Lập trình Java

BÀI 3

Hướng đối tượng

Các khái niệm cơ bản

❖ **Đối tượng (object)**: trong thế giới thực khái niệm đối tượng có thể xem như một thực thể: người, vật, bảng dữ liệu,…

- ✓ Đối tượng giúp hiểu rõ thế giới thực
- ✓ Cơ sở cho việc cài đặt trên máy tính
- ✓ Mỗi đối tượng có định danh, thuộc tính, hành vi

Ví dụ: đối tượng sinh viên

MSSV: "TH0701001"; Tên sinh viên: "Nguyễn Văn A"

❖ **Hệ thống các đối tượng**: là 1 tập hợp các đối tượng

- ✓ Mỗi đối tượng đảm trách 1 công việc
- ✓ Các đối tượng có thể quan hệ với nhau
- ✓ Các đối tượng có thể trao đổi thông tin với nhau
- ✓ Các đối tượng có thể xử lý song song, hay phân tán

Các khái niệm cơ bản

❖ **Lớp (class)**: là khuôn mẫu (template) để sinh ra đối tượng. Lớp là sự trừu tượng hóa của tập các đối tượng có các thuộc tính, hành vi tương tự nhau, và được gom chung lại thành 1 lớp.

Ví dụ: lớp các đối tượng ***Sinhviên***

- ✓ Sinh viên "Nguyễn Văn A", mã số TH0701001 → 1 đối tượng thuộc lớp ***Sinhviên***
- ✓ Sinh viên "Nguyễn Văn B", mã số TH0701002 → là 1 đối tượng thuộc lớp ***Sinhviên***

❖ **Đối tượng (object) của lớp**: một đối tượng cụ thể thuộc 1 lớp là 1 thể hiện cụ thể của 1 lớp đó.

Lớp và đối tượng trong java

❖ **Khai báo lớp**

```
class <ClassName>
{
        <danh sách thuộc tính>
        <Phương thức khởi tạo>
        <danh sách các phươngthức>
}
```


Ths. Mai Xuân Hùng

Thuộc tính của đối tượng

❖ các đặc điểm mang giá trị của đối tượng, là vùng dữ liệu được khai báo bên trong lớp

```
class <ClassName>      {
        <Tiền tố> <kiểu dữ liệu> <tên thuộc tính>;
}
```

❖ Phạm vi truy xuất đối với thuộc tính

* **public**: có thể truy xuất từ bất kỳ 1 lớp khác.
* **protected**: có thể truy xuất được từ những lớp con.
* **private**: không thể truy xuất từ 1 lớp khác.
* **static:** dùng chung cho mọi thể hiện của lớp.
* **final:** hằng
* **default:** (không phải từ khóa) có thể truy cập từ các class trong cùng gói

Phương thức của đối tượng

- Chức năng xử lý, hành vi của các đối tượng.

```
class <ClassName>     {
        …
        <Tiền tố> <kiểu trả về> <tên phương thức>(<các đối số>){
                …
        }
}
```

Phạm vi truy xuất của phương thức

*__public__: có thể truy cập được từ bên ngoài lớp khai báo.

*__protected__: có thể truy cập được từ lớp khai báo và các lớp dẫn xuất (lớp con).

*__private__: chỉ được truy cập bên trong lớp khai báo.

*__static__: phương thức lớp dùng chung cho tất cả các thể hiện của lớp, có thể được thực hiện kể cả khi không có đối tượng của lớp

*__final__: không được khai báo chồng ở các lớp dẫn xuất.

*__abstract__: không có phần source code, sẽ được cài đặt trong các lớp dẫn xuất.

*__synchoronized__: dùng để ngăn những tác động của các đối tượng khác lên đối tượng đang xét trong khi đang đồng bộ hóa. Dùng trong lập trình miltithreads.

Ví dụ khai báo lớp

Ví dụ 1: ***class*** *Sinhvien {*

```
// Danh sách thuộc tính
String    maSv, tenSv, dcLienlac;
int       tuoi;
…
// Danh sách các khởi tạo
Sinhvien(){}
Sinhvien (…)       { …}
…
// Danh sách các phương thức
public void capnhatSV (…) {…}
public void xemThongTinSV() {…}
…
```

}

Ví dụ khai báo lớp (tt)

```
…
// Tạo đối tượng mới thuộc lớp Sinhvien
Sinhvien sv = new Sinhvien();
…
// Gán giá trị cho thuộc tính của đối tượng
sv.maSv = “TH0601001”;
sv.tenSv = “Nguyen Van A”;
sv.tuoi = “20”;
sv.dcLienlac = “KP6, Linh Trung, Thu Duc”;
…
// Gọi thực hiện phương thức
sv.xemThongTinSV();
```

Ví dụ khai báo lớp (tt)

Ví dụ 2:

```
class Sinhvien {
    // Danh sách thuộc tính
    private String   maSv;
    String   tenSv, dcLienlac;
    int      tuoi;
    …
}
…
Sinhvien sv = new Sinhvien();
sv.maSv = "TH0601001"; /* Lỗi truy cập thuộc tính private từ
                          bên ngoài lớp khai báo */
Sv.tenSv = "Nguyen Van A";
…
```

Lớp và đối tượng trong java

❖ **Khởi tạo (constructor):** là một loại phương thức đặc biệt của lớp, dùng để khởi tạo một đối tượng.

