Chương 3. Kế thừa

Nội dung

- Khái niệm
- Khai báo kế thừa
- Phạm vi kế thừa
- Đối tượng super
- Định nghĩa lại phương thức
- Liên kết muộn
- Ứng dụng liên kết muộn
- Lớp Object

Nội dung (tiếp)

- Lớp trừu tượng
- Giao diện
- Một số lớp tiện ích trong Java
  - Giao diện Enumeration
  - Lớp Vector
  - Lớp Hashtable
  - Lớp StringTokenizier
- Lớp trong (Inner Class)

1. Khái niệm

- Kế thừa cho phép định nghĩa một lớp mới qua một lớp đã có.
- Lớp dùng để kế thừa gọi là lớp cha (lớp cơ sở).
- Lớp kế thừa gọi là lớp con (lớp dẫn xuất).
- Lớp con có một số thành phần của lớp cha mà không cần định nghĩa.
- Lớp con có thể định nghĩa thêm các thành phần riêng của mình.

2. Khai báo kế thừa

- Cú pháp:

```
class <TênLớpCon>extends<TênLớpCha>
{
   Khai báo các thành phần bổ sung của lớp con
}
```

- Ví dụ: lớp SV kế thừa từ lớp CONNGUOI

```
class SV extends ConNguoi{
   ....
}
```

3. Phạm vi kế thừa

- Lớp con được phép kế thừa các thành phần của lớp cha với phạm vi:
  - public
  - protected
- Thành phần protected: được phép kế thừa nhưng không được phép truy xuất bên ngoài lớp.

Ví dụ kế thừa

```
class ConNguoi {
        protected String hoTen;
        protected  int namSinh;
        public ConNguoi(){ hoTen=“”; namSinh=1900;}
        public ConNguoi(String ht, int ns){ ...}
        public void ganHoTen(String ht){...}
        public void ganNamSinh(int ns){...}
        public String layHoTen(){...}
        public int layNamSinh(){...}
        public void hienThi()
        {  System.out.print(hoTen+” “ + namSinh);} }
```

Ví dụ kế thừa (tiếp)

```
class SV extends ConNguoi {
        protected double dtb;
        public void ganDTB(double d){...}
        public double layDTB(){...}
        public void hienThi(){...}
    }
```

Lớp SV có những thành phần nào?

4. Đối tượng super

- Đối tượng super dùng để truy xuất đến đối tượng thuộc lớp cha trong phạm vi lớp con.
- Sử dụng super:
  - Gọi phương thức khởi tạo lớp cha: super(...)
  - Gọi phương thức thuộc lớp cha: super.tenPhươngThức(...)

- *Lời gọi super() phải được gọi đầu tiên trong PTKT lớp con (sau khai báo).*
- *Nếu trong lớp con không gọi PTKT của lớp cha thì tự động gọi PTKT ngầm định của lớp cha.*

Ví dụ sử dụng super

```
class SV extends ConNguoi {
    protected double dtb;
    public SV() { super(); dtb=0.0;}
    public SV(String ht, int ns, double d)
     { super(ht,ns); dtb=d; }
    public void ganDtb(double d){...}
    public double layDtb(){...}
    public void hienThi(){...}
  }
```

5. Định nghĩa lại phương thức

- Trong lớp con được phép định nghĩa lại các phương thức kế thừa từ lớp cha.
- Đối tượng của lớp con sẽ sử dụng phương thức đã định nghĩa lại.
- Ví dụ: định nghĩa lại phương thức hienThi() trong lớp SV:

```
class SV extends ConNguoi
    {
       protected double dtb;
       …
       public void hienThi()
       {
           super.hienThi();
           System.out.println(“ ” + dtb);
       }
    }
```

- Sử dụng:

```
SV s = new SV(“Ng Van A”,1985,7.5);
s.hienThi(); //gọi phương thức đã được định nghĩa lại.
```

5. Định nghĩa lại phương thức (tiếp)

- Phương thức không được định nghĩa lại:
  - Là phương thức của lớp cha không cho phép các lớp con định nghĩa lại.
  - Khai báo phương thức trong lớp cha bằng cách thêm từ khóa final.
  - Ví dụ:
    final public void finalMethod(...)
    {...}

Lớp không được kế thừa

- Để không cho một lớp được kế thừa ta thêm từ khóa final trước khai báo lớp.

