Trường Cao đẳng Công nghệ Thông tin TP.HCM

Khoa Công nghệ Thông tin – Điện tử

Chương 5:

KẾ THỪA

Giảng viên: Hà Mỹ Trinh

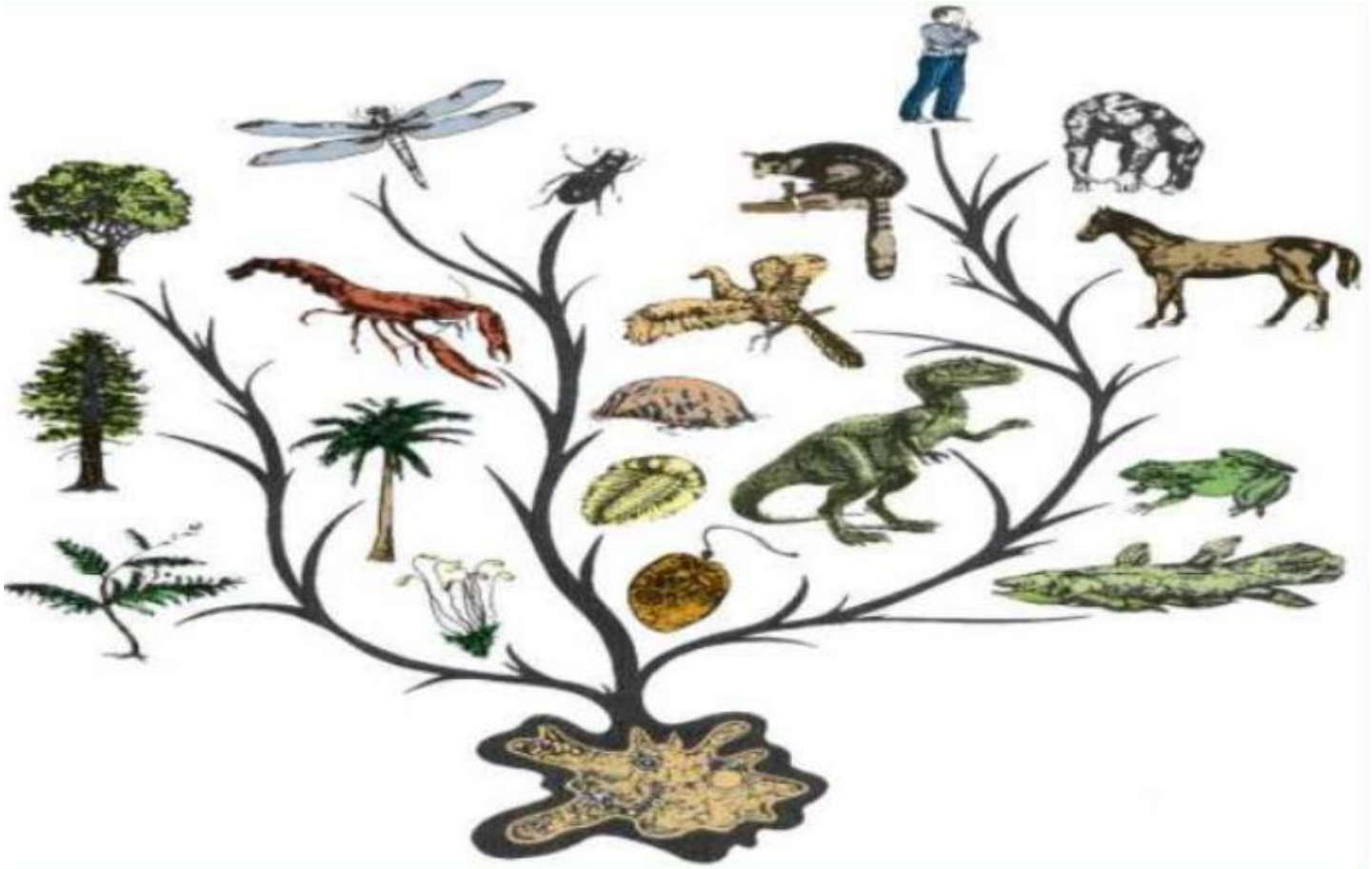
Email: trinhhm@itc.edu.vn

Tham khảo từ tài liệu Khoa

Nội dung

1. Khái niệm về kế thừa
2. Biểu diễn quan hệ kế thừa trong biểu đồ lớp
3. Nguyên lý kế thừa
4. Khởi tạo và hủy bỏ đối tượng
5. GHI ĐÈ PHƯƠNG THỨC (overriding)
6. Lớp vô sinh

1.1 Kế Thừa là gì?



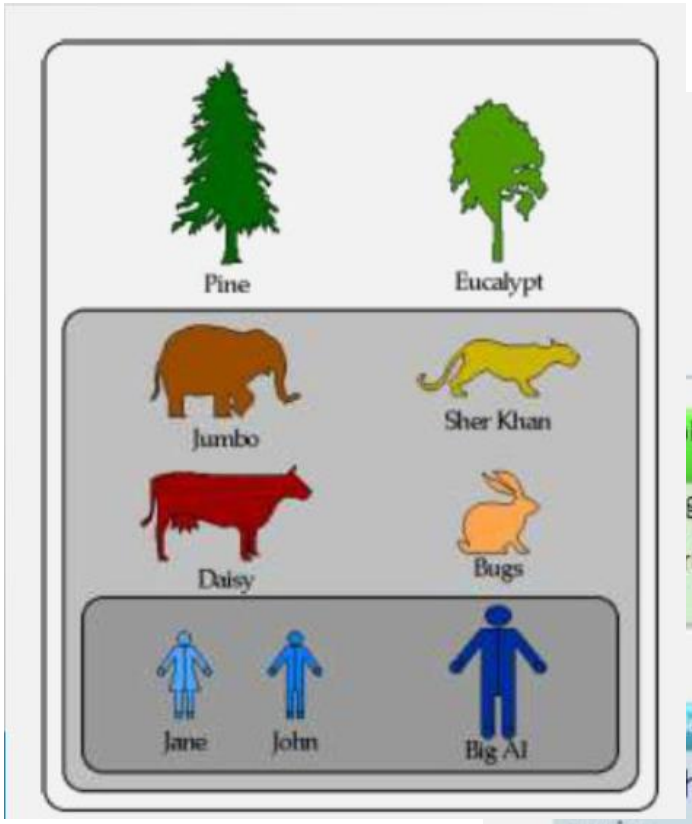
"Xây dựng các lớp mới có sẵn các đặc tính của lớp cũ, đồng thời chia sẻ hay mở rộng các đặc tính sẵn có"

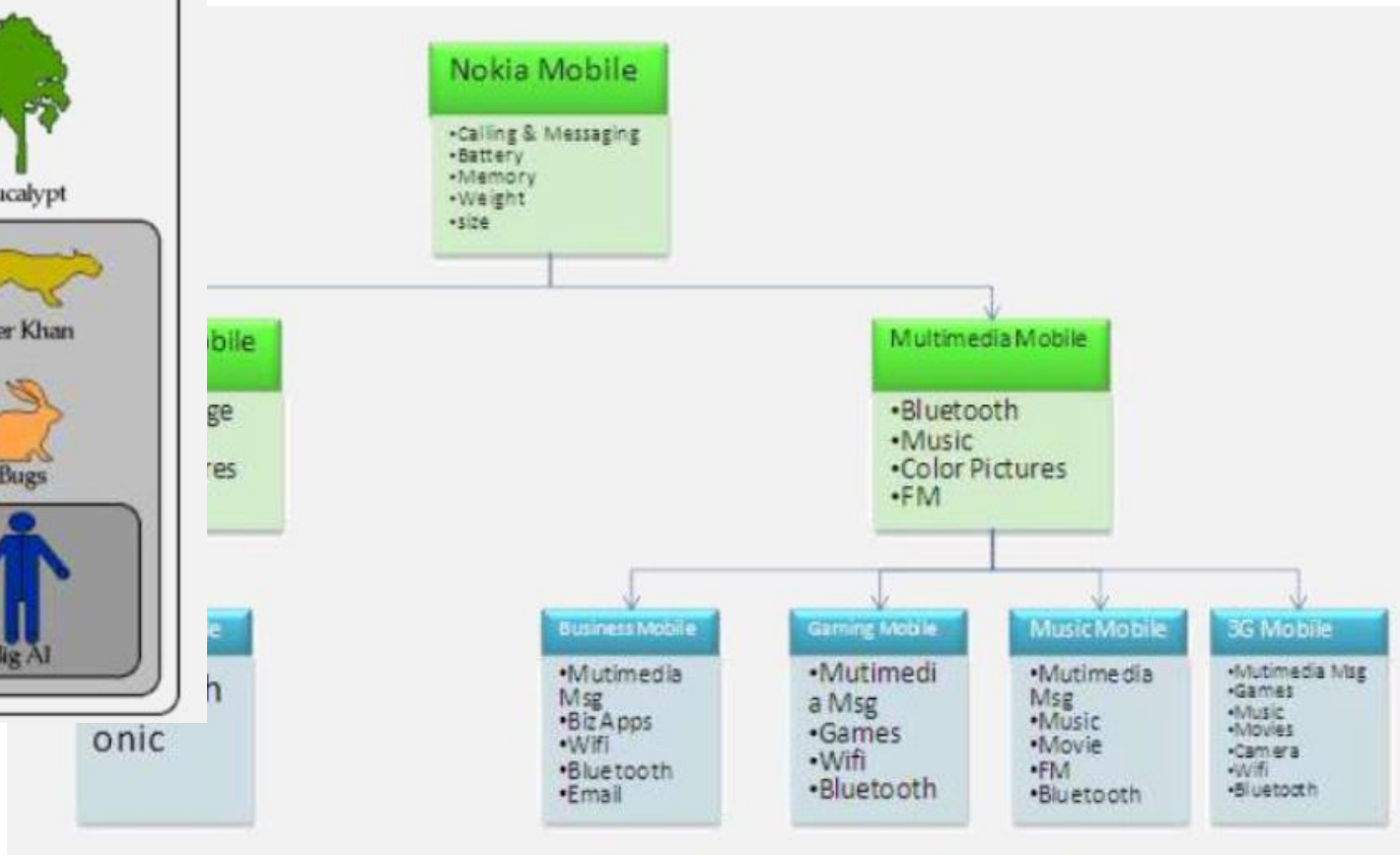
1.2 Bản chất của kế thừa

- Phát triển lớp mới dựa trên các lớp đã có
- Ví dụ
  - Lớp Người có các thuộc tính như tên, tuổi, chiều cao, cân nặng…; các phương thức như ăn, ngủ, chơi…
  - Lớp Sinh Viên thừa kế từ lớp Người, thừa kế được các thuộc tính tên, tuổi, chiều cao, cân nặng…; các phương thức ăn, ngủ, chơi…
  - Bổ sung thêm các thuộc tính như mã số sinh viên, số tín chỉ tích lũy…, các phương thức như tham dự lớp học, thi…