- ✓ Dùng để khởi tạo giá trị cho các thuộc tính của đối tượng.
- ✓ Cùng tên với lớp.
- ✓ Không có giá trị trả về.
- ✓ Tự động thi hành khi tạo ra đối tượng (new)
- ✓ Có thể có tham số hoặc không.

❖ **Lưu ý:** Mỗi lớp sẽ có 1 constructor mặc định (nếu ta không khai báo constructor nào). Ngược lại nếu ta có khai báo 1 constructor khác thì constructor mặc định chỉ dùng được khi khai báo tường minh.

Khai báo Constructor

- ***Ví dụ 1***

```
class Sinhvien
{
        ...
        // Không có định nghĩa constructor nào
}
...
// Dùng constructor mặc định
Sinhvien sv = new Sinhvien();
```

Khai báo Constructor (tt)

Ví dụ 2:

```
class Sinhvien
{
    …
    // không có constructor mặc định
    Sinhvien(<các đối số>) {…}

}

…
Sinhvien sv = new Sinhvien();
// lỗi biên dịch
```

```
class Sinhvien
{
    …
    // khai báo constructor mặc định
    Sinhvien(){}
   // constructor có đối số
      Sinhvien(<các đối số>) {…}
}

…
Sinhvien sv = new Sinhvien();
```

Ví dụ phương thức khởi tạo

```
package constructor;
class SinhVien {
private String Ten;
    public void In()
    {
            System.out.println("Ten:"+Ten);
    }
}
```

```
package constructor;
public class Constructor {
public static void main(String[] args) {
        SinhVien s= new SinhVien();
        s.In();
    }
}
```

Ten:null

Kết quả?

```
class SinhVien {
    private String Ten;
    public SinhVien()
    {
        Ten="Nguyen Van Tung";
    }
    public void In()
    {
        System.out.println("Ten:"+Ten);
    }
}
```

Ten:Nguyen Van Tung

Ví dụ phương thức khởi tạo

```
package constructor;
class SinhVien {
    private String MSSV;
    private String Ten;
public SinhVien()
    {
        Ten="Nguyen Van Tung";
    }
 public SinhVien(String str)
    {
        Ten=str;
    }
```

```
public void In()
    {
      System.out.println("Ten:"+Ten);
    }
}
```

```
package constructor;
public class Constructor {
public static void main(String[] args) {
            SinhVien s= new SinhVien(“Anh Thu”);
        s.In();
    }
}
```

Ten: Anh Thu

Lớp và đối tượng trong java

❖ **Overloading method:** Việc khai báo trong một lớp nhiều phương thức có cùng tên nhưng khác tham số (khác kiểu dữ liệu, khác số lượng tham số) gọi là khai báo chồng phương thức.

Ví dụ:

```
class Sinhvien {
        ...
        public void xemThongTinSV() {
                ...
        }
        public void xemThongTinSV(String psMaSv) {
                ...
        }
}
```

Lớp và đối tượng trong java

❖**Tham chiếu *this:*** là một biến ẩn tồn tại trong tất cả các lớp, ***this*** được sử dụng trong khi chạy và tham khảo đến bản thân lớp chứa nó.

Ví dụ:

```
class Sinhvien    {
        String    maSv, tenSv, dcLienlac;
        int       tuoi;
        …
        public void xemThongTinSV(
        {
                System.out.println(this.maSv);
                System.out.println(this.tenSv);
                        …
        }
}
```

Tính đóng gói

❖ **Đóng gói:** nhóm những gì có liên quan với nhau vào thành một và có thể sử dụng một cái tên để gọi.

Ví dụ:

✓ Các phương thức đóng gói các câu lệnh.

✓ Đối tượng đóng gói dữ liệu và các hành vi/phương thức liên quan.

(Đối tượng = Dữ liệu + Hành vi/Phương thức)

Tính đóng gói

❖ **Đóng gói:** dùng để che dấu một phần hoặc tất cả thông tin, chi tiết cài đặt bên trong với bên ngoài.

- Ví dụ: khai báo các lớp thuộc cùng gói trong java

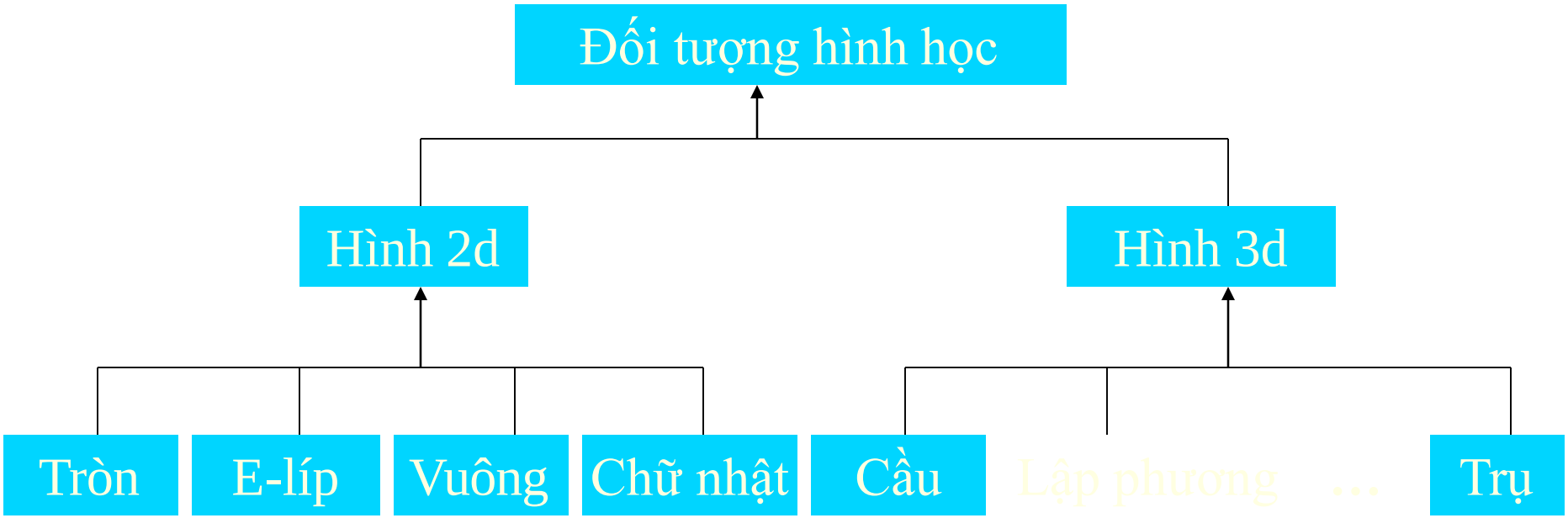
package *<tên gói>; // khai báo trước khi khai báo lớp*

