Types of Inheritance

In the previous lessons, you covered the basics of inheritance. In this lesson, you will learn about the various types of inheritance in Java.

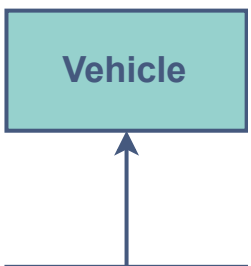
WE'LL COVER THE FOLLOWING

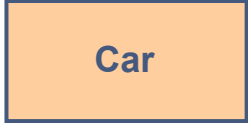
Based upon superclasses and subclasses, there are the following **five** types of inheritance in general:

1. **Single**
2. **Multi-level**
3. **Hierarchical**
4. **Multiple**
5. **Hybrid**

Single Inheritance

In single inheritance, there is only a single class extending from another class. We can take the example of the `Vehicle` class (Super class) and the `Car` class (Sub class). Let's implement these classes below:

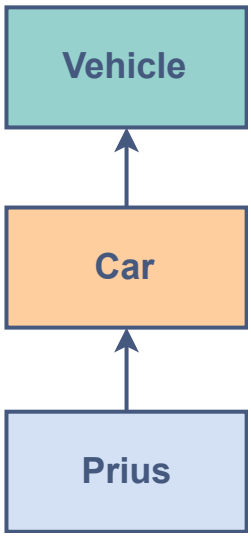





```
class Vehicle {          //Base Vehicle class

  private int topSpeed;
  public void setTopSpeed(int speed) {
    this.topSpeed=speed;
    System.out.println("The top speed is set to: "+ topSpeed);
  }

}

class Car extends Vehicle { // sub class Car extending from Vehicle

  public void openTrunk() {
    System.out.println("The Car trunk is Open Now");
  }

}

class Main {

  public static void main(String[] args) {
    Car corolla = new Car();
    corolla.setTopSpeed(220);
    corolla.openTrunk();
  }

}
```

Multi-level Inheritance

When a class is derived from such a class which itself is derived from another class, this type of inheritance is called Multilevel Inheritance. Classes can be extended to any further levels as per the requirement of the model.



Let's implement the three classes illustrated above:

- A `Car` **IS A** `Vehicle`
- A `Prius` **IS A** `Car`

```
class Vehicle {          //Base Vehicle class

  private int topSpeed;

  public void setTopSpeed(int speed) {
    this.topSpeed=speed;
    System.out.println("The top speed is set to: "+ topSpeed);
  }

}

class Car extends Vehicle { // Derived from Vehicle Base for Prius

  public void openTrunk() {
    System.out.println("The Car trunk is Open Now!");
  }

}

class Prius extends Car {// Derived from Prius & can be base to any further class

  public void turnOnHybrid() {
    System.out.println("The Hybrid mode is turned on!");
  }

}

class Main {

  public static void main(String[] args) {
    Prius priusPrime = new Prius();
    priusPrime.setTopSpeed(220);
    priusPrime.openTrunk();
    priusPrime.turnOnHybrid();
  }

}
```

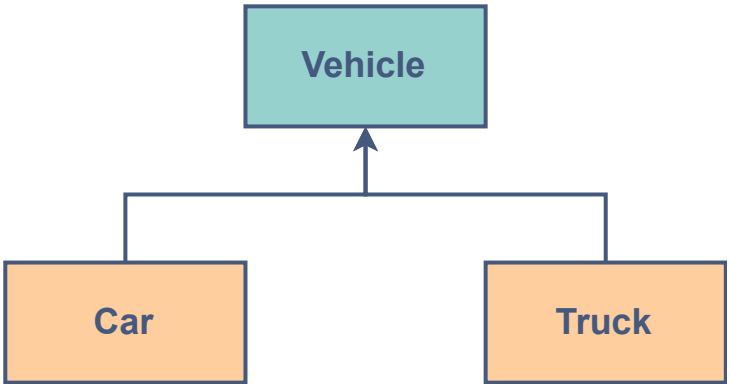
Hierarchical Inheritance

When more than one classes inherit from the same class, it is referred to as hierarchical inheritance. In hierarchical inheritance, more than one classes extend, as per the requirement of the design, from the same base class. The common attributes of these child classes are implemented inside the base class.

class.

