Kế thừa – Inheritance

Trừu tượng - Abstract

ĐẠI HỌC SÀI GÒN

19/09/2023

MỤC TIÊU

- *Khái niệm kế thừa.*
- *Mối quan hệ IS-A*
- *Lớp trừu tượng và phương thức trừu tượng.*

Kế thừa (Inheritance)

Tính Kế Thừa

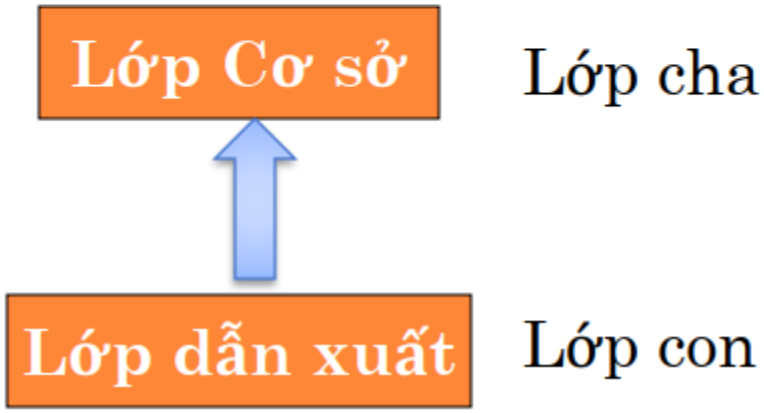
- Kế thừa là một trong các đặc điểm chính của Lập trình hướng đối tượng
- Là cơ chế cho phép một lớp tiếp nhận thuộc tính và phương thức từ một lớp khác, giúp tái sử dụng và mở rộng mã nguồn mà không cần phải viết lại.
- Quan hệ kế thừa giữa 2 lớp gọi là quan hệ IS - A

Ví dụ

- Lớp Canhan:
  - Thuộc tính: madinhdanh, ho, tenlot, tenchinh, gioitinh, tenTat, thuongtru, ngaysinh.
- Lớp SinhVien Kế Thừa Canhan
  - Tất cả thuộc tính của Canhan và thêm thuộc tính cho SinhVien như: mã sinh viên, lớp, ngành học...

Lớp cơ sở và lớp dẫn xuất

- Lớp cơ sở (base class): là lớp đã được xây dựng
- Lớp dẫn xuất(derived class): là lớp cần được xây dựng dựa trên lớp cơ sở



Lớp cha

Lớp con

Khai báo lớp kế thừa

```
public class LopCon extends LopCha {

    // Nội dung của lớp

}
```

Ví dụ

```
public class Sinhvien extends Canhan {
    private String maSinhVien;
    private String lopHoc;
    private String nganhHoc;

    // Constructor, getters, setters...
}
```

Truy xuất

Lớp con được phép truy xuất các thành phần protected và public của lớp cha.

Lớp Object là lớp gốc, tất cả các lớp đều kế thừa từ lớp Object.

Từ khóa super

- Từ khóa super trong Java là một biến tham chiếu được sử dụng để tham chiếu đến đối tượng của lớp cha gần nhất.
- Khi nào sử dụng?
  - Để gọi phương thức của lớp cha.
  - Để truy cập vào biến thành viên của lớp cha.
  - Để gọi constructor của lớp cha.

Ví dụ về hàm khởi tạo

- Từ khóa super phải luôn là câu lệnh đầu tiên trong constructor nếu được sử dụng để gọi constructor của lớp cha.

```
class LopCha {
    LopCha() {
        System.out.println("Constructor của lớp cha.");
    }
}

class LopCon extends LopCha {
    LopCon() {
        super();
        System.out.println("Constructor của lớp con.");
    }
}
```

Ví dụ

```
class LopCha {
    void hienThi() {
        System.out.println("Đây là phương thức trong lớp cha.");
    }
}

class LopCon extends LopCha {
    void hienThi() {
        super.hienThi();
        System.out.println("Đây là phương thức trong lớp con.");
    }
}
```

Ghi Đè (Method Overriding Overriding)

- Khi một lớp con cung cấp một triển khai cụ thể cho một phương thức đã được định nghĩa trong lớp cha của nó.
- Tính chất:
  - Phương thức phải có cùng tên như trong lớp cha.
  - Phương thức phải có cùng kiểu trả về (hoặc là kiểu con của kiểu trả về của lớp cha).