class *<tên lớp> {*

...

}

Tính kế thừa



- Thừa hưởng các thuộc tính và phương thức đã có
- Bổ sung, chi tiết hóa cho phù hợp với mục đích sử dụng mới
  - ✓ **Thuộc tính**: thêm mới
  - ✓ **Phương thức**: thêm mới hay hiệu chỉnh

Tính kế thừa

- ✓ **Lớp dẫn xuất hay lớp con (SubClass)**
- ✓ **Lớp cơ sở hay lớp cha (SuperClass)**
- ✓ Lớp con có thể kế thừa tất cả hay một phần các thành phần dữ liệu (thuộc tính), phương thức của lớp cha (public, protected, default)
- ✓ Dùng từ khóa

extends. ***Ví dụ:***

```
class nguoi { …
}
class sinhvien extends nguoi { …
}
```

Lưu ý: default không phải là 1 từ khóa

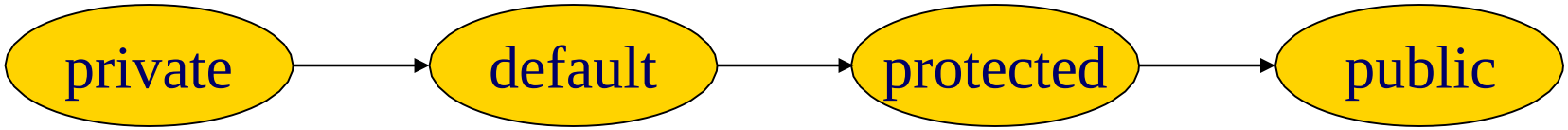
Tính kế thừa

❖ **Overriding Method**

- Được định nghĩa trong lớp con

- Có tên, kiểu trả về & các đối số giống với phương thức của lớp cha

- Có kiểu, phạm vi truy cập không "nhỏ hơn" phương thức trong lớp cha



Tính kế thừa

- ***Ví dụ:***

```
class Hinhhoc { …
        public float tinhdientich() {
                return 0;
        }
        …
}

class HinhVuong extends Hinhhoc {
        private int canh;
        public float tinhdientich() {
                return canh*canh;
        }
        …
}
```

Chỉ có thể **public** *do phương thức tinhdientich() của lớp cha là* **public**

Tính kế thừa

```
class HinhChuNhat extends HinhVuong {
        private int cd;
        private int cr;
        public float tinhdientich() {
                return cd*cr;
        }
        …
}
```

Chỉ có thể ***public*** *do phương thức tinhdientich() của lớp cha là* ***public***

Từ khóa super

- Gọi constructor của lớp cha
- Nếu gọi tường minh thì phải là câu lệnh đầu tiên

Sự thừa kế trong hàm khởi tạo
Constructor Inheritance

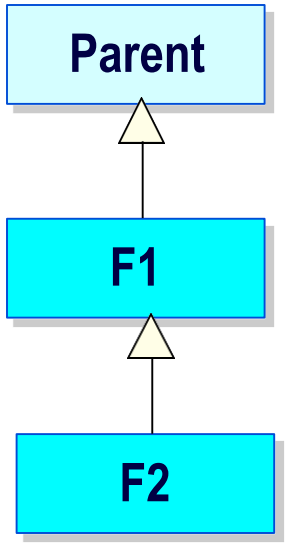
- Khai báo về thừa kế trong hàm khởi tạo
- Chuỗi các hàm khởi tạo (Constructor Chaining)
- Các nguyên tắc của hàm khởi tạo (Rules)
  - Triệu hồi tường minh hàm khởi tạo của lớp cha

Sự thừa kế trong hàm khởi tạo

- Khi tạo một thể hiện của lớp dẫn xuất , trước hết phải gọi đến hàm khởi tạo của lớp cha, tiếp đó mới là hàm khởi tạo của lớp con.
- Có thể triệu hồi hàm dựng của lớp cha bằng cách sử dụng từ khóa super trong phần khai báo hàm dựng của lớp con.

