Phát Triển Ứng Dụng CSDL 2

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI NGÔN NGỮ JAVA

GVLT: LNHNAM
BMHTTT- **ĐHKHTN TPHCM**

Khai báo lớp

- Quyền truy xuất lớp từ package khác →kèm theo tiền tố khi khai báo lớp

[public] class <tênlớp>{

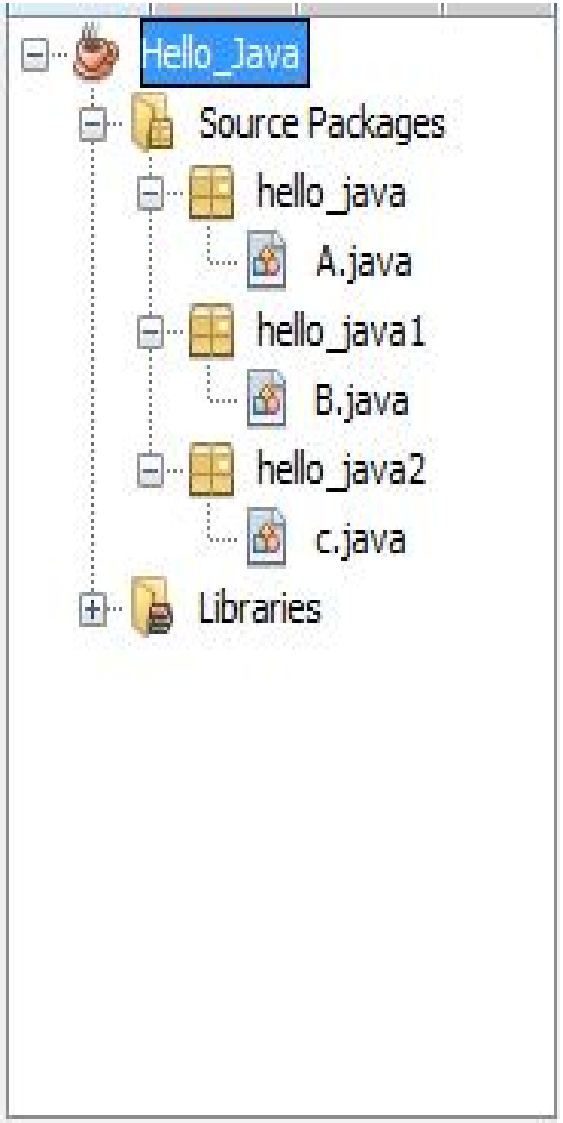
// khai báo các thuộc tính

// khai báo các phương thức

}

- public: lớp này có thể được sử dụng bên trong package khác

khai báo lớp




```java
package hello_java;
import hello_java1.*;
/**
 *
 * @author vinature
 */
public class A {
    private B b;
}
```

```java
package hello_java1;
import hello_java.*;
import hello_java2.*;

/**
 *
 * @author vinature
 */
public class B {
    private A a;
    private C c;
}
```

```java
package hello_java2;
import hello_java1.*;
import hello_java.*;
/**
 *
 * @author vinature
 */
class c {
    private A a;
    private B b;
}
```

Khai báo thuộc tính

- Quyền truy xuất thuộc tính từ đối tượng khác
  →kèm theo tiền tố khi khác báo thuộc tính

[public] class <tênlớp>{

[public][private][protected] <kiểudữliệu> <tênthuộc tính>

……..

// khai báo các phương thức

……..

}

Khai báo phương thức

- Quyền truy xuất phương thức từ đối tượng khác →kèm theo tiền tố khi khác báo phương thức

class <tênlop>{

[public][private][protected] <kiểudữliệu> <tênthuộc tính>

……..

[public][private][protected] <kiểudữliệutrảvề> <tênphươngthức> (<kiểudữliệu> tênthamsố, <kiểudữliệu> tênthamsố,…) {

….

