HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
Posts and Telecommunications Institute of Technology

RIPT

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

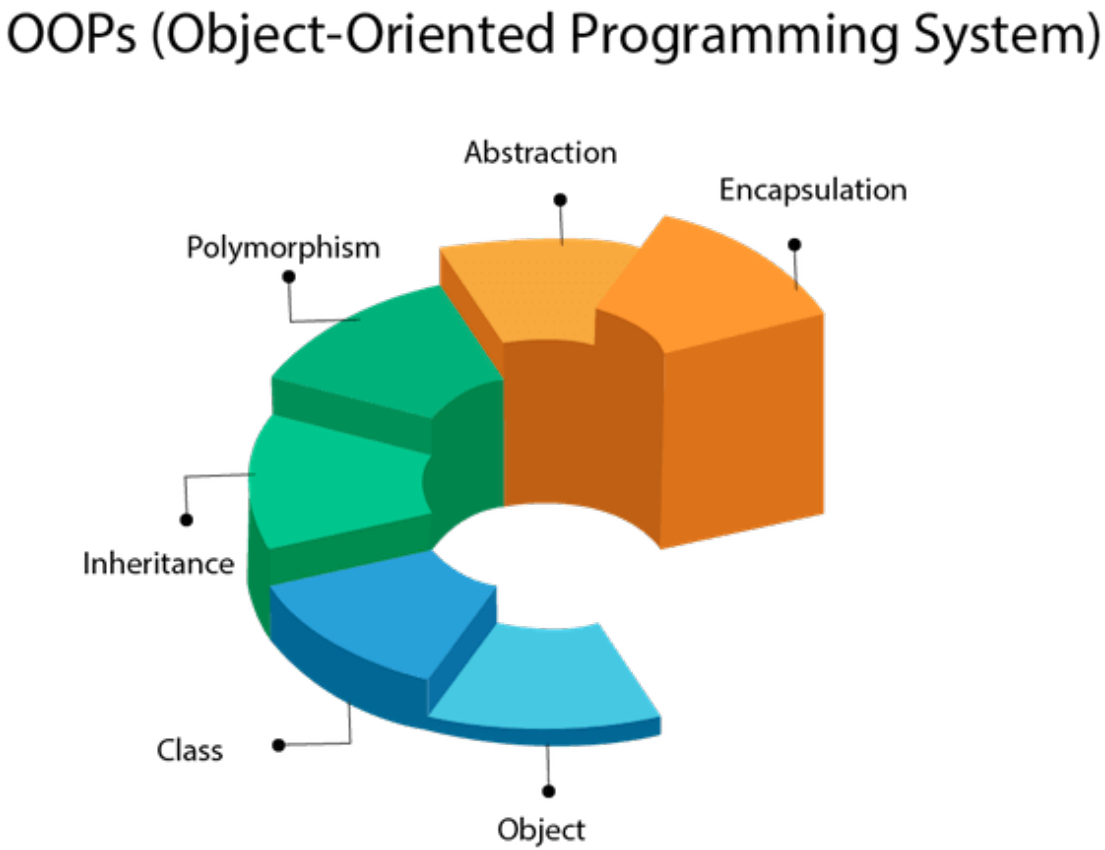
Nguyễn Việt Dũng

dungnv1@ptit.edu.vn

KHÁI NIỆM CƠ BẢN

Lập trình hướng đối tượng

- Đối tượng
- Lớp đối tượng
- Trừu tượng hóa đối tượng
- Kế thừa
- Đóng gói
- Đa hình
- Trừu tượng



Lập trình hướng đối tượng – Đối tượng

Đối tượng **(Object)** là một **thực thể hoạt động khi chương trình đang chạy** và được xác định bằng 3 yếu tố:

- **Định danh**: Xác định **duy nhất** một đối tượng trong hệ thống
- **Trạng thái:** Tổ hợp các giá trị **thuộc tính** của đối tượng
- **Hoạt động**: Các **hành động** mà đối tượng có thể thực hiện

Lập trình hướng đối tượng – Đối tượng

B23DCCC001

Họ tên: Nguyễn Văn A
Năm sinh: 2004
Khóa: D23
Lớp tín chỉ: INT1223-20241, INT1413-20241

Xem hồ sơ cá nhân
Đăng ký tín chỉ
Xem danh sách lớp

VKH10003

Họ tên: Nguyễn Văn B
Năm sinh: 1980
Đơn vị công tác: Học viện Công nghệ Bưu chính Viễn thông
Vị trí công việc: Cán bộ

Xem hồ sơ cá nhân
Xem vị trí công việc
Cập nhật vị trí công việc

Lập trình hướng đối tượng – Lớp đối tượng

Lớp đối tượng **(Class)** là một khái niệm trừu tượng, dùng để chỉ một tập hợp các đối tượng có mặt trong hệ thống.

Thuộc tính ←
Họ tên: Nguyễn Văn A
Năm sinh: 2004
Khóa: D23
Lớp tín chỉ: INT1223-20241, INT1413-20241

Phương thức ←
Xem hồ sơ cá nhân
Đăng ký tín chỉ
Xem danh sách lớp

Lập trình hướng đối tượng – Lớp đối tượng

Lớp **Sinh viên**

Thuộc tính:
- Mã sinh viên (Chữ)
- Họ tên (Chữ)
- Năm sinh (Số)
- Khóa (Chữ)
- Lớp tín chỉ (Mảng chữ)

Phương thức:
- Xem hồ sơ cá nhân
- Đăng ký tín chỉ
- Xem danh sách lớp

Lớp **Cán bộ**

Thuộc tính:
- Họ tên (Chữ)
- Năm sinh (Số)
- Đơn vị công tác (Đơn vị)
- Vị trí công việc (Chữ)

