

# 操作题

操作题，无需表现在作业答案中，自己琢磨和练习即可。（如有留下痕迹的必要，我会具体给出。）

1. 查看如下代码，回答问题，问题写在注释部分。该题目可以练习Debug的使用。

```
/**
 * 问题1: 思考注释一下面的代码能否放开?会不会报错?为了更好的代码可读性,
可以怎么做?
 * 问题2: 思考控制台输出的顺序,提供必要的文字说明解释代码为何如此执行。
 * 问题3: Homework building和Homework constructor会不会输出,为什么?
 */
public class Homework {
    static Student s = new Student();
    {
        System.out.println("Homework building");
        s = null;
    }
    public static void main(String[] args) {
        System.out.println("main");
        //注释一
        //System.out.println(s.age);
        Person p = new Person("刘备");
        System.out.println(p.name);
    }

    public Homework() {
        System.out.println("Homework constructor");
    }
}
class Person{
    {
        name = "赵云";
        System.out.println("Person building");
    }
    String name = "曹操";
    static Student s = new Student();

    public Person() {
    }

    public Person(String name) {
        System.out.println("Person constructor");
        this.name = name;
    }
}
```

```

}

class Student{
    int age = 10;
    {
        System.out.println("Student building");
        age = 20;
    }

    public Student(int age) {
        this.age = age;
    }

    public Student(){
        System.out.println("Student constructor");
    }
}

```

问题1：思考注释一下面的代码能否放开？会不会报错？为了更好的代码可读性，可以怎么做？

可以，不会报错。把 Person 中的静态成员变量定义在 Homework 的 main 方法中。  
如：

```

System.out.println("main");
Student s = new Student();
System.out.println(s.age);
Person p = new Person("刘备");
System.out.println(p.name);

```

问题2：思考控制台输出的顺序，提供必要的文字说明解释代码为何如此执行。

一开始 Homework 类的 main 方法进栈会启动类加载，Homework 类会加载，此时静态成员变量 s 会被 new 出来，然后因为 s 的 new 会导致 Student 类的加载，然后 age 被初始化成 10，接着构造代码块运行输出

(1) Student building;

然后再无参构造输出

(2) Student constructor;

然后回 main 输出 main；接着注释代码会输出

(2) 20；接着 new 一个 Person 先静态变量然后静态方法快然后构造快然后再无参构造器按顺序输出

(4) Student building

(5) Student constructor

(6) Person building

(7) Person constructor

再回 main 输出 p.name

(8) 曹操

问题3：Homework building 和 Homework constructor 会不会输出，为什么？

不会，没有 new Homework 实例，不会调用构造块和构造器。

# 简答题

- 以下简答题，直接将答案写在题目下面即可。（都是一些概念，虽然我们学得是技术，但
- 基本的概念还是需要记忆的）

## 1. 简述一下构造代码块和静态代码块的作用和执行时机。

构造代码块会在new对象的时候在类加载后，与非静态成员变量按照先后顺序，构造器方法之前执行  
静态代码块会在类加载的时候在静态成员变量自动初始化后，与其显式赋值按照先后顺序执行。

## 2. 总结目前为止，成员变量和静态成员变量的赋值以及顺序。

成员变量赋值：

- （1）类加载jvm的自动初始值
- （2）显示赋值与构造块：按代码先后顺序
- （3）构造器赋值最后
- （4）setter方法，调用的时候

静态成员变量：

- （1）类加载jvm的自动初始值
- （2）显示赋值与静态代码块：按代码先后顺序
- （3）构造器也可以赋值（new 对象最后），但一般不这么搞
- （4）setter方法可以，一般不这么搞