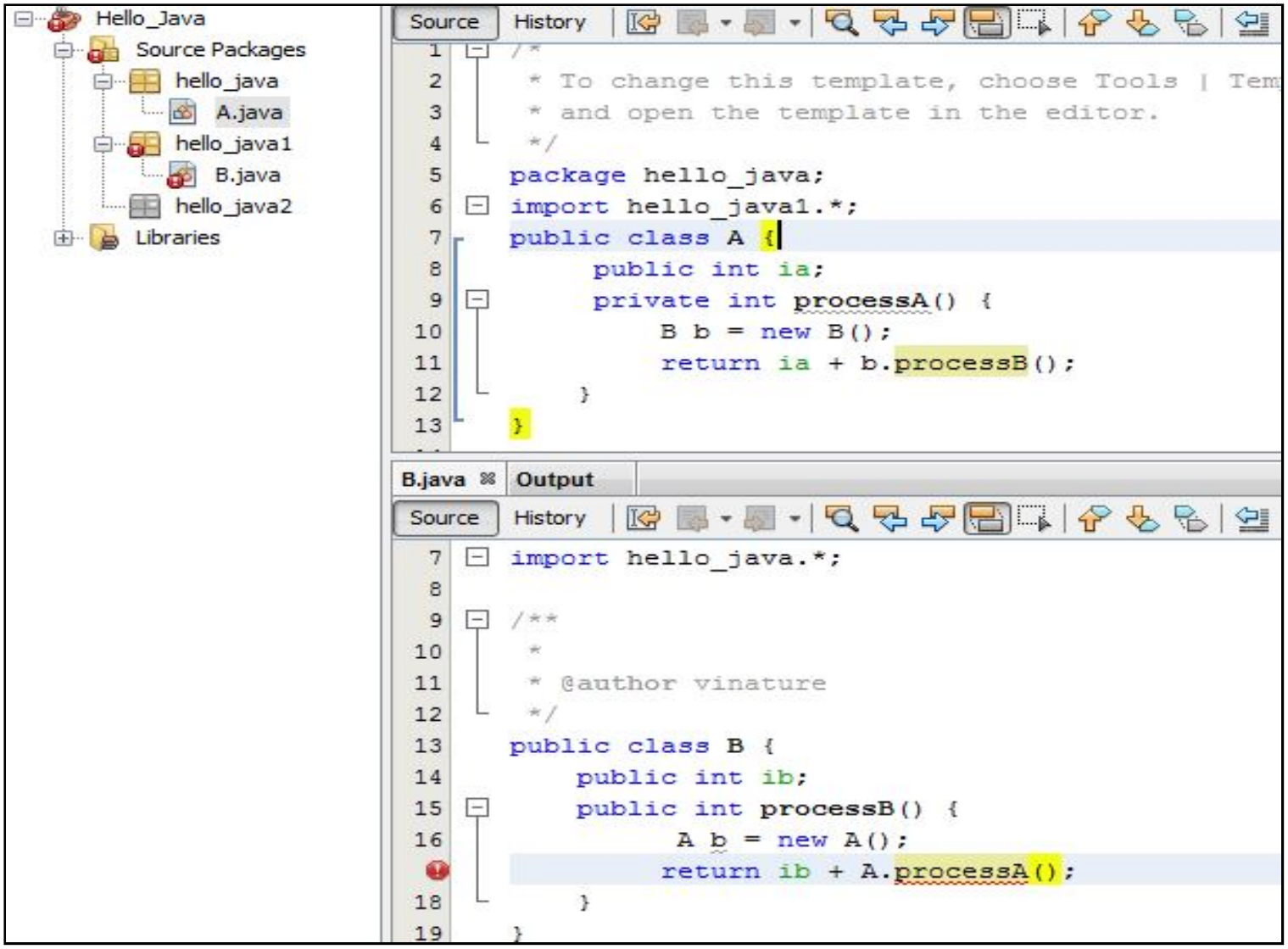
}

}

Phạm vi thành phần

- Private:
  - Chỉ cho phép truy xuất bên trong đối tượng.
- Public:
  - Không ràng buộc về phạm vị truy xuất
- Protected:
  - Đối với các lớp trong cùng package, không ràng buộc truy xuất
  - Đối với các lớp ngoài package, chỉ có lớp con mới truy xuất được.