Phương thức:
- Xem hồ sơ cá nhân
- Xem vị trí công việc
- Cập nhật vị trí công việc

Lập trình hướng đối tượng – Phân biệt

Lớp	Đối tượng
Khái niệm trừu tượng	Thực thể vật lý
Là khuôn mẫu **(blueprint)** của đối tượng, sử dụng để định nghĩa và khởi tạo ra các đối tượng	Đối tượng là sự thể hiện **(instance)** của một lớp
Không chiếm dụng bộ nhớ khi khởi tạo lớp	Chiếm dụng bộ nhớ ngay khi khởi tạo đối tượng
Khai báo lớp 1 lần	Có thể khởi tạo nhiều đối tượng trong 1 lớp, theo yêu cầu

Lập trình hướng đối tượng – Trừu tượng hóa đối tượng

Trừu trượng hóa là quá trình tổ chức một bài toán phức tạp thành những đối tượng có cấu trúc chặt chẽ, trong đó các dữ liệu và hành động của đối tượng được định nghĩa, có sự gắn kết với nhau

Theo dữ liệu	Theo chức năng
Thuộc tính	Phương thức

Lập trình hướng đối tượng – Trừu tượng hóa đối tượng

1. Tập hợp tất cả các thông tin/hành động có thể có của các đối tượng.
2. Nhóm các đối tượng có các thông tin/hoạt động tương tự nhau, loại bỏ bớt các thông tin/hoạt động cá biệt, tạo thành một nhóm đối tượng chung.
3. Mỗi nhóm đối tượng đề xuất một lớp đối tượng tương ứng.
4. Các thông tin/hành động chung của nhóm đối tượng sẽ cấu thành các thuộc tính/phương thức của lớp tương ứng

Lập trình hướng đối tượng – Trừu tượng hóa đối tượng

B23DCCC001

Họ tên: Nguyễn Văn A
Năm sinh: 2004
Khóa: D23
Lớp tín chỉ: INT1223-20241, INT1413-20241
Sở thích: Thể thao, đọc sách
Tình trạng việc làm: Thực tập sinh

Xem hồ sơ cá nhân
Đi học
Đi thực tập
Đọc sách

B23DCCC002

Họ tên: Trần Thị C
Năm sinh: 2004
Khóa: D23
Lớp tín chỉ: INT1223-20241, INT1413-20241
Chiều cao: 170 cm
Tình trạng hôn nhân: Đã có người yêu
Thông tin người yêu: Nguyễn Văn D

Xem hồ sơ cá nhân
Đi học
Đăng ký tín chỉ
Hẹn hò

Lập trình hướng đối tượng – Trừu tượng hóa đối tượng

B23DCCC001

Họ tên: Nguyễn Văn A
Năm sinh: 2004
Khóa: D23
Lớp tín chỉ: INT1223-20241, INT1413-20241
~~Sở thích: Thể thao, đọc sách~~
~~Tình trạng việc làm: Thực tập sinh~~

Xem hồ sơ cá nhân
Đi học
~~Đi thực tập~~
~~Đọc sách~~

B23DCCC002

Họ tên: Trần Thị C
Năm sinh: 2004
Khóa: D23
Lớp tín chỉ: INT1223-20241, INT1413-20241
~~Chiều cao: 170 cm~~
~~Tình trạng hôn nhân: Đã có người yêu~~
~~Thông tin người yêu: Nguyễn Văn D~~

Xem hồ sơ cá nhân
Đi học
Đăng ký tín chỉ
~~Hẹn hò~~

Lập trình hướng đối tượng – Trừu tượng hóa đối tượng

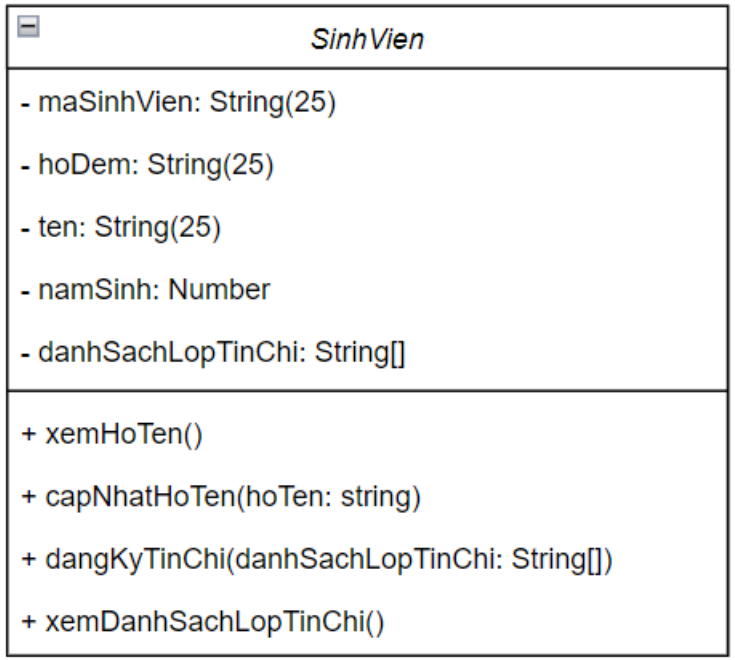
Lớp **Sinh viên**

Thuộc tính:

- Mã sinh viên (Chữ)
- Họ tên (Chữ)
- Năm sinh (Số)
- Khóa (Chữ)
- Lớp tín chỉ (Mảng chữ)

Phương thức:

