

# Relationship Between Classes-Inheritance

Ali Haider

[syedalihaider.ciit@gmail.com](mailto:syedalihaider.ciit@gmail.com)

Department of Computer Science IUB

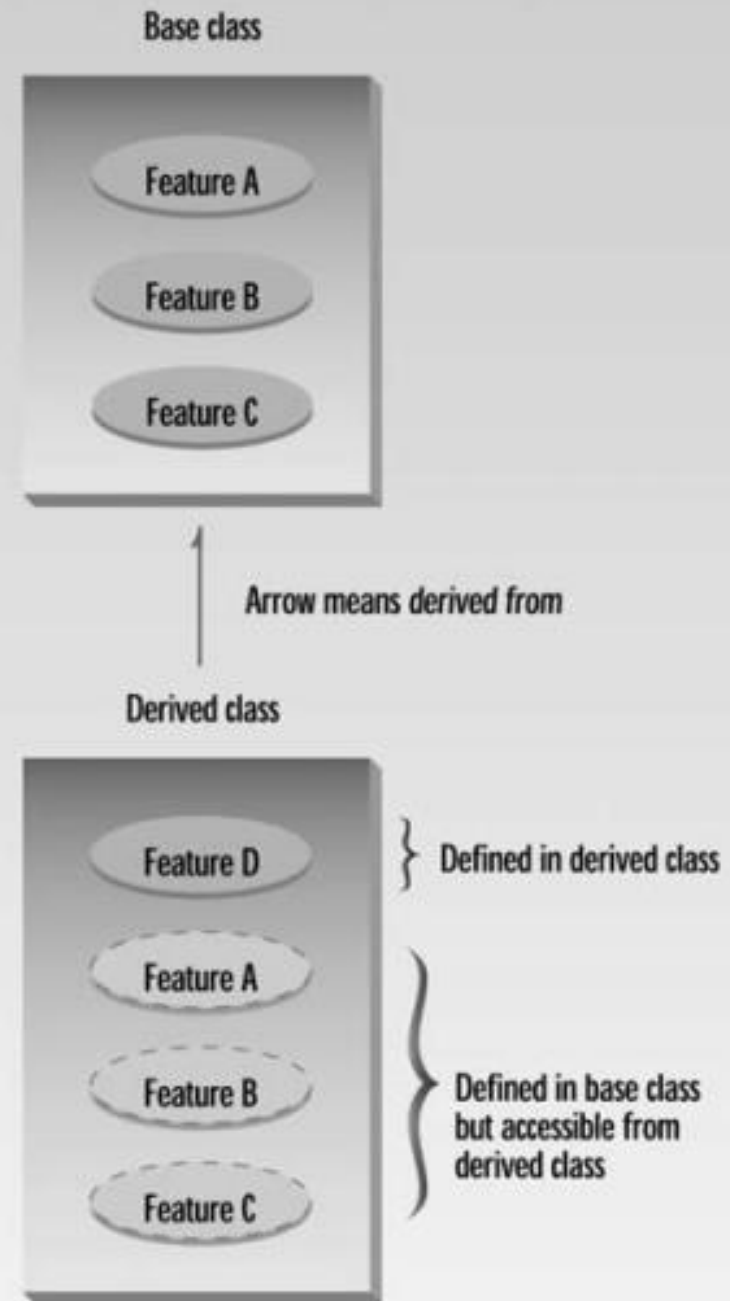
# Overview

- Inheritance
- Types of Inheritance
  - Single Inheritance
  - Multiple Inheritance
  - Multilevel Inheritance
  - Hierarchical Inheritance
  - Hybrid (Virtual) Inheritance

# Inheritance-1

- Inheritance is probably the most powerful feature of object-oriented programming, after classes themselves.
- Inheritance is the process of creating new classes, called derived classes, from existing or base classes.
- The derived class inherits all the capabilities of the base class but can add embellishments and refinements of its own.
- The base class is unchanged by this process.
- Inheritance is an essential part of OOP. Its big payoff is that it permits code reusability.

# Inheritance-2



# Types of Inheritance

- **Single Inheritance:** In single inheritance, a class is allowed to inherit from only one class. i.e. one sub class is inherited by one base class only.
- **Multiple Inheritance:** Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. i.e one sub class is inherited from more than one base classes.
- **Multilevel Inheritance:** In this type of inheritance, a derived class is created from another derived class.
- **Hierarchical Inheritance:** In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class.
- **Hybrid (Virtual) Inheritance:** Hybrid Inheritance is implemented by combining more than one type of inheritance.
- For example: Combining Hierarchical inheritance and Multiple Inheritance.