Chương Trình Dịch



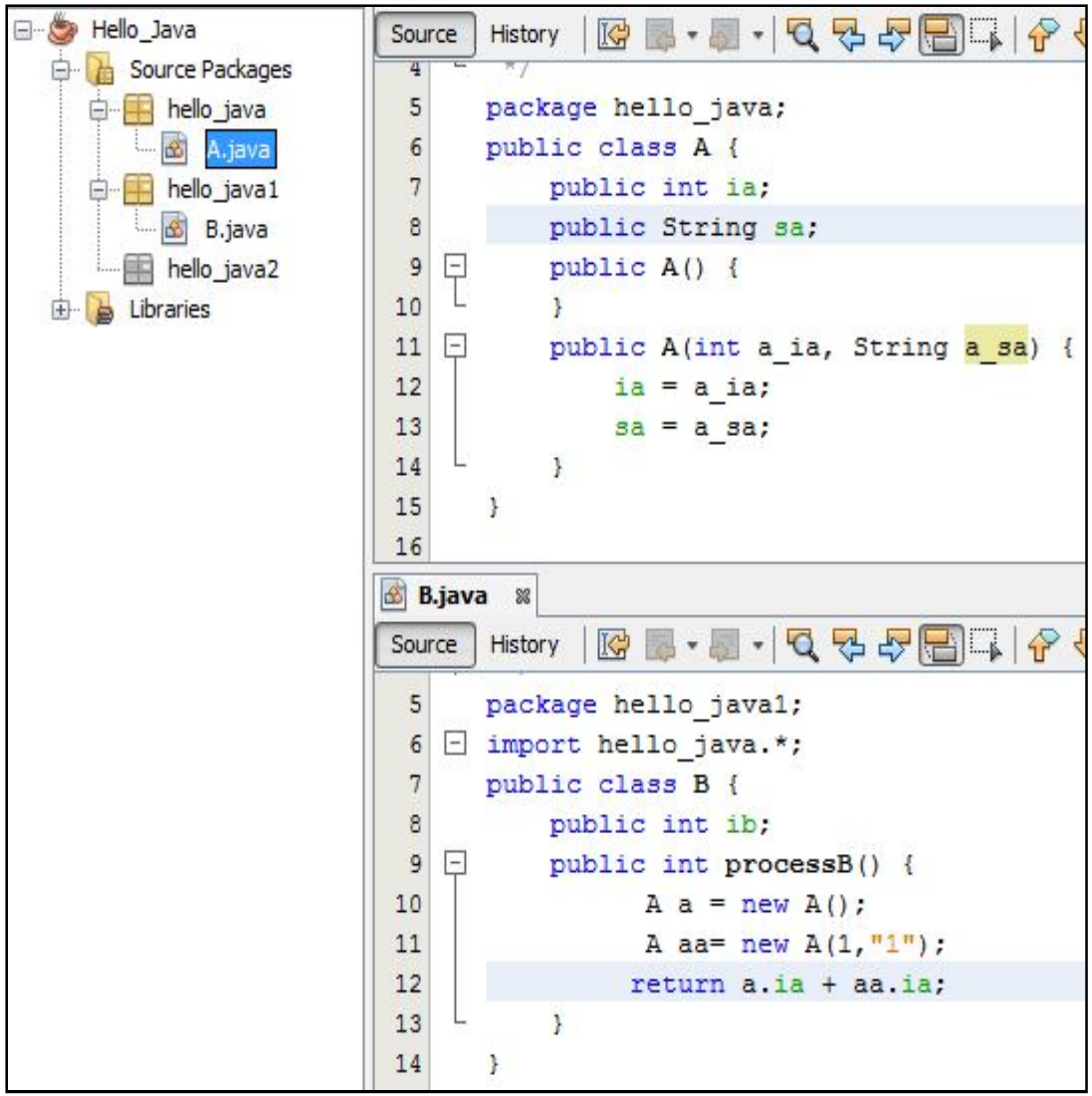
Nạp chồng phương thức (overload)

- Overload
  - Khai báo nhiều phương thức có cùng tên
  - Khác nhau về kiểu dữ liệu tham số, khác số lượng tham số.

```
public class A {
    public int ia;
    public String sa;
    public A() {
    }
    public A(int ia, String sa) {
        this.ia = ia;
        this.sa = sa;
    }
    public int processA() {
        A a = new A(1, "1");
        ia = 3;
        return ia + a.ia;
    }
    public String processA() {
        return sa;
    }
    public int processA(int b) {
        A a = new A(1, "1");
        ia = b;
        return ia + a.ia + b;
    }
    public String processA(String b) {
        return sa + b;
    }
}
```

