Kế thừa

Nội dung

1. Khái niêm Kế thừa
2. Sử dụng Kế thừa trong Java
3. Nguyên lý kế thừa
4. Overloading
5. Sử dụng Super

Khái niêm kế thừa

Kế thừa?

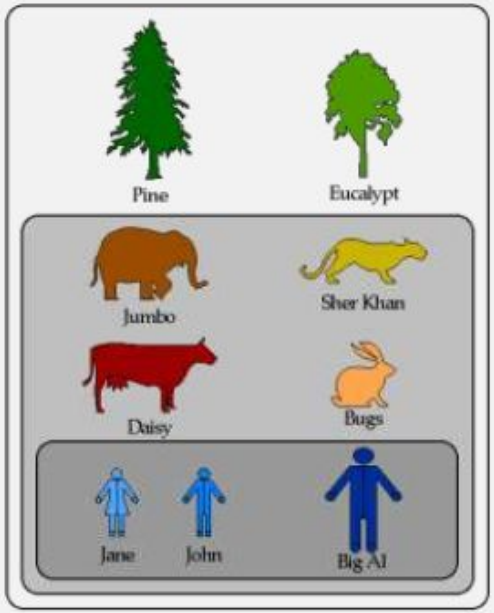
"Xây dựng các lớp mới có sẵn các đặc tính của lớp cũ, đồng thời chia sẻ hay mở rộng các đặc tính sẵn có"

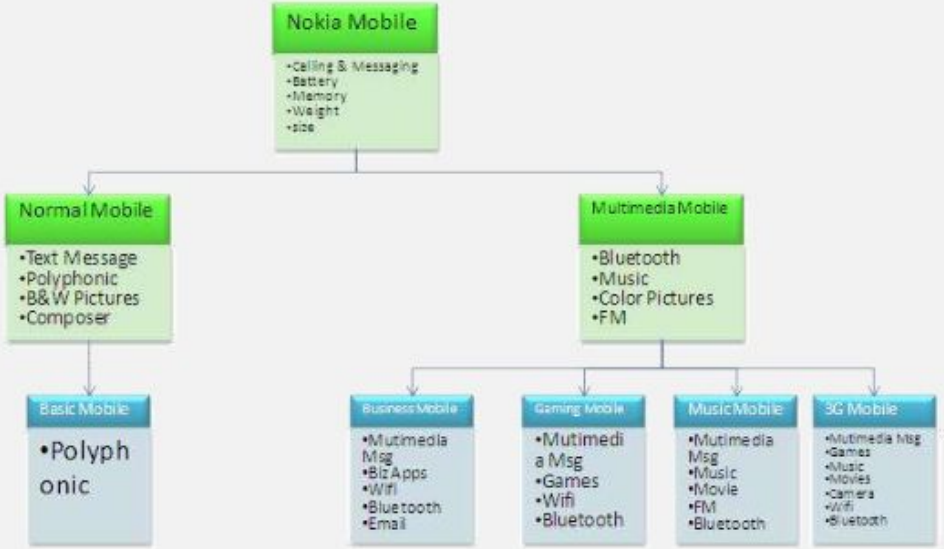
Bản chất Kế thừa

- Phát triển lớp mới dựa trên các lớp đã có
- Ví dụ
  + Lớp Người: có thuộc tính **tên, tuổi, chiều cao, cân nặng**.
  + Lớp Sinh Viên kế thừa lớp Người sẽ được thừa kế các thuộc tính **tên , tuổi, chiều cao, cân nặng.**
  + Có thể bổ sung thêm các thuộc tính như **mãSV, email,...**