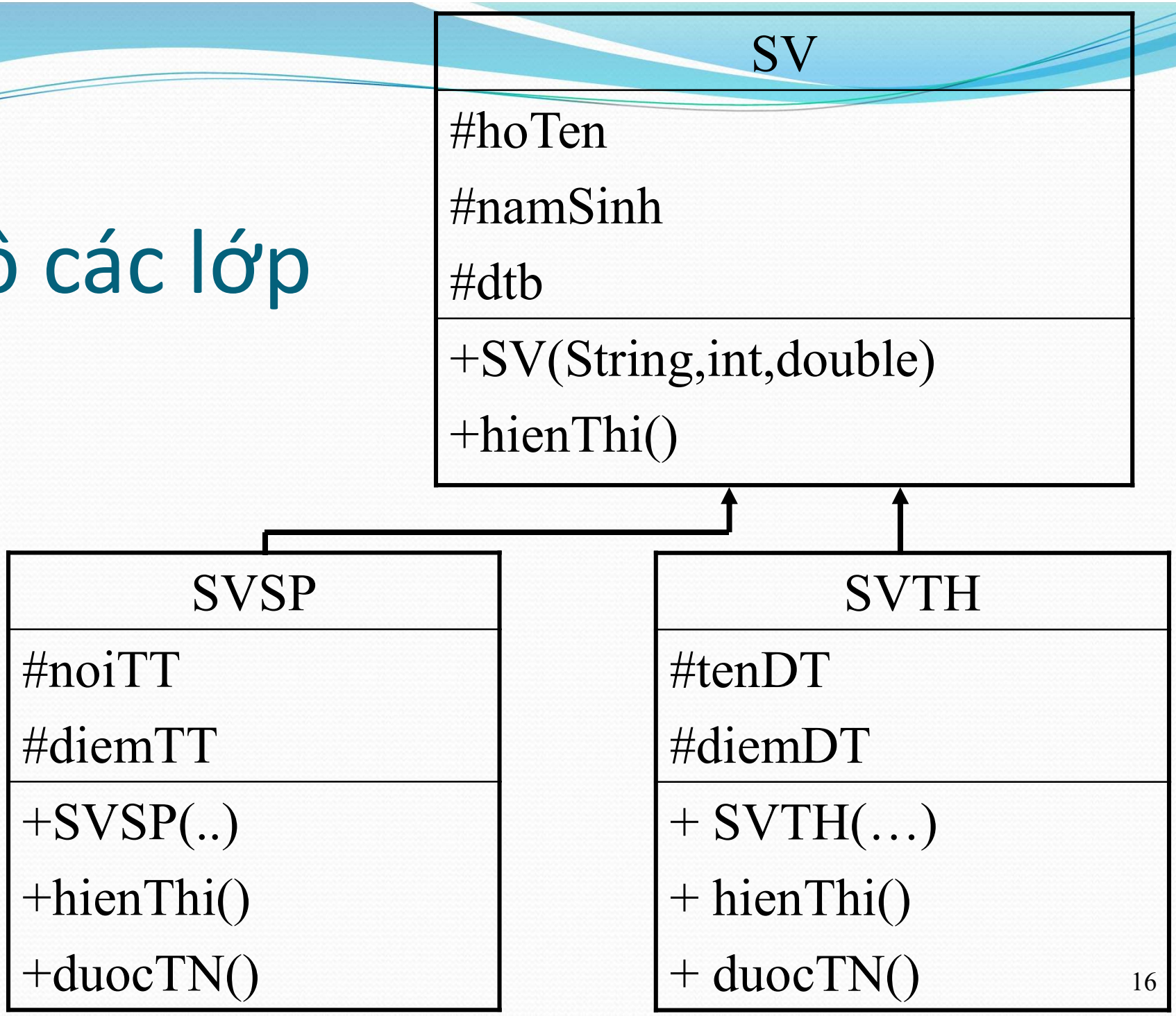
```
final class FinalClass
{
    ...
}
```

- Trong Java lớp Math không được kế thừa.