Phương thức khởi tạo (constructor)

- Phương thức khởi tạo:
  - không có giá trị trả về
  - cùng tên với tên lớp
  - Có thể có tham số hoặc không có tham số
  - Được gọi đến khi dùng từ khóa new.
  - java sẽ cung cấp cho lớp một constructor mặc định nhưng nếu định nghĩa 1 phương thức khởi tạo, phương thức khởi tạo mặc định sẽ không còn tác dụng



Từ khóa this

- Biến this
  - Một biến ẩn tồn tại trong tất cả các lớp trong ngôn ngữ java
  - Tham khảo đến bản thân lớp chứa nó

```
package hello_java;

public class A {
    public int ia;
    public String sa;
    public A() {
    }
    public A(int ia, String sa) {
        this.ia = ia;
        this.sa = sa;
    }
    public A(A a) {
        this.ia = a.ia;
        this.sa = a.sa;
    }
    public int processA() {
        A a = new A(1,"1");
        ia=3;
        A aa = new A(this);
        return aa.ia+a.ia;
    }

}
```

Thành phần tĩnh (static)

- Thành phần tĩnh
  - Khai báo bằng từ khóa static
  - Thành phần chung của lớp, không phải của bất kỳ đối tượng nào
  - Truy cập thành phần tĩnh có thể dùng: tênlớp.tênthànhphần
  - Bên trong phương thức tĩnh không được truy cập thuộc tính không tĩnh

```
public class HocSinh {
    private double toan;
    private double ly;
    private double hoa;
    private String hoten;
    public static double nguongGioi = 8.0;
    public static double nguongKha = 7.0;
    public static void KhoangCachNguongGioiKha() {
        System.out.println(nguongGioi - nguongKha);
    }
    public void XepLoai() {
        double diemtb = (toan + ly + hoa) / 3;
        if (diemtb >= nguongGioi) {
            System.out.println("gioi");
            return;
        }
        if (diemtb >= nguongKha) {
            System.out.println("kha");
            return;
        }
        System.out.println("khac");
    }
    public HocSinh(String hocten, double toan,
        double ly, double hoa) {
        this.hoten = hoten;
        this.toan = toan;
        this.ly = ly;
        this.hoa = hoa;
    }
}
```

Thành phần tĩnh (static)

```java
public class HocSinh {
    private double toan;
    private double ly;
    private double hoa;
    private String hoten;
    public static double nguongGioi = 8.0;
    public static double nguongKha = 7.0;
    public static void KhoangCachNguongGioiKha() {
        System.out.println(nguongGioi - nguongKha);
    }
    public void XepLoai() {
        double diemtb = (toan + ly + hoa) / 3;
        if (diemtb >= nguongGioi) {
            System.out.println("gioi");
            return;
        }
        if (diemtb >= nguongKha) {
            System.out.println("kha");
            return;
        }
        System.out.println("khac");
    }
    public HocSinh(String hocten, double toan,
        double ly, double hoa) {
        this.hoten = hoten;
        this.toan = toan;
        this.ly = ly;
        this.hoa = hoa;
    }
}
```

```java
package hello_java;
public class A {

    static void main() {
        HocSinh h = new HocSinh("A", 7, 8, 9);
        System.out.println(HocSinh.nguongGioi);
        System.out.println(HocSinh.nguongKha);
        HocSinh.KhoangCachNguongGioiKha();
        h.XepLoai();

        HocSinh.nguongGioi = 9;
        HocSinh.nguongKha = 8;
        h.XepLoai();
    }
}
```

Kế thừa

- Một lớp (lớp con/dẫn xuất) có thể kế thừa tất cả các thuộc tính và phương thức của một lớp khác (lớp cha/cơ sở):

```
class Lớpcha {
.....
}
class Lớpcon extends lớpcha {
        ......
}
```

→ Lớp con bao gồm:

- Thành phần được khai báo trong lớp con
- Thành phần được khai báo trong lớp cha
- Không truy cập thành phần được khai báo private trong lớp cha

Kế thừa

```
5   package hello_java;
6   public class SinhVien {
7       //Các phương thức và thuộc tính sinh viên
8   }
9
```

```
5   package hello_java1;
6   import hello_java.*;
7   public class SinhVienCaoDang extends SinhVien{
8       //Các phương thực và thuộc tính của sinh viên cao dẳng
9   }
10
```