1.2 Bản chất của kế thừa

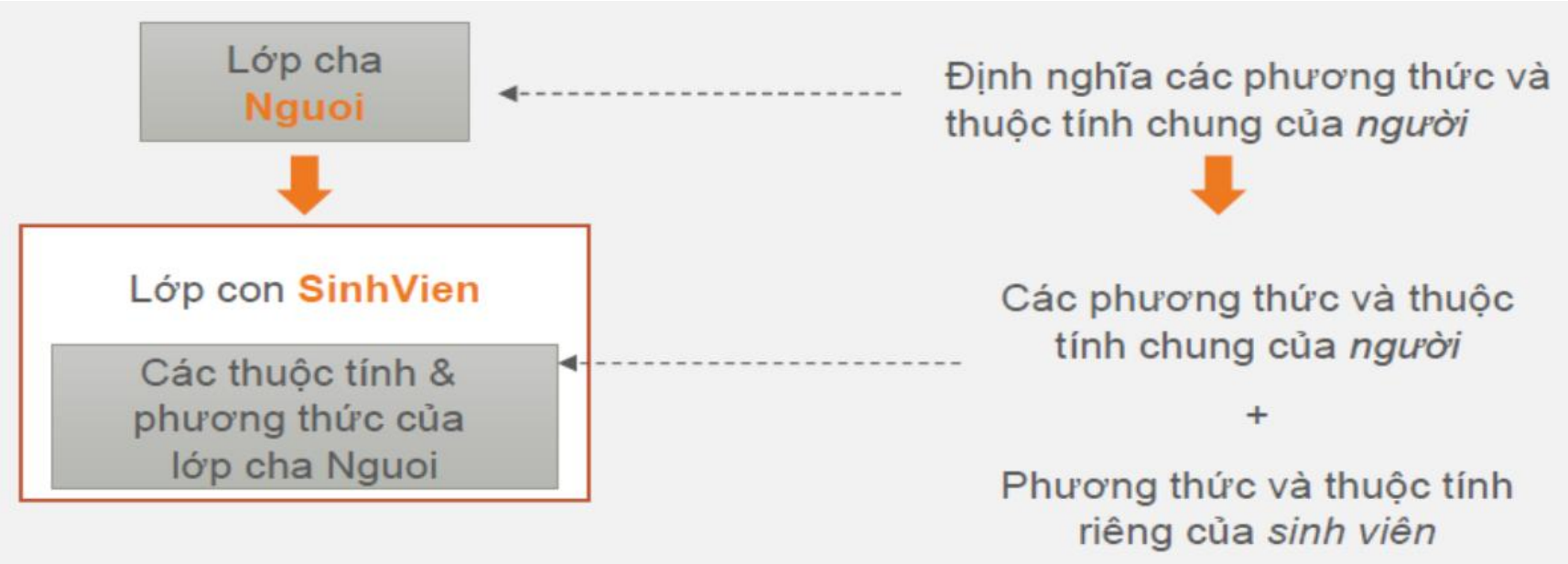
- Chính là nguyên lý phân cấp trong trừu tượng hóa





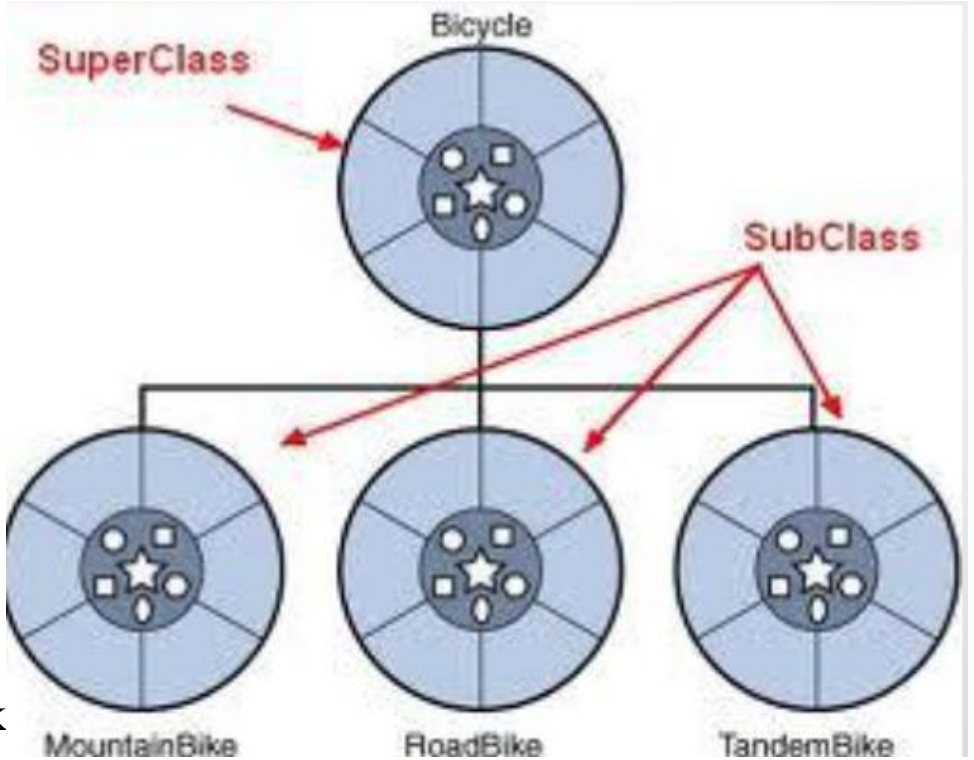
1.3 Khái niệm

- Khái niệm
  - Lớp cũ: Lớp cha (parent, superclass), lớp cơ sở (base class)
  - Lớp mới: Lớp con (child, subclass), lớp dẫn xuất (derived class)
  - Ví dụ: SinhVien thừa kế (dẫn xuất) từ lớp Nguoi