# Single Inheritance Example

## Syntax

```
class subclass_name : access_mode base_class
{
    //body of subclass
};
```

```
// Single inheritance
#include <iostream>
using namespace std;

// base class
class Vehicle {
public:
    Vehicle()
    {
        cout << "This is a Vehicle" << endl;
    }
};

// sub class derived from two base classes
class Car: public Vehicle{

};

// main function
int main()
{
    // creating object of sub class will
    // invoke the constructor of base classes
    Car obj;
    return 0;
}
```

# Inheritance Example

- Base class **Shape**
- And its derived class **Rectangle**

```
int main(void) {  
    Rectangle Rect;  
    Rect.setWidth(5);  
    Rect.setHeight(7);  
    // Print the area of the object.  
    cout << "Total area: " << Rect.getArea() << endl;  
    return 0;  
}
```

```
class Triangle: public Shape {  
    public:  
        Triangle( int a = 0, int b = 0):Shape(a, b) { }  
  
        int area () {  
            cout << "Triangle class area :" <<endl;  
            return (width * height / 2);  
        }  
};  
  
// Main function for the program  
int main() {  
    Shape *shape;  
    Rectangle rec(10,7);  
    Triangle tri(10,5);  
    // store the address of Rectangle  
    shape = &rec;  
    // call rectangle area.  
    shape->area();  
    // store the address of Triangle  
    shape = &tri;  
    // call triangle area.  
    shape->area();  
    return 0;  
}
```

# Multiple Inheritance

- A C++ class can inherit members from more than one class and here is the extended syntax
- class derived-class: access baseA, access baseB{
- //class body
- }



# Example-1

```
// C++ program to explain
// multiple inheritance
#include <iostream>
using namespace std;

// first base class
class Vehicle {
public:
    Vehicle()
    {
        cout << "This is a Vehicle" << endl;
    }
};

// second base class
class FourWheeler {
public:
    FourWheeler()
    {
        cout << "This is a 4 wheeler Vehicle" << endl;
    }
};

// sub class derived from two base classes
class Car: public Vehicle, public FourWheeler {

};

// main function
int main()
{
    // creating object of sub class will
    // invoke the constructor of base classes
    Car obj;
    return 0;
}
```

# Multiple Inheritance Example-2

```
#include <iostream>
using namespace std;
// Base class Shape
class Shape {
protected:
    int width;
    int height;
public:
    void setWidth(int w) {
        width = w;
    }
    void setHeight(int h) {
        height = h;
    }
};
// Base class PaintCost
class PaintCost {
public:
    int getCost(int area) {
        return area * 70;
    }
};
```

1

```
// Derived class
class Rectangle: public Shape, public PaintCost {
public:
    int getArea() {
        return (width * height);
    }
};
int main(void) {
    Rectangle Rect;
    int area;
    Rect.setWidth(5);
    Rect.setHeight(7);
    area = Rect.getArea();
    // Print the area of the object.
    cout << "Total area: " << Rect.getArea() << endl;
    // Print the total cost of painting
    cout << "Total paint cost: $" << Rect.getCost(area) << endl;
    return 0;
}
```

2

# Multi Level Inheritance

In this type of inheritance, a derived class is created from another derived class.

```
// Multilevel Inheritance
#include <iostream>
using namespace std;

// base class
class Vehicle
{
    public:
    Vehicle()
    {
        cout << "This is a Vehicle" << endl;
    }
};

class fourWheeler: public Vehicle
{
    public:
    fourWheeler()
    {
        cout<<"Objects with 4 wheels are vehicles"<<endl;
    }
};

// sub class derived from two base classes
class Car: public fourWheeler{
    public:
    car()
    {
        cout<<"Car has 4 Wheels"<<endl;
    }
};

// main function
int main()
{
    //creating object of sub class will
    //invoke the constructor of base classes
    Car obj;
    return 0;
}
```

# Hierarchical Inheritance

In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class.

```
// Hierarchical Inheritance
#include <iostream>
using namespace std;

// base class
class Vehicle
{
    public:
        Vehicle()
        {
            cout << "This is a Vehicle" << endl;
        }
};

// first sub class
class Car: public Vehicle
{
};

// second sub class
class Bus: public Vehicle
{
};

// main function
int main()
{
    // creating object of sub class will
    // invoke the constructor of base class
    Car obj1;
    Bus obj2;
    return 0;
}
```

# Hybrid (Virtual) Inheritance

Hybrid Inheritance is implemented by combining more than one type of inheritance.

```
// C++ program for Hybrid Inheritance

#include <iostream>
using namespace std;

// base class
class Vehicle
{
    public:
    Vehicle()
    {
        cout << "This is a Vehicle" << endl;
    }
};

//base class
class Fare
{
    public:
    Fare()
    {
        cout<<"Fare of Vehicle\n";
    }
};

// first sub class
class Car: public Vehicle
{
};

// second sub class
class Bus: public Vehicle, public Fare
{
};

// main function
int main()
{
    // creating object of sub class will
    // invoke the constructor of base class
    Bus obj2;
    return 0;
}
```