The Syntax and Terminologies

In this lesson, you will learn how to use inheritance syntactically and the terminologies related to it.

The Terminologies

As we know that a new class is created *based* on an *existing* class in Inheritance, hence we use the terminology below for the new class and the existing class:

- **SuperClass (Mother Class or Base Class):** This class allows the *re-use* of its `non-private` members in another class.
- **SubClass (Child Class or Derived Class):** This class is the one that *inherits* from the superclass.

> 💡 A *child* class has **all non-private** characteristics of the *mother* class.

What does a Child have?

An object of the child class can use:

- All `non-private` members defined in the **child** class.
- All `non-private` members defined in the **mother** class.

> ⚠️ Some classes cannot be inherited. Such classes are defined with the keyword, `final`. An example of such a class is the built-in `Integer class`

> - this class cannot have derived classes.

The `extends` Keyword

In Java, to implement inheritance we have to use the keyword `extends` to implement inheritance:

```
SubClass extends SuperClass{
//contents of SubClass

}
```

Let's take an example of a `Vehicle class` as a *base class* and implement a `Car class` that will extend from this `Vehicle class`. As a *Car **IS A**, Vehicle* the implementation of inheritance relation between these classes will stand valid.

```java
// Base Class Vehicle
class Vehicle {

  // Private Fields
  private String make;
  private String color;
  private int year;
  private String model;


  // Parameterized Constructor
  public Vehicle(String make, String color, int year, String model) {
    this.make = make;
    this.color = color;
    this.year = year;
    this.model = model;
  }

  // public method to print details
  public void printDetails() {
    System.out.println("Manufacturer: " + make);
    System.out.println("Color: " + color);
    System.out.println("Year: " + year);
    System.out.println("Model: " + model);
  }

}

// Derived Class Car
class Car extends Vehicle {

  // Private field
  private String bodyStyle;

  // Parameterized Constructor
  public Car(String make, String color, int year, String model, String bodyStyle) {
    super(make, color, year, model);  //calling parent class constructor
    this.bodyStyle = bodyStyle;
```

```java
    }

  public void carDetails() {  //details of car

    printDetails();            //calling method from parent class
    System.out.println("Body Style: " + bodyStyle);
  }

}

class Main {

  public static void main(String[] args) {
    Car elantraSedan = new Car("Hyundai", "Red", 2019, "Elantra", "Sedan"); //creation of car
    elantraSedan.carDetails(); //calling method to print details
  }

}
```

In the code above, ignore the **line 37** for now, you will get to know about it in the next lesson.

> **Note:** In Java, a class can extend from only one other class at a time and a class cannot extend itself.

Let's move on to the description of a very important keyword `super` in Java inheritance mechanism.