- Xem hồ sơ cá nhân
- Điểm danh
- Đăng ký tín chỉ
- Xem danh sách lớp



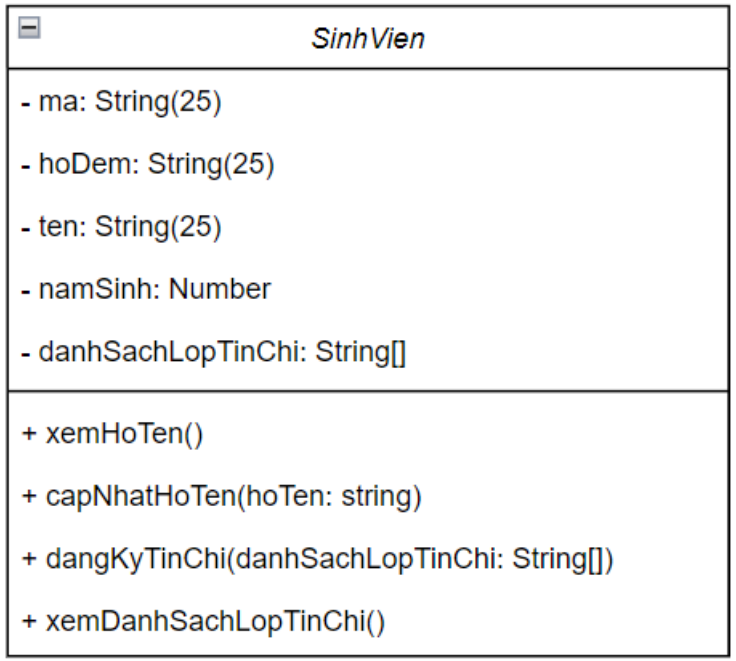
Lập trình hướng đối tượng – Trừu tượng hóa đối tượng

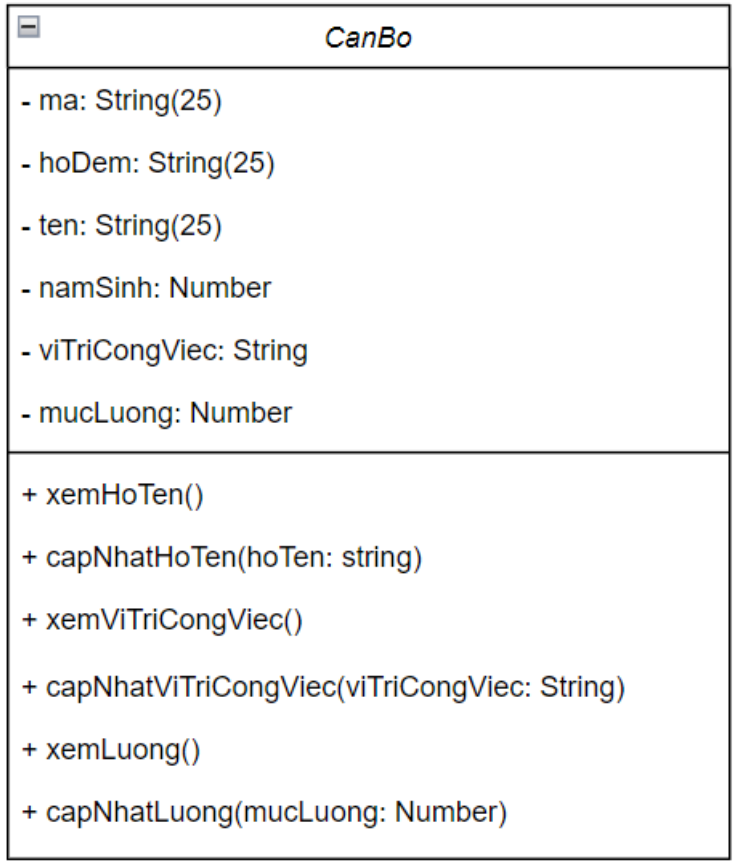
- Tập trung vào vấn đề cần quan tâm
- Xác định những đặc tính thiết yếu và những hành động cần thiết
- Giảm thiểu những chi tiết không cần thiết

```java
import java.util.ArrayList;

public class SinhVien {
    // Thuộc tính (Attributes)
    private String maSinhVien;
    private String hoDem;
    private String ten;
    private int namSinh;
    private ArrayList<String> danhSachLopTinChi;

    // Phương thức khởi tạo (Constructor)
    public SinhVien(String maSinhVien, String hoDem, String ten, int namSinh) {
        this.maSinhVien = maSinhVien;
        this.hoDem = hoDem;
        this.ten = ten;
        this.namSinh = namSinh;
        this.danhSachLopTinChi = new ArrayList<>();
    }

    // Phương thức xem họ tên (Method xemHoTen)
    public String xemHoTen() {
        return hoDem + " " + ten;
    }

    // Phương thức cập nhật họ tên (Method capNhatHoTen)
    public void capNhatHoTen(String hoTen) {
        String[] parts = hoTen.split(" ");
        this.hoDem = String.join(" ", parts, 0, parts.length - 1);
        this.ten = parts[parts.length - 1];
    }

    // Phương thức đăng ký tín chỉ (Method dangKyTinChi)
    public void dangKyTinChi(ArrayList<String> danhSachLopTinChi) {
        this.danhSachLopTinChi.addAll(danhSachLopTinChi);
    }

    // Phương thức xem danh sách lớp tín chỉ (Method xemDanhSachLopTinChi)
    public ArrayList<String> xemDanhSachLopTinChi() {
        return this.danhSachLopTinChi;
    }
}
```

```java
Codeium: Refactor | Explain
public class Main {
    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        // Tạo một đối tượng của lớp SinhVien
        SinhVien sv = new SinhVien("SV001", "Nguyen Van", "A", 2004);

        // Gọi phương thức xemHoTen()
        System.out.println("Họ tên: " + sv.xemHoTen());

        // Cập nhật họ tên
        sv.capNhatHoTen("Tran Van Binh");
        System.out.println("Họ tên sau khi cập nhật: " + sv.xemHoTen());

        // Đăng ký lớp tín chỉ
        ArrayList<String> dsLopTinChi = new ArrayList<>();
        dsLopTinChi.add("LTC001");
        dsLopTinChi.add("LTC002");
        sv.dangKyTinChi(dsLopTinChi);

        // Xem danh sách lớp tín chỉ đã đăng ký
        System.out.println("Danh sách lớp tín chỉ đã đăng ký: " + sv.xemDanhSachLopTinChi());
    }
}
```