Chuỗi hàm dựng - Constructor Chaining

```
class Parent
{  public Parent(){
        System.out.println("This is constructor of Parent class");
    }
}
class F1 extends Parent
{    public F1() {
        System.out.println("This is constructor of F1 class");
    }
}
class F2 extends F1
{   public F2(){
        System.out.println("This is constructor of F2 class");
    }
}
```



Chuỗi khởi tạo

```
public static void main(String[] args) {
        F2 f1=new F2();
   }
```

```
This is constructor of Parent class
This is constructor of F1 class
This is constructor of F2 class
```

1. Object
2. Parent() call **super()**
3. F1() call **super ()**
4. F2() call **super()**
5. Main() call new F2()

- Khi tạo một thể hiện của lớp dẫn xuất , trước hết phải gọi đến hàm khởi tạo của lớp cha, tiếp đó là hàm khởi tạo của lớp con.

Các nguyên tắc của hàm khởi tạo

- Hàm dựng mặc nhiên (default constructor) sẽ tự động sinh ra bởi trình biên dịch nếu lớp không khai báo hàm dựng.
- Hàm dựng mặc nhiên luôn luôn không có tham số
- Nếu trong lớp có định nghĩa hàm dựng, hàm dựng mặc nhiên sẽ không còn được sử dụng.
- Nếu không có lời gọi tương minh đến hàm dựng của lớp cha tại lớp con, trình biên dịch sẽ tự động chèn lời gọi tới hàm dựng mặc nhiên (implicity) hoặc hàm dựng không tham số (explicity) của lớp cha trước khi thực thi đoạn code khác trong hàm dựng lớp con.

Có 1 vấn đề?

```
public class Parent
{
    private int a;

    public Parent(int value)
    {
        a = value;
        System.out.println("Invoke parent parameter constructor");
    }
}
```

```
public class F1 extends Parent
{
    public F1()
    {
        System.out.println("Invoke F1 default constructor");
    }
}
```

Sửa như thế nào?

```
public class Parent
{
    private int a;

    public Parent()
    {
        a = 0;
        System.out.println("Invoke parent default constructor");
    }

    public Parent(int value)
    {
        a = value;
        System.out.println("Invoke parent parameter constructor");
    }
}
public class F1 extends Parent
{
    public F1()
    {
        System.out.println("Invoke F1 default constructor");
    }
}
```

Triệu hồi tường minh hàm khởi tạo lớp cha

```
public class Parent
{
    private int a;

    public Parent(int value)
    {
        a = value;
        System.out.println("Invoke parent parameter constructor");
    }
}

public class F1 extends Parent
{
    public F1(int value)
    {
        super (value);
        System.out.println("Invoke F1 default constructor");
    }
}
```

Ví dụ thừa kế (tt)

```
class Person {
    private String CMND;
    private String Name;
    private int age;
    public Person(String cm, String na, int a){  CMND=cm;
        Name=na;
        age=a;
    }
    public void Print()
    {   System.out.println("Chung minh"+"\t"+"Tên"+"\t"+"Tuoi");
        System.out.print(CMND+"\t"+Name+"\t"+age);
    }
}
```

Ví dụ thừa kế (tt)

```
class Employee extends Person
{  private double salary;
   public Employee(String cm, String na, int a, double sa)
   {
      super(cm,na,a);
      salary=sa;
   }
   public void Print()
   {   super.Print();
      System.out.print("Luong thang:"+salary);
   }
}
```

Ví dụ thừa kế (tt)

```
class Maneger extends Employee {
    private double allowance;
    public Maneger(String cm, String na, int a, double sa, double allow)
    {  super(cm,na,a,sa);
       allowance=allow;
    }
    public void Print()
    {  super.Print();
       System.out.print("Phu cap:"+allowance);
    }
}
```

Ví dụ thừa kế (tt)

```
class Maneger extends Employee {
    private double allowance;
    public Maneger(String cm, String na, int a, double sa, double allow)
    {  super(cm,na,a,sa);
       allowance=allow;
    }
    public void Print()
    {  super.Print();
       System.out.print("Phu cap:"+allowance);
    }
}
```

Ví dụ thừa kế (tt)

```
public class Nhanvien {
    public static void main(String[] args) {
    Person  p=new Person("1234", "NGuyen Huu Đat",23);
    Employee e=new Employee("2345","Tran Ngoc Tuan", 24,10000000);
   Maneger mng= new Maneger("3456", "Lê Văn Toàn",
25,10000000,2000000);
            System.out.println("Thong tin nguoi:");
            p.Print();
            System.out.println();
            System.out.println("Thong nhan vien:");
            e.Print();
            System.out.println();
            System.out.print("Thong tin quan ly");
            mng.Print();}}
```

Kết quả

Thong tin nguoi:

Chung minh Tên Tuoi

1234 NGuyen Huu Đat 23

Thong nhan vien:

Chung minh Tên Tuoi

2345 Tran Ngoc Tuan 24Luong thang:1.0E7

Thong tin quan lyChung minh Tên Tuoi

3456 Lê Văn Toàn 25Luong thang:1.0E7Phu cap: 2000000.0

Phương thức ghi đè - Overriding Methods

- Overriding Methods: Lớp con định nghĩa một hay nhiều phương thức đã được định nghĩa ở lớp cha.