Ví dụ

```
class Animal {
    void sound() {
        System.out.println("Animal makes a sound");
    }
}
class Dog extends Animal {
    void sound() {
        System.out.println("Dog barks");
    }
}
```

Mở Rộng Phương Thức

- Khi một lớp con kế thừa một phương thức từ lớp cha và muốn sử dụng phần chức năng gốc của phương thức đó cùng với việc thêm các chức năng mới, có thể ghi đè phương thức và trong đó gọi lại phương thức của lớp cha bằng từ khóa super."

Ví dụ

```
class Animal {
    void sound() {
        System.out.println("Animal makes a sound");
    }
}
class Dog extends Animal {
    void sound() {
        super.sound();
        System.out.println("Dog barks");
    }
}
```

Sự Chuyển Kiểu (Type Casting)

- Chuyển kiểu là quá trình biến đổi một biến từ một kiểu dữ liệu này sang một kiểu dữ liệu khác. Trong lập trình, có hai loại chuyển kiểu chính: chuyển kiểu ngầm định và chuyển kiểu tường minh.

Chuyển kiểu ngầm định (Implicit Casting):

- Xảy ra khi biến từ một kiểu dữ liệu có kích thước nhỏ hơn được chuyển thành kiểu dữ liệu có kích thước lớn hơn.
- Ví dụ: chuyển từ int sang double trong Java.

Chuyển kiểu tường minh (Explicit Casting):

- Xảy ra khi biến từ một kiểu dữ liệu có kích thước lớn hơn được chuyển thành kiểu dữ liệu có kích thước nhỏ hơn.
- Cần phải chỉ định kiểu dữ liệu mục tiêu.
- Ví dụ: chuyển từ double sang int trong Java.

Upcasting

- Upcasting là việc chuyển một đối tượng từ lớp con (subclass) lên lớp cha (superclass).
- Điều này luôn an toàn và thường xảy ra một cách tự động (ngầm định).
- Upcasting giúp sử dụng đa hình - cho phép một đối tượng được xử lý như là một đối tượng của lớp cha của nó.

Upcasting

- Upcasting là việc chuyển một đối tượng từ lớp con (subclass) lên lớp cha (superclass).
- Điều này luôn an toàn và thường xảy ra một cách tự động (ngầm định).
- Upcasting giúp sử dụng đa hình - cho phép một đối tượng được xử lý như là một đối tượng của lớp cha của nó.
- Ví dụ:
  - Sinhvien sinhVien = new SinhvienNoitru();
  - Canhan canhan = sinhVien; // Upcasting

Downcasting

- Downcasting là việc chuyển một đối tượng từ lớp cha (superclass) xuống lớp con (subclass).
- Trái ngược với upcasting, downcasting không an toàn và luôn cần phải được thực hiện một cách tường minh. Nếu không, nó có thể dẫn đến lỗi thời gian chạy nếu đối tượng không phải là một thể hiện của lớp con.
- Trước khi thực hiện downcasting, thường sử dụng instanceof để kiểm tra.

Ví dụ

- Canhan canhan = new SinhvienNoitru();
- if(canhan instanceof SinhvienNoitru) {
- SinhvienNoitru sinhVienNoiTru = (SinhvienNoitru) canhan; // Downcasting
- }

Ví dụ lỗi downcasting

```
public static void main(String[] args) {
    Canhan banMinh = new Canhan();

    // Cố gắng downcasting Canhan
    //(không phải SinhvienNoitru) sang SinhvienNoitru
    SinhvienNoitru svMinh = (SinhvienNoitru) banMinh;
    // Điều này sẽ gây ra lỗi ClassCastException
```

Lớp trừu tượng (Abstract class)

Khái niệm

- Lớp trừu tượng là một cơ sở đặc biệt trong lập trình hướng đối tượng. Khác biệt chính của nó so với các lớp thông thường là **không thể tạo ra một đối tượng** trực tiếp từ nó.
- Trong lớp trừu tượng, có thể tồn tại những phương thức không có phần thực thi chi tiết - gọi là **phương thức trừu tượng**. Tuy nhiên, lớp trừu tượng cũng có thể chứa phương thức bình thường với đầy đủ xử lý.
- Mục tiêu chính của lớp trừu tượng là đề ra một bản mẫu, một **"hợp đồng"** mà các lớp con sẽ phải tuân theo khi kế thừa từ nó.