Ghi đè phương thức (overide)

- Hiện tượng trong lớp cơ sở và lớp dẫn xuất có hai phương thức giống hệt nhau.

```
class lớpcha
{
        public void process(){…};
}
class lớpcon extends lớpcha
{
        public void process(){…};
}
```

Ghi đè phương thức (overide)

```java
package hello_java;
public class SinhVien {
    protected String hoTen;
    protected int soTc;
    protected double diemTb;
    protected String maSo;
    public boolean kiemTraTotNghiep()
    {
        if(soTc> 140 && diemTb >= 5)
            return true;
        return false;
    }

}
```

```java
package hello_java;
public class SinhVienHoanChinh extends SinhVien{
    public boolean kiemTraTotNghiep()
    {
        if(soTc> 110 && diemTb >= 6)
            return true;
        return false;
    }
}
```

```java
package hello_java1;
import hello_java.*;
public class SinhVienCaoDang extends SinhVien{
    public boolean kiemTraTotNghiep()
    {
        if(soTc> 100 && diemTb >= 6)
            return true;
        return false;
    }
}
```

Ghi đè phương thức (override)

- Khi có hiện tượng ghi đè, phương thức bị ghi đè bên trong lớp cơ sở sẽ bị ẩn đi bên trong dẫn xuất

→ Dùng từ khóa super đề truy cập phương thức bị ghi đè của lớp cơ sở

```
Class lớpcha
{
        public void process1(){…}
}
class lớpcon extends lớpcha
{
        public void process1(){…super.process1()…}
        public void process2(){…super.process1()…}
}
```

- Nếu phương thức ơ lớp cơ sở đã bị overload thì không được override ở lớp dẫn xuất

Tứ khóa Final

- Final public void TênPhươngThức(){…}

→Phương thức không được phép override ở lớp dẫn xuất

```
5   package hello_java;
    public class SinhVien {
7       private String hoTen;
8       protected int soTc;
9       protected double diemTb;
10      protected String maSo;
11      final public String getTen()
12      {
13          return hoTen;
14      }
15      final public void setTen(String hoTen)
16      {
17          this.hoTen=hoTen;
18      }
        public boolean kiemTraTotNghiep()
20      {
            if(soTc> 140 && diemTb >= 5)
22              return true;
23          return false;
24      }
25
26  }
27
```

```
1   package hello_java;
2   public class SinhVienHoanChinh extends SinhVien{
        public boolean kiemTraTotNghiep()
4       {
5           if(soTc> 110 && diemTb >= 6)
6               return true;
7           return false;
8       }
        public String getTen()
10      {
11          return "";
12      }
        public void setTen(String hoTen)
14      {
15
16      }
17  }
```

Tứ khóa Final

- Final class tênlớp {...}

→Lớp không được phép kế thừa

```
5   package hello_java;
6   public final class SinhVien {
7       private String hoTen;
8       protected int soTc;
9       protected double diemTb;
10      protected String maSo;
11      public String getTen()
12      {
13          return hoTen;
14      }
15      public void setTen(String hoTen)
16      {
17          this.hoTen=hoTen;
18      }
19      public boolean kiemTraTotNghiep()
20      {
21          if(soTc> 140 && diemTb >= 5)
22              return true;
23          return false;
24      }
25
26  }
27
```

```
1   package hello_java;
2   public class SinhVienHoanChinh extends SinhVien{
3       public boolean kiemTraTotNghiep()
4       {
5           if(soTc> 110 && diemTb >= 6)
6               return true;
7           return false;
8       }
9   }
10
```

Lớp trừu tượng (abstract class)

- Phương thức trừu tượng là phương thức không có cài đặt

  public abstract tênphươngthức(…);

- Lớp trừu tượng là lớp chứa phương thức có chứa phương thức trừu tượng hoặc phương thưc có cài đặt

```
public abstract class tênlớp
{
        public void process1(...){....};
        public abstract void process1(...);
 }
```

Lớp trừu tượng (abstract class)

- Không được phép tạo một đối tượng của lớp trừu tượng
- Lớp trừu tượng có thể được sử dụng để giữ tham chiếu của các lớp dẫn xuất

```
public abstract class tênlớpA {
        public abstract void process1(...);
 }
Public class tênlớpB extends tênlớpA {
        public void process1(...){....};
```