Lập trình hướng đối tượng – Tính kế thừa

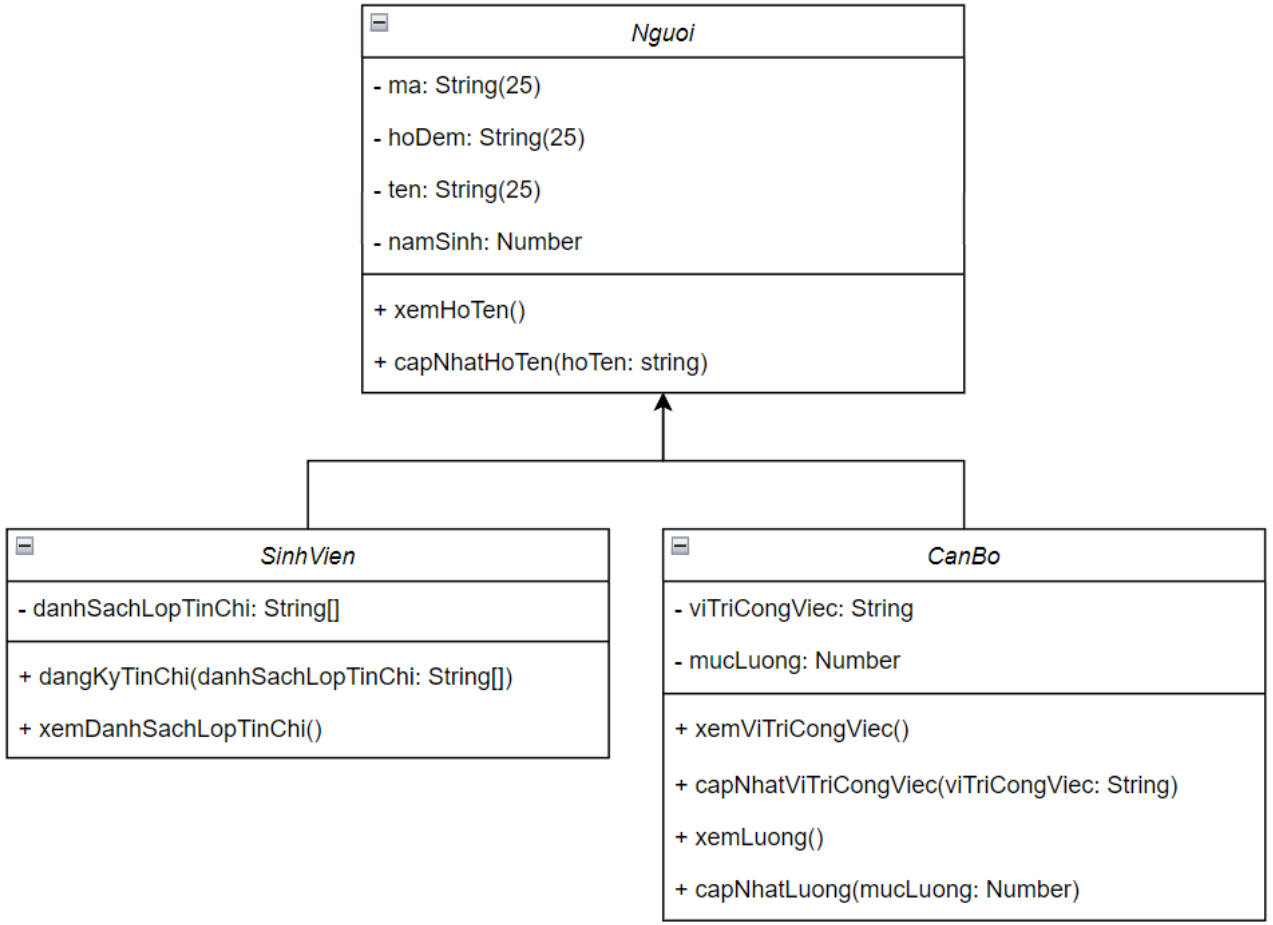




```java
Codeium: Refactor | Explain
public class CanBo {
    // Thuộc tính (Attributes)
    private String ma;
    private String hoDem;
    private String ten;
    private int namSinh;
    private String viTriCongViec;
    private double mucLuong;

    // Phương thức khởi tạo (Constructor)
    public CanBo(String ma, String hoDem, String ten, int namSinh, String viTriCongViec, double mucLuong) {
        this.ma = ma;
        this.hoDem = hoDem;
        this.ten = ten;
        this.namSinh = namSinh;
        this.viTriCongViec = viTriCongViec;
        this.mucLuong = mucLuong;
    }

    // Phương thức xem họ tên (Method xemHoTen)
    Codeium: Refactor | Explain | ×
    public String xemHoTen() {
        return hoDem + " " + ten;
    }

    // Phương thức cập nhật họ tên (Method capNhatHoTen)
    Codeium: Refactor | Explain | ×
    public void capNhatHoTen(String hoTen) {
        String[] parts = hoTen.split(" ");
        this.hoDem = String.join(" ", parts, 0, parts.length - 1);
        this.ten = parts[parts.length - 1];
    }

    // Phương thức xem vị trí công việc (Method xemViTriCongViec)
    Codeium: Refactor | Explain | ×
    public String xemViTriCongViec() {
        return this.viTriCongViec;
    }

    // Phương thức cập nhật vị trí công việc (Method capNhatViTriCongViec)
    Codeium: Refactor | Explain | ×
    public void capNhatViTriCongViec(String viTriCongViec) {
        this.viTriCongViec = viTriCongViec;
    }
```

```java
Codeium: Refactor | Explain
public class Main {
    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        // Tạo một đối tượng của lớp CanBo
        CanBo cb = new CanBo("CB001", "Tran Van", "Hung", 1985, "Giam doc", 1500.0);

        // Gọi phương thức xemHoTen()
        System.out.println("Họ tên: " + cb.xemHoTen());

        // Cập nhật họ tên
        cb.capNhatHoTen("Nguyen Van Hoang");
        System.out.println("Họ tên sau khi cập nhật: " + cb.xemHoTen());

        // Xem vị trí công việc
        System.out.println("Vị trí công việc: " + cb.xemViTriCongViec());

        // Cập nhật vị trí công việc
        cb.capNhatViTriCongViec("Truong phong");
        System.out.println("Vị trí công việc sau khi cập nhật: " + cb.xemViTriCongViec());

        // Xem mức lương
        System.out.println("Mức lương: " + cb.xemLuong());

        // Cập nhật mức lương
        cb.capNhatLuong(1800.0);
        System.out.println("Mức lương sau khi cập nhật: " + cb.xemLuong());
    }
}
```