Ví dụ Overriding Methods

```
public class Parent
{
    public void printInfor()
    {
        System.out.println("Calling printInfor()method of Parent");
    }
}
```

```
public class F1 extends Parent
{
    public void printInfor()
    {
        System.out.println("Calling printInfor()method of F1");
    }
}
```

Biến, phương thức và lớp Final

- **Biến Final - Final Variables**
- **Phương thức Final - Final Methods**
- **Lớp Final - Final Classes**

Biến final

- Từ khóa "final" được sử dụng với biến để chỉ rằng giá trị của biến là hằng số.
- Hằng số là giá trị được gán cho biến vào thời điểm khai báo và sẽ không thay đổi về sau.

```
public final int MAX_COLS =100;
```

Phương thức hằng (Final)

- Được sử dụng để ngăn chặn việc ghi đè (override) hoặc che lấp (hidden) trong các lớp Java.
- Phương thức được khai báo là private hoặc là một thành phần của lớp final thì được xem là phương thức hằng.
- Phương thức hằng không thể khai báo là trừu tượng (abstract).

```
public final void find()
{
      //.....
}
```

Lớp hằng - Final Classes

- Là lớp không có lớp con.
- Được sử dụng để hạn chế việc thừa kế và ngăn chặn việc sửa đổi một lớp.
- Là lớp có thể hoặc không có các phương thức hằng.
- Lớp hằng có thể tạo đối tượng

```
public final class Student {
    // ...
}
```

Tính đa hình

- **Đa hình:** Cùng một phương thức có thể có những cách thi hành khác nhau tại những thời điểm khác nhau. Trong Java tự động thể hiện tính đa hình
- **Abstract**: Lớp trừu tượng, hàm trừu tượng
- **Interface:** được cài đặt bởi các lớp con để triển khai các phương thức mà lớp muốn có.

Tính đa hình – Ví dụ (tt)

```
package tronvuong;

class Hinh {

    public void Ve()

    {

        System.out.println("Ve hinh");

    }

}
```

Tính đa hình – Ví dụ (tt)

```
package tronvuong;
class HinhTron extends Hinh {
    public void Ve()
    {
        System.out.println("Ve tron");
    }
}
```

Tính đa hình – Ví dụ (tt)

```
package tronvuong;
class HinhVuong extends Hinh{
    public void Ve()
    {
        System.out.println("Ve vuong");
    }
}
```

Tính đa hình – Ví dụ (tt)

```
package tronvuong;
 public class   TronVuong {
    public static void main(String[] args)
    {  Hinh h=new Hinh();
       h.Ve();
       Hinh h1 = new HinhVuong();  h1.Ve();
       Hinh h2 = new HinhTron();  h2.Ve(); }
     }
```

Tính đa hình – Ví dụ (tt)

- ***Kết quả xuất ra màn hình***:

```
Ve hinh
Ve vuong
Ve tron
```

Lớp trừu tượng

- Dùng để định nghĩa làm lớp cha cho các lớp khác
- Dùng để định nghĩa các phương thức và thuộc tính chung cho các lớp con của nó
- *Ví dụ*: Lớp "hình tròn", "hình vuông" thừa kế từ lớp "hình vẽ".
- Dùng từ khóa abstract để khai báo một lớp trừu tượng
- Lớp abstract không thể tạo ra đối tượng.
- Có thể khai báo 0,1 hoặc nhiều phương thức trừu tượng bên trong lớp
- Các lớp con của lớp trừu tượng phải cài đặt các phương thức trừu tượng của lớp trừu tượng, nếu không nó trở thành trừu tượng
- Không thể tạo ra một đối tượng thuộc lớp trừu tượng, nhưng có thể khai báo biến thuộc lớp trừu tượng để tham chiếu đến các đối tượng thuộc lớp con của nó

Khai báo lớp trừu tượng

```
abstract class Tênlớp{
    <các thuộc tính>
    <các phương thức trừu tượng>
}
```

Ví dụ lớp trừu tượng

```
package tronvalapphuong;
 astract class Hinh
 {
        static final double PI=3.1415;
        public abstract double DienTich();
        public abstract double TheTich();
  }
```

Ví dụ lớp trừu tượng (tt)

```
package tronvalapphuong;  class
HinhTron extends Hinh {
    private double R;
    public HinhTron(double r) {
        R=r;
    }
    @Override
    public double DienTich() {return PI*R*R; }
   @Override
    public double TheTich() {  return 0;
                        }
    }
```

Ví dụ lớp trừu tượng (tt)

```
package tronvalapphuong;
class HinhLapPhuong extends Hinh {
    private double a;private double b;private double c;
    public HinhLapPhuong(double aa, double bb, double cc) {
        a=aa;b=bb;c=cc;
    }
    @Override
    public double DienTich() {return(2*(a*b+b*c+a*c));   }
    @Override
    public double TheTich() {
        return a*b*c;
    }
    }
```

Ví dụ lớp trừu tượng (tt)

```
package tronvalapphuong;  public class
TronVaLapPhuong {
    public static void main(String[] args){  Hinh
                hr= new HinhTron(5.5);
        System.out.println("Hinh tron");
        System.out.println("Dien Tich: "+hr.DienTich());
        System.out.println("The Tich: "+hr.TheTich());  Hinh
        hlp=new HinhLapPhuong(2,3,4);
        System.out.println("Hinh lap phuong: ");
        System.out.println("Dien Tich: "+hlp.DienTich());
        System.out.println("The Tich: "+hlp.TheTich());
    }
}
```

Kết quả

Hinh tron

Dien Tich: 95.03037499999999

The Tich: 0.0

Hinh lap phuong:

Dien Tich: 52.0

The Tich: 24.0

Giao tiếp (giao diện – Interfaces)