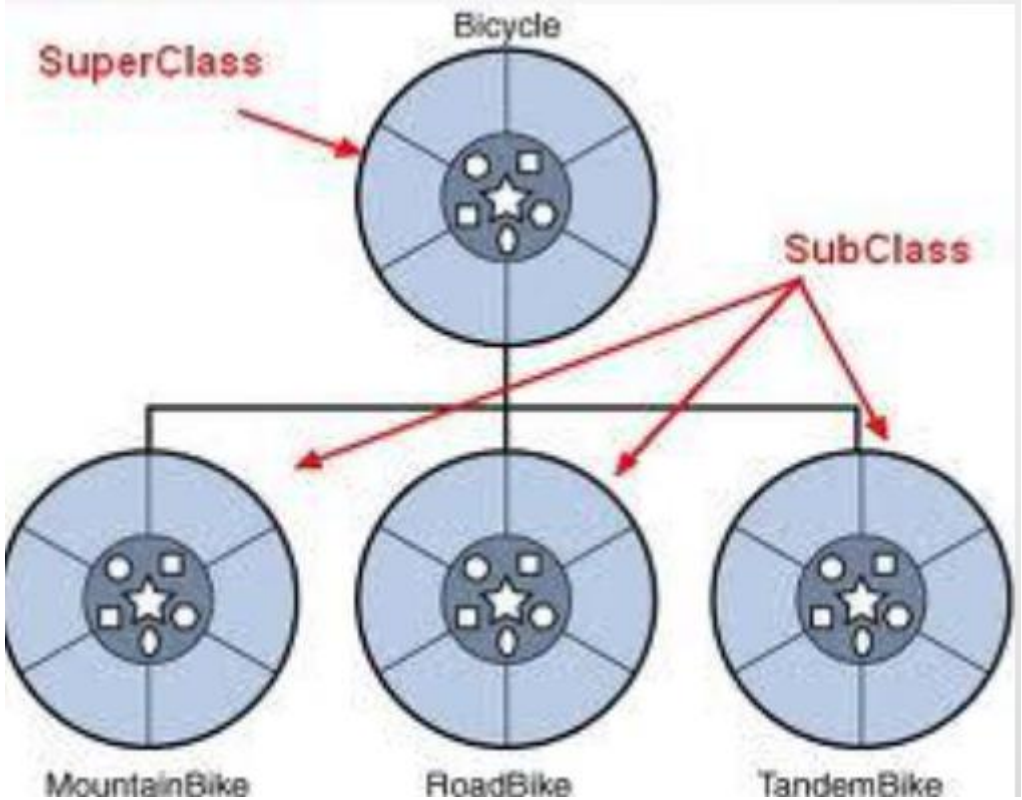


1.4 Sự phân cấp kế thừa

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



1.4 Sự phân cấp kế thừa



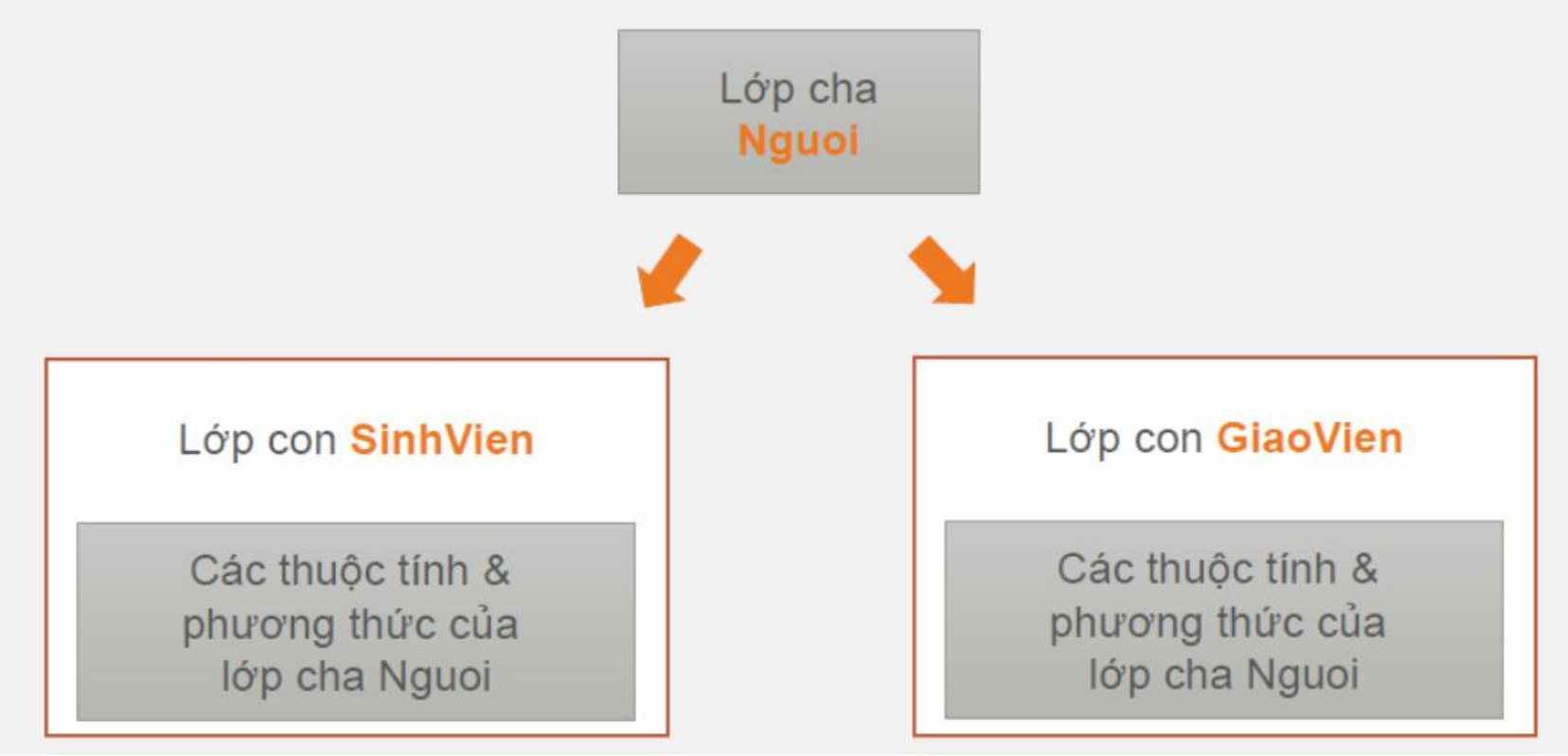
- class Bicycle{…}
- class MountainBike extends Bicycle{…}
- class RoadBike extends Bicycle{…}
- class TandemBike extends Bicycle{…}

1.5 Mối quan hệ kế thừa

- Lớp con và lớp cha có tính tương đồng
  - Lớp SinhVien **kế thừa** từ lớp Nguoi.
  - Một sinh viên **là** một người

1.5 Mối quan hệ kế thừa

- Cả GiaoVien và SinhVien đều có quan hệ là (is-a) với lớp Nguoi
- Cả giáo viên và sinh viên đều có một số hành vi thông thường của con người



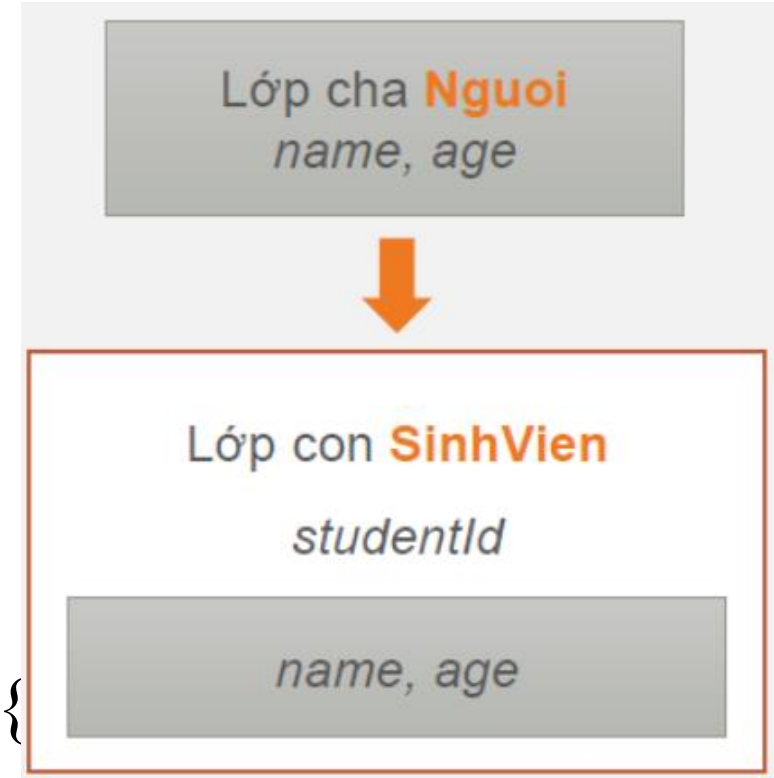
1.5 Mối quan hệ kế thừa

- Lớp con
  - Là một loại (is-a-kind-of) của lớp cha
  - Kế thừa các thành phần dữ liệu và các hành vi của lớp cha
  - Chi tiết hóa cho phù hợp với mục đích sử dụng mới
    - Extension: Thêm các thuộc tính/hành vi mới
    - Redefinition (Method Overriding): Chỉnh sửa lại các hành vi kế thừa từ lớp cha

1.6 Cú pháp

- Cú pháp (Java):
- <Lớp con> extends <Lớp cha>
- Ví dụ

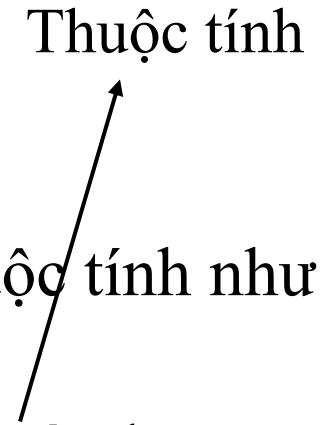
```
class Nguoi {
        String name; int age;
}
class SinhVien extends Nguoi{
        int studentId;
}
```



- Lớp con mở rộng các đặc tính của lớp cha

1.7 Bản chất của kế thừa

- Là một kỹ thuật tái sử dụng mã nguồn
  - Tái sử dụng mã nguồn thông qua **lớp**
- **Ví dụ:** Lớp SinhViên tái sử dụng được các thuộc tính như tên, tuổi… và các phương thức của lớp Người.



- Lớp con được phép sở hữu các tài sản (trường và phương thức) của lớp cha
  - Lớp con **được** phép sở hữu các tài sản public hoặc protected của lớp cha
  - Lớp con cũng **được** phép sở hữu các tài sản mặc định {default} của lớp cha nếu lớp con và lớp cha được định nghĩa cùng gói (package)
  - Lớp con **không thể** truy cập thành viên **private** của lớp cha
- Lớp con **không** kế thừa các **hàm tạo** của lớp cha

1.7 Bản chất của kế thừa

```
package poly.ho;
public class NhanVien{
    public String hoTen;
    protected double luong;
    public NhanVien(String hoTen, double luong){...}
    void xuat(){...}
    private double thueThuNhap(){...}
}
```

A. super.hoTen
B. super.luong
C. super.xuat()
D. super.thueThuNhap()

```
package poly.hcm;
public class TruongPhong extends NhanVien{
    public double trachNhiem;
    public TruongPhong (String hoTen, double luong, double trachNhiem){...}
    public void xuat(){
        // Mã ở đây có thể sử dụng những tài sản nào của lớp cha
    }
}
```

1.8 Sử dụng Super

- Truy cập đến các thành viên của lớp cha bằng cách sử dụng từ khóa super
- Có thể sử dụng super để gọi hàm tạo của lớp cha
- Gọi phương thức khởi tạo
- super(danh sách tham số);
  - Bắt buộc nếu lớp cha không có phương thức khởi tạo mặc định
  - Phải được khai báo tại dòng lệnh đầu tiên trong phương thức khởi tạo của lớp con

```
public class Parent{
   public String name;
   public void
method(){}
}
```

```
public class Child extends Parent{
   public String name;
   public void method(){
     this.name = super.name;
     super.method()
   }
}
```

1.8 Sử dụng Super

- super được dùng để thay cho superclass, tương tự như this thay cho đối tượng được gọi.
- Dùng super để:
  - gọi 1 contructor của superclass
  - gọi 1 phương thức của superclass
- Gọi superclass Constructor
  - *super(), hoặc super(tham_số)*
  - Lệnh trên phải được *đặt tại dòng đầu tiên* của subclass constructor và là cách duy nhất để gọi 1 superclass constructor.

```
public Cylinder() {
}
```

```
public Cylinder() {
    super();
}
```

1.8 Sử dụng Super

- Gọi superclass method

  super.method(tham_số)

- Vd:

```
double findVolume() {
    return super.findArea() * length;
}
```

1.8 Sử dụng Super

- Ví dụ: Không sử dụng super
- Ví dụ: Sử dụng super

```
class Vehicle {
    int speed = 50;
}

 public class Bike extends Vehicle {
    int speed = 100;

    void display() {
        System.out.println(speed); //in speed của lớp Bike
    }

    public static void main(String args[]) {
        Bike b = new Bike();
        b.display();
    }
}
```

Kết quả:

```
100
```

```
class Vehicle {
    int speed = 50;
}

public class Bike2 extends Vehicle {
    int speed = 100;

    void display() {
        System.out.println(super.speed);//in speed của lớp Vehicle
    }

    public static void main(String args[]) {
        Bike2 b = new Bike2();
        b.display();

    }
}
```

