary: $"
         ↪  + monthlySalary;
    }
}
```