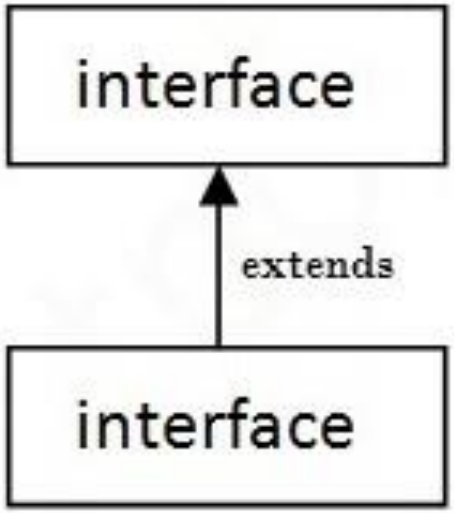
- Giới thiệu về giao tiếp.
- Hiện thực nhiều giao tiếp

Giao tiếp - interface

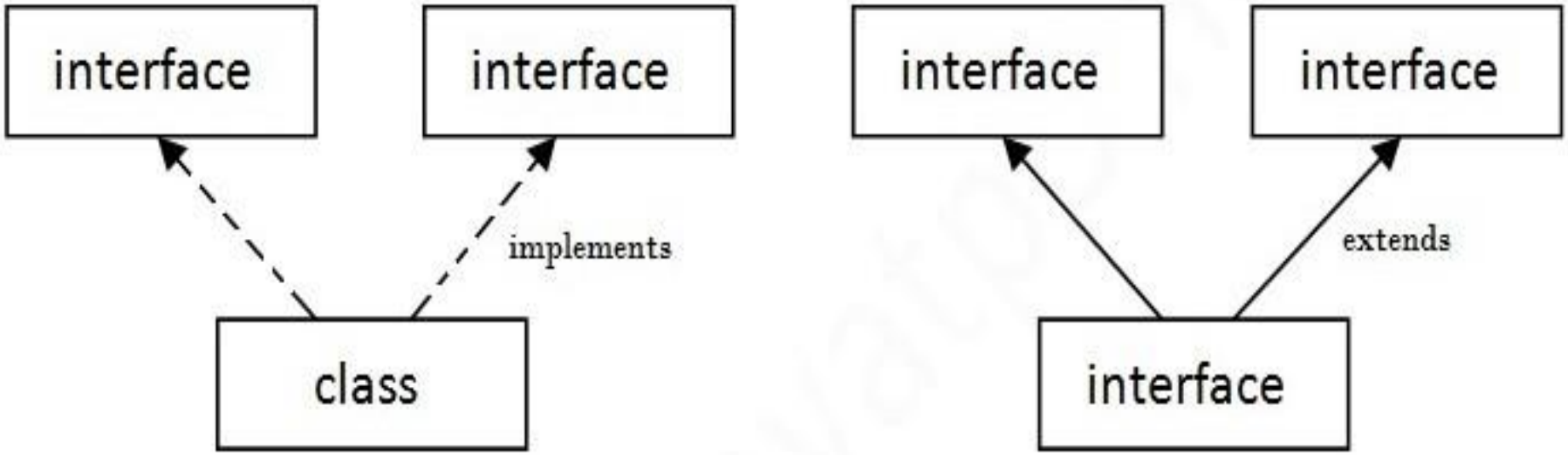
❖ **Interface:** giao tiếp của một lớp**, là phần đặc tả** (không có phần cài đặt cụ thể) của lớp, nó chứa các khai báo phương thức và thuộc tính để bên ngoài có thể truy xuất được. (java, C#, …)

- ✓ Lớp sẽ cài đặt các phương thức trong interface.
- ✓ Trong lập trình hiện đại các đối tượng không đưa ra cách truy cập cho một lớp, thay vào đó cung cấp các interface. Người lập trình dựa vào interface để gọi các dịch vụ mà lớp cung cấp.
- ✓ Thuộc tính của **interface** là các **hằng** và **các phương thức của** lớp giao tiếp là trừu tượng (mặc dù không có từ khóa abstract).
- ✓ Một lớp implements nhiều interface