## 3. 描述一下各种访问权限修饰符的访问级别。（排除protected，因为没讲）

public：无权限限制

缺省：同包下的其他类及同类

private：同类下

## 4. 说一下自己对面向对象语言特性和面向对象开发思想的理解。（开放性问题，说出自己的理解即可）

一切皆可为对象，将方法和变量封装成一个个类

## 5. Getter/Setter方法的好处是什么？定义类时需要无脑加上Getter/Setter方法吗？

好处：（1）隐藏信息，实现细节（2）实现了专业的分工  
不需要，比如工具类不用

## 6. 封装的好处是什么？（从代码设计者和使用者角度分别阐述）

代码设计者可以自由发挥不受限制，使用者可以不用去管如何实现只管调用api。  
将变化隔离；便于使用；提高重用性；安全性。

# 编程题

## 导包相关练习

导包是很基本的操作，某些细节也需要注意，请完成以下练习：

```
// 在com.cskaoyan.a包下，定义一个名为MyClazz的类如下
public class MyClazz {
    public void hello() {
        System.out.println("a包");
    }
}

// 同时，在com.cskaoyan.b包下，一个类名也为MyClazz
public class MyClazz {
    public void hello() {
        System.out.println("b包");
    }
}

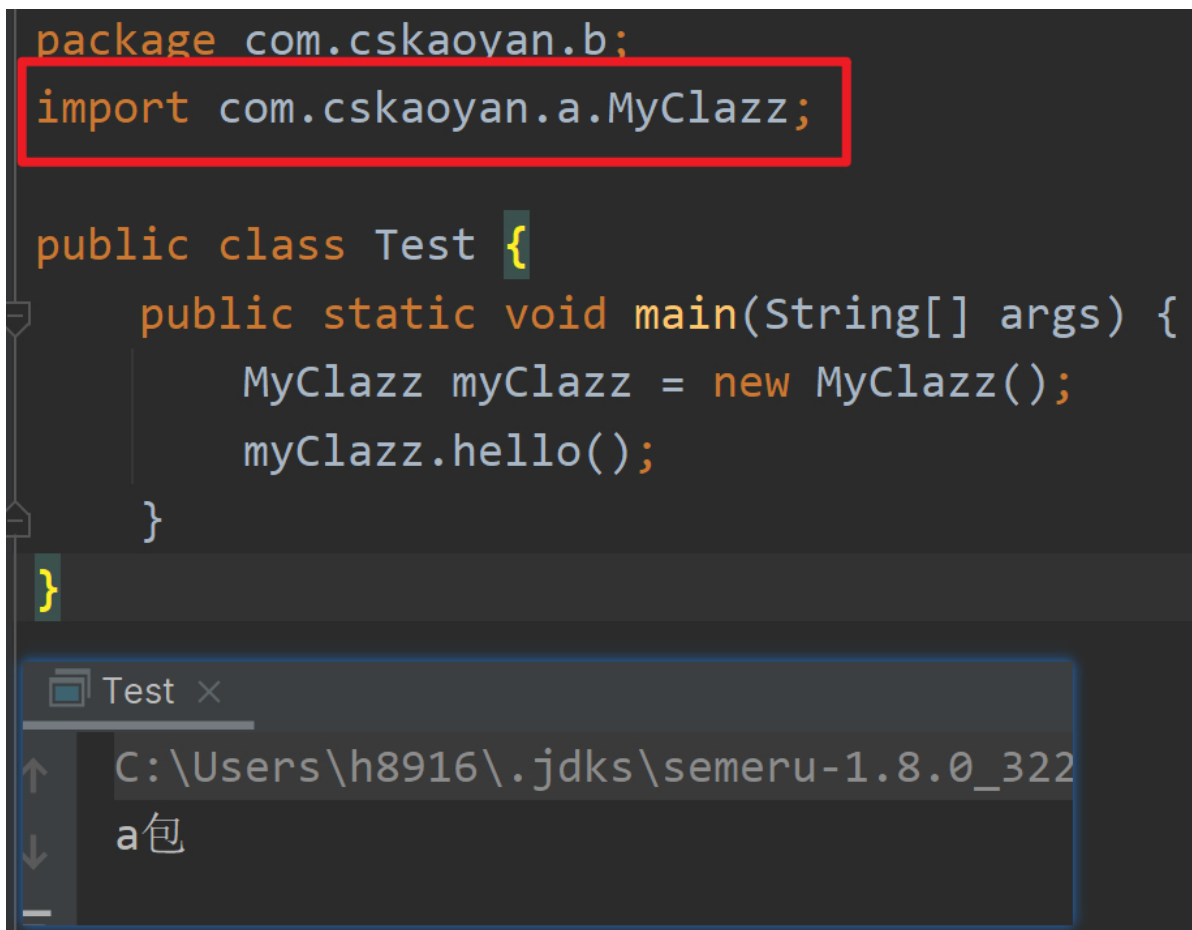
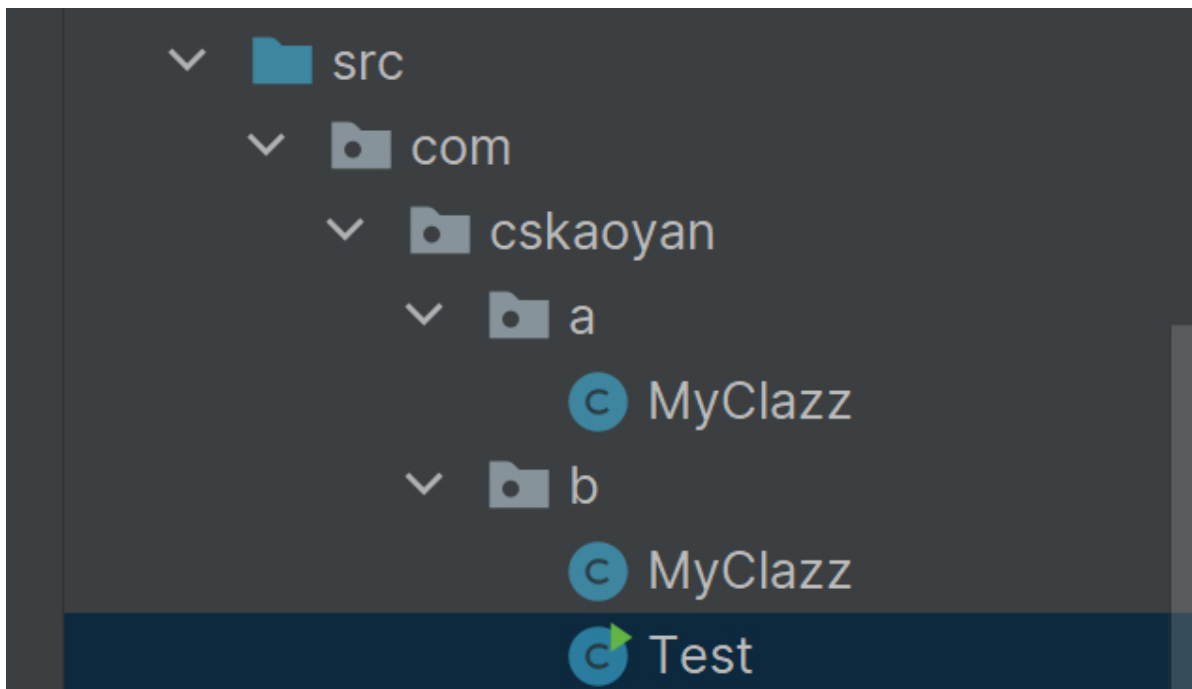
// 同时在com.cskaoyan.b包下定义一个Test类如下：
public class Test {
    public static void main(String[] args) {
        MyClazz myClazz = new MyClazz();
        myClazz.hello();
    }
}
```

