

Khoá học: Game Developer

# Bài 05: Kế thừa (Inheritance)





# Mục tiêu bài học

Kết thúc bài học này, bạn có thể

- Hiểu khái niệm và sự phân cấp thừa kế.
- Tái sử dụng các lớp có sẵn.
- Hiểu khái niệm sự đa hình và cơ chế nạp chồng phương thức.
- Hiểu khái niệm về lớp và phương thức niêm phong.

# Nội dung

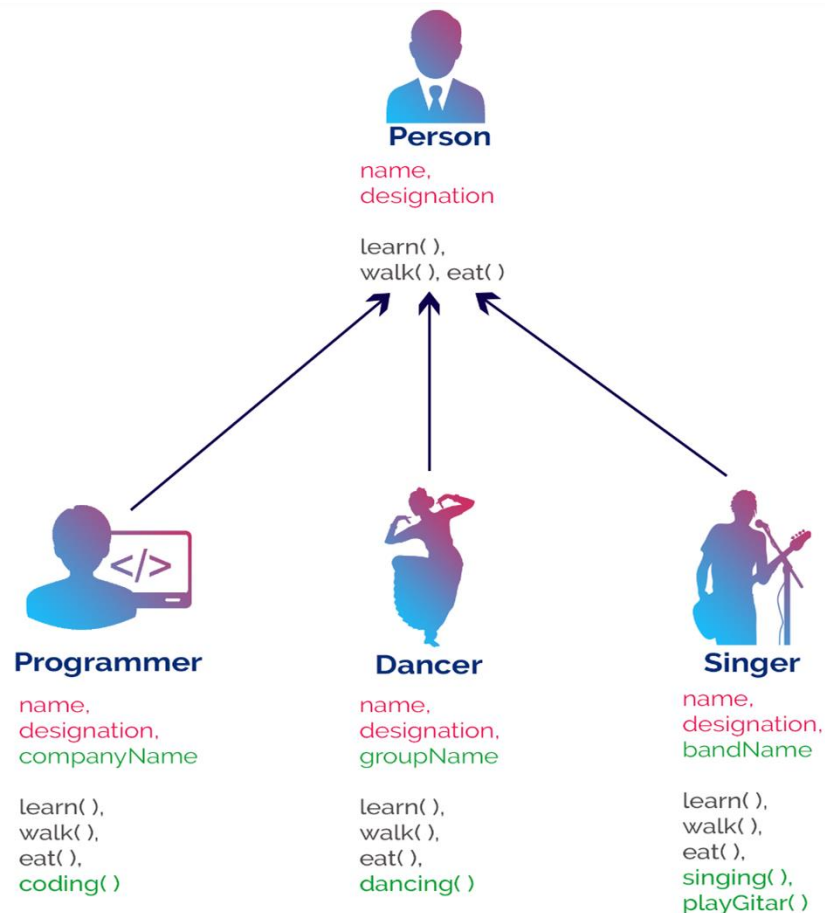


1. Khái niệm Kế thừa
2. Từ khoá base
3. Cơ chế nạp chồng và che giấu của phương thức
4. Sealed class

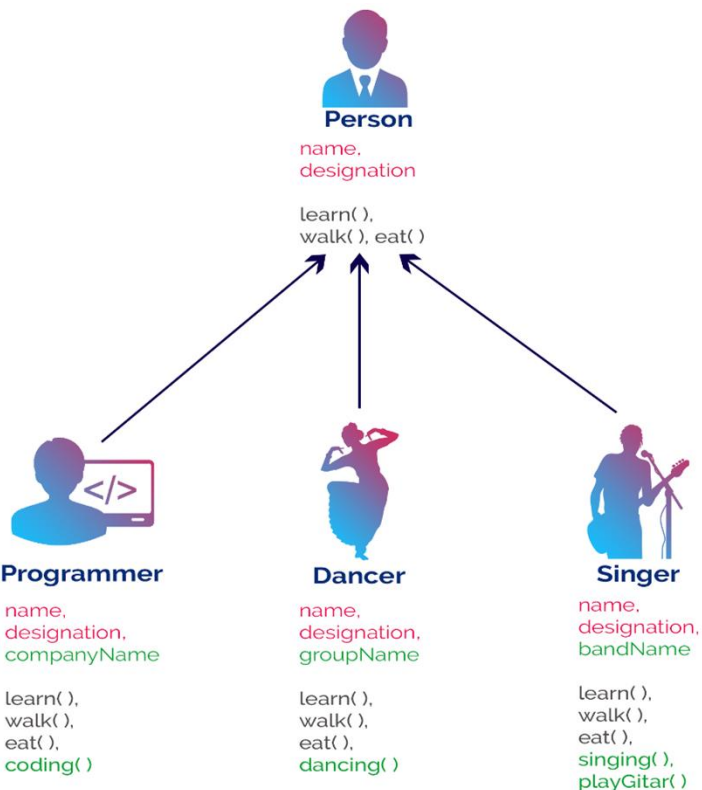
**Tham khảo:** <https://docs.oracle.com/javase/tutorial/java/javaOO/classvars.html>