Luyện tập

- Khai báo lớp SV (theo sơ đồ)
- Khai báo các lớp SVSP, SVTH kế thừa từ lớp SV.
- Chương trình:
  - Tạo 1 sinh viên SP
  - Tạo 1 sinh viên TH
  - Hiển thị thông tin của 2 SV trên

Sơ đồ các lớp


SV
#hoTen
#namSinh
#dtb
+SV(String,int,double)
+hienThi()
SVSP
#noiTT
#diemTT
+SVSP(..)
+hienThi()
+duocTN()
SVTH
#tenDT
#diemDT
+ SVTH(…)
+ hienThi()
+ duocTN()

Cài đặt lớp SV

```
class SV{
        protected String hoTen;
        protected int namSinh;
        protected double dtb;
        public SV(String ht, int ns, double d)
        {hoTen=ht; namSinh=ns; dtb=d;}
        public void hienThi()
        {  System.out.print(hoTen+" "+namSinh+" "+dtb);}
    }
```

Cài đặt lớp SVSP

Cài đặt lớp SVTH

6. Liên kết muộn

- Tham chiếu của lớp cha có thể tham chiếu đến đối tượng của lớp con.
- Ví dụ:

```
SV s1,s2;
s1= new SVSP("Ng V A",1986,7.5,"Trg A",7.0);
s2 = new SVTH("Le Th B",1987,7,"VB", 8.0);
s1.hienThi(); ??
s2.hienThi(); ??
```

6. Liên kết muộn (tiếp)

- Cơ chế liên kết muộn: khi tham chiếu đến đối tượng thuộc lớp nào thì sử dụng các thành phần của lớp đó định nghĩa lại từ lớp cha hoặc được kế thừa từ lớp cha.
- Ví dụ: với s1, s2 như trên

```
s1.hienThi(); // sử dụng phương thức hienThi() của lớp SVSP
s2.hienThi(); // sử dụng phương thức hienThi() của lớp SVTH
```

6. Liên kết muộn (tiếp)

- Tham chiếu của lớp cơ sở chỉ sử dụng được những phương thức định nghĩa lại.
- Ví dụ: không thực hiện được

```
boolean d1 = s1.duocTN();
boolean d2 = s2.duocTN();
```

- Để thực hiện được phải khai báo phương thức duocTN() ở lớp SV.

7. Ứng dụng liên kết muộn

- Chương trình quản lý danh sách sinh viên gồm 2 loại SVSP và SVTH. Thực hiện các thao tác:
  - Tạo lập và lưu trữ danh sách SV.
  - Liệt kê danh sách sv.
  - Liệt kê danh sách svsp.
  - Liệt kê danh sách sv được tốt nghiệp.

Sơ đồ các lớp

SV
#hoTen #namSinh #dtb
+SV(…) +hienThi() +loaiSV() +duocTN()

DSSV
- ds[] - soSV
+DSSV(int) +them(SV) +lietKe() +lietKe(String) +dsTN()

SVSP
#noiTT #diemTT
+SVSP(…) +hienThi() +loaiSV() +duocTN()

SVTH
#tenDT #diemDT
+SVTH(…) +hienThi() +loaiSV() +duocTN()

Lớp SV

```
class SV{
        protected String hoTen;
        protected int namSinh;
        protected double dtb;
        public SV(String ht, int ns, double d){...}
        public void hienThi(){...}
        public String loaiSV(){ return “”;}
        public boolean duocTN() { return true;}
    }
```

Lớp SVSP

```
class SVSP extends SV{
        protected String noiTT;
        protected double diemTT;
        public SVSP(...) {...}
        public void hienThi(){...}
        public String loaiSV(){ return "SP";}
        public boolean duocTN()
        { return dtb>=5.0 && diemTT>=7.0;}
    }
```

Cài đặt lớp DSSV

```
class DSSV{
        private SV[] ds;
        private int soSV;
        public DSSV(int n)
            {ds = new SV[n]; soSV=0;}
        public void them(SV s)
            { if (soSV<ds.length) ds[soSV++]=s; }
        public void lietKe() {
             for(int i=0;i<soSV;i++)
             ds[i].hienThi();
       }
```

Cài đặt lớp DSSV (tiếp)

```
public void lietKe(String loai) {
        for(int i=0; i<soSV; i++)
           if (ds[i].loaiSV().equals(loai))
                ds[i].hienThi();
}
public void dsTN()
{
    for(int i=0; i<soSV; i++)
       if (ds[i].duocTN())
                ds[i].hienThi();
}
```

Sử dụng lớp DSSV

```
class SDDSSV{
   public static void main(String args[])
   {  DSSV k1=new DSSV(80); SV s;
      s = new SVSP("Nguyen Van A",1987,7.0,"Truong X",7.0);
      k1.them(s);
      s = new SVTH("Le Thi B",1987,7.5,"Java",7.0);
      k1.them(s);
      …
      k1.lietKe();
      k1.lietKe("SP");
      k1.dsTN(); }
}
```

Luyện tập

- Cần quản lý một danh sách nhân viên của 1 cơ quan gồm 2 loại: Nhân viên biên chế và nhân viên hợp đồng.
- Các thông tin chung: họ tên, phòng.
  - NVBC: hệ số lương, số năm CT.
  - NVHD: lương hợp đồng, Loại HĐ(NH,DH).
- Các thao tác trên danh sách nhân viên:
  - Tạo lập và lưu các nhân viên
  - Liệt kê danh sách nhân viên.
  - Liệt kê danh sách nhân viên theo loại: HĐ, BC
  - Tính tổng lương toàn bộ nhân viên.
  - Liệt kê danh sách nhân viên hợp đồng dài hạn.