首先根据上述注释，将三个类定义出来

毫无疑问，直接执行Test中的main方法会输出的是b包，思考以下问题（独立的三个问题）：

1. 不改变Test类中main方法的基础上，让main方法运行之后输出"a包"，应该怎么做？
2. 不做任何导包操作的基础上，修改main方法，让main方法运行之后同时输出"a包"和"b包"，应该怎么做？
3. 在Test类中添加导包语句import com.cskaoyan.a.\*，不修改任何代码，执行main方法，输出的是什么？为什么？

- 1.
2. 加一句：System.out.println("a包");
3. b包



```
package com.cskaoyan.b;

public class Test {
    public static void main(String[] args) {
        MyClazz myClazz = new MyClazz();
        System.out.println("a包");
        myClazz.hello();
    }
}
```

Test x

C:\Users\h8916\.jdk\semeru-1.8.0\_322\

a包

b包

```
package com.cskaoyan.b;
import com.cskaoyan.a.*;

public class Test {
    public static void main(String[] args) {
        MyClazz myClazz = new MyClazz();
        // System.out.println("a包");
        myClazz.hello();
    }
}
```

Test x

C:\Users\h8916\.jdk\semeru-1.8.0\_322\

b包

Process finished with exit code 0

# 工具类禁止创建对象

普遍来说，工具类不应该创建对象，应该私有化构造方法

```
public static void main(String[] args) {  
    ArrayTool a = new ArrayTool();  
    double[] numbers = getNumbers();  
}
```

D:\code\homework\src\com\cskaoyan\homework\day07\ToolTest.java:22:23  
java: ArrayTool() 在 com.cskaoyan.homework.day07.ArrayTool 中是 private 访问

需求一：在昨天自己的作业，数组工具类的基础上，私有化它的构造方法

需求二：

定义一个Scanner工具类ScannerUtils，提供以下方法：

- 1，键盘录入字符串
  - 2，键盘录入int整数
  - 3，键盘录入一个Person对象（Person类中有age和name属性）
- 不要忘记私有化构造器

注：录入对象，无非就是先录入成员变量取值，然后构造器创建对象。

需求一：

```
private ArrayTool(){  
}
```

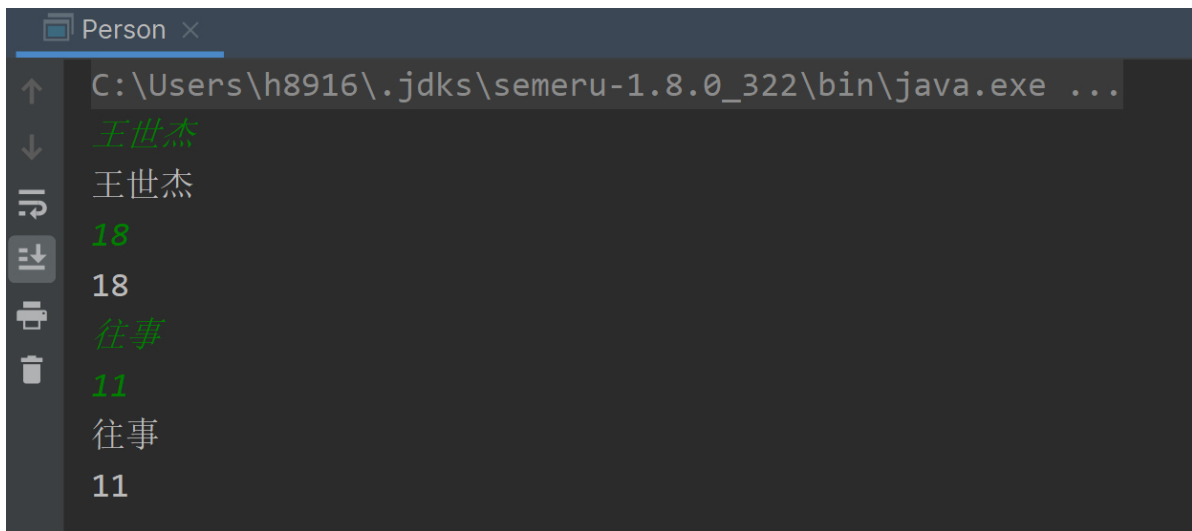
需求二：

```
package com.cskaoyan.homework.day07.arraytool;  
  
import java.util.Scanner;  
  
public class ScannerUtils {  
    private ScannerUtils() {  
    }  
  
    public static String nextString() {  
        return new Scanner(System.in).nextLine();  
    }  
  
    public static int nextInt() {  
        return new Scanner(System.in).nextInt();  
    }  
  
    public static Person nextPerson() {  
        return new Person(nextString(), nextInt());  
    }  
}
```

```
class Person {  
    private String name;  
    private int age;  
  
    public Person(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
  
    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 static void main(String[] args) {  
        String name = ScannerUtils.nextString();  
        System.out.println(name);  
        int age = ScannerUtils.nextInt();  
        System.out.println(age);  
        Person person = ScannerUtils.nextPerson();  
        System.out.println(person.getName());  
        System.out.println(person.getAge());  
    }  
}
```

```
class ArrayTool {  
    private ArrayTool(){  
    }  
  
    public static void printNum(double[] numbers) {  
        System.out.println(Arrays.toString(numbers));  
    }  
}
```





