Inheritance

Overview

Inheritance is an important pillar of object-oriented programming. Inheritance is the process of inheriting the properties and behavior of an existing class into a new class. When we inherit the class, we can reuse the existing class's methods and fields into a new class. Inheritance can also be defined as the **Is-A** relationship, which is also known as the parent-child relationship.

Let's understand the inheritance with the help of a real-world example.

Let's assume that **Human** is a class that has properties such as **height**, **weight**, **age**, etc. and functionalities (or methods) such as **eating()**, **sleeping()**, **dreaming()**, **working()**, etc.

Now we want to create **Male** and **Female** classes. Both males and females are humans, and they share some common properties (like **height**, **weight**, **age**, etc.) and behaviors (or functionalities like **eating()**, **sleeping()**, etc.), so they can inherit these properties and functionalities from the **Human** class. Both males and females also have specific characteristics (like men having short hair and females have long hair). Such properties can be added to the **Male** and **Female** classes separately.

This approach makes us write less code as both the classes inherited several properties and functions from the superclass; thus, we didn't have to re-write them. Also, this makes it easier to read the code.

Why we use inheritance:

- The main advantage of inheritance is code reusability. We can reuse the code when we inherit the existing class's methods and fields into a new class.
- The runtime polymorphism (method overriding) can be achieved by inheritance only.

Important terminology of inheritance

- **Sub Class:** The class that inherits properties from another class is called Subclass or Derived Class.
- **Super Class:** The class whose properties are inherited by subclass is called Base Class or Superclass.

C++ inheritance syntax:

```
class parent_class {
    //Body of parent class
};
class child_class: access_modifier parent_class {
    //Body of child class
};
```

Here, child_class is the name of the subclass, access_mode is the mode in which you want to inherit this sub-class, for example, public, private, etc., and parent_class is the name of the superclass from which you want to inherit the subclass.

Modes of Inheritance

1. **Public mode**: If we derive a subclass from a public base class. Then, the base class's public members will become public in the derived class, and protected class members will become protected in the derived class.
2. **Protected mode**: If we derive a subclass from a Protected base class. Then both public members and protected members of the base class will become protected in the derived class.

3. **Private mode**: If we derive a subclass from a Private base class. Then both public members and protected members of the base class will become Private in the derived class.

Base Class member Access Specifier	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not Accessible	Not Accessible	Not Accessible