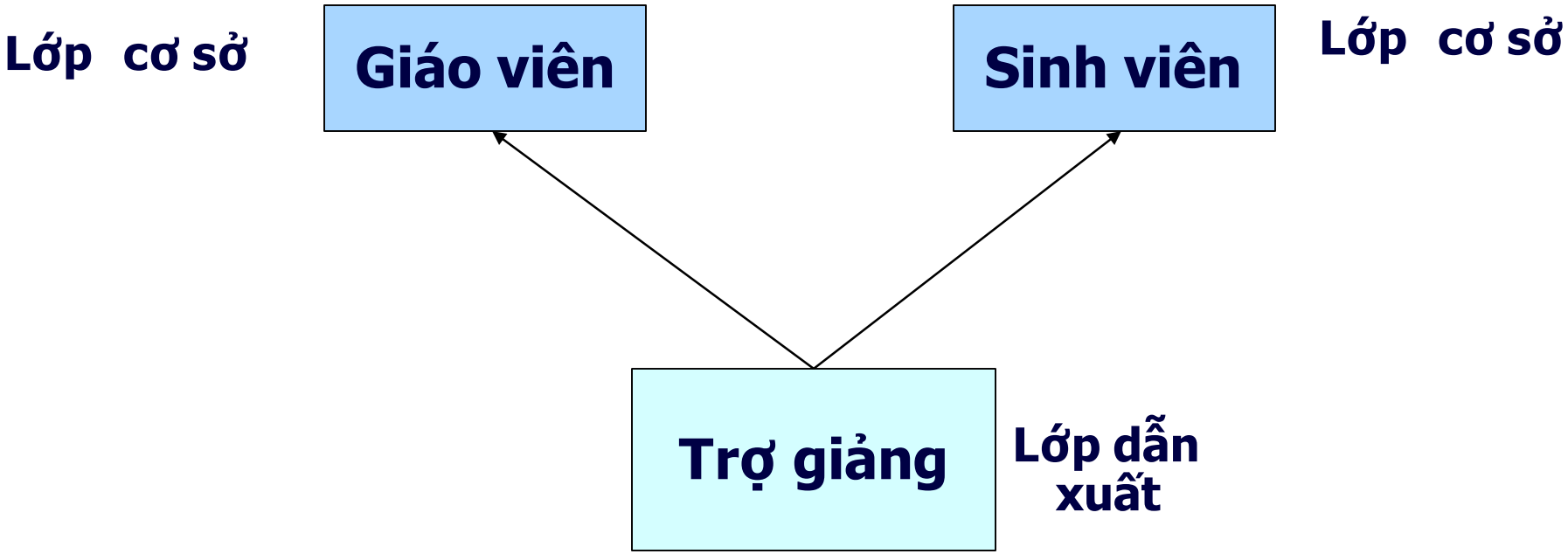
Quan hệ giữa các lớp interface



Đa thừa kế trong Java



Đa thừa kế (tiếp)



Ví dụ - interface

***Ví dụ**:*

```
// Định nghĩa một interface "Hinh" trong tập tin
package vidu1;
public interface Hinh
{  final double PI=3.1415;
   public double DienTich();
   public double ChuVi();
   public String LayTenHinh();
   public void Nhap();
}
```

Ví dụ- interface

- **Định nghĩa lớp "HinhTron" implement từ lớp "Hinh"**

```
package vidu1;
import java.util.Scanner;
class HinhTron implements Hinh{
    private double R;
    @Override
    public double DienTich() {
        return PI*R*R;
       }
    @Override
    public double ChuVi() {
        return 2*PI*R;
    }
```

```
@Override
    public String LayTenHinh() {
        return ("Hình tròn");
    }
    @Override
    public void Nhap(){
        System.out.print("Nhap R=");
        Scanner scan = new Scanner
(System.in);
        R=scan.nextDouble();
    }
}
```

Ví dụ- interface

- **Định nghĩa lớp "HinhVuong" implement từ lớp "Hinh"**

```
package vidu1;
import java.util.Scanner;
class HinhVuong implements Hinh
{
    private double canh;
    @Override
    public double DienTich() {
        return canh*canh;
    }
    @Override
    public double ChuVi() {
        return canh*4;
    }
```

```
@Override
    public String LayTenHinh() {
        return ("Hình vuông");
    }
    @Override
    public void Nhap()
    {
        System.out.print("Nhap
canh=");
        Scanner scan = new Scanner
(System.in);
        canh=scan.nextDouble();
    }
}
```

Ví dụ- interface

- **Sử dụng các lớp**

```
package vidu1;
public class Vidu1 {
  public static void main(String[] args)
  {    Hinh h=new HinhTron();
       h.Nhap();
       System.out.println("Dien tich hình tron:"+h.DienTich());
       System.out.println("Chu vi hình tron:"+h.ChuVi());
       h= new HinhVuong();
       h.Nhap();
       System.out.println("Dien tich hình vuong:"+h.DienTich());
       System.out.println("Chu vi hình :"+h.ChuVi());
    }
}
```

Kết quả

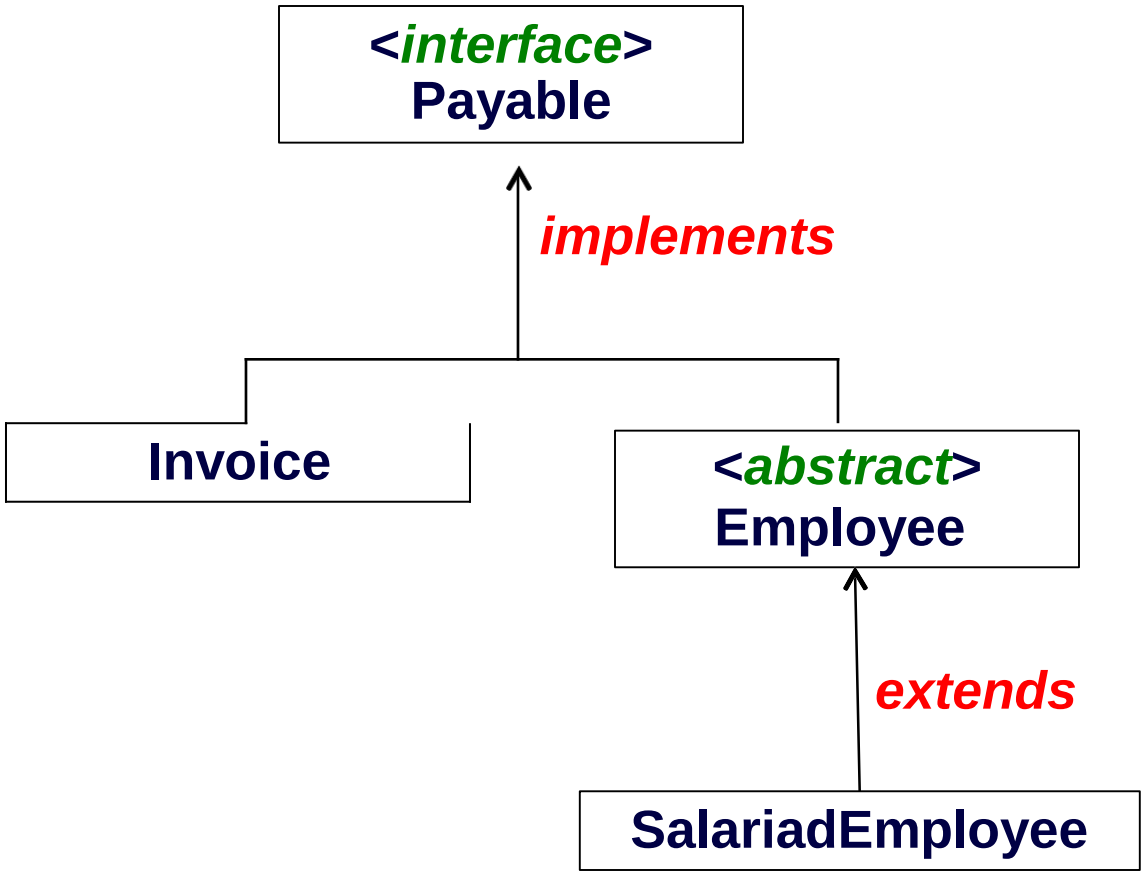
```
Nhap R=2
Dien tich hình tron:12.566
Chu vi hình tron:12.566
Nhap canh=4
Dien tich hình vuong:16.0
Chu vi hình :16.0
```