## 禁止创建对象

- 综合访问权限修饰符，static等知识点，做一个综合的练习。

定义一个Student类，并要求在其他类中，最多只能创建10个Student类的对象。

分析：

- 1，如果允许外部直接创建对象，显然无法控制创建对象的个数
- 2，需要计数器指示外部创建对象的个数

```
package com.cskaoyan.homework.day07.forbid;
```

```
/*定义一个Student类，并要求在其他类中，最多只能创建10个Student类的对象。
```

```
分析：
```

- 1，如果允许外部直接创建对象，显然无法控制创建对象的个数
- 2，需要计数器指示外部创建对象的个数

```
*/
```

```
public class Test {  
    public static void main(String[] args) {  
        for (int i = 0; i < 12; i++) {  
            System.out.println("创建对象: "+Student.getInstance());  
        }  
    }  
}
```

```
class Student {  
  
    private static int count = 1;  
  
    private String name;  
  
    private int age;
```

```

private Student() {

}

public static Student getInstance() {
    Student student = null;
    if (count <= 10) {
        student = new Student();
        System.out.println("第"+Student.count+"个学生");
        count++;
        return student;
    }
    System.out.println("失败");
    return student;
}
}

```

```

Test (2) x
第5个学生
创建对象: com.cskaoyan.homework.day07.forbid.Student@3c9486aa
第4个学生
创建对象: com.cskaoyan.homework.day07.forbid.Student@1716c313
第5个学生
创建对象: com.cskaoyan.homework.day07.forbid.Student@275f61ee
第6个学生
创建对象: com.cskaoyan.homework.day07.forbid.Student@daa86c64
第7个学生
创建对象: com.cskaoyan.homework.day07.forbid.Student@e130a15
第8个学生
创建对象: com.cskaoyan.homework.day07.forbid.Student@733dac00
第9个学生
创建对象: com.cskaoyan.homework.day07.forbid.Student@247eedbb
第10个学生
创建对象: com.cskaoyan.homework.day07.forbid.Student@979ecea
失败
创建对象: null
失败
创建对象: null

```

## Getter/Setter方法

- 当属性私有化，又有外界访问需求时，提供Getter/Setter方法

创建两个类，分别用来表示长方形和正方形。

同时定义所需的成员变量（边长），代表长方形或者正方形的边长（私有化成员变量，并提供相应的Getter/Setter方法，获取以及改变长方形和正方形的边长。）

然后在两个类中分别定义两个成员方法，用于求对应图形的面积和周长。

最后，写代码测试一下创建对象，方法调用等。

注：Getter/Setter方法可以选择手写一个，其余的用快捷键自动生成。

```
package com.cskaoyan.homework.day07.area;

public class Cal {
    public static void main(String[] args) {
        Rectangle rectangle1 = new Rectangle(1,2);
        System.out.print("第一个长方形：
长: "+rectangle1.getLength()+"宽: "+rectangle1.getWidth());
        System.out.println("第一个长方形面积: "+rectangle1.getArea()+"周
长为: "+rectangle1.getPerimeter());
        Rectangle rectangle2 = new Rectangle();
        rectangle2.setLength(1.2);
        rectangle2.setWidth(2.4);
        System.out.println("第二个长方形面积: "+rectangle2.getArea()+"周
长为: "+rectangle2.getPerimeter());

        Square square1 = new Square(22.22);
        System.out.print("第一个正方形的边
长: "+square1.getSideLength());
        System.out.println("第一个正方形面积: "+square1.getArea()+"周长
为: "+square1.getPerimeter());
        Square square2 = new Square();
        square2.setSideLength(11.11);
        System.out.print("第二个正方形的边
长: "+square2.getSideLength());
        System.out.println("第一个正方形面积: "+square2.getArea()+"周长
为: "+square2.getPerimeter());

    }
}

class Rectangle {
    private double length;
    private double width;

    public Rectangle() {
    }

    public Rectangle(double length, double width) {
```

```

        this.length = length;
        this.width = width;
    }

    public double getLength() {
        return length;
    }

    public void setLength(double length) {
        this.length = length;
    }

    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        this.width = width;
    }

    public double getArea() {
        return this.length * this.width;
    }

    public double getPerimeter() {
        return (this.length + this.width) * 2;
    }
}

class Square {
    private double sideLength;

    public Square() {
    }

    public Square(double sideLength) {
        this.sideLength = sideLength;
    }

    public double getSideLength() {
        return sideLength;
    }

    public void setSideLength(double sideLength) {
        this.sideLength = sideLength;
    }

    public double getArea() {

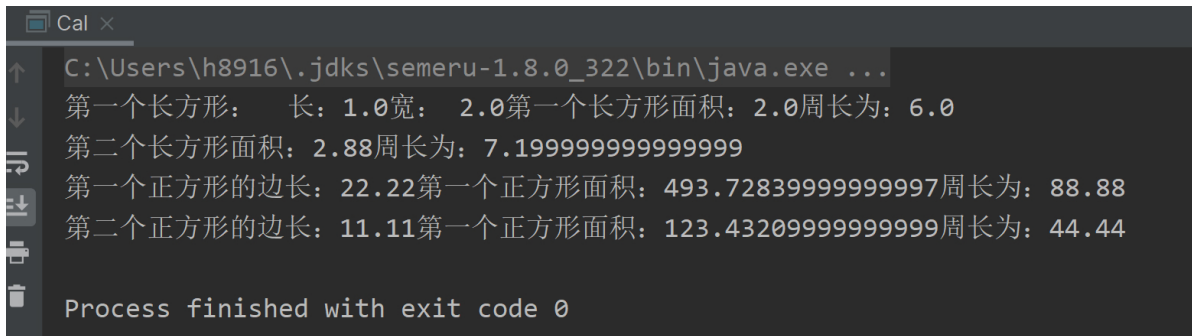
```

```

        return sideLength * sideLength;
    }

    public double getPerimeter() {
        return sideLength * 4;
    }
}