Bản chất Kế thừa

- Chính là nguyên lý phân cấp trong trừu tượng hoá

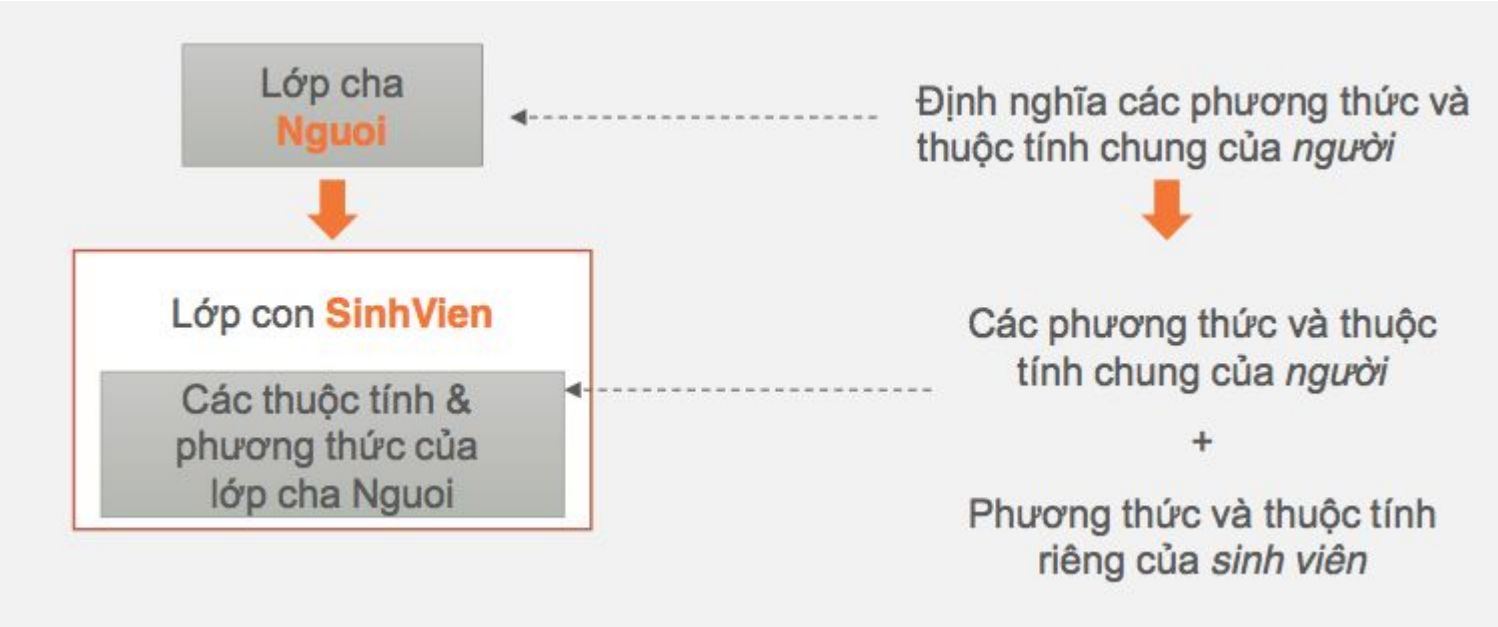




Bản chất Kế thừa

- Là kỹ thuật tái sử dụng mã nguồn
  + Tái sử dung mã nguồn thông qua lớp
- Ví dụ: Lớp sinhvien tái sử dụng thuộc tính tên, tuổi, chiều cao, cân nặng và phương thức của lớp Người

Ví dụ


Lớp cha
Nguoi
Định nghĩa các phương thức và thuộc tính chung của *người*
Lớp con SinhVien
Các thuộc tính & phương thức của lớp cha Nguoi
Các phương thức và thuộc tính chung của *người*
+
Phương thức và thuộc tính riêng của *sinh viên*

Cú pháp trong Java

- Cú pháp (Java):

```
<Lớp con> extends <Lớp cha>
```

- Ví dụ

```
class Nguoi {
  String name; int age;
}
class SinhVien extends Nguoi {
  int studentId;
}
```

– Lớp con *mở rộng* các đặc tính của lớp cha

Nguyên lý Kế thừa

- Lớp con có thể kế thừa được gì từ lớp cha?
  + Kế thừa được các thành viên được khai báo là public và protected của lớp cha
  + Không kế thừa được các thành viên private

Nguyên lý Kế thừa

	public	protected	*mặc định*	private
Cùng lớp	✔	✔	✔	✔
Lớp bất kỳ cùng gói	✔	✔	✔	✘
Lớp con khác gói	✔	✔	✘	✘
Lớp bất kỳ khác gói	✔	✘	✘	✘