→ Không hợp lệ: *tênlớpA t = new tênlớpA();*

→Hợp lệ: *tênlớpA t = new tênlớpB();*

Giao diện (interface)

- Giao diện là một lớp cơ sở thuần ảo dùng làm khuôn mẫu hoàn toàn cho các lớp dẫn xuất hiện thực
- Giao diện chỉ bao gồm
  - Hằng
  - Các phương thức trừu tượng (không cài đặt)

```
public interface têngiaodiện{
    public static final double tênhằng1=3.14;
    public static final String tênhằng2="PI";
    public void tênphươngthức1(...);
    public void tênphươngthức2(...);
}
```

Giao diện (interface)

- Dùng từ khóa implements để hiện thực hóa giao diện
- Một lớp hiện hóa giao diện phải hiện thực tất cả các phương thức trừu tượng của giao diện đó

```
public interface Hinh{
   public static final double PI=3.14;
   public void tinhDienTich();
   public void  tinhChuVi();
}
public class HinhTamGiac implements Hinh{
   public double tinhDienTich(){ ... }
   public double tinhChuVi(){.....}
 }
```

Đa Hình Thái

- Nhiều phương thức bị override ở nhiều cấp cấp bậc khác nhau

→Quyết định trong lúc runtime phương thức nào sẽ được thực thi.

```
Class A_Object
{
      void method_1() { ...};
}
Class B_Object extendsA_Object
{
       void method_1() { ...} ;
}
```

```
 public static void main(String[] args)
{
    A_Object arr_Object = new A_Object[2];
    arr_Object[0] = new  B_Object();
    arr_Object[1] = new A_Object();
    for (int i=0; i<2; i++)
    {
            arr_Object[i] .method_1();
     }
}
```

Đa Hình Thái

- Các bước đa hình thái:
  - Xây dừng lớp cơ sở (lớp bình thường, lớp trìu tượng, giao diện)
  - Xây dựng các lớp dẫn xuất ghi đè các phương thức bình thường của lớp cơ sở, hiện thực chi tiết các phương thức trừu tượng
  - Thực hiện tạo ra đa hình thái:
    - Con trỏ lớp cơ sở có thể giữ địa chỉ lớp dẫn xuất và quyết định phương thức thực thi sẽ là phương thức đã ghi đè, hiện thực hóa ở lớp dẫn xuất

Đa Hình Thái

```java
package hello_java;
public abstract class SinhVien {
    private String hoTen;
    protected int soTc;
    protected double diemTb;
    protected String maSo;
    public final String getTen()
    {
        return hoTen;
    }
    public final void setTen(String hoTen)
    {
        this.hoTen=hoTen;
    }
    public abstract void kiemTraTotNghiep();
}
```

```java
package hello_java;

public class SinhVienHoanChinh extends SinhVien {

    public void kiemTraTotNghiep() {
        if (soTc > 110 && diemTb >= 6) {
            System.out.print("Ok");
        } else {
            System.out.print("Fail");
        }
    }
    public void capNhapmaSo(String maNhom) {
        maSo = maSo +"_"+ maNhom;
    }
}
```

```java
package hello_java1;

import hello_java.*;

public class SinhVienCaoDang extends SinhVien {

    public void kiemTraTotNghiep() {
        if (soTc > 100 && diemTb >= 6) {
            System.out.print("Ok");
        } else {
            System.out.print("Fail");
        }
    }
}
```

Đa Hình Thái

```
5   package hello_java1;
6
7   import hello_java.*;
8
9   public class B {
10
11      public static void main() {
12          SinhVien arrobject[] = new SinhVien[2];
13          arrobject[0] = new SinhVienCaoDang();
14          arrobject[1] = new SinhVienHoanChinh();
15
16
17          for (int i = 0; i < 2; i++) {   //gọi phương thực đã ghi đè của lớp dẫn xuất
18              arrobject[i].kiemTraTotNghiep();
19          }
20          //gọi phương thức riêng của lớp dẫn xuất
21          SinhVienHoanChinh s = (SinhVienHoanChinh) arrobject[1];
22          s.capNhapmaSo("1");
23      }
24  }
25
```