Sơ đồ các lớp

DSNV
- ds[] - soNV
+DSNV(int) +them(NV) +lietKe() +lietKe(String) +tongLuong() +dsHDDH()

NV
#hoTen #phong
+NV(…) +hienThi() +loaiNV() +layLuong() +laNVHDDH()

NVBC
#hsLuong #soNamCT
+NVBC(..) +hienThi() +loaiNV() +layLuong()

NVHD
#luong #loaiHD
+NVHD(…) +hienThi() +loaiNV() +layLuong() +laNVHDDH()

Bài thực hành số 4

- Bài 1. Thực hành bài DSSV
- Bài 2. Thực hành bài Thùng thư
- Bài 3. Thực hành bài cửa hàng CD.

8. Lớp Object

- Đây là lớp đối tượng cấp cao nhất của các lớp trong Java.
- Mọi lớp trong Java đều kế thừa từ lớp này.
- Đối tượng Object có một số phương thức:
  - public boolean equals(Object)
  - public String toString()
- Ta có thể dùng tham chiếu của lớp Object để tham chiếu đến một đối tượng thuộc lớp bất kỳ.
- Ví dụ: Object o = new SV(...);

9. Lớp trừu tượng

- Phương thức trừu tượng là phương thức không cài đặt chi tiết.
- Khai báo PTTT:

abstract <khai báo phương thức>;

- Ví dụ: khai báo phương thức duocTN() của lớp SV.

abstract boolean duocTN();

9. Lớp trừu tượng (tiếp)

- Lớp trừu tượng (Abstract Class) là lớp chứa ít nhất một PTTT.
- Lớp trừu tượng dùng để làm cơ sở định nghĩa các lớp khác.
- Khai báo lớp trừu tượng:

```
abstract class <TênLớp>
{
   khai báo các thành phần của lớp
}
```

9. Lớp trừu tượng (tiếp)

- Ví dụ: khai báo lớp trừu tượng SV

```
abstract class SV
{
    …
    abstract public boolean duocTN();
}
```

- *Lưu ý: không thể tạo đối tượng từ lớp trừu tượng.*
- Ví dụ: không thể tạo đối tượng từ lớp SV

```
SV s = new SV(…);
```

9. Lớp trừu tượng (tiếp)

- Lớp kế thừa từ lớp trừu tượng phải khai báo tường minh các PTTT nếu không cũng là lớp trừu tượng.
- Ví dụ: khai báo lớp SVSP kế thừa từ lớp SV

```
class SVSP extends SV
{
    …
    public boolean duocTN(){…}
}
```

9. Lớp trừu tượng (tiếp)

- Ví dụ: lớp sinh viên tại chức (SVTC) kế thừa từ lớp SV, là lớp trừu tượng.

```
abstract class SVTC extends SV
{
    protected String noiCT;
    ...
    abstract public boolean duocTN();
}
```

9. Lớp trừu tượng (tiếp)

- Ví dụ: xây dựng các lớp tính diện tích các hình: tròn, tam giác, chữ nhật.
- Chương trình minh họa gồm một mảng các đối tượng và tính tổng diện tích của các hình trong mảng.

Lớp HINH

```
abstract class HINH
{
    abstract double dienTich();
}
```

Lớp HinhTron

```
class HinhTron extends HINH {
    double bk;
    public HinhTron(double b)
    { bk = b;}
    public double dienTich()
    {
       return bk*bk*Math.PI;
    }
}
```

Lớp HinhCN

```
class HinhCN extends HINH {
    double dai,rong;
    public HinhCN(double d, double r)
    { dai = d; rong = r;}
    public double dienTich()
    {
       return dai*rong;
    }
}
```

