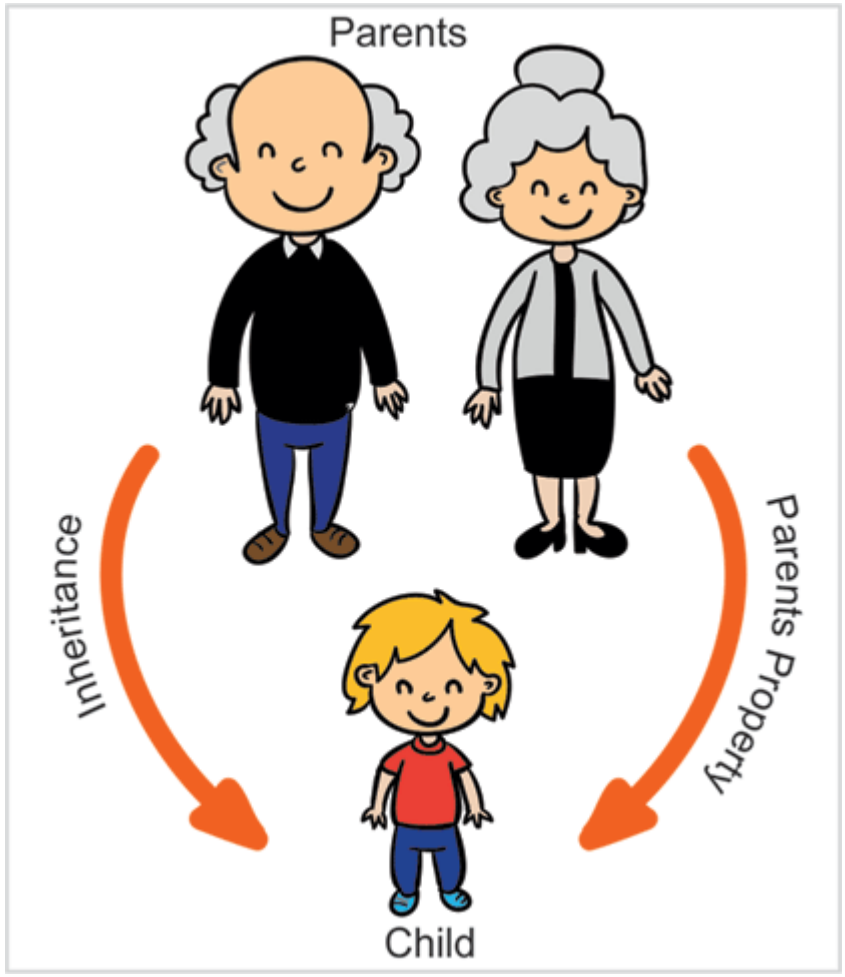


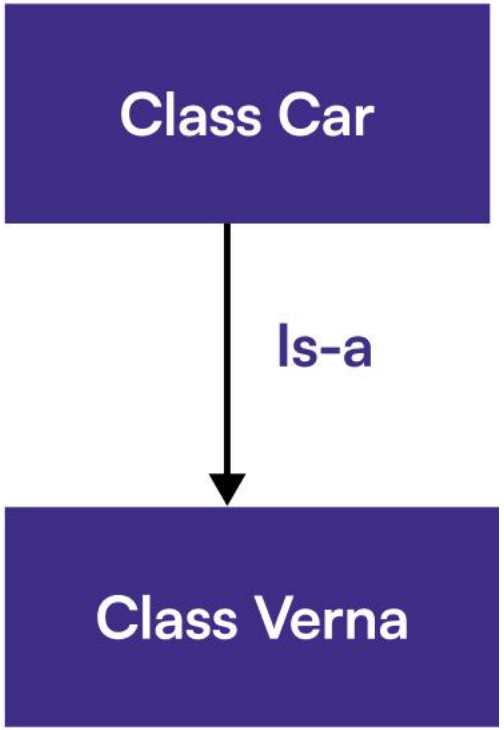
Inheritance in Java

Nimesha Hewawasam
nimesha.h@nsbm.ac.lk

What Is Inheritance?

- Inheritance allows a class (**subclass**) to **inherit fields and methods** from another class (**superclass**).
- Uses '**extends**' keyword.
- Represents an '**is-a**' relationship (*Car is-a Vehicle*)

```
class A {
  .........
        Is-A Relationship
  }
class  B  extends A
{
  .........
  .........
  }
```



Why Use Inheritance?