Kết quả:

```
50
```

1.8 Sử dụng Super

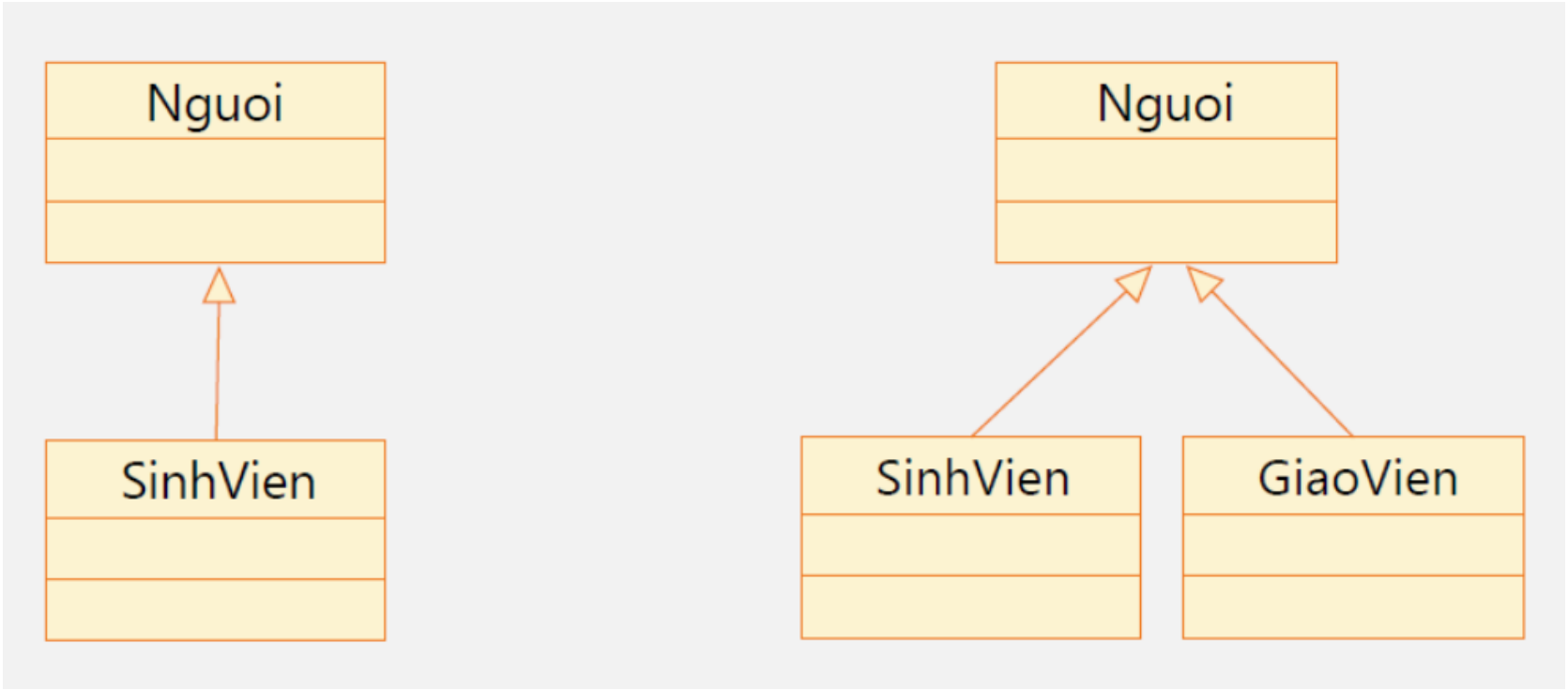
```
package poly.ho;
public class NhanVien{
    public NhanVien(String hoTen, double luong){…}
    public void xuat(){…}
}
```

```
package poly.hcm;
public class TruongPhong extends NhanVien{
    public double trachNhiem;
    public TruongPhong (String hoTen, double luong, double
trachNhiem){
        super(hoTen, luong);
        this.trachNhiem = trachNhiem
    }
    public void xuat(){
        super.xuat()
        System.out.println(trachNhiem)
    }}
```

2. Biểu diễn quan hệ kế thừa trong biểu đồ lớp

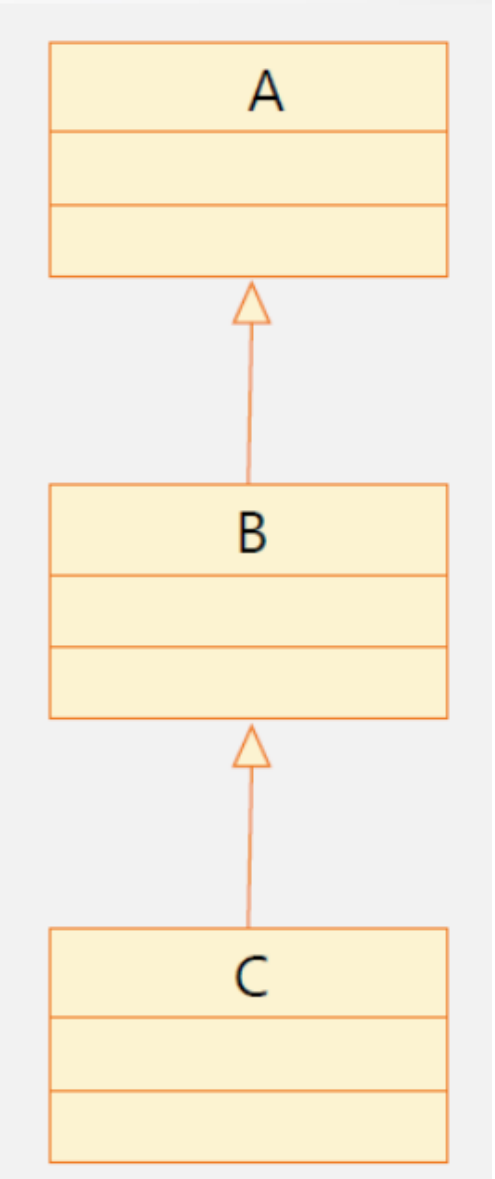
2.1 Cây phân cấp kế thừa

- Dùng mũi tên với tam giác rỗng ở đầu



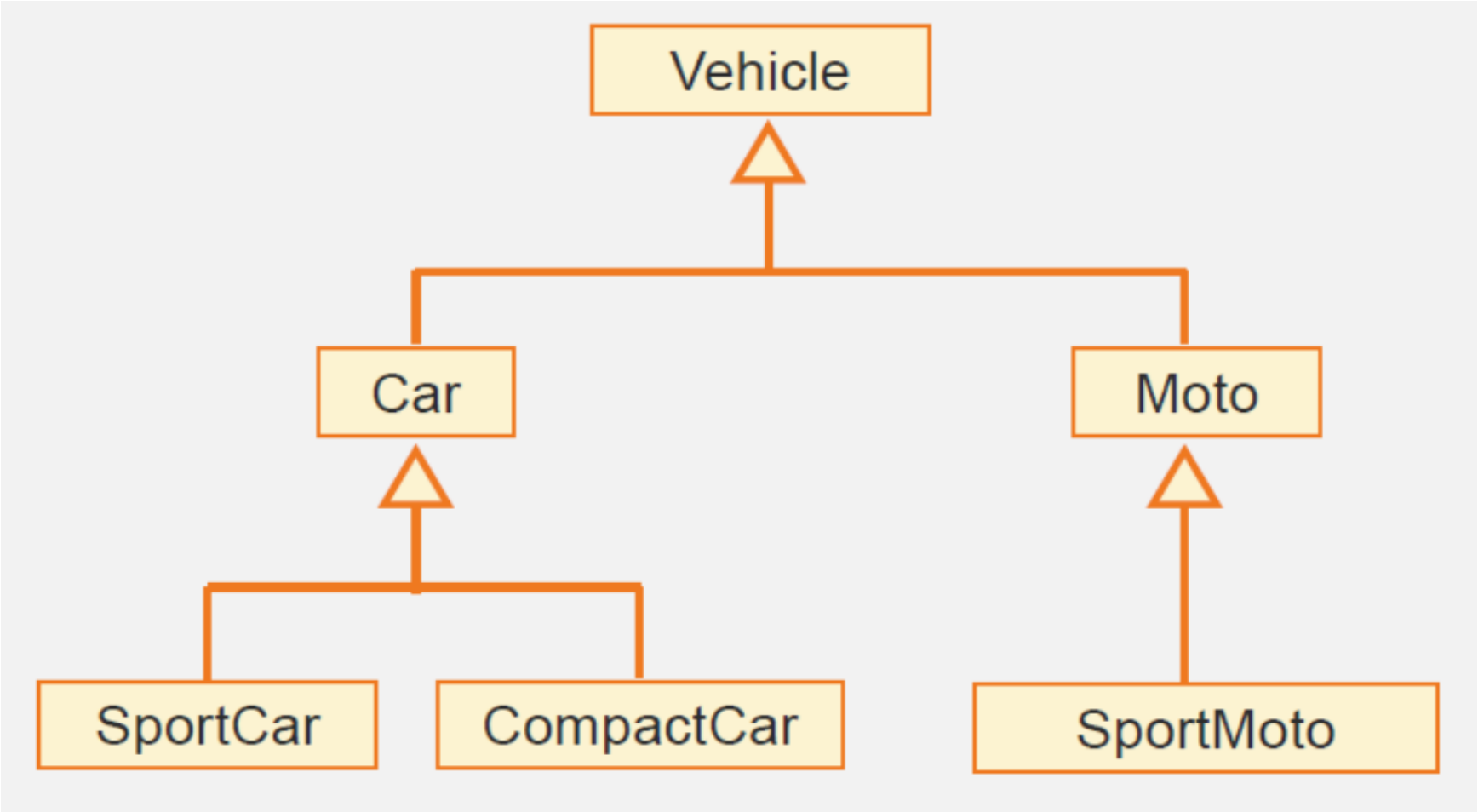
2.1 Cây phân cấp kế thừa

- Cấu trúc phân cấp hình cây, biểu diễn mối quan hệ kế thừa giữa các lớp.
  - Dẫn xuất trực tiếp
    - B dẫn xuất trực tiếp từ A
  - Dẫn xuất gián tiếp
    - C dẫn xuất gián tiếp từ A