Lớp TamGiac

```
class TamGiac extends HINH {
    double c1,c2,c3;
    public TamGiac(double a, double b, double c)
    { c1 = a; c2 = b; c3 = c;}
    public double dienTich()
    {  double p = (c1+c2+c3)/2
       return Math.sqrt(p*(p-c1)*(p-c2)*(p-c3));
    }
}
```

Lớp DTHINH

```
class DTHINH {
        public static void main(String args[])
        {    HINH ds[]=new HINH[5];
             ds[0]=new HinhTron(1.3);
             ds[1]=new TamGiac(3,4,5);
             ds[2]=new HinhCN(2,5);
             ds[3]=new HinhTron(3.0);
             ds[4]=new HinhCN(4,3);
            //Tính tổng diện tích các hình
            double tongDT=0;
            for(int i=0;i<5;i++) tongDT+=ds[i].dienTich();
            System.out.println("Tong dien tich "+ tongDT);
        }
    }
```

10. Đa kế thừa

- Đa kế thừa là sự kế thừa từ nhiều lớp.
- Đa kế thừa tạo ra nhiều sự nhập nhằng và phức tạp nên trong Java không hỗ trợ đa kế thừa.

11. Giao diện (Interface)

- Giao diện là một giải pháp của Java nhằm thay thế cho đa kế thừa.
- Một giao diện là một tập các hằng, các mẫu phương thức (prototype) mà không có cài đặt chi tiết.
- Khai báo giao diện:

```
interface <TênGiaoDiện>
{
   khai báo các nội dung của giao diện
}
```

11. Giao diện (tiếp)

- Ví dụ: Giao diện tính toán trên các hình.

```
interface TinhToanHinh
{
        double PI = 3.1415;
        double dienTich();
        double chuVi();
}
```

11. Giao diện (tiếp)

- Có thể coi giao diện như một lớp trừu tượng đặc biệt: các phương thức đều trừu tượng.
- Khai báo lớp cài đặt giao diện: phải tường minh các phương thức của giao diện.
- Cú pháp:

```
class <TênLớp> implements <các giao diện>
    {
        Khai báo các thành phần của lớp
        Chi tiết các phương thức của các giao diện
    }
```

11. Giao diện (tiếp)

- Ví dụ: lớp hình chữ nhật cài đặt giao diện TinhToanHinh

```
class HinhChuNhat implements TinhToanHinh{
    private double dai, rong;
    public HinhChuNhat(double d, double r)
        {dai = d; rong = r;}
    public double chuVi() { return 2*(dai+rong);}
    public double dienTich() {return dai*rong;}
}
```

11. Giao diện (tiếp)

- Lớp hình tròn cài đặt giao diện TinhToanHinh

```
class HinhTron implements TinhToanHinh
{
   private double banKinh;
   public HinhTron(double d) { banKinh = d;}
   public double chuVi() { return 2*banKinh*PI;}
   public double dienTich()
   { return banKinh*banKinh*PI; }
}
```

11. Giao diện (tiếp)

- Lưu ý: một lớp cài đặt một giao diện thì phải cài đặt tất cả các phương thức mà giao diện khai báo (có thể là cài đặt rỗng).
- Một giao diện có thể mở rộng (kế thừa) từ một giao diện khác. Khi đó giao diện mới sẽ có đầy đủ các thành phần được khai báo trong giao diện cơ sở.

11. Giao diện (tiếp)

- Ví dụ: giao diện GiaoDienHinh mở rộng từ giao diện TinhToanHinh

```
interface GiaoDienHinh extends TinhToanHinh
{
   //khai báo các thành phần bổ sung
   void ve();
   void xoa();
   void dichuyen(int x, int y);
}
```

11. Giao diện (tiếp)

- Phạm vi của các thành phần trong giao diện:
  - Thành phần là hằng thì có phạm vi là **public static final**.
  - Thành phần là phương thức có phạm vi **public**.
- Sử dụng giao diện như tham chiếu: tương tự như lớp cơ sở trừu tượng, có thể dùng giao diện để tạo ra tham chiếu đến đối tượng cài đặt giao diện.
- Ví dụ:

```
TinhToanHinh x = new HinhTron(3.0);
```

11. Giao diện (tiếp)

- Ví dụ:

```
class GiaoDien1{
    public static void main(String args[]){
        TinhToanHinh h;
        h = new HinhChuNhat(1,2);
        System.out.println("Chu vi HCN = " + h.chuVi());
        System.out.println("Dien tich HCN = " + h.dienTich());
        h = new HinhTron(1);
        System.out.println("Chu vi hinh tron = " + h.chuVi());
        System.out.println("Dien tich hinh tron = "+ h.dienTich());
    }
}
```