Lập trình hướng đối tượng – Tính kế thừa



```java
Codelum: Refactor | Explain
public class Nguoi {
    // Thuộc tính (Attributes)
    protected String ma;
    protected String hoDem;
    protected String ten;
    protected int namSinh;

    // Phương thức khởi tạo (Constructor)
    public Nguoi(String ma, String hoDem, String ten, int namSinh) {
        this.ma = ma;
        this.hoDem = hoDem;
        this.ten = ten;
        this.namSinh = namSinh;
    }

    // Phương thức xem họ tên (Method xemHoTen)
    Codelum: Refactor | Explain | ×
    public String xemHoTen() {
        return hoDem + " " + ten;
    }

    // Phương thức cập nhật họ tên (Method capNhatHoTen)
    Codelum: Refactor | Explain | ×
    public void capNhatHoTen(String hoTen) {
        String[] parts = hoTen.split(" ");
        this.hoDem = String.join(" ", parts, 0, parts.length - 1);
        this.ten = parts[parts.length - 1];
    }
}
```

```java
Codeium: Refactor | Explain
public class SinhVien extends Nguoi {
    private ArrayList<String> danhSachLopTinChi;

    // Phương thức khởi tạo (Constructor)
    public SinhVien(String ma, String hoDem, String ten, int namSinh) {
        super(ma, hoDem, ten, namSinh);
        this.danhSachLopTinChi = new ArrayList<>();
    }

    // Phương thức đăng ký tín chỉ (Method dangKyTinChi)
    Codeium: Refactor | Explain | ×
    public void dangKyTinChi(ArrayList<String> danhSachLopTinChi) {
        this.danhSachLopTinChi.addAll(danhSachLopTinChi);
    }

    // Phương thức xem danh sách lớp tín chỉ (Method xemDanhSachLopTinChi)
    Codeium: Refactor | Explain | ×
    public ArrayList<String> xemDanhSachLopTinChi() {
        return this.danhSachLopTinChi;
    }
}
```

main3.java

```java
Codeium: Refactor | Explain
1  public class Main {
       Codeium: Refactor | Explain | Generate Javadoc | ×
2      public static void main(String[] args) {
3          // Tạo một đối tượng SinhVien
4          SinhVien sv = new SinhVien("SV001", "Nguyen Van", "An", 2000);
5          System.out.println("Sinh viên: " + sv.xemHoTen());
6
7          // Tạo một đối tượng CanBo
8          CanBo cb = new CanBo("CB001", "Tran Van", "Hung", 1985, "Giam doc", 1500.0);
9          System.out.println("Cán bộ: " + cb.xemHoTen());
10     }
11 }
12
```

Lập trình hướng đối tượng – Tính kế thừa

Tính kế thừa cho phép xây dựng một lớp mới (lớp Con/lớp dẫn xuất), kế thừa và tái sử dụng các thuộc tính, phương thức dựa trên lớp cũ (lớp Cha) đã có trước đó. Từ đó:

- Cho phép lớp dẫn xuất có thể sử dụng các thuộc tính và phương thức của lớp cơ sở tương tự như sử dụng các thuộc tính và phương thức của mình.
- Cho phép việc chỉ cần cài đặt phương thức ở một lớp cơ sở, mà có thể sử dụng được ở tất cả các lớp dẫn xuất.
- Cho phép tránh sự cài đặt trùng lặp mã nguồn của chương trình.
- Cho phép chỉ phải thay đổi một lần khi cần phải thay đổi dữ liệu của các lớp

Lập trình hướng đối tượng – Tính đóng gói

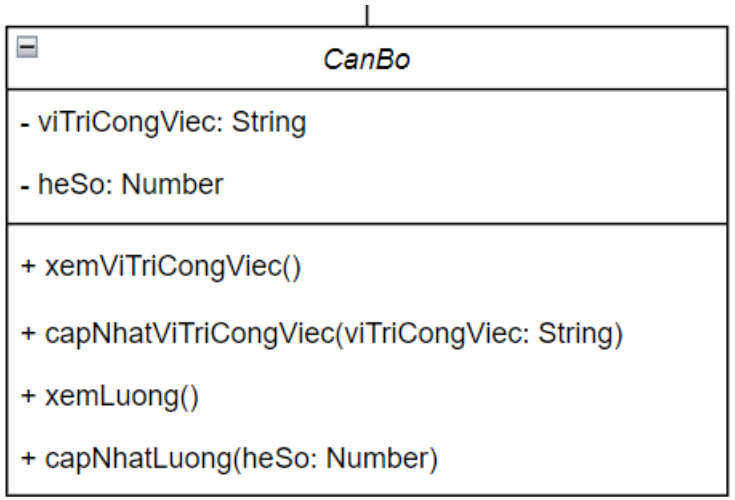
Lớp **CanBo** thể hiện thông tin cán bộ, giảng viên trong trường. Mỗi cán bộ, giảng viên có thông tin cá nhân, vị trí công việc đảm nhiệm và nhận mức lương tùy theo công việc.