Ví dụ 1

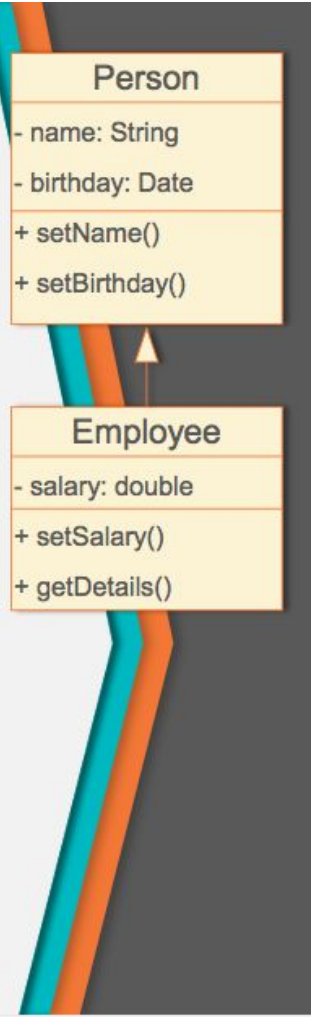
```
public class TuGiac {
  protected Diem d1, d2, d3, d4;
  public void setD1(Diem _d1) {d1=_d1;}
  public Diem getD1(){return d1;}
  public void printTuGiac(){...}
}
public class HinhVuong extends TuGiac {
        public HinhVuong(){
                d1 = new Diem(0,0); d2 = new Diem(0,1);
                d3 = new Diem(1,0); d4 = new Diem(1,1);
        }
}

public class Test{
        public static void main(String args[]){
                HinhVuong hv = new HinhVuong();
                hv.printTuGiac();
        }
}
```

Sử dụng các thành phần protected của lớp cha trong lớp con

Gọi phương thức public của lớp cha trong đối tượng lớp con

Ví dụ 2

```
class Person {
    private String name;
    private Date bithday;
    public String getName() {return name;}
}
class Employee extends Person {
    private double salary;
    public boolean setSalary(double sal) {
        salary = sal;
        return true;
    }
    public String getDetail() {
        String s = name + ", " + birthday +
                   ", " + salary; // ERROR
    }
}
```



Ví dụ 2 (tiếp)

```
public class Test{
    public static void main(String args[]){
        Employee e = new Employee();
        e.setName("John");
        e.setSalary(3.0);
    }
}
```

Ví dụ 3

- Cùng gói
- Khác gói

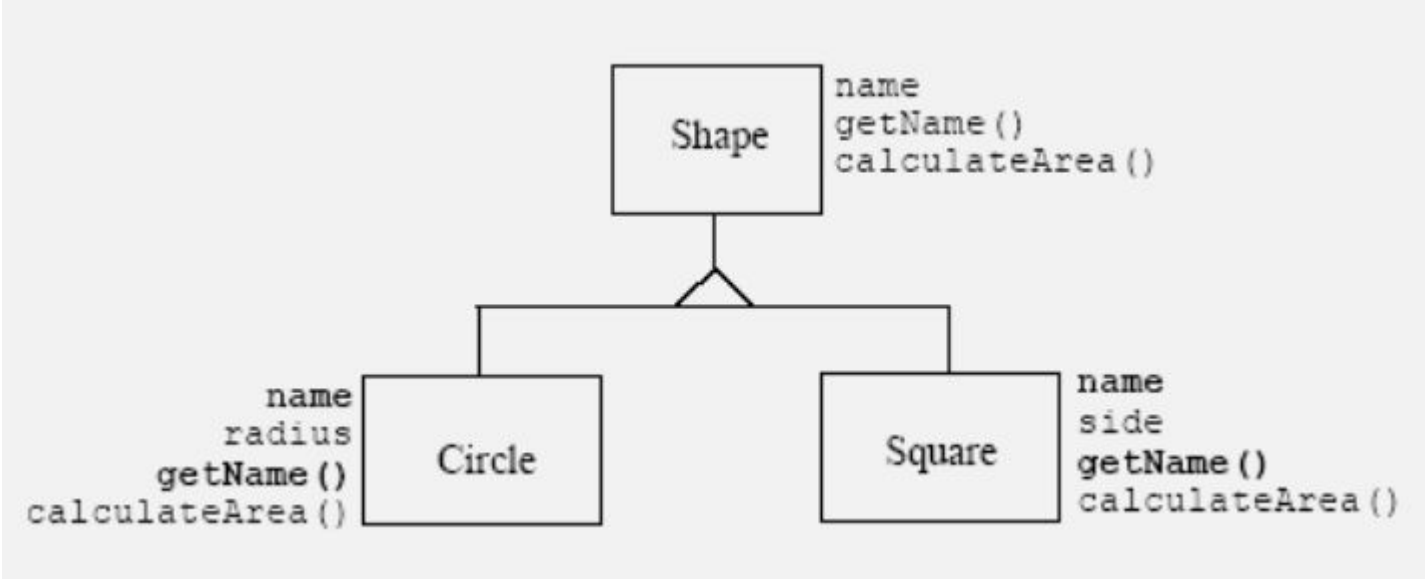
```
package abc;
public class Person {
    Date birthday;
    String name;
}

package abc.Person;
public class Employee extends Person {
    double salary;
    public String getDetail() {
        String s;
        s = name + "," + birthday + "," + salary;
        return s;
    }
}
```