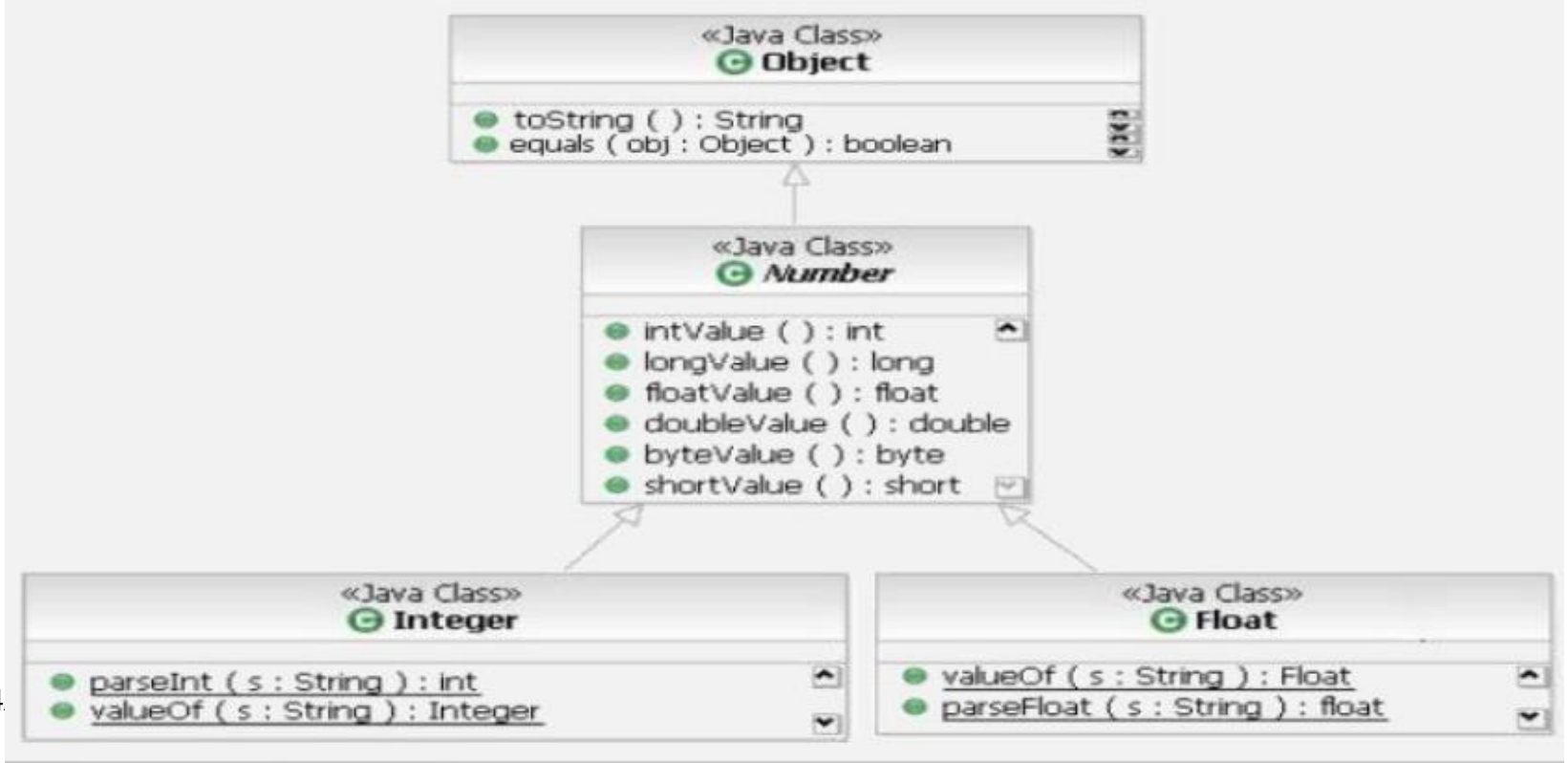
2.1 Cây phân cấp kế thừa

- Cây phân cấp kế thừa
  - Các lớp con có cùng lớp cha gọi là các lớp anh chị em (siblings)



2.2 Lớp Object

- Lớp Object là lớp gốc trên cùng của tất cả các cây phân cấp kế thừa
  - Nếu một lớp không được định nghĩa là lớp con của một lớp khác thì mặc định nó là lớp con trực tiếp của lớp Object.
- Được định nghĩa trong package chuẩn java.lang
- Chứa một số phương thức hữu ích kế thừa lại cho tất cả các lớp, ví dụ: toString(), equals()...



3. Nguyên lý kế thừa

- Lớp con có thể thừa kế 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.
- Thành viên protected trong lớp cha được truy cập trong:
  - Các thành viên lớp cha
  - Các thành viên lớp con
  - Các thành viên các lớp cùng thuộc 1 package với lớp cha

3. 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	✔	✘	✘	✘

3. Nguyên lý kế thừa

- Các phương thức không được phép kế thừa:
  - Các phương thức khởi tạo và hủy
    - Làm nhiệm vụ khởi đầu và gỡ bỏ các đối tượng
    - Chúng chỉ biết cách làm việc với từng lớp cụ thể
  - Toán tử gán =
    - Làm nhiệm vụ giống như phương thức khởi tạo

3. Nguyên lý kế thừa

```
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

3. Nguyên lý kế thừa

```
run:
(vidu_silde4_muc3.Diem@15db9742;vidu_silde4_muc3.Diem@6d06d69c) (vidu_silde4_muc3.Diem@7852e922;vidu_silde4_muc3.Diem@4e25154f)
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Nguyên lý kế thừa

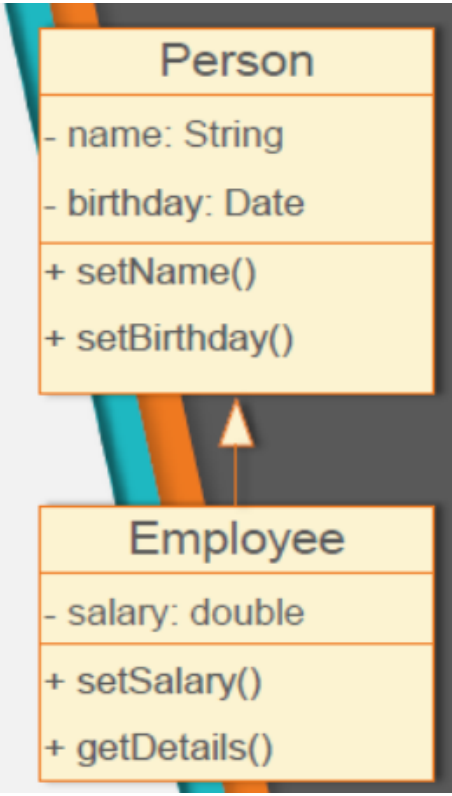
Ví dụ 2

```
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
    }
}
```

3. Nguyên lý kế thừa

Ví dụ 2 (tiếp)

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



3. Nguyên lý kế thừa

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;
    }
}
```

3. Nguyên lý kế thừa

Ví dụ 3

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

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

4. Khởi tạo và hủy bỏ đối tượng

- Khởi tạo đối tượng:
  - Lớp cha được khởi tạo trước lớp con.
  - Các phương thức khởi tạo của lớp con luôn gọi phương thức khởi tạo của lớp cha ở câu lệnh đầu tiên
    - Tự động gọi (ngầm định - implicit): Khi lớp cha **CÓ** phương thức khởi tạo mặc định (hoặc ngầm định)
    - Gọi trực tiếp (tường minh - explicit)
- Hủy bỏ đối tượng:
  - Ngược lại so với khởi tạo đối tượng
- Tái sử dụng các đoạn mã của lớp cha trong lớp con

Sử dụng từ khóa Super

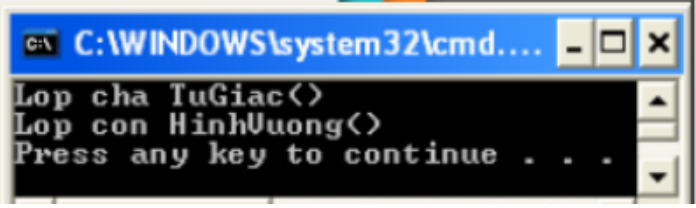
```
class Diem {
    private int x, y;
    public Diem(){}
    public Diem(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public void setX(int x){
        this.x = x;
    }
    public int getX() {
        return x;
    }
    public void printDiem(){
        System.out.print("(" + x + ", " + y + ")");
    }
}
```

4. Khởi tạo và hủy bỏ đối tượng

Ví dụ

```
public class TuGiac {
   protected Diem d1, d2;
   protected Diem d3, d4;
   public TuGiac() {
     System.out.println("Lop cha TuGiac()");
   }
}
public class HinhVuong extends TuGiac {
   public HinhVuong() {
    // Tu dong goi TuGiac()
    System.out.println("Lop con HinhVuong()");
   }
}
public class Test {
  public static void main(String arg[]) {
    HinhVuong hv = new HinhVuong();
  }
}
```

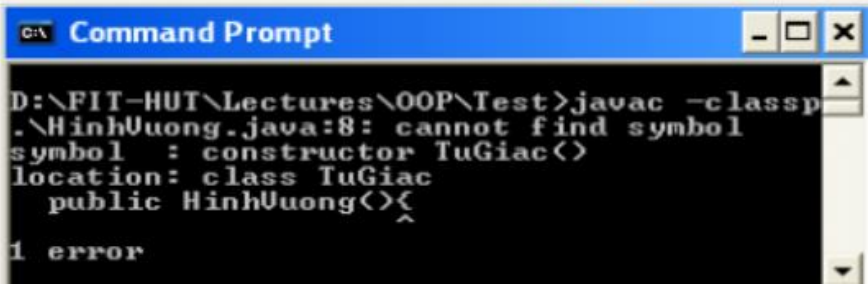
Tự động gọi (ngầm định - implicit): Khi lớp cha **CÓ** phương thức khởi tạo mặc định



4. Khởi tạo và hủy bỏ đối tượng

Ví dụ

```
public class TuGiac {
  protected Diem d1, d2;
  protected Diem d3, d4;
  public TuGiac(Diem d1, Diem d2, Diem d3, Diem d4) {
   System.out.println("Lop cha TuGiac(d1, d2, d3, d4)");
   this.d1 = d1; this.d2 = d2;
   this.d3 = d3; this.d4 = d4;
  }
}
public class HinhVuong extends TuGiac {
  public HinhVuong() {
    System.out.println("Lop con HinhVuong()");
  }
}
public class Test {
  public static void main(String arg[]) {
    HinhVuong hv = new HinhVuong();
  }
}
```

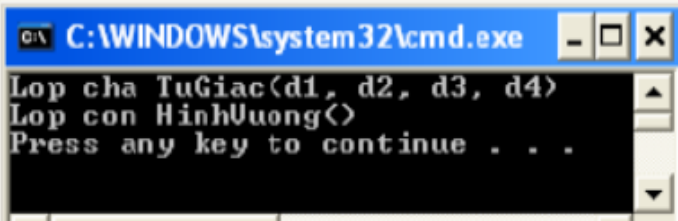


4. Khởi tạo và hủy bỏ đối tượng

Ví dụ