Lương của cán bộ, giảng viên được tính theo công thức:

<Tiền lương> = <Hệ số lương> * <Lương cơ sở> + <Phụ cấp công việc>

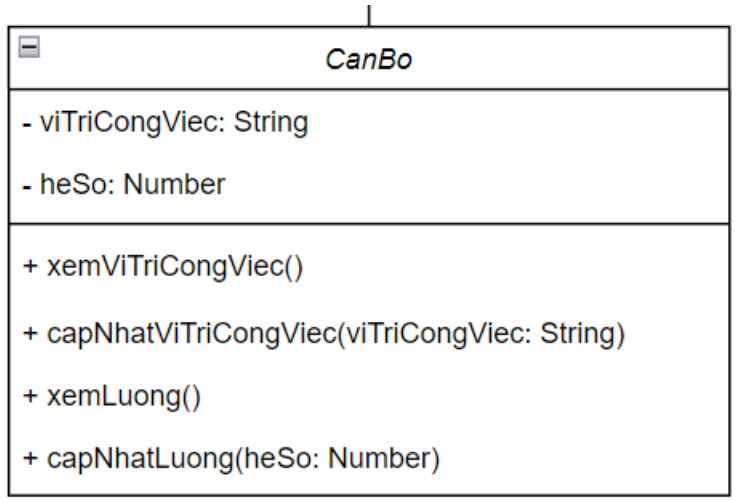
Trong đó:

- Phụ cấp công việc với cán bộ hành chính là mức 1.000.000 đồng, với giảng viên là 500.000 đồng;
- Hệ số lương không được vượt quá hệ số lương tối đa trong hệ thống



Lập trình hướng đối tượng – Tính đóng gói

1. Làm sao để khi hệ thống hoặc 1 module khác xem thông tin mức lương nhận của một cán bộ thì không thể biết cách tính lương?
2. Làm sao để khi hệ thống hoặc 1 module khác cập nhật hệ số lương cho cán bộ, chúng ta không bị vượt quá hệ số lương tối đa?



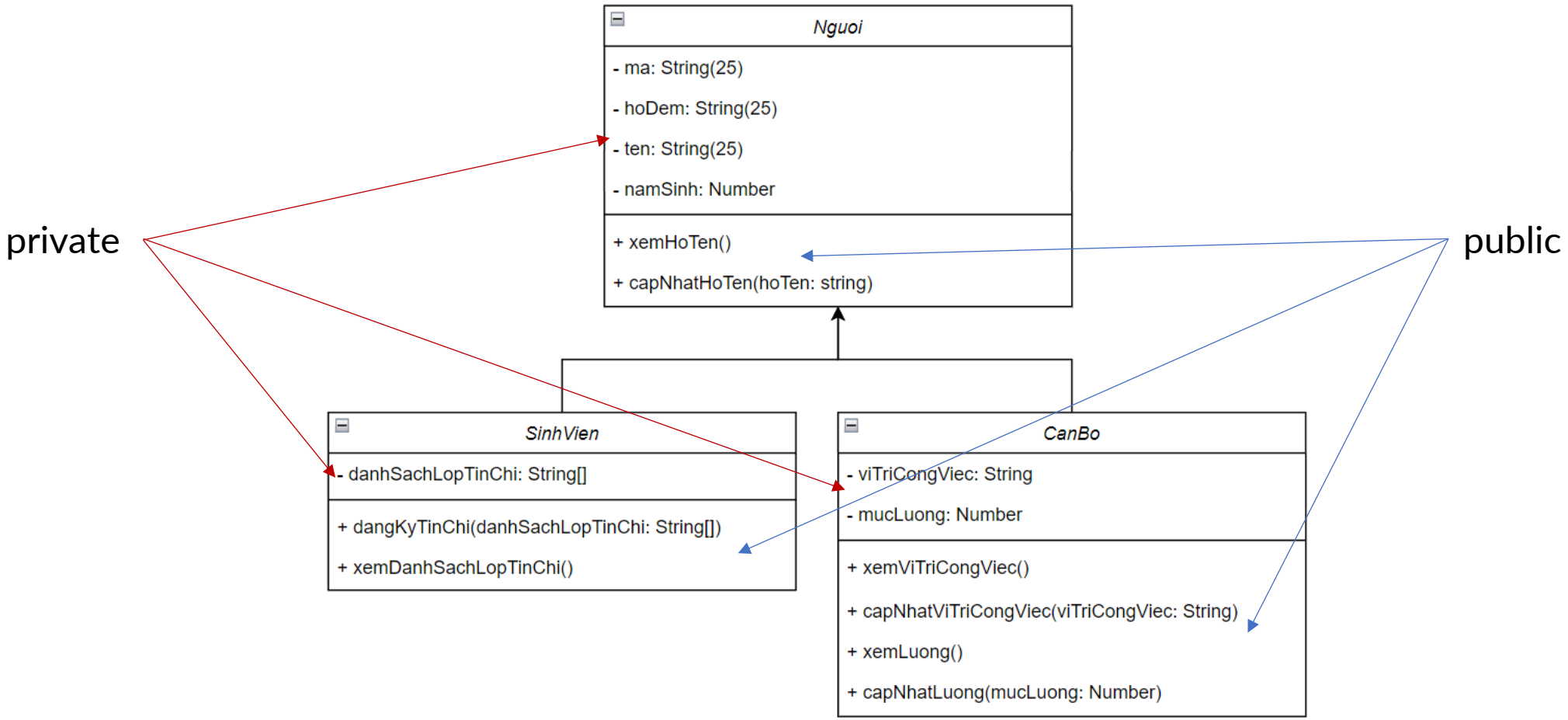
Lập trình hướng đối tượng – Tính đóng gói