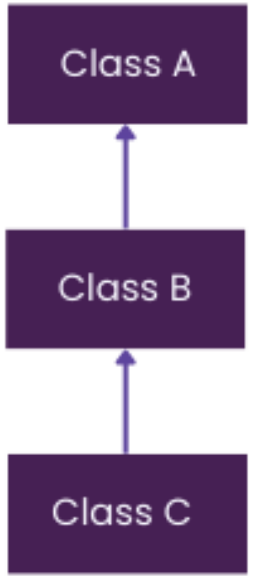
- **Code Reusability** – subclasses reuse base class functionality.
- **Structural Clarity** – reflects natural hierarchies.
- **Ease of Maintenance** – updates propagate from superclass to subclasses.

Real-World Analogy – Vehicles

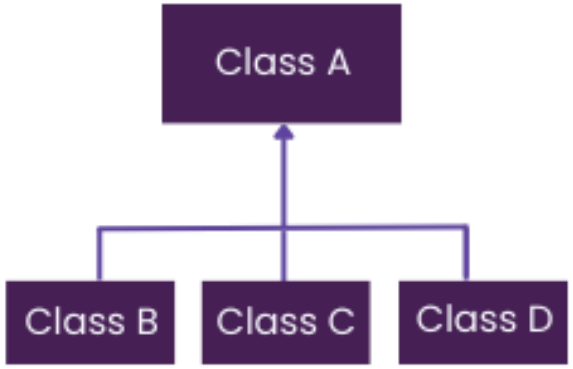
- **Superclass: Vehicle**
  - Common attributes: Number of wheels, engine type;
  - Common methods: start(), stop().
- **Subclasses**:
  - Car : 4 wheels, doors, seats
  - Bus : 6 wheels , doors, seats
  - Motorcycle: 2 wheels, no doors

Types of Inheritance in Java

Single Inheritance



Multilevel Inheritance



Hierarchical Inheritance

Types of Inheritance in Java

- Single Inheritance – one subclass from one superclass.

```java
package vehicleP;


public class Vehicle {
    void start(){
        System.out.println("Vehicle Starting");
    }

}
```

```java
package vehicleP;


public class Car extends Vehicle{
    void openDoor(){
        System.out.println("Door Open");
    }

    public static void main(String[] args) {
        Car obj = new Car();
        obj.openDoor(); // specific to the car
        obj.start(); // inherits from Vehicle class
    }

}
```

Types of Inheritance in Java

- Multilevel Inheritance – chain of inheritance.

```java
package vehicleP;


public class Vehicle {
    void start(){
        System.out.println("Vehicle Starting");
    }

}
```

```java
package vehicleP;


public class Car extends Vehicle{
    void openDoor(){
        System.out.println("Door Open");
    }

    public static void main(String[] args) {
        Car obj = new Car();
        obj.openDoor(); // specific to the car
        obj.start(); // inherits from Vehicle class
    }

}
```

```java
package vehicleP;


public class ElectricCar extends Car{
    void chargeBattery(){
        System.out.println("Electric car charging");
    }

    public static void main(String[] args) {
        ElectricCar obj3 = new ElectricCar();
        obj3.start(); // Inherit from Vehicle
        obj3.openDoor(); // Inherit from Car
        obj3.chargeBattery(); // Specific for this class
    }
}
```

Types of Inheritance in Java

- Hierarchical Inheritance – one superclass, many subclasses.

```java
package vehicleP;


public class Vehicle {
    void start(){
        System.out.println("Vehicle Starting");
    }

}
```

```java
package vehicleP;

public class Bick extends Vehicle{
    void kickStart() {
        System.out.println("Bike kick-started...");
    }
}
```

```java
package vehicleP;

public class Truck extends Vehicle{
     void loadCargo() {
        System.out.println("Truck loading cargo...");
     }

}
```

```java
package vehicleP;

public class Car extends Vehicle{
    void openDoor(){
        System.out.println("Door Open");
    }

}
```

Types of Inheritance in Java

```java
package vehicleP;

public class Main {
    public static void main(String[] args) {
        Car obj1 = new Car();
        Bick obj2 = new Bick();
        Truck obj3 = new Truck();

        //all share vehicle class start method()
        obj1.start();
        obj2.start();
        obj3.start();

        //Each has its own behaviours
        obj1.openDoor();
        obj2.kickStart();
        obj3.loadCargo();
    }
}
```

Vehicle : Super class
Car, Bick, Truck : Sub Classes

Types of Inheritance in Java

- ***Multiple inheritance*** allows a class to be derived from two or more classes, inheriting the members of all parents
- Java does not support multiple inheritance
- In most cases, the use of **interfaces** gives us aspects of multiple inheritance without the overhead

Interactive Exercise

- Identify real-world 'is-a' relationships (e.g., Animal → Dog, Cat).
- Bring examples from everyday life.
- Write small Java code demonstrating inheritance.

Summary

- Inheritance enables code reuse and logical hierarchies.
- Java supports multiple inheritance via interfaces only.
- Real-world analogies: vehicles, people, cups.