11. Giao diện (tiếp)

- Một lớp có thể cài đặt nhiều giao diện.
- Ví dụ:

  class HinhTron implements TinhToanHinh, VeHinh

  { … }
- Phân biệt giữa lớp trừu tượng và giao diện:
  - Lớp trừu tượng có thể chứa các thuộc tính trong khi giao diện chỉ chứa các hằng và khai báo phương thức.
  - Một lớp có thể cài đặt nhiều giao diện trong khi chỉ kế thừa một lớp trừu tượng.
- Khi xây dựng một lớp trừu tượng không có dữ liệu và mọi phương thức đều là trừu tượng thì nên dùng giao diện để thay thế.

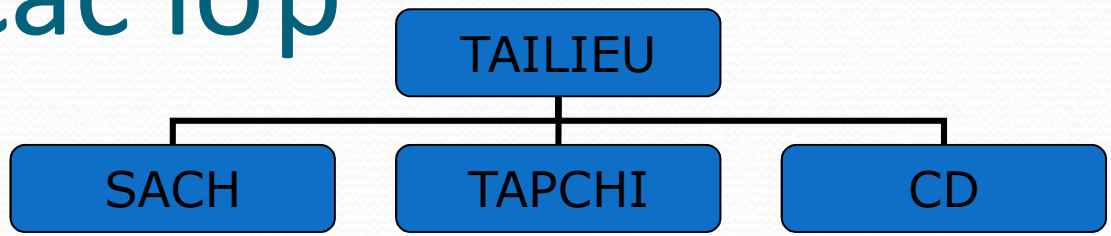
Bài tập

- Quản lý thư viện: Một thư viện gồm các loại tài liệu sau:
  - Sách(Mã sách, Tên Sách, Tác giả, NXB, Năm XB, Vị trí)
  - Tạp chí (Mã tạp chí, Tên tạp chí, Chuyên ngành, Số, Năm, Vị trí)
  - CD(Mã CD, Tên CD, Số thứ tự, Nội dung, Vị trí)

Quản lý thư viện

- Hãy tổ chức các lớp sao cho có thể lập trình để thực hiện được các chức năng sau:
  - Lưu danh sách các tài liệu có trong thư viện.
  - Liệt kê toàn bộ tài liệu có trong thư viện.
  - Liệt kê từng loại tài liệu có trong thư viện.
  - Xem thông tin về tài liệu khi biết mã tài liệu.
  - Tìm kiếm một tài liệu theo: Tên và tác giả đối với sách; Tên tạp chí, Chuyên ngành, số, năm đối với tạp chí, Tên CD, Số thứ tự và nội dung đối với CD

Sơ đồ các lớp



TAILIEU
#maTL #tenTL #viTri
TAILIEU(…) hienThi() …

SACH
#tacGia #nhaXB #namXB
SACH(…) hienThi() …

Sơ đồ các lớp

TAPCHI
#chuyenNganh #so #nam
TAPCHI(…) hienThi() …

CD
#soTT #noiDung
CD(…) hienThi() …

DSTL
-ds[] -soTL
DSTL(…) them() lietKe() lietKe(…) xem(…) tim(ten,tg) tim(ten,cn,so,nam) tim(ten,stt,nd)

Mở rộng bài toán quản lý thư viện

- Bổ sung vào lớp SV đã có những phương thức cần thiết để có thể quản lý việc mượn, trả tài liệu ở thư viện.
- Cài đặt lớp NKTV (nhật ký thư viện) dùng để quản lý việc mượn và trả sách của sinh viên. Lớp này phải thực hiện được các chức năng:
  1. Cho mượn tài liệu: cho 1 sv mượn 1 tài liệu nào đó.
  2. Trả tài liệu: ghi nhận việc trả 1 tài liệu của 1 sv.
  3. Xem danh sách những sv còn mượn tài liệu.
  4. Xem danh sách những tài liệu mà sv đang mượn.
  5. Liệt kê những tài liệu mà 1 sv nào đó đang mượn.

