Kế thừa

Kế thừa trong lập trình hướng đối tượng là một trong những chủ đề rất quan trọng. **Kế thừa trong OOP** cho phép chúng ta định nghĩa một class mới dựa trên thuộc tính và phương thức của một class đã có trước. Giúp chúng ta tối ưu được thời gian công sức phát triển ứng dụng, đồng thời cũng dễ dàng nâng cấp và chỉnh sửa mã nguồn. Điều đó có nghĩa là, khi chúng ta định nghĩa một class mới, không nhất thiết phải viết lại tất cả mã nguồn cho nó, mà có thể kế thừa dựa trên 1 class đã tồn tại.

class [TÊN CLASS]: [KIỂU KẾ THỪA] [LỚP CƠ SỞ] {

//Khai báo các phương thức và thuộc tính

}

Kế thừa

```
class Shape {
   public:
      void printShape();
};

class Rectangle: public Shape {
   public:
      void printRectangle();
      int getArea();
};
```

Các kiểu kế thừa

- public: các phương thức và thuộc tính của lớp cha khi được kế thừa từ lớp con sẽ được giữ nguyên kiểu.
- protected: các phương thức và thuộc tính của lớp cha khi được kế thừa từ lớp con sẽ bị thay đổi kiểu.
- private: các phương thức và thuộc tính của lớp cha khi được kế thừa từ lớp con sẽ được chuyển thành private.

Các kiểu kế thừa

	public	protected	private
Kế thừa public	public	protected	
Kế thừa protected	protected	private	
Kế thừa private	private	private	

Kế thừa

```
class Shape {
   public:
      void printShape();
};

class Rectangle: private Shape {
   public:
      void printRectangle() {
            this->printShape();
      }
};
```

Kế thừa

```
class Shape {
   protected:
      void printShape();
};

class Rectangle: private Shape {
   public:
      void printRectangle() {
            this->printShape();
      }
};
```

Kế thừa

```
class Shape {
   private:
      void printShape();
};

class Rectangle: private Shape {
   public:
      void printRectangle() {
            this->printShape();
      }
};
```

Kế thừa

```
class Shape {
   public:
      void printShape();
};

class Rectangle: public Shape {
   public:
      void printRectangle();
};

Rectangle r;

r.printShape();

r.printRectangle();
```

Kế thừa

```
class Shape {
   public:
      void printShape();
};

class Rectangle: protected Shape {
   public:
      void printRectangle();
};

Rectangle r;

~~r.printShape();~~

r.printRectangle();
```

Kế thừa

```
class Shape {
   public:
      void printShape();
};

class Rectangle: private Shape {
   public:
      void printRectangle();
};

Rectangle r;

r.printShape();

r.printRectangle();
```

Kế thừa

```
class Shape {
   public:
      void printShape();
};

class Rectangle: private Shape {
   public:
      void printRectangle();
};

Rectangle r;

r.printShape();

r.printRectangle();
```

Kế thừa

```
class Shape {
   public:
      Shape();
};

class Rectangle: private Shape {
   public:
      Rectangle(): Shape::Shape() {


       }
};
```

Kế thừa

```
class Shape {
   public:
       Shape();
       Shape(int x, int y);
};
class Rectangle: private Shape {
   public:
       Rectangle(): Shape::Shape(0, 0) {
       }
};
```

Đa kế thừa

```
class PaintCost {
   protected:
      void printPaintCost();
};

class Shape {
   public:
      void printShape();
};

class Rectangle: public Shape, protected PaintCost {
   public:
      int getArea() {
         return (width * height);
      }
};
```

Kế thừa

```
class Shape {
   protected:
        int width, height;
   public:
        Shape(int a, int b) {
                width = a;
                height = b;
        }
        void area() {
                cout<<"Area: "<<width*height;
        }
};
```

Kế thừa

```
class Rectangle: public Shape {
   public:
        Rectangle( int a, int b): Shape::Shape(a, b) {



        }

       void area () {
                cout << "Area : " <<(width * height);
       }
};
```

Kế thừa

```
class Rectangle: public Shape {
   public:
        Rectangle( int a, int b): Shape::Shape(a, b) {



        }

      void area () {
               Shape::area();
      }
};
```