Inheritance

- It is a mechanism which acquires the properties and behaviors of a class from another class.
- The class whose members are inherited is called the base class, and the class that inherits those members is called the derived/child/subclass.
- The TypeScript uses class inheritance through the extends keyword

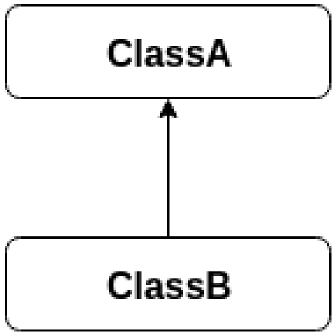
Why use inheritance?

- ➔ Code Reusability.

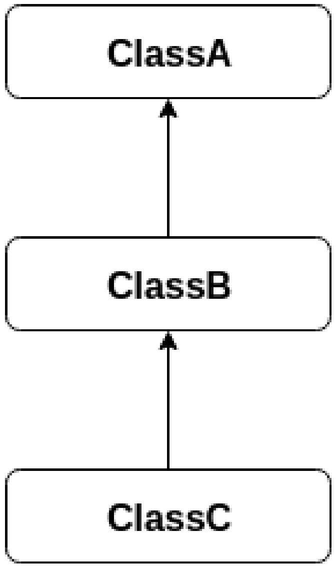
Types of Inheritance

We can classify the inheritance into the five types. These are:

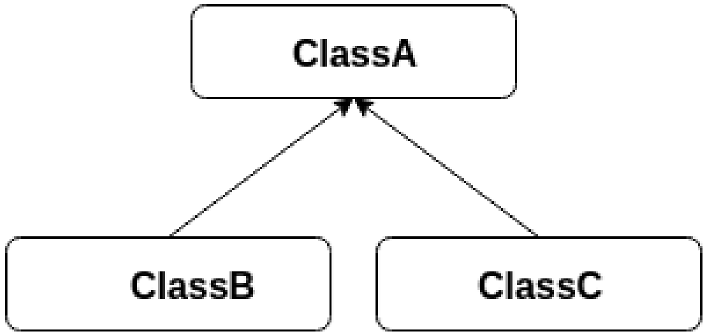
1. Single Inheritance
2. Multilevel Inheritance
3. Multiple Inheritance
4. Hierarchical Inheritance
5. Hybrid Inheritance



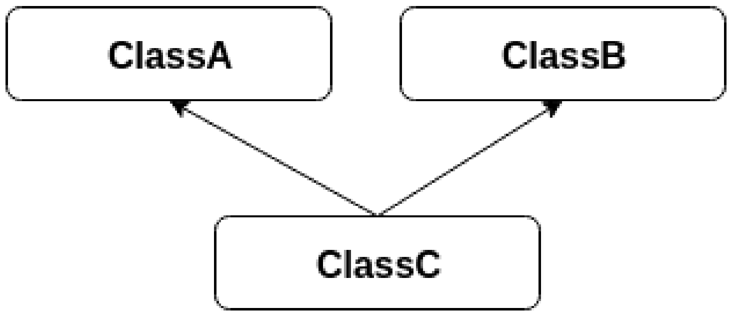
Single Inheritance



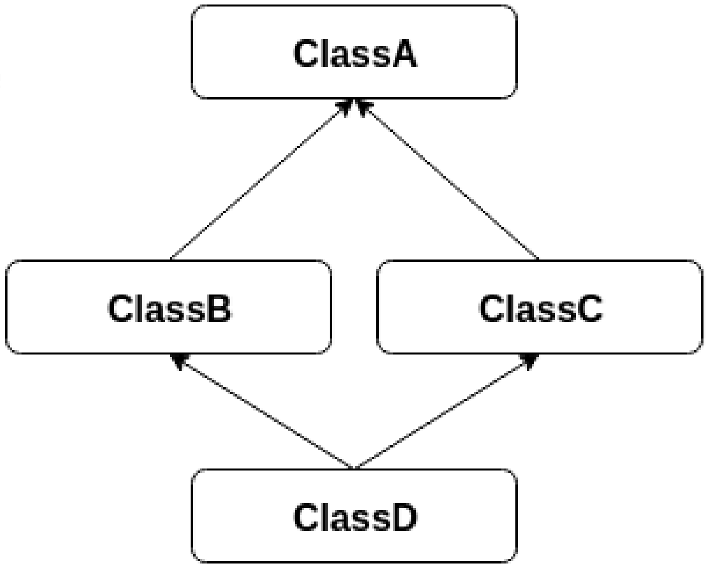
Multilevel Inheritance



Hierarchical Inheritance



Multiple Inheritance



Hybrid Inheritance

Function overloading

Function overloading is a mechanism or ability to create multiple methods with the **same name** but different parameter types and **return type**.

However, it can have the same number of parameters. Function overloading is also known as method overloading.

How to achieve -

- The function name is the same
- The number of parameters is different in each overloaded function.
- The number of parameters is the same, and their type is different.
- The all overloads function must have the same return type.

Eg.

- Suppose we have to perform **multiplication** of the numbers, which has a different number of parameters. We write the **two** methods such as mul_a(number, number) for **two parameters**, and mul_b(number, number, number) for **three parameters**.
- Now, it may be difficult for us as well as other programmers to understand the behavior of the method because its name **differs**. In that case, we need to use function overloading, which increases the readability of the program.

Advantage of function overloading

- It saves the memory space so that program execution becomes fast.
- It provides code reusability, which saves time and efforts.
- It increases the readability of the program.
- Code maintenance is easy.

JavaScript

```javascript
//Function with string type parameter
function add(a:string, b:string): string;
//Function with number type parameter
function add(a:number, b:number): number;
//Function Definition
function add(a: any, b:any): any {
    return a + b;
}
//Result
console.log("Addition: " +add("Hello ", "David"));
console.log("Addition: "+add(30, 20));
```

In the above example:

- The first **two** lines are the function overload **declaration.** It has two overloads:
  - A Function which accepts a **string** parameter.
  - A Function which accepts a **number** parameter.
- The third line is the **function definition**, where the data type of the parameters is set to **any**.
- The last two statements **invoke** the overloaded function.

Function overloading in a class

JavaScript

```
class A
{
    public sample(s: string): number;
    public sample(n: number): string;
    public sample(arg: any): any
    {
        if (typeof(arg) === 'number')
            return arg.toString();
        if (typeof(arg) === 'string')
            return arg.length;
    }
}
let obj = new A();
console.log("Result: " +obj.sample(101));
console.log("Length of String: " +obj.sample("David"));
```

Function overloading with a different number of parameters and different types along with the same function name is not supported.

JavaScript

```
function display(x:number, y:number):void //Compiler Error: Duplicate function implementation
{
    console.log(x + x);
}

function display(x:string): void //Compiler Error: Duplicate function implementation
{
    console.log(x);
}
```