Override – Ghi đè

Định nghĩa

- Lớp con khai báo phương thức có phần chữ ký hoàn toàn giống với lớp cha nhưng nội dung thay đổi
- Phương thức ghi đè sẽ thay thế hoặc làm rõ hơn phương thức ở lớp cha
- Các đối tượng con sẽ sử dụng phương thức ghi đè thay cho phương thức ở lớp cha

Định nghĩa

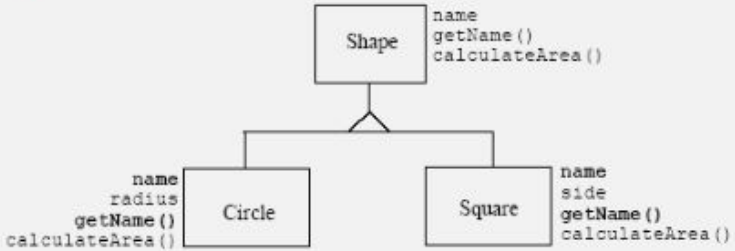

Shape
name
getName()
calculateArea()
Circle
name
radius
getName()
calculateArea()
Square
name
side
getName()
calculateArea()

Ví dụ

```
class Shape {
    protected String name;
    Shape(String n) { name = n; }
    public String getName() { return name; }
    public float calculateArea() { return 0.0f; }
}

class Circle extends Shape {
    private int radius;
    Circle(String n, int r){
        super(n);
        radius = r;
    }

    public float calculateArea() {
        float area = (float)(3.14 * radius * radius);
        return area;
    }
}
```

Lớp Square

```
class Square extends Shape {
    private int side;
    Square(String n, int s) {
        super(n);
        side = s;
    }
    public float calculateArea() {
        float area = (float) side * side;
        return area;
    }
}
```



Quy định trong ghi đè

- Phương thức ghi đè trong lớp con phải
  - Có danh sách tham số giống hệt phương thức kế thừa trong lớp cha.
  - Có cùng kiểu trả về với phương thức kế thừa trong lớp cha
- Các chỉ định truy cập không giới hạn chặt hơn phương thức trong lớp cha
  - Ví dụ, nếu ghi đè một phương thức protected, thì phương thức mới có thể là protected hoặc public, mà không được là private.

Ví dụ

```
class Parent {
    public void doSomething() {}
    protected int doSomething2() {
        return 0;
    }
}

class Child extends Parent {
    protected void doSomething() {}
    protected void doSomething2() {}
}
```

Không ghi đè được do không cùng kiểu trả về

Không ghi đè được do chỉ định truy cập yếu hơn (public -> protected)

Quy định trong ghi đè

- Không được phép ghi đè
  + Các phương thức static trong lớp cha
  + Các phương thức private trong lớp cha
  + Các phương thức hằng (final) trong lớp cha

Sử dụng từ khoá super

- Từ khoá `super` được sử dụng để gọi đến các phương thức của lớp cha
- Mục đích nhằm tái sử dụng lại code đã được viết trong lớp cha

Sử dụng từ khoá super

Cú pháp

- Gọi phương thức khởi tạo

```
super(danh sách tham số);
```

  - Bắt buộc nếu lớp cha không có phương thức khởi tạo mặc định

- Gọi các phương thức của lớp cha

```
super.tênPt(danh sách tham số);
```

Ví dụ

```
package abc;
public class Person {
    protected String name;
    protected int age;
    public String getDetail() {
        String s = name + "," + age;
        return s;
    }
}

import abc.Person;
public class Employee extends Person {
  double salary;
  public String getDetail() {
      String s = super.getDetail() + "," + salary;
      return s;
    }
}
```