```
public class TuGiac {
  protected Diem d1, d2, d3, d4;
  public TuGiac(Diem d1, Diem d2, Diem d3, Diem d4) {
    System.out.println("Lop cha TuGiac(d1, d2, d3, d4)");
    this.d1 = d1; this.d2 = d2;
    this.d3 = d3; this.d4 = d4;
  }
}
public class HinhVuong extends TuGiac {
  public HinhVuong() {
    super(new Diem(0,0), new Diem(0,1),
          new Diem(1,1),new Diem(1,0));
    System.out.println("Lop con HinhVuong()");
  }
}
public class Test {
  public static void main(String arg[]) {
    HinhVuong hv = new HinhVuong();
  }
}
```

Tự động gọi (ngầm định - implicit): Khi lớp cha **CÓ** phương thức khởi tạo ngầm định

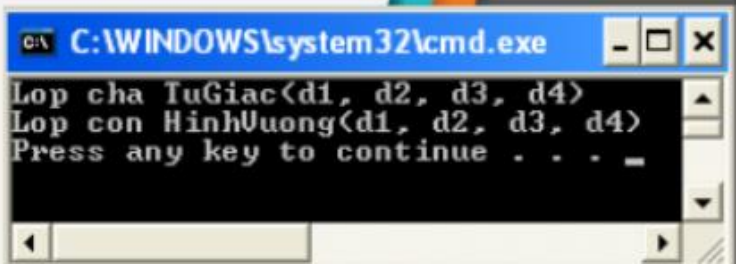


4. Khởi tạo và hủy bỏ đối tượng

Ví dụ

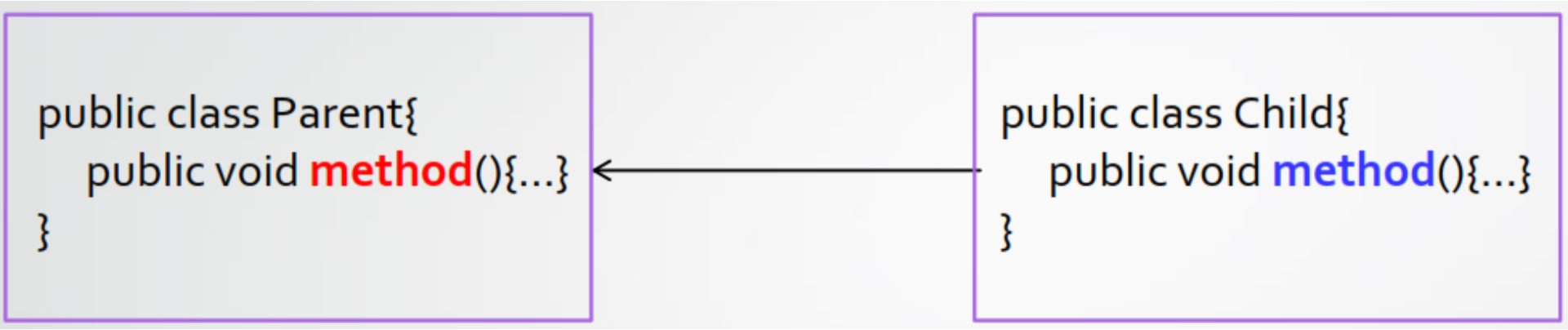
```
public class TuGiac {
  protected Diem d1, d2, d3, d4;
  public TuGiac(Diem d1, Diem d2, Diem d3, Diem d4) {
   System.out.println("Lop cha TuGiac(d1, d2, d3, d4)");
    this.d1 = d1; this.d2 = d2;
    this.d3 = d3; this.d4 = d4;
  }
}
public class HinhVuong extends TuGiac {
  public HinhVuong(Diem d1, Diem d2, Diem d3, Diem d4) {
    super(d1, d2, d3, d4);
    System.out.println("Lop con HinhVuong(d1, d2, d3, d4)");
  }
}
public class Test {
  public static void main(String arg[]) {
    HinhVuong hv = new HinhVuong(
      (new Diem(0,0), new Diem(0,1),
       new Diem(1,1),new Diem(1,0)));
  }
}
```

Tự động gọi (ngầm định - implicit): Khi lớp cha **CÓ** phương thức khởi tạo ngầm định

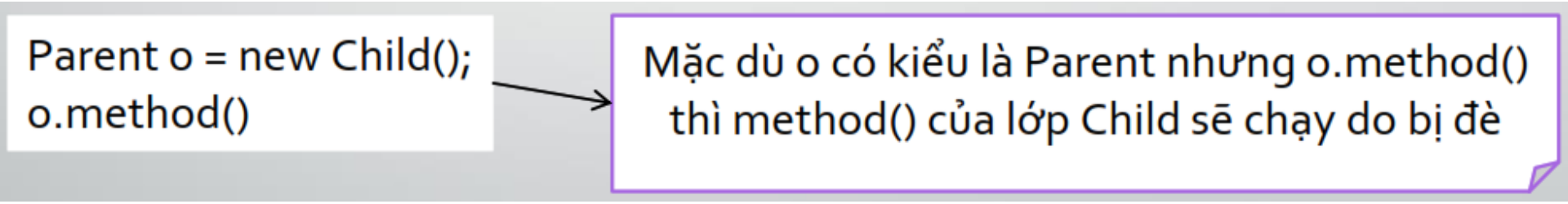


5. Ghi đè phương thức (Overriding)

- Overriding là trường hợp lớp con và lớp cha có phương thức cùng cú pháp.



- Lớp Parent và Child đều có phương thức method() cùng cú pháp nên method() trong Child sẽ đè lên method() trong Parent



5. Ghi đè phương thức (Overriding)

- **Vấn đề ghi đè.**
  - Lớp con ghi đè phương thức của lớp cha thì sẽ che dấu phương thức của lớp cha
  - Mục đích của ghi đè là để sửa lại phương thức của lớp cha trong lớp con
  - Sử dụng từ khóa **super** để truy cập đến phương thức đã bị ghi đè của lớp cha.
  - Đặc tả truy xuất của phương thức lớp con phải có độ công khai **bằng hoặc hơn** đặc tả truy xuất của phương thức lớp cha.

6. Lớp vô sinh

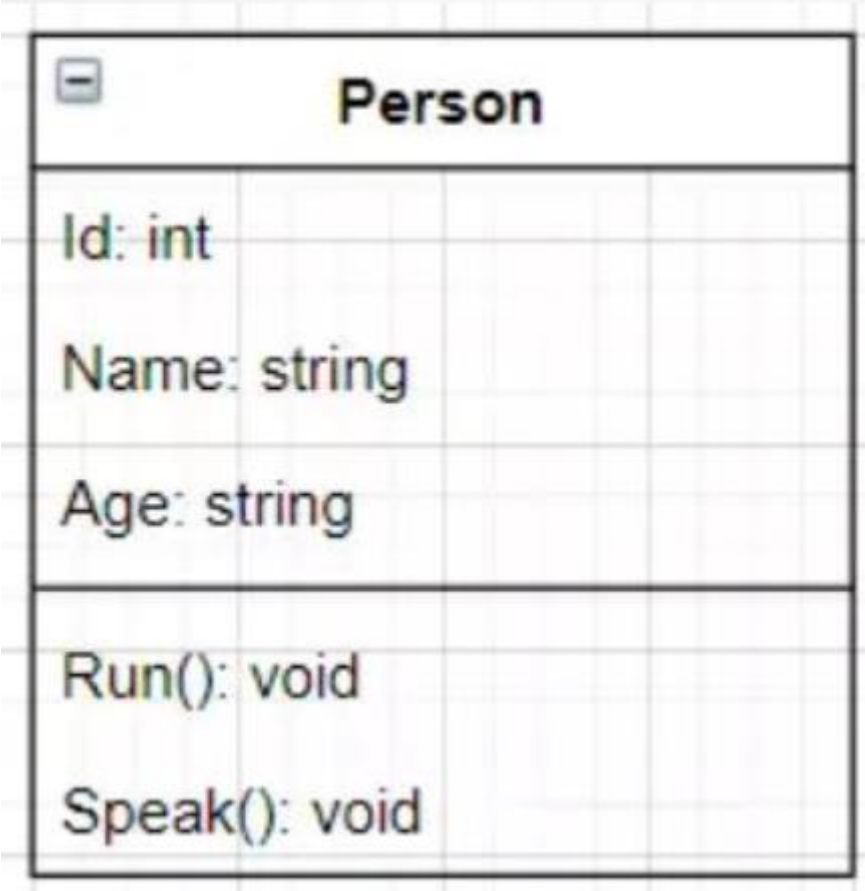
- Lớp mà ta không thể có lớp dẫn xuất từ nó (không có lớp con) gọi là lớp "vô sinh", hay nói cách khác không thể kế thừa được từ một lớp "vô sinh". Lớp "vô sinh" dùng để hạn chế, ngăn ngừa các lớp khác dẫn xuất từ nó.
- Để khai báo một lớp là lớp "vô sinh", chúng ta dùng từ khóa final class.
- Tất cả các phương thức của lớp vô sinh đều vô sinh, nhưng các thuộc tính của lớp vô sinh thì có thể không vô sinh.

Ví dụ:

