面向对象的三大特征

封装

通过类把一些细节和私有信息隐藏起来，暴露一些必要方法给用户使用。为了实现低耦合，高内聚

所有属性一般都是private，属性的修改都要通过方法来实现，提供给用户使用的方法才用public， 内部使用的方法也要private

访问控制符使用原则：

- 局部变量不用修饰符
- 属性除static之外，尽量用private，内部使用的方法也用private
- 要被继承，又不想公开给其它类调用的方法，用protected
- 其它的方法用public，构造器一般也有public

继承

父类的方法子类可以直接调用，为了实现代码重用，缩小修改代码范围。

关键字extends， Java里只能继承一个父类（单继承），但是可以使用所有超类（父类的父类...）的方法。java里面每个类都默认继承Object，Object是所有类根类。

- 重写（覆盖）override父类的方法

方法名跟父类方法一样，参数列表一样，返回类型一样； 修饰符比父类方法更大；返回类型一样； 抛出异常比父类小

重写后的方法覆盖父类方法, 如果要调用父类的方法，用super.xxx()

```
//父类的方法
public int grow(String water) {
        System.out.println("父类生长");
        return 10;
    }



    /*
     * 方法名跟父类方法一样，参数列表一样，返回类型一样；  修饰符比父类方法更大； 抛出异常比父类小
     * 重写后的方法覆盖父类方法
     */
    public int grow(String water) {
        super.grow(""); //用super调用父类的方法，它可以放在任意位置
        System.out.println("苹果树的生长");

        return 0;
    }
```

多态

类型定义和赋值可以是不同类，把子类实例赋给父类变量自动转换类型（向上造型）；把父类实例赋给子类变量，需要强制类型转换（向下造型），如果转换不成，会报类型转换失败异常。

向上造型是自动的；向下造型需要强制类型转换，只有原类型向上造型后再向下造型才能成功。

为了程序更加健壮，我们在向下造型的时候可以先用instanceof来判断，再造型

```
if(apple instanceof Grape) {//instanceof判断一个对象的类型，返回布尔值
            ((Grape)apple).getSeeds();
        }
```

继承与组合

继承一般用于扩展方法时候，不能随意用，否则破坏封装特性。有些类不了不被继承，类的修饰符会加上final，比如String

在一个类里面定义其它类的属性，叫做组合

```
public class Car {
    //在一个类里面定义其它类的属性，叫做组合
    private Engin engin;
    private Wheel wheel;

    public Car(Engin engin, Wheel wheel) {
        super();
        this.engin = engin;
        this.wheel = wheel;
    }

    public void drive() {
        engin.start();
        wheel.run();
    }

    public static void main(String[] args) {
        Car car = new Car(new Engin(), new Wheel());
        car.drive();
    }
}
```

继承关系是is-a;　组合关系has-a

Object类的方法

==和equals（）

- ==对于基本类型判断数值；对于引用类型判断地址

```
        Person person1 = new Person("123456789123456789","Tom","男",22);
        Person person2 = new Person("123456789123456789","Tom","男",22);
        Person person3 = person1;
        System.out.println(person1 == person2);//false,   ==对于基本类型判断数值；对于引用类型判断地址
        System.out.println(person1 == person3);//true


        int i=65;
        float f = 65.0F;
        char c = 'A';
        System.out.println(i == f);//true
        System.out.println(f == c);//true
```

- 引用类型，判断内容是否相等，用重写后的equals

```
public class Person {
    private String id;
    private String name;
    private String sex;
    private int age;

    public Person(String id, String name, String sex, int age) {
        super();
        this.id = id;
        this.name = name;
        this.sex = sex;
        this.age = age;
    }

    public String getId() {
        return id;
    }

    @Override
    public boolean equals(Object obj) {
        if(((Person)obj).getId().equals(id)) {
            return true;
        }else {
            return false;
        }
    }

}
```

- 字符串字面量直接赋值，如果内容相同，实际是常量池中的同一个对象

```
        //指向常量池中的同一个对象
        String str1 = "abc";
        String str2 = "abc";
        System.out.println("相同的字面量赋值的字符串，equals判断：" +
str1.equals(str2));
        System.out.println("相同的字面量赋值的字符串，==equals判断：" + (str1==str2));

        //地址不同，内容相同
        String str3 = new String("abc");
        String str4 = new String("abc");
        System.out.println("new的相同内容字符串，equals判断：" + str3.equals(str4));
        System.out.println("new的相同内容字符串，==equals判断：" + (str3==str4));
```

toString

如果用Print语句打印一个对象，默认会调用toString方法内容供打印，Object类的toString返回"类名@hashcode"，如果要打印对象属性相关的内容，需要重写toString()方法

```
    @Override
    public String toString() {
        return "Person [id=" + id + ", name=" + name + ", sex=" + sex + ", age="
+ age + "]";
    }
```

抽象类

有abstract修饰的方法叫抽象方法，抽象方法在继承必须添加实现（重写）

一旦有方法是抽象的，类也必须写成抽象类。 加上abstract修饰类叫抽象类，抽象类是不能new 实例对象

子类继承抽象类必须实现所有抽象方法

抽象类：

```
package com.hqyj;

/*加上abstract修饰类叫抽象类，抽象类是不能new 实例对象
 * 强迫继承该抽象类，然后用子类实例化
*/
public abstract class Shape {
    public abstract int calcArea();//有abstract修饰的方法叫抽象方法，抽象方法在继承必须
添加实现（重写）
}
```

子类：

```
package com.hqyj;

public class Rectangle extends Shape{
    private int length;
    private int width;

    public Rectangle() {
```

```
        super();
    }


    public Rectangle(int length, int width) {
        super();
        this.length = length;
        this.width = width;
    }


    @Override
    public int calcArea() {
        return length*width;
    }

}
```

今日作业

1. 面向对象的三大特征是什么？你是怎么理解三个特征的？
2. 组合练习。写一个窗户类Window, 它有通风； 再写一个门类Door，它有开门，关门的方法； 再组合Window和Door实现教室类ClassRoom, 教室有开门，关门，通风等功能。
3. equals练习。写Dog类， 重写equals和toString方法，创建多个内容相同和不相同的实例，用equals打印比较结果，打印每个实例的信息
4. 抽象类练习。写个抽象类Person，抽象方法讲话Speak(). 创建中国人，美国人等子类，继承Person。