Kế thừ interface

❖ **Kế thừa interface**

```
public interface InterfaceName extends interface1, interface2, interface3
{
        // ...
}
```

Ví dụ - interface



Ví dụ- interface

❖ *Lớp tính lương (interface)*

```
package salary;
public interface TienTra
{
     double TongTienTra();
    void In();
}
```

Lớp "hoadon"

```
package salary;
class HoaDon implements TienTra
{       private String MaHD;
        private String MoTa;
        private int SoLuong;
        private double Gia;
        public HoaDon(String Ma, String Mo, int So, double gia )
        {
            MaHD=Ma;
            MoTa=Mo;
            SetSoLuong(So);
            Gia=gia;
        }
```

Lớp "hoadon"

```
public void SetSoLuong(int soluong)
{  SoLuong=(soluong<0)?0:soluong;
}
public void SetGia(double gia)
{  Gia=(gia<0)?0:gia;
}
@Override  public void In(){
        System.out.printf("%s:\n%s:%s\n%s:%s\n%s:%d\n%s:%.2f\n%s:%.2f
\n","Hóa đơn","Mã hóa đơn",MaHD,"Mô Tả",MoTa,"Số
Lượng",SoLuong,"Giá",Gia,"Tổng tiền",this.TongTienTra());
      }
@Override
   public double TongTienTra() {
      return SoLuong*Gia;
   }
```

Lớp "congnhan"

```
package salary;
public abstract class CongNhan implements TienTra {
    private String CMND;
    private String Ten;
    public CongNhan(String cm, String ten){
        CMND=cm;
        Ten=ten;
    }
    protected String GetCM(){
        return CMND;
    }
    protected String GetTen(){
        return Ten;
    }
}
```

Lớp "LuongNV"

```
package salary;
/*Tính luong nhan vien*/
public class LuongNV extends CongNhan {
    private double LuongTuan;
    public LuongNV(String cm, String ten, double luong)
    {
        super(cm,ten);
        LuongTuan=luong;
        //SetLuongTuan(luong);
    }
```

Lớp "LuongNV"

```
public void SetLuongTuan(double luong)
{  LuongTuan = (luong<0) ? 0:luong;
}
@Override
public double TongTienTra()
{  return LuongTuan;
}
@Override  public void In(){
        System.out.printf("%s\n%s:%s\n%s:%s\n%s:%.2f\n","Công
nhân","Chứng minh ND",super.GetCM(), "Tên", super.GetTen
(),"Luong Tuan",LuongTuan);
    }
```

Lớp chứa hàm "main"

```
package salary;
public class Salary {
    public static void main(String[] args
    {  TienTra tr[] = new TienTra[4];
        tr[0] = new HoaDon("123", "Ghế", 2, 3000);
        tr[1] = new HoaDon("234","Lốp",3,2000);
        tr[2]=new LuongNV("CN123","Nguyễn Văn Tùng",9000000);
        tr[3]=new LuongNV("CN123","Nguyễn Văn Tùng",10000000);
        for(TienTra tientra:tr)
            tientra.In();
    }
```

Kết quả

Hóa đơn:
Mã hóa đơn:123 Mô
Tả:Ghế
Số Lượng:2
Giá:3000.00
Tổng tiền:6000.00 Hóa đơn:
Mã hóa đơn:234 Mô
Tả:Lốp
Số Lượng:3
Giá:2000.00
Tổng tiền:6000.00
Công nhân
Chứng minh ND:CN123
Tên:Nguyễn Văn Tùng Luong
Tuan:9000000.00 Công nhân
Chứng minh ND:CN123
Tên:Nguyễn Văn Tùng Luong
Tuan:10000000.00

Quan hệ giữa Class và Interface

	Class	Interface
Class	extends	implements
Interface		extends