```
public final class A
{
    public final int x;
    private int y;
    public final void method_1()
    {
        // …
    }
    public final void method_2()
    {
        // …
    }
}
```

Vẽ sơ đồ cây phân cấp kế thừa

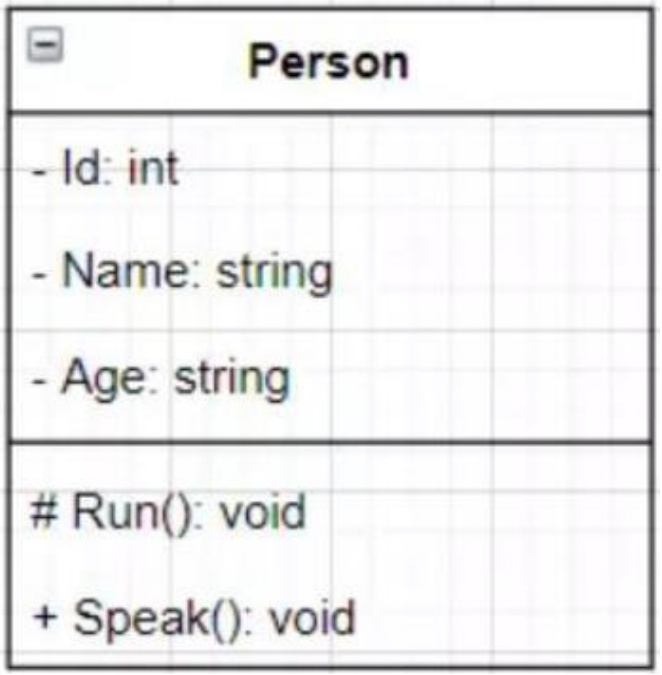
- Các tính chất cơ bản:
  - Tên class
  - Thuộc tính
  - Phương thức
- Ví dụ khai báo tên, thuộc tính, phương thức kèm theo kiểu trả về của 1 class:



Vẽ sơ đồ cây phân cấp kế thừa

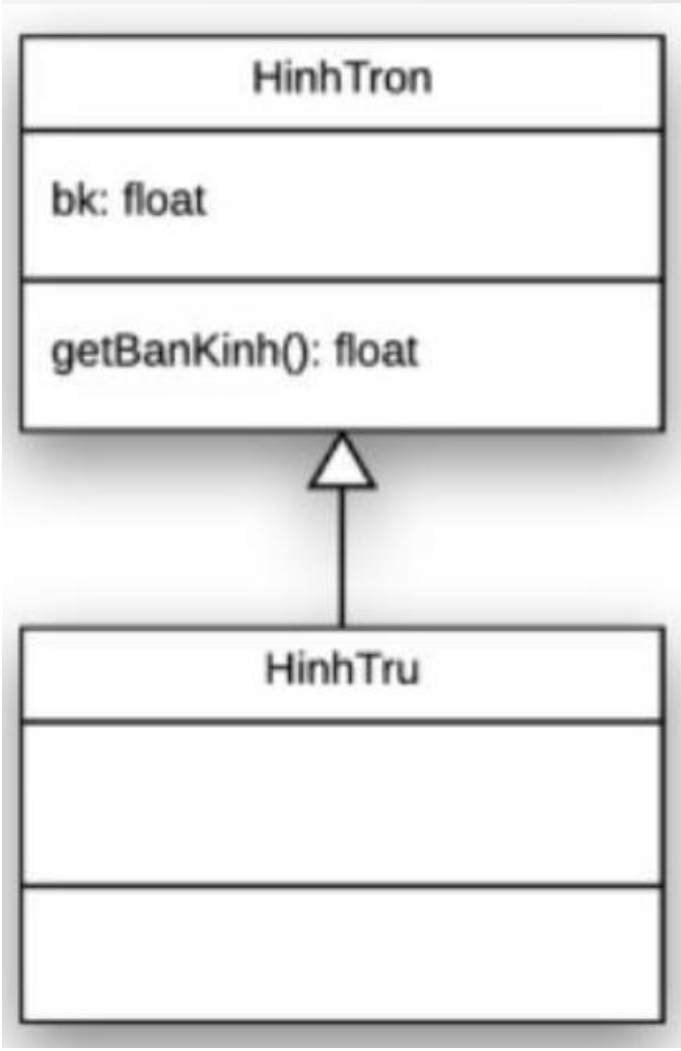
Ký hiệu tiền tố:

- Sử dụng để đặc tả phạm vi truy cập cho các Thuộc tính và Phương thức của 1 class (Cấp quyền cho các class khác sử dụng Thuộc tính và Phương thức của class này).
- 4 lựa chọn phạm vi truy cập
  - Private (–): Chỉ mình các đối tượng được tạo từ class này có thể sử dụng.
  - Public (+): Mọi đối tượng đều có thể sử dụng.
  - Protected (#): Chỉ các đối tượng được tạo từ class này và class kế thừa từ class này có thể sử dụng.
- Package/Default: Các đối tượng được tạo từ class trong lớp cùng gói có thể sử dụng.



Vẽ sơ đồ cây phân cấp kế thừa

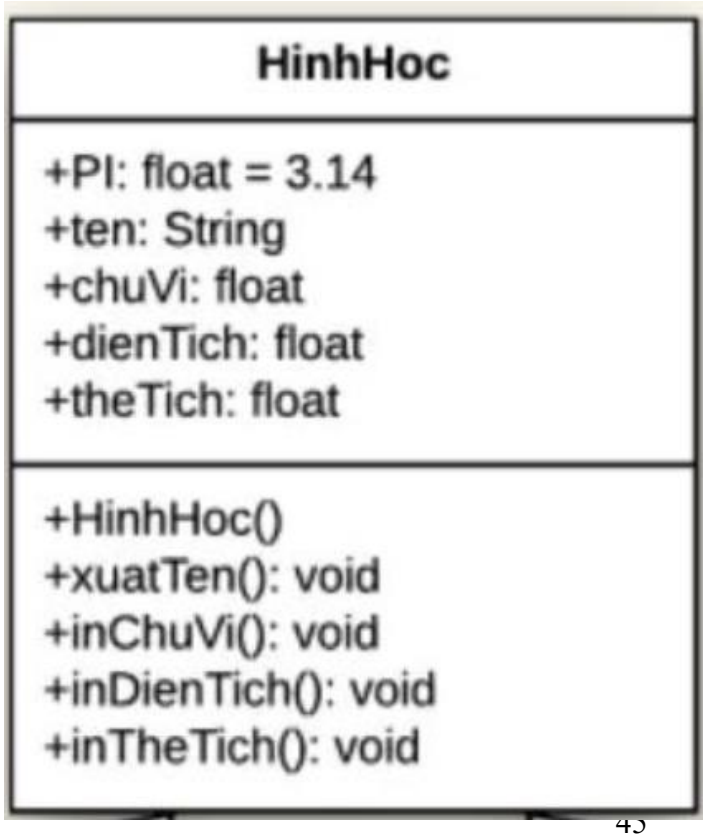
Khai báo lớp cùng các thuộc tính và phương thức



Khai báo lớp con kế thừa (chú ý tam giác rỗng)

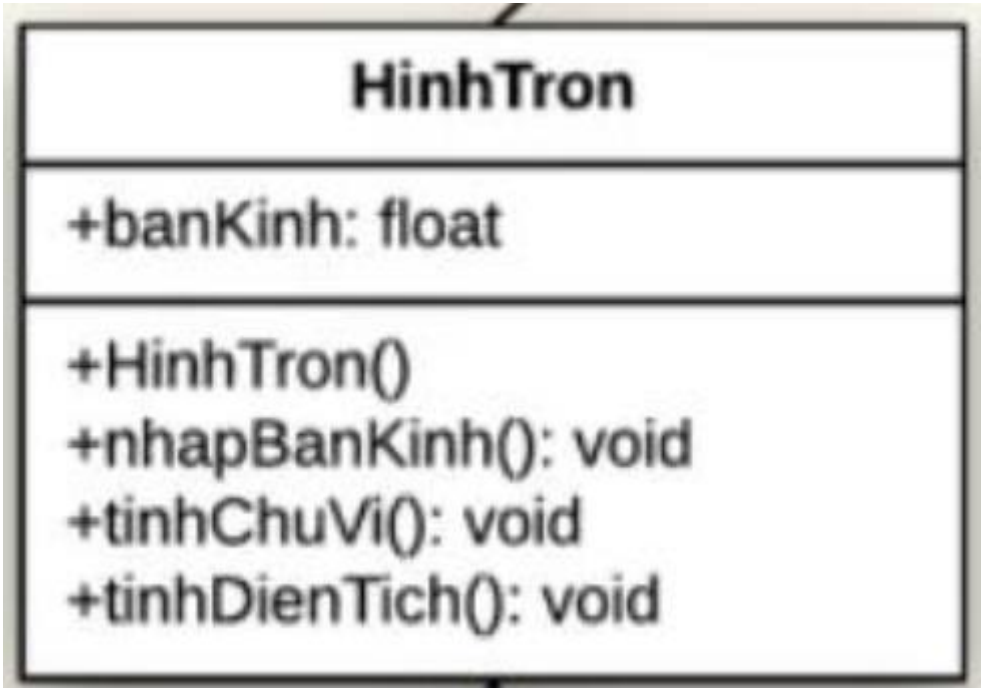
Bài tập 1

- Xây dựng lớp HìnhHoc có các
  - thuộc tính: double - PI = 3.14, chuVi, dienTich, theTich; String – ten
  - phương thức: HinhHoc(), xuatTen, inChuVi(), inDienTich(), inTheTich()
- Tất cả đều dùng tiền tố public



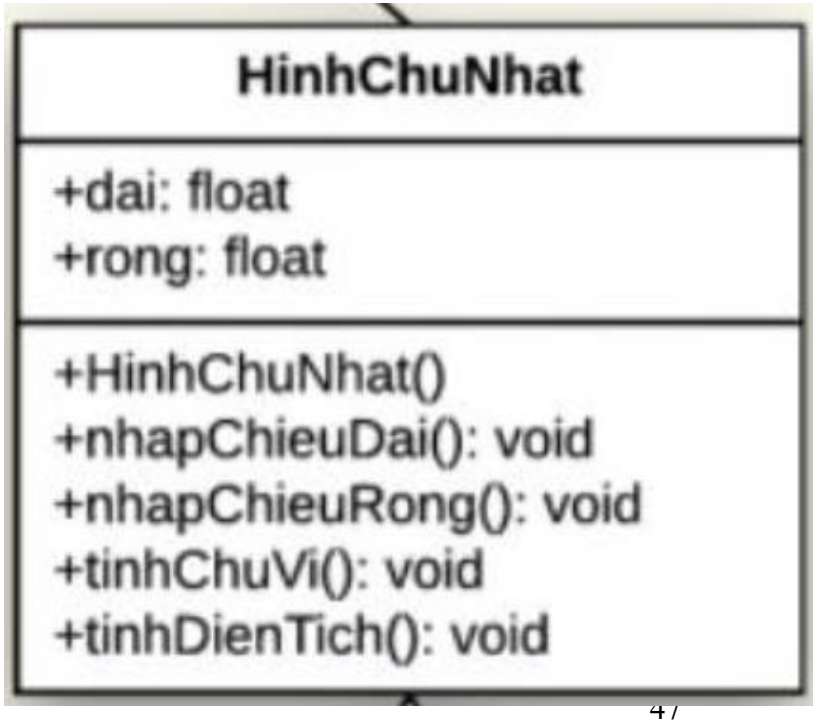
Bài tập

- Xây dựng lớp con HìnhTron kế thừa lớp HinhHoc có các
  - thuộc tính: double - banKinh
  - phương thức: HinhTron(), nhapBanKinh(), tinhChuVi(), tinhDienTich()
- Tất cả đều dùng tiền tố public



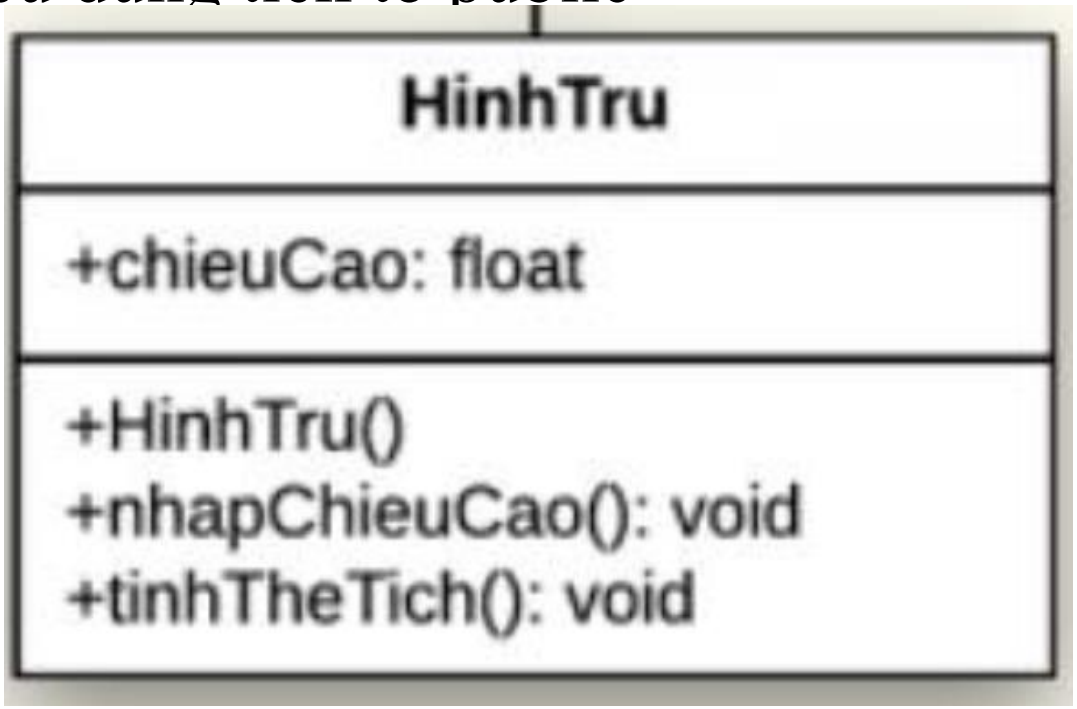
Bài tập

- Xây dựng lớp con HìnhChuNhat kế thừa lớp HinhHoc có các
  - thuộc tính: double – dai, rong
  - phương thức: HinhChuNhat(), nhapChieuDai(), nhapChieuRong(), tinhChuVi(), tinhDienTich()