- Chương trình chính:
  - Tạo ra 3 sinh viên s1, s2, s3
  - Tạo ra 5 tài liệu: t1, t2, t3, t4, t5
  - Sinh viên s1 mượn tài liệu t2
  - Sinh viên s2 mượn tài liệu t4
  - Sinh viên s3 mượn tài liệu t1
  - Sinh viên s2 mượn tài liệu t3
  - Sinh viên s1 mượn tài liệu t5
  - Sinh viên s2 trả tài liệu t4
  - Sinh viên s3 mượn tài liệu t4
  - Xem danh sách các sinh viên mượn tài liệu
  - Xem danh sách tài liệu mà sinh viên đang mượn
  - Xem danh sách tài liệu mà sinh viên s1 đang mượn

Bài thực hành số 5

1. Thực hành bài quản lý tài liệu thư viện.
2. Thực hành bài Thùng thư bằng cách dùng giao diện hoặc lớp trừu tượng.

12. Một số lớp tiện ích

Nội dung

- Giới thiệu
- Giao diện Enumeration
- Lớp Vector
- Lớp HashTable
- Lớp StringToken

Giới thiệu

- Java cung cấp gói java.util.* chứa các giao diện và lớp tiện ích khi lập trình.

1. Giao diện Enumeration

- Giao diện Enumeration: dùng để duyệt các phần tử trong các danh sách.
- Giao diện chứa hai phương thức:
  - boolean hasMoreElements(): cho biết đã duyệt hết các phần tử trong danh sách chưa.
  - Object nextElement(): trả về phần tử tiếp theo trong danh sách.

2. Lớp Vector

- Lớp Vector dùng để lưu trữ và xử lý danh sách các đối tượng.
- Phương thức khởi tạo:

  Vector()

  Vector(int initialCapacity)

  Vector(int initialCapacity, int capacityIncrement)

2. Lớp Vector (tiếp)

- Một số phương thức:

  void addElement(Object obj)

  boolean contains(Object obj)

  Object elementAt(int index)

  int indexOf(Object element)

  int indexOf(Object element, int index)

  void removeElement(Object obj)

  void removeElementAt(int index)

  void removeAllElements()

2. Lớp Vector (tiếp)

void setElementAt(Object obj, int index)

int size()

String toString()

void trimToSize()

Enumeration elements()

- Ví dụ: chương trình sử dụng lớp Vector quản lý một danh sách đơn giản.

Ví dụ

```
import java.util.*;
class VectorExample
{
   public static void main(String args[])
   {
         Vector ds = new Vector();
         //Thêm các phần tử vào ds
         ds.addElement("Nguyen Van A");
         ds.addElement("Tran Thi B");
         ds.addElement("Le Thi C");
         ds.addElement("Ho D");
```

```
//Tìm một họ tên
   String ht="Le Thi B";
   if (ds.contains(ht))
      System.out.println("Có SV "+ht+ " tại vị trí " + ds.indexOf(ht));
   else
      System.out.println("Không có SV" + ht);
//In danh sách
   Enumeration e = ds.elements();
   while (e.hasMoreElements())
         System.out.println(e.nextElement());
//Xóa phần tử
    ds.removeElement("Nguyen Van A");
    System.out.println(ds);
   }
}
```

Vector của những đối tượng do người dùng định nghĩa

- Một đối tượng Vector có thể chứa đối tượng do người dùng định nghĩa.
- Để sử dụng được các phương thức: contains, indexOf, removeElement thì trong lớp do người dùng định nghĩa phải có phương thức:

boolean equals(Object)

Ví dụ vector các phân số

- Sử dụng lớp Vector để quản lý các phân số.

```
class PS{
    private int tu, mau;
    ...
    boolean equals(Object p)
    {
       PS q = (PS) p;
       return (double)tu/mau==(double) q.tu/q.mau;
    }
}
```

```
class VTPS
{
   public static void main(String args[])
   {   Vector v = new Vector();
       v.addElement(new PS(1,2));
       v.addElement(new PS(5,3));
       v.addElement(new PS(2,4));
       //Tìm phân số 2/4
       PS p = new PS(2,4);
       if (v.contains(p))
          System.out.println("Vị trí: " + v.indexOf(p));
      // Xóa phân số 2/4
          v.removeElement(p);
     //In vector
          System.out.println(v);        }
}
```

Luyện tập

- Cần quản lý một danh sách các loại sinh viên: SVTH, SVSP và thực hiện các thao tác trên danh sách:
  - Thêm 1 SV vào danh sách có kiểm tra trùng lặp.
  - Tìm một SV trong danh sách theo họ tên
  - Xóa một SV theo họ tên.
  - In danh sách sinh viên.
  - In danh sách theo loại SV.