Tính đóng gói **che giấu thông tin và những tính chất xử lý bên trong** của đối tượng. Các đối tượng khác **không thể tác động trực tiếp** đến dữ liệu bên trong và **làm thay đổi trạng thái** của đối tượng mà bắt buộc phải **thông qua các phương thức công khai** do đối tượng đó cung cấp

Phạm vi truy nhập (Scope) đảm bảo tính công khai và khả năng truy cập, sử dụng các thuộc tính, phương thức ở các đối tượng khác .

- private: Nội bộ lớp
- protected: Nội bộ lớp và các lớp dẫn xuất
- public: Công khai

Lập trình hướng đối tượng – Tính đóng gói



```java
public class CanBo extends Nguoi {
    private String viTriCongViec;
    private double heSoLuong;
    private double luongCoSo;
    private double phuCapCongViec;
    private static final double HE_SO_LUONG_TOI_DA = 10.0; // Hệ số lương tối đa

    // Phương thức khởi tạo (Constructor)
    public CanBo(String ma, String hoDem, String ten, int namSinh,
            String viTriCongViec, double heSoLuong, double luongCoSo) {
        super(ma, hoDem, ten, namSinh);
        this.viTriCongViec = viTriCongViec;
        this.luongCoSo = luongCoSo;
        setHeSoLuong(heSoLuong); // Gọi phương thức để đảm bảo không vượt quá hệ số lương tối đa
        this.phuCapCongViec = tinhPhuCapCongViec(viTriCongViec);
    }
```

```java
// Cập nhật hệ số lương
giangVien.setHeSoLuong(12.0);
System.out.println("Lương giảng viên sau khi cập nhật hệ số lương: "
    + giangVien.xemLuong());
```

```java
    // Phương thức tính phụ cấp công việc dựa trên vị trí công việc
    Codelum: Refactor | Explain | ×
    private double tinhPhuCapCongViec(String viTriCongViec) {
        if (viTriCongViec.equalsIgnoreCase("Cán bộ hành chính")) {
            return 1000000;
        } else if (viTriCongViec.equalsIgnoreCase("Giảng viên")) {
            return 500000;
        } else {
            return 0;
        }
    }

    // Phương thức thiết lập hệ số lương với kiểm tra giới hạn tối đa
    Codelum: Refactor | Explain | ×
    public void setHeSoLuong(double heSoLuong) {
        if (heSoLuong ⩽ HE_SO_LUONG_TOI_DA) {
            this.heSoLuong = heSoLuong;
        } else {
            System.out.println("Hệ số lương vượt quá giới hạn tối đa!");
            this.heSoLuong = HE_SO_LUONG_TOI_DA;
        }
    }

    // Phương thức xem lương (Method xemLuong)
    Codelum: Refactor | Explain | ×
    public double xemLuong() {
        return this.heSoLuong * this.luongCoSo + this.phuCapCongViec;
    }
}
```

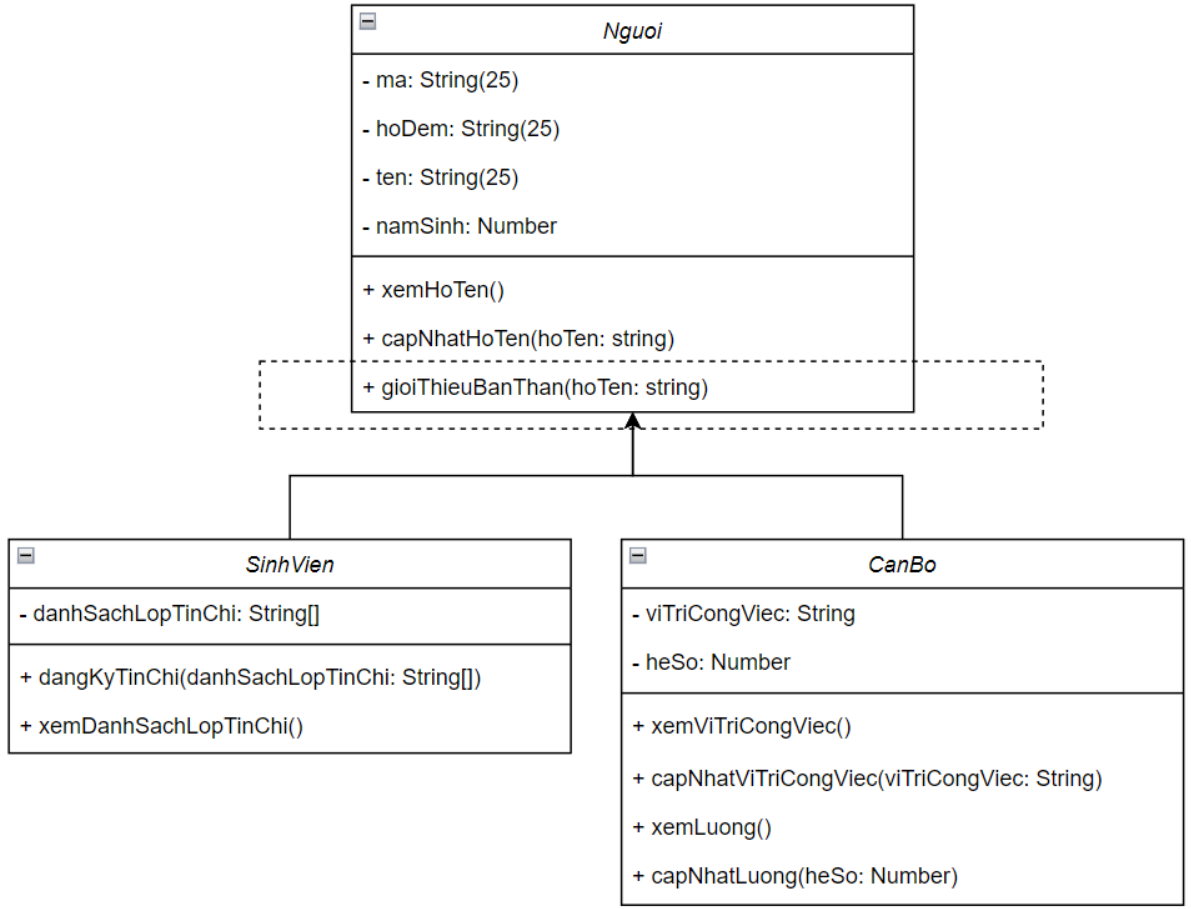
Lập trình hướng đối tượng – Tính đóng gói