Example:

- A `Car` **IS A** `Vehicle`
- A `Truck` **IS A** `Vehicle`



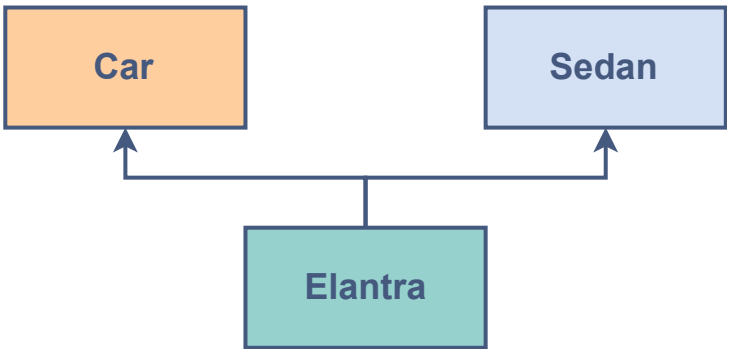

```
class Vehicle {          //Base Vehicle class

  private int topSpeed;

  public void setTopSpeed(int speed) {
    this.topSpeed=speed;
    System.out.println("The top speed of "+getClass().getSimpleName()+" is set to: "+ topSpee
  }

}

class Car extends Vehicle { // Derived from Vehicle Base for Prius

  //implementation of Car class
}

class Truck extends Vehicle {// Derived from Prius can be base to any further class

  //implementation of Truck class
}

class Main {

  public static void main(String[] args) {
    Car corolla = new Car();
    corolla.setTopSpeed(220);

    Truck volvo = new Truck();
    volvo.setTopSpeed(120);
  }

}
```

Multiple Inheritance

When a class is derived from more than one base class, i.e. when a class has more than one immediate parent classes, this type of inheritance is called Multiple Inheritance.

Example:

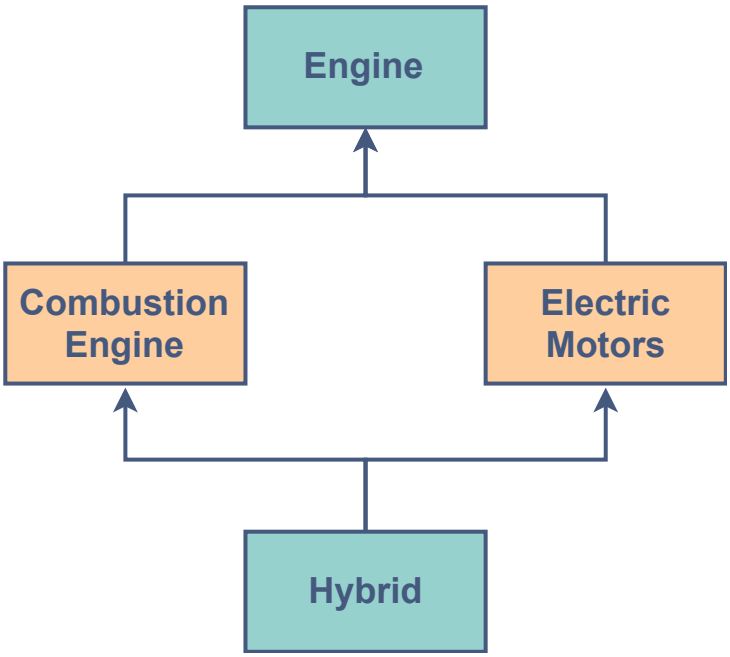
- A Hyundai `Elantra` **IS A** `Car`.
- A Hyundai `Elantra` **IS A** `Sedan` also.



Hybrid Inheritance

A type of inheritance which is a combination of **Multiple** and **Multi-level** inheritance is called *hybrid inheritance*.

- A combustion engine is an **engine**
- An electric motors engine is an **engine**
- A **Hybrid** engine combines both combustion engine and electric motors.



> **Note:** In Java, **Multiple** and **Hybrid** inheritance are applicable using interfaces only.

This lesson was about the different types of Inheritance. In the next lesson, we will discuss the advantages of inheritance.