1 面向对象

封装、继承、多态

什么是封装?

在java中，使用权限访问修饰符对类的成员进行控制。

权限访问修饰符: public、protected、default、private

- 属性私有化
- 提供属性访问器

```
package com.highcom;

public class Person {

    //封装第一步:属性私有化
    private String pid;
    private String pname;
    private int page;
    //封装第二步:提供属性访问器  get、set
    public String getPid() {
        return pid;
    }
```

```
    public void setPid(String pid) {
        this.pid = pid;
    }
    public String getPname() {
        return pname;
    }
    public void setPname(String pname) {
        this.pname = pname;
    }
    public int getPage() {
        return page;
    }
    public void setPage(int page) {
        if(page>=0 && page<=150){
            this.page = page;
        }else{
            System.out.println("您的年龄超出范
围!");
        }
    }
    //在类中，成员方法
    public void playGame(String s){
        System.out.println(this.getPname()+",
会玩"+s+"!");
    }

}
```

什么是继承?

子类继承父类，子类可以使用父类的所有属性和方法。

类和类之间的关系：单继承关系，但是可以实现多层继承。

继承的语法：

```
public class 子类名 extends 父类名{



}
```

继承满足一个规则： is a

继承的好处：代码的复用性

```
package com.highcom;

public class Animal {

    private String name;
    private int age;
    private String color;

    public Animal(){
```

```
        System.out.println("这是父类构造器");
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getColor() {
        return color;
    }
    public void setColor(String color) {
        this.color = color;
    }
}
```

```
package com.highcom;

public class Cat extends Animal{

    public Cat(){
        System.out.println("这是子类构造器");
    }
}
```

```
package com.highcom;

public class TestCat {

    public static void main(String[] args) {
        //子类实例化的过程?
        Cat c = new Cat();
        c.setName("小强");
        c.setAge(2);
        c.setColor("红");

System.out.println(c.getName()+","+c.getAge()
+","+c.getColor());
    }
}
```

权限访问修饰符：

	private	default	protected	public
同一个类下	√	√	√	√
同一个包下的类		√	√	√
不同包中的子类			√	√
其他包中的类				√

super和this的使用：

super:

super. :调用父类的属性和方法 : 写在子类的构造器和成员方法中.

super()：调用父类的构造器:只能写在子类构造器的第一行.

this

this. :调用本类的属性和方法 : 写在本类的构造器和成员方法中.

this()：调用本类的构造器:只能写在本类构造器的第一行.

什么是抽象类?

在java中，用abstract修饰的类就是抽象类.

作用：天生就是作为父类存在的.

原因：因为可能有多个子类实现父类的方法是不一样的，要把这样的方法定义成抽象方法。

定义抽象方法的格式：

```
public abstract 返回值类型 方法名（参数列表）;
```

什么是接口?

在java中，就是功能的集合.

接口和接口的关系：多继承

```
interface 接口名1{}
interface 接口名2{}
interface 接口名3 extends 接口名1，接口名2{}
```

接口和类的关系：实现

语法:

```
public class 实现类名 implements 接口名，接口名..
{

    重写接口中的抽象方法;
}
```

开发中常见使用语法:

```
public class 子类名 extends 父类名 implements 接口,接口{

}
```