3. Lớp Hashtable

- Lớp Hashtable dùng để quản lý những danh sách bằng bảng băm, phục vụ cho truy xuất nhanh.
- Một phần tử trong Hashtable gồm 2 thành phần: Key và Value.
- Phương thức khởi tạo:

  Hashtable()

  Hashtable(int initialCapacity)

  Hashtable(int initialCapacity, float factor)

3. Lớp Hashtable (tiếp)

- Một số phương thức:

  Object put(Object key, Object value)

  boolean contains(Object obj)

  boolean containsKey(Object key)

  Object get(Object key)

  boolean isEmpty()

  Object remove(Object key)

3. Lớp Hashtable (tiếp)

int size()

void clear()

String toString()

Enumeration elements()

Enumeration keys()

- Ví dụ: Chương trình minh họa thao tác tra từ trên từ điển dùng Hashtable.

```
import java.util.*;
class HashtableExample
{
    public static void main(String args[])
    {
        Hashtable tdav = new Hashtable();
        tdav.put("Hello","Chao");
        tdav.put("Begin","Bat dau");
        tdav.put("End","Ket thuc");
        //Tim tu trong tu dien
        String tu="Begin";
        if (tdav.containsKey(tu))
          System.out.println(tu+" : "+tdav.get(tu));
        else
          System.out.println("Khong co tu:"+tu);
    }
}
```

Luyện tập

- Tạo một từ điển Anh-Việt cho biết nghĩa của từ, phân loại từ và ví dụ minh họa. Các chức năng bao gồm:
  - Tạo một từ điển rỗng, tạo từ điển từ một tệp văn bản hoặc tệp định kiểu.
  - Thêm một từ vào từ điển.
  - Tra từ
  - Liệt kê toàn bộ các từ của từ điển.

4. Lớp StringTokenizer

- Dùng để phân tách các xâu ký tự thành các thành phần đơn vị (token) dựa vào các ký tự ngăn cách.
- Phương thức khởi tạo:

  StringTokenizer(String str)

  StringTokenizer(String str, String delimit)

  StringTokenizer(String str, String delimit, boolean returnTokens)

4. Lớp StringToken (tiếp)

- Các phương thức:

  int countTokens()

  boolean hasMoreElements()

  boolean hasMoreTokens()

  Object nextElement()

  String nextToken()

- Ví dụ: chương trình liệt kê các từ trong một câu.

```
import java.util.*;
class StringTokenExample
{
    public static void main(String args[])
    {
    String s = "Day la mot dong van ban. Dong nay co the chua dau
    phay; dau cham cau;";
    StringTokenizer st=new StringTokenizer(s," ,.;");
    System.out.println("So tu:"+st.countTokens());
    while (st.hasMoreTokens()) {
            System.out.println(st.nextToken());
    }
}
```

Luyện tập

- Viết chương trình đếm số lần xuất hiện của từng từ trong một câu.
- Ví dụ cho câu “to be or not to be” thì kết quả:
- to: 2, be: 2, or: 1, not: 1

13. Lớp trong (Inner Class)

- Lớp trong là lớp được khai báo bên trong một lớp khác.
- Lớp trong được khai báo như các lớp thông thường.
- Ví dụ:

```
class OuterClass{
    class InnerClass{ ... }
    ...
}
```

13. Lớp trong (tiếp)

- Lớp trong được xem là sở hữu riêng của lớp ngoài.
- Chỉ được sử dụng trong phạm vi lớp ngoài.
- Lớp trong có thể thao tác các thành phần (thuộc tính, phương thức) của lớp ngoài.
- Ví dụ minh họa.

```
class Outer {
   int outer_x = 100;
   void test() {
   Inner inner = new Inner();
   inner.display();}
   class Inner {
         void display() {
         System.out.println("display: outer_x = " + outer_x); }
   }
}
class InnerClassDemo {
public static void main(String args[]) {
   Outer outer = new Outer();
   outer.test();}
}
```

13. Lớp trong (tiếp)

- Lớp ngoài không được truy xuất trực tiếp đến các thành phần của lớp trong.

```
class Outer {
    class Inner {
        int y = 10;
    }
    void showy() {
        System.out.println(y); // lỗi
    }
}
```

Bài thực hành số 6

- Xây dựng lớp DSSV bằng cách dùng Vector
- Tổ chức các lớp sao cho có thể xây dựng lớp chứa các đối tượng số nguyên và phân số và cho phép tính tổng, sắp xếp các số theo thứ tự tăng.
- Xây dựng một từ điển Anh-Việt trong đó một từ tiếng Anh có nhiều nghĩa tiếng Việt thuộc các loại từ khác nhau.