- Tất cả đều dùng tiền tố public



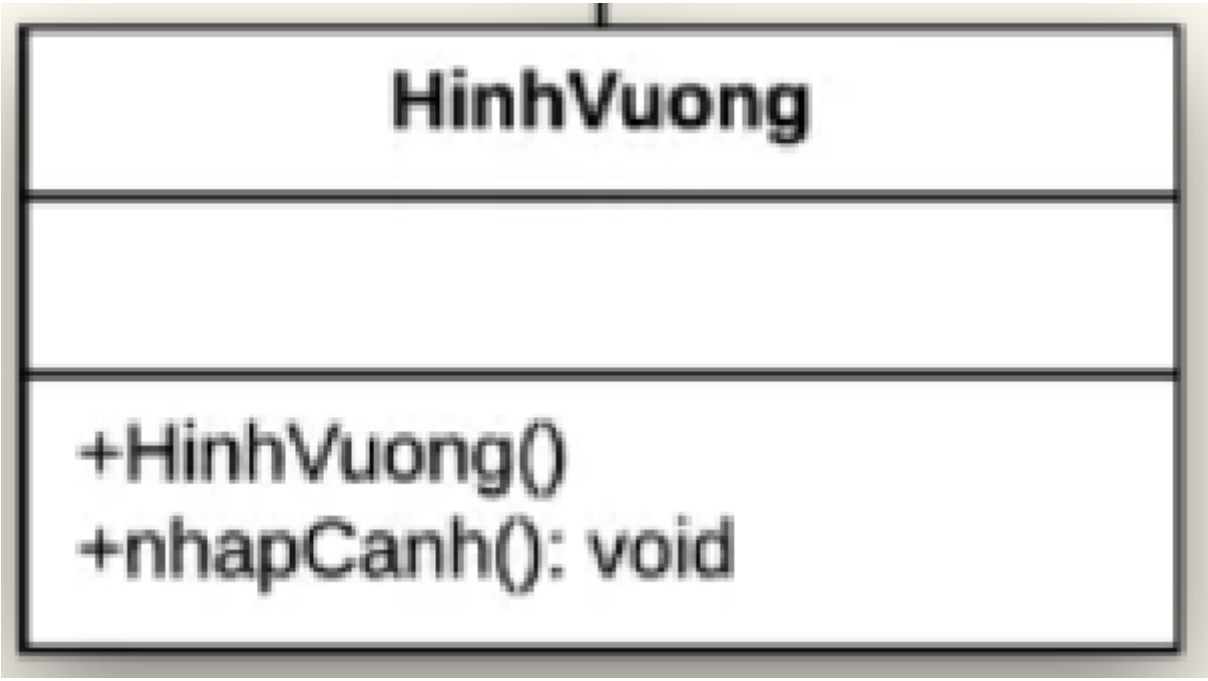
Bài tập

- Xây dựng lớp con HìnhTru kế thừa lớp HinhTron có các
  - thuộc tính: double – chieuCao
  - phương thức: HinhTru(), nhapChieuCao(), tinhTheTich()
- Tất cả đều dùng tiền tố public

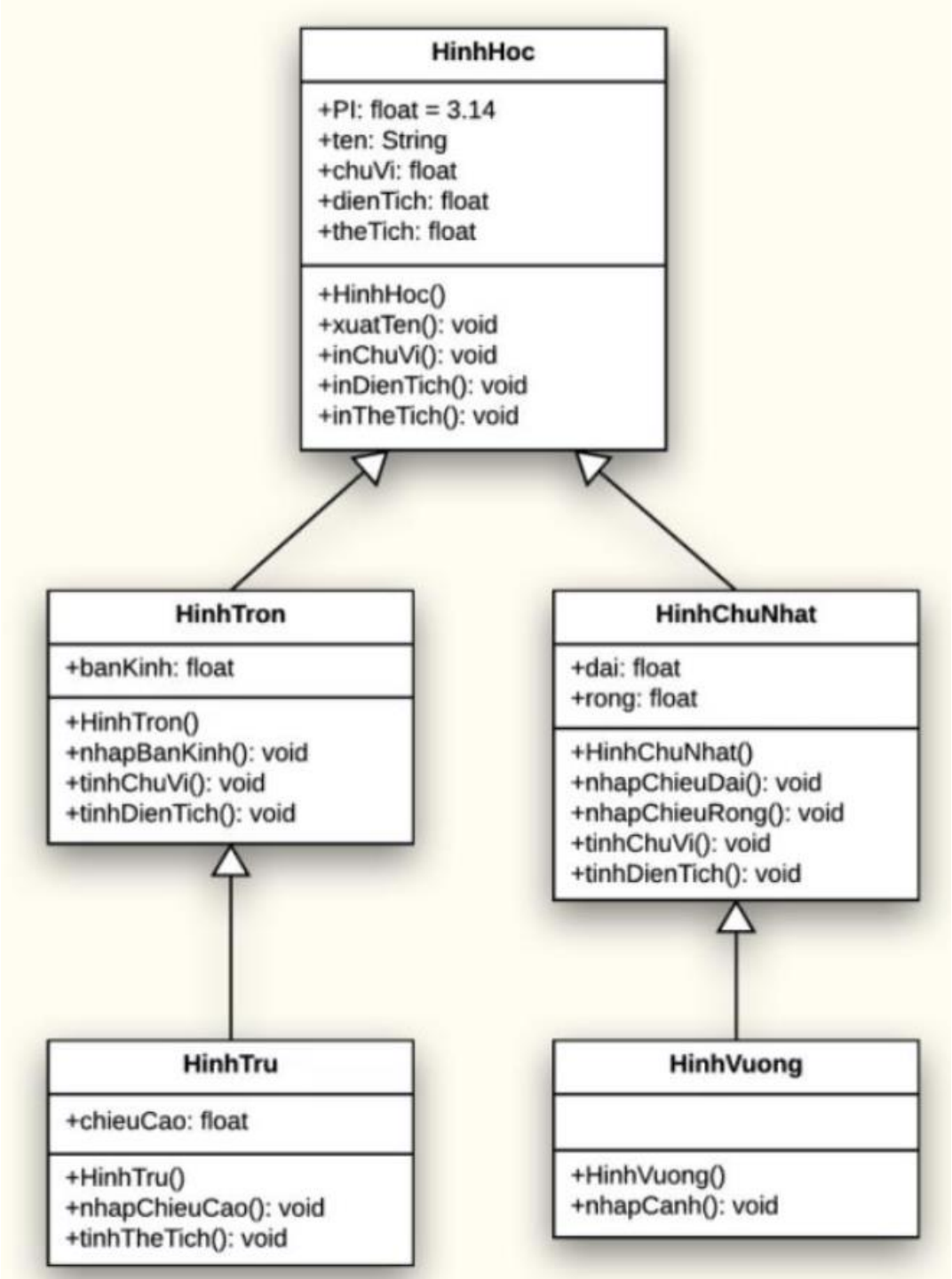


Bài tập

- Xây dựng lớp con HìnhVuong kế thừa lớp HinhChuNhat có các
  - phương thức: HinhVuong(), nhapCanh()
- Tất cả đều dùng tiền tố public



Bài tập


HinhHoc
+PI: float = 3.14
+ten: String
+chuVi: float
+dienTich: float
+theTich: float
+HinhHoc()
+xuatTen(): void
+inChuVi(): void
+inDienTich(): void
+inTheTich(): void
HinhTron
+banKinh: float
+HinhTron()
+nhapBanKinh(): void
+tinhChuVi(): void
+tinhDienTich(): void
HinhChuNhat
+dai: float
+rong: float
+HinhChuNhat()
+nhapChieuDai(): void
+nhapChieuRong(): void
+tinhChuVi(): void
+tinhDienTich(): void
HinhTru
+chieuCao: float
+HinhTru()
+nhapChieuCao(): void
+tinhTheTich(): void
HinhVuong
+HinhVuong()
+nhapCanh(): void

Bài tập 2

- Thiết kế chương trình quản lý các đối tượng sau trong một Viện khoa học: nhà khoa học, nhà quản lý và nhân viên phòng thí nghiệm. Một nhà khoa học cũng có thể làm công tác quản lý. Các thành phần dữ liệu của các đối tượng trên:
  - Nhà khoa học: họ tên, năm sinh, số bài báo đã công bố, số ngày công trong tháng, bậc lương
  - Nhà quản lý: họ tên, năm sinh, số ngày công trong tháng, bậc lương
  - Nhân viên phòng thí nghiệm: họ tên, năm sinh, lương trong tháng.
- Thực hiện các yêu cầu sau:
  - 1. Vẽ sơ đồ cây phân cấp kế thừa
  - 2. Cài đặt các phương thức thiết lập để nhập liệu, biết rằng nhân viên phòng thí nghiệm lãnh lương khoán, còn lương của nhà khoa học và nhà quản lý bằng số ngày công trong tháng * bậc lương * Lương cơ bản (giả sử = 400,000).
  - 3. Xuất dữ liệu ra màn hình
  - 4. In tổng lương đã chi trả cho từng loại đối tượng


Q&A