Cú pháp

```
public abstract class TenLopTruuTuong {
    // Phần dữ liệu
    // Phương thức không trừu tượng
    public void phuongThucKhongTruuTuong() {
        // ...
    }
    // Phương thức trừu tượng
    public abstract void phuongThucTruuTuong();
}
```

Đặc điểm của Lớp Trừu Tượng

- Không thể tạo đối tượng trực tiếp từ lớp trừu tượng.
- Có thể có biến, phương thức, và các khối mã.
- Cần phải được kế thừa và phương thức trừu tượng của nó phải được triển khai trong lớp con.

Lợi ích của Lớp Trừu Tượng

- Cung cấp một bản mẫu cho các lớp con.
- Tăng sự tái sử dụng mã.
- Giúp quản lý và mở rộng mã dễ dàng hơn.

Ví dụ

```
public abstract class Nhanvien {
    private String maNhanVien;
    private String hoVaTen;

    public Nhanvien(String maNhanVien, String hoVaTen) {
        this.maNhanVien = maNhanVien;
        this.hoVaTen = hoVaTen;
    }
```

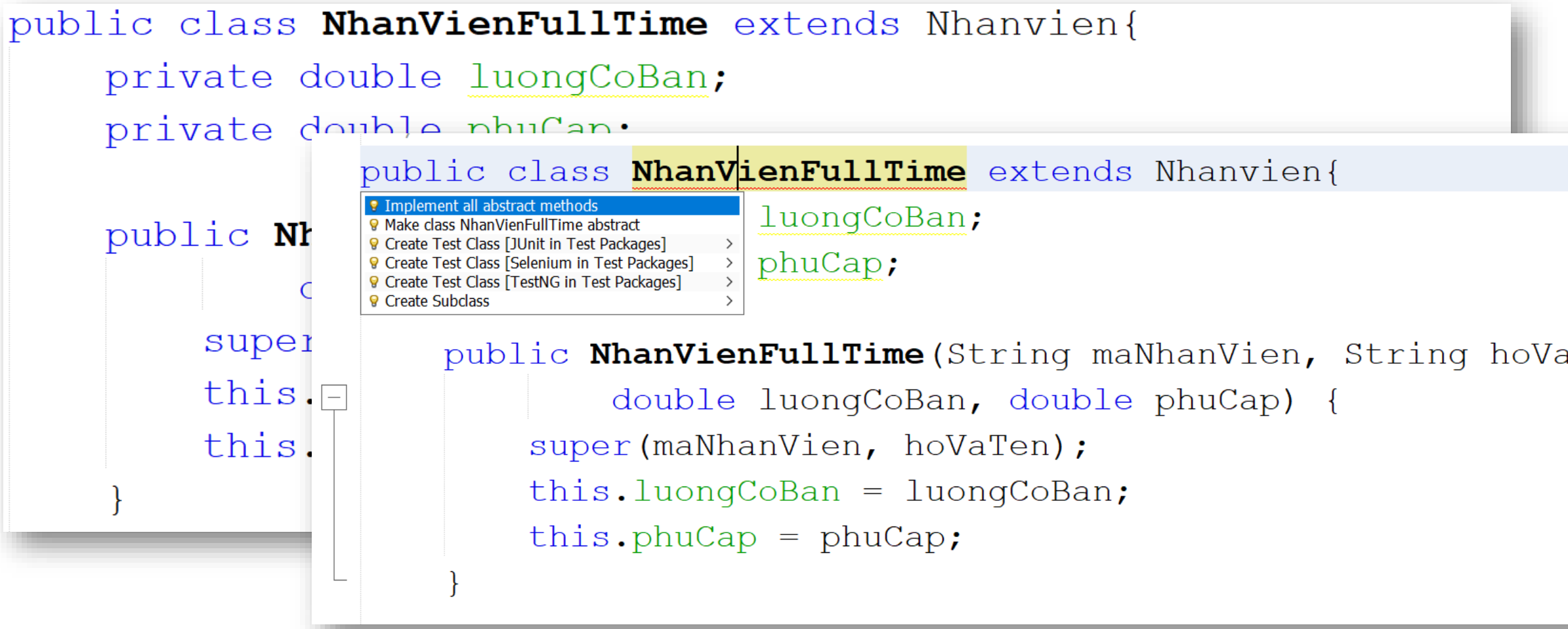
Ví dụ

```
    // Phương thức trừu tượng
    public abstract double tinhLuong();

    // Phương thức trừu tượng
    public abstract double tinhThuong();

    // Phương thức không trừu tượng
    public void inThongTin() {
        System.out.println("Mã nhân viên: " + maNhanVien);
        System.out.println("Họ và tên: " + hoVaTen);
    }
}
```

