Inheritance

Get familiar with the concept of inheritance and its type with implementation.

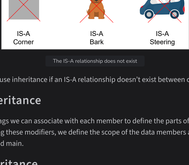
We'll cover the following

Definition

Inheritance provides a way to create a new class from an existing class. The new class is a specialized version of the existing class such that it inherits all the public attributes (variables) and methods of the existing class. The existing class is used as a starting point or base to create the new class.

The IS-A relationship

After reading the definition above, the next question that comes to mind is, "when do we use inheritance?" Wherever we come across an IS-A relationship between objects, we can use inheritance.

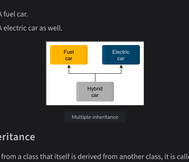


The IS-A relationship

So, from the description of inheritance above, we can conclude that we can build new classes by extending existing classes.

Existing Class	Derived Class
Shape	Square
Animal	Dog
Vehicle	Car

Let's look at the figure below to visualize some examples where an IS-A relationship doesn't exist.



The IS-A relationship does not exist

Remember, we cannot use inheritance if an IS-A relationship doesn't exist between classes.

Modes of inheritance

Access modifiers are tags we can associate with each member to define the parts of the program they can access directly. By using these modifiers, we define the scope of the data members and member functions for the other classes and main.

Types of inheritance

Based on parent classes and child classes, there are *five* types of inheritance in general, which are explained below.

Single inheritance

In single inheritance, there is only a single class extending from a single parent class.

Example:

- A fuel car IS-A vehicle

Single inheritance

Multiple inheritance

When a class is derived from more than one base class, i.e., when a class has more than one immediate parent class, it is called multiple inheritance.

Example:

- The hybrid car IS-A fuel car.
- The hybrid car IS-A electric car as well.

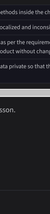
Multiple inheritance

Multi-level inheritance

When a class is derived from a class that itself is derived from another class, it is called multi-level inheritance. We can extend the classes to as many levels as we want.

Example:

- A fuel car IS-A vehicle
- A gasoline car IS-A fuel car



Multi-level inheritance

Hierarchical inheritance

In hierarchical inheritance, more than one class extends, 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.

Example:

- A fuel car IS-A vehicle
- An electric car IS-A vehicle

Hierarchical inheritance

Hybrid inheritance

A type of inheritance that is a combination of more than one type of inheritance is called **hybrid inheritance**.

Example:

- A fuel car IS-A vehicle.
- An electric car IS-A vehicle.
- A hybrid car IS-A fuel car and IS-A electric car.

Hybrid inheritance

Implementation

Let's take an example of a `vehicle` class and implement different classes that will extend from it. We will also implement hierarchical, multi-level, and multiple inheritances from this example.

> **Note:** Some languages, such as Java, C# and JavaScript, do not support multiple inheritance through classes.

```
// Base class (Parent)
class Vehicle {
    private String name;
    private String model;
    Vehicle(String name, String model) {
        this.name = name;
        this.model = model;
    }
    public void getName() {
        System.out.print("The car is a " + name + " " + model);
    }
}

// Single inheritance
// FuelCar class extending from Vehicle class
// Derived class (Child)
class FuelCar extends Vehicle {
    private String combustType;
    FuelCar(String name, String model, String combustType) {
        super(name, model);
        this.combustType = combustType;
    }
    public void getFuelCar() {
        getName();
        System.out.print(", combust type is " + combustType);
    }
}

// Hierarchical inheritance
// Alongside the FuelCar class, the ElectricCar class is also extending from Vehicle class
```

The implementation of various classes using inheritance

Advantages of inheritance

The following are four main advantages of inheritance:

Advantages	Description
Reusability	We don't need to duplicate methods inside the child classes that also occur in the parent classes.
Code modification	Ensures that all changes are localized and inconsistencies are avoided.
Extensibility	We can extend the base class as per the requirements of the derived class. It provides an easy way to upgra enhance specific parts of a product without changing the core attributes.
Data hiding	A base class can keep some data private so that the derived class cannot alter it. This concept is called encapsulation.

Let's learn about polymorphism in the next lesson.