# Sự phân cấp

- Các lớp trong C# tồn tại trong một hệ thống thứ bậc phân cấp, gọi là **cây thừa kế**.
- Lớp bậc trên gọi là **lớp cha (super class)**.
- Lớp bậc dưới gọi là **lớp con (sub class)**.
- Trong C# một lớp con chỉ có **một lớp cha duy nhất (đơn thừa kế)**.



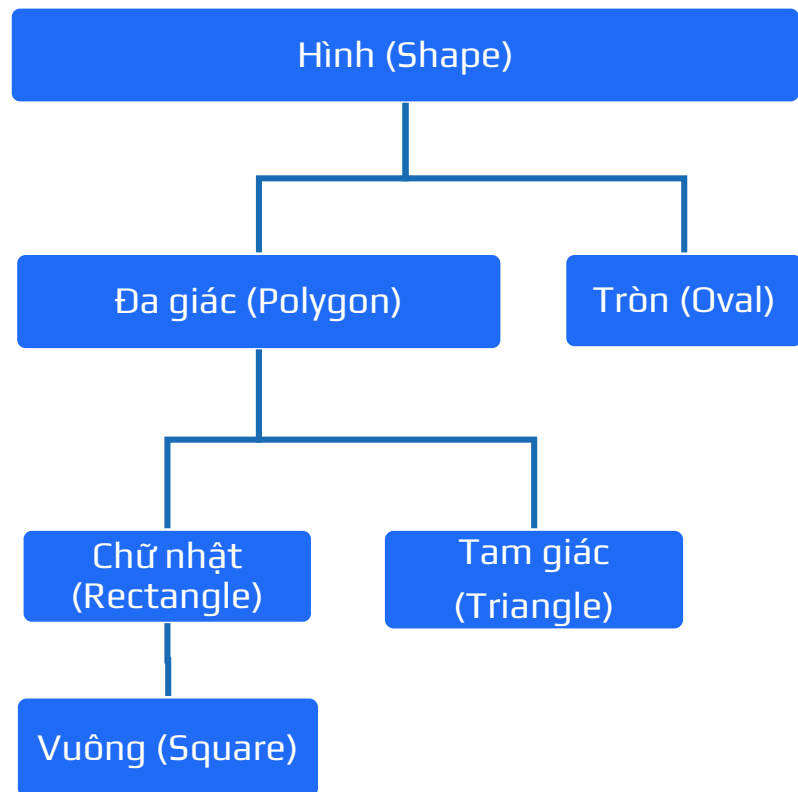
# Phân cấp thừa kế



```
class Person {...}  
class Programmer : Person {...}  
class Dancer : Person {...}  
class Singer : Person {...}
```

: là ký tự để **lớp con** (sub class) kế thừa từ **lớp cha** (super class)

# Phân cấp thừa kế



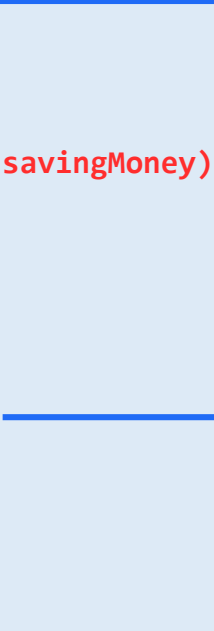
```
class Shape {...}
class Polygon : Shape {...}
class Oval : Shape {...}
class Rectangle : Polygon {...}
class Triangle : Polygon {...}
class Square : Rectangle {...}
```

# Thừa kế

## Inheritance

- **Kế thừa** là cơ chế tạo 1 lớp mới dựa trên kết quả từ lớp đã có. Mục đích của thừa kế là **tái sử dụng**.
- Lớp con **ĐƯỢC PHÉP** sở hữu các tài sản (properties và methods) của lớp cha
  - Nếu tài sản của lớp cha là **public** hoặc **protected**.
- Lớp con **KHÔNG THỂ** truy cập thành viên **private** của lớp cha.
- Lớp con **KHÔNG** kế thừa các **phương thức khởi tạo (constructor)** của lớp cha.

```
public class Father {  
    protected string brandName;  
    int numOfBakeries;  
    private int savingMoney;  
    public Father(string brandName, int savingMoney)  
    {...}  
    private void makingCake(){...}  
    void manageBakery(){...}  
    protected void buyIngredient(){...}  
}  
  
public class Son : Father {...}
```

A blue line starts from the right side of the 'Father' class definition, goes down, and then turns left as an arrow pointing to the 'Son' class definition, indicating that 'Son' inherits from 'Father'.