Ví dụ

```
public class NhanVienFullTime extends Nhanvien{
    private double luongCoBan;
    private double phuCap;

    public NhanVienFullTime(String maNhanVien, String hoVaTen,
            double luongCoBan, double phuCap) {
        super(maNhanVien, hoVaTen);
        this.luongCoBan = luongCoBan;
        this.phuCap = phuCap;
    }
```

Ví dụ


public class NhanVienFullTime extends Nhanvien{
private double luongCoBan;
private double phuCap;
public Nh
super
this.
this.
}
public class NhanVienFullTime extends Nhanvien{
Implement all abstract methods
Make class NhanVienFullTime abstract
Create Test Class [JUnit in Test Packages]
Create Test Class [Selenium in Test Packages]
Create Test Class [TestNG in Test Packages]
Create Subclass
luongCoBan;
phuCap;
public NhanVienFullTime(String maNhanVien, String hoVa
double luongCoBan, double phuCap) {
super(maNhanVien, hoVaTen);
this.luongCoBan = luongCoBan;
this.phuCap = phuCap;
}

Ví dụ

```java
@Override
public double tinhLuong() {
    return luongCoBan + phuCap;
}

@Override
public double tinhThuong() {
    return luongCoBan * 0.1;
}
```

Ví dụ

```
@Override
public void inThongTin() {
    super.inThongTin(); // gọi phương thức từ lớp cha
    System.out.println("Lương: " + tinhLuong());
    System.out.println("Thưởng: " + tinhThuong());
}
```

Lưu ý

- Một lớp trừu tượng (abstract class) không nhất thiết phải có phương thức trừu tượng (abstract method), nhưng nếu một lớp có có phương thức trừu tượng, thì nó phải được khai báo là abstract.

Các "lớp nội tuyến" (Inner Class)

- Inner Class : Được định nghĩa trong một lớp khác.
- Local Inner Class (lớp nội tuyến cục bộ): Được định nghĩa trong một phương thức.
- Anonymous Inner Class (lớp nội tuyến vô danh): Không có tên và thường được sử dụng để tạo đối tượng ngay lập tức từ một interface hoặc lớp trừu tượng.
- Static Nested Class (lớp lồng nhưng là static): Được định nghĩa như một static member trong một lớp khác.

Khai báo đối tượng từ lớp trừu tượng

- Cách "gián tiếp" là thông qua việc kế thừa.
  - tạo một lớp con kế thừa từ lớp trừu tượng, triển khai tất cả các phương thức trừu tượng của lớp cha (nếu có) và từ lớp con này tạo đối tượng.
- Sử dụng các lớp nội tuyến hoặc biểu thức lambda, có thể xem xét việc "khởi tạo" một đối tượng từ lớp trừu tượng, nhưng thực chất đó là việc tạo một **lớp vô danh (**Anonymous Inner Class) triển khai lớp trừu tượng đó.

Anonymous Inner Class

```
    LopTruuTuong obj = new LopTruuTuong() {
    @Override
    void phuongThucTruuTuong() {
        System.out.println("Triển khai phương thức trừu tượng"
                + " trong lớp vô danh.");
    }
};
```

Ví dụ

```
public class QuanLyNhanVien {
  private ArrayList<Nhanvien> danhSachNhanVien = new ArrayList<>();
```

```
public void themNhanVienFullTime(String maNhanVien, String hoten) {
    Nhanvien fullTime = new Nhanvien(maNhanVien,hoten) {
        @Override
        public double tinhLuong() {
            return 5000;
        }

        @Override
        public double tinhThuong() {
            return 100;
        }
    };
```

Ví dụ

```
public void hienThiLuongChoTungNhanVien() {
    for (Nhanvien nv : danhSachNhanVien) {
        System.out.println("Nhan vien " + nv.getHoVaTen() +
                " co luong: " + nv.tinhLuong());
    }
}
```

```
public static void main(String[] args) {
    QuanLyNhanVien ql = new QuanLyNhanVien();
    ql.themNhanVienFullTime("10063","Nguyen");
    ql.themNhanVienFullTime("10064","Minh");

    ql.hienThiLuongChoTungNhanVien();
}
```

TRAO ĐỔI


SGU
TRƯỜNG ĐẠI HỌC SÀI GÒN
SAIGON UNIVERSITY
ĐẠI HỌC SÀI GÒN
CẤM ĐẬU XE