```



Cal x

```

C:\Users\h8916\.jdk\semeru-1.8.0_322\bin\java.exe ...
第一个长方形： 长： 1.0宽： 2.0第一个长方形面积： 2.0周长为： 6.0
第二个长方形面积： 2.88周长为： 7.199999999999999
第一个正方形的边长： 22.22第一个正方形面积： 493.72839999999997周长为： 88.88
第二个正方形的边长： 11.11第一个正方形面积： 123.43209999999999周长为： 44.44

Process finished with exit code 0

```

## Debug和代码执行顺序练习

读下列代码，指出代码的执行流程，并给出必要的说明。

```

public class TestStaticDemo {
    public static void main(String[] args) {
        staticMethod();
    }

    static TestStaticDemo ts = new TestStaticDemo();

    static {
        System.out.println("静态代码块");
    }

    {
        System.out.println("构造代码块");
    }

    public TestStaticDemo() {
        System.out.println("无参构造器");
        System.out.println("a=" + a + ",b=" + b);
    }

    public static void staticMethod() {
        System.out.println("静态成员方法");
    }

    int a = 666;
    static int b = 777;
    static TestStaticDemo ts = new TestStaticDemo();
}

```

做一下练习即可。

调用main方法前，类加载，静态成员变量显式赋值和静态代码块按顺序执行

```
static TestStaticDemo ts = new TestStaticDemo();
new 对象会 构造块和构造器按顺序执行：
{
    System.out.println("构造代码块"); ①
}

public TestStaticDemo() {
    System.out.println("无参构造器");②
    System.out.println("a=" + a + ",b=" + b); // a=666 //b=0 ③
}

static {
    System.out.println("静态代码块");④
}

static int b = 777;
static TestStaticDemo ts1 = new TestStaticDemo();
new 对象会 构造块和构造器按顺序执行：
{
    System.out.println("构造代码块"); ⑤
}

public TestStaticDemo() {
    System.out.println("无参构造器"); ⑥
    System.out.println("a=" + a + ",b=" + b); // a=666 //b=777 ⑦
}

最后执行静态方法:staticMethod();⑧
```

System.out.println("构造代码块"); ①	构造代码块
System.out.println("无参构造器");②	无参构造器
System.out.println("a=" + a + ",b=" + b); // a=666 //b=0 ③ a=666 ,b=0	
System.out.println("静态代码块");④	静态代码块
System.out.println("构造代码块"); ⑤	构造代码块
System.out.println("无参构造器"); ⑥	无参构造器
System.out.println("a=" + a + ",b=" + b); // a=666 //b=777 ⑦ a=666 ,b=0	
staticMethod();⑧	静态成员方法