# Phương thức khởi tạo ở lớp con

## Subclass constructor

- Lớp con **KHÔNG** kế thừa constructor của lớp cha. Lớp con phải tự thiết lập constructor cho riêng mình
- Constructor của lớp cha được thực hiện trước rồi mới đến constructor của lớp con.
- Chỉ định constructor:
  - **Nếu lớp cha có nhiều constructor** thì lớp con được quyền chỉ định 1 constructor của lớp cha.
  - **Nếu lớp con không chỉ định** thì constructor mặc định của lớp cha sẽ tự động thực hiện.

```
public class Person {  
    string name, designation;  
    public Person(string name, string designation) {  
        this.name = name;  
        this.designation = designation;  
    }  
}
```

```
public class Programmer : Person {  
    string companyName;  
    public Programmer(string name, string designation, string  
        companyName) {  
        base(name, designation);  
        this.companyName = companyName;  
    }  
}
```



# Từ khoá base

## Inheritance

- Khi cả lớp cha và lớp con có các thành viên trùng tên thì phân biệt bằng cách sử dụng từ khoá:
  - **base.tênThànhViên**: truy cập đến thành phần của **lớp cha**.
  - **(this.)tênThànhViên**: truy cập đến thành phần của **lớp hiện hành**.
- Từ khoá **base** để tái sử dụng phương thức khởi tạo của lớp cha.

```
public class Parent {  
    public string field;  
    public void method() {...}  
}
```

```
public class Child : Parent {  
    public string field;  
    public void method(){  
        this.field = base.field;  
        base.method();  
    }  
}
```

# Từ khoá base

Ví dụ

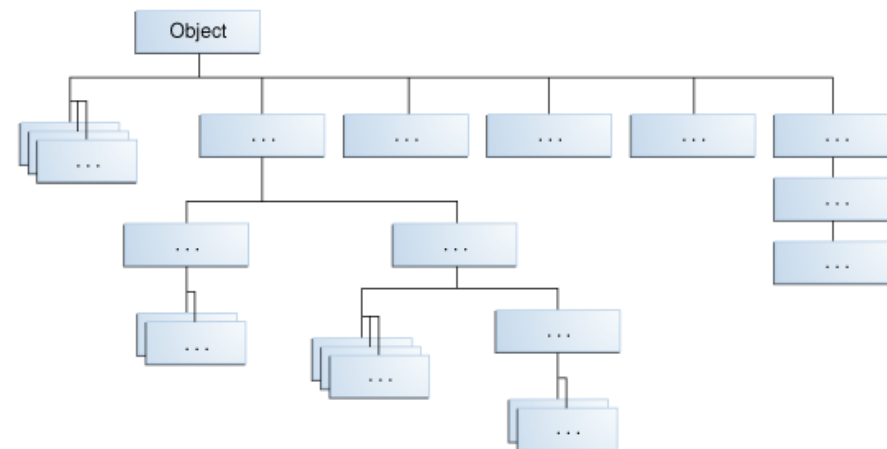
```
public class Person {  
    string name, designation;  
    public Person(string name, string designation) {  
        this.name = name;  
        this.designation = designation;  
    }  
    public void showInfo() {  
        Console.WriteLine("Name: " + name);  
        Console.WriteLine("Designation: " + designation);  
    }  
}
```

```
public class Programmer : Person {  
    string companyName;  
    public Programmer(string name, string designation, string  
        companyName) : base(name, designation)  
    {  
        this.companyName = companyName;  
    }  
    public void showInfo() {  
        base.showInfo();  
        Console.WriteLine("Company: " + companyName);  
    }  
}
```

# Lớp Object

## The Object class

- Bất kỳ các lớp do lập trình viên định nghĩa trên nền tảng Java đều là hậu duệ từ **Object**.
- Tất cả các lớp tạo ra đều kế thừa các phương thức từ Object nhưng có thể tự định nghĩa lại các phương thức này.



Method	Behavior
<b>Object clone()</b>	Makes a copy of an object.
<b>boolean equals(Object obj)</b>	Compares this object to its argument.
<b>int hashCode()</b>	Returns an integer hash code value for this object.
<b>string toString()</b>	Returns a string that textually represents the object.

# Lớp niêm phong

## Sealed Class

- **Lớp niêm phong** là lớp không cho phép lớp khác kế thừa từ nó.
- Từ khoá **sealed** được thêm vào trước class để tạo niêm phong.
- Từ khoá **sealed** cũng được thêm vào trước method để ngăn không cho các lớp override lại phương thức đó.