- Cho phép che dấu dữ liệu bên trong của đối tượng.
- Cho phép che dấu sự cài đặt chi tiết bên trong của phương thức.
- Khi sử dụng chỉ cần gọi các phương thức theo một cách thống nhất, mặc dù các phương thức có thể được cài đặt khác nhau cho các trường hợp khác nhau.
- Cho phép hạn chế tối đa việc sửa lại mã chương trình.

Lập trình hướng đối tượng – Tính đa hình

Giới thiệu bản thân:

- Cả sinh viên và giảng viên khi giới thiệu bản thân sẽ hiển thị thông tin mã, họ tên, năm sinh
- Sinh viên khi giới thiệu bản thân sẽ hiển thị thêm đang học lớp nào
- Cán bộ khi giới thiệu bản thân sẽ hiển thị thêm đang ở vị trí công việc nào



```java
public class Nguoi {
    protected String ma;
    protected String hoDem;
    protected String ten;
    protected int namSinh;

    public Nguoi(String ma, String hoDem, String ten, int namSinh) {
        this.ma = ma;
        this.hoDem = hoDem;
        this.ten = ten;
        this.namSinh = namSinh;
    }

    Codelum: Refactor | Explain | Generate Javadoc | X
    public String gioiThieuBanThan() {
        return "Mã: " + ma + ", Họ tên: " + hoDem + " " + ten + ", Năm sinh: " + namSinh;
    }
}
```

```java
Codeium: Refactor | Explain
public class SinhVien extends Nguoi {
    private String lopHoc;

    public SinhVien(String ma, String hoDem, String ten, int namSinh, String lopHoc) {
        super(ma, hoDem, ten, namSinh);
        this.lopHoc = lopHoc;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    @Override
    public String gioiThieuBanThan() {
        return super.gioiThieuBanThan() + ", Đang học lớp: " + lopHoc;
    }
}
```

```java
public class CanBo extends Nguoi {
    private String viTriCongViec;

    public CanBo(String ma, String hoDem, String ten, int namSinh, String viTriCongViec) {
        super(ma, hoDem, ten, namSinh);
        this.viTriCongViec = viTriCongViec;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    @Override
    public String gioiThieuBanThan() {
        return super.gioiThieuBanThan() + ", Vị trí công việc: " + viTriCongViec;
    }
}
```

```java
public class Main {
    Codelum: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        SinhVien sv = new SinhVien("SV001", "Nguyen Van", "A", 2000, "CNTT1");
        CanBo cb = new CanBo("CB001", "Tran Van", "B", 1985, "Giam doc");

        System.out.println(sv.gioiThieuBanThan());
        System.out.println(cb.gioiThieuBanThan());
    }
}
```

```
Mã: SV001, Họ tên: Nguyen Van A, Năm sinh: 2000, Đang học lớp: CNTT1
Mã: CB001, Họ tên: Tran Van B, Năm sinh: 1985, Vị trí công việc: Giam doc
```

Lập trình hướng đối tượng – Tính đa hình

- Tính đa hình cho phép các đối tượng khác nhau thực thi phương thức giống nhau theo những cách khác nhau.
- Cho phép các lớp được định nghĩa các phương thức trùng nhau: cùng tên, cùng số lượng và kiểu tham số, cùng kiểu trả về.
- Nạp chồng phương thức **(override)**. Khi gọi các phương thức trùng tên, dựa vào đối tượng đang gọi mà chương trình sẽ thực hiện phương thức của lớp tương ứng, và do đó, sẽ cho các kết quả khác nhau.

Lập trình hướng đối tượng – Bài tập

1. Liệt kê tất cả các thuộc tính và hành động của đối tượng Xe ô tô. Đề xuất lớp Car (Ô tô).

2. Liệt kê tất cả các thuộc tính và hành động của đối tượng Xe buýt. Đề xuất lớp Bus.

3. Từ hai lớp Car và Bus của bài 1 và 2. Đề xuất một lớp Phương tiện (Vehicle) cho hai lớp trên kế thừa, để tránh trùng lặp dữ liệu giữa hai lớp Car và Bus.

4. Khai báo các lớp Car, Bus, Vehicle bằng một chương trình Java