

动手动脑4

1、求平方数的静态类方法Square.java,不用static但仍想在main中调用的处理方法

```
public class SquareInt {  
  
    public static void main(String[] args) {  
        int result;  
  
        for (int x = 1; x <= 10; x++) {  
            result = square(x);  
            // Math库中也提供了求平方数的方法  
            // result=(int)Math.pow(x,2);  
            System.out.println("The square of " + x + " is " + result + "\n");  
        }  
    }  
  
    // 自定义求平方数的静态方法  
    public static int square(int y) {  
        return y * y;  
    }  
}
```

//类的对象实例化

//王荣荣 2016/10/16

```
public class SquareIntTest {  
    public static void main(String[] args) {  
        for (int x=1; x <= 10; x++) {  
            SquareIntTest obj;    //创建类的示例obj  
            obj=new SquareIntTest();  
            int result = obj.square(x);  
            // Math库中也提供了求平方数的方法  
            // result=(int)Math.pow(x,2);  
            System.out.println("The square of " + x + " is " + result + "\n");  
        }  
    }  
  
    // 自定义求平方数的静态方法  
    public int square(int y) {  
        return y * y;  
    }  
}
```

结果:

```
Console [X]
<terminated> SquareIntTest [Java Application]
The square of 1 is 1

The square of 2 is 4

The square of 3 is 9

The square of 4 is 16

The square of 5 is 25

The square of 6 is 36

The square of 7 is 49

The square of 8 is 64

The square of 9 is 81

The square of 10 is 100
```


2.编写一个方法，使用以下算法生成指定数目（比如**1000个**）的随机整数

//王荣荣 2016/10/15

```
import java.util.Random;
public class Random1000 {
    public static void main(String[] args)
    {
        Random r1 = new Random(1000);
        System.out.println("第一个种子为1000的Random对象");
        System.out.println("r1.nextBoolean():\t" + r1.nextBoolean());
        System.out.println("r1.nextInt():\t\t" + r1.nextInt());
        System.out.println("r1.nextDouble():\t" + r1.nextDouble());
        System.out.println("r1.nextGaussian():\t" + r1.nextGaussian());
        System.out.println("-----");
    }
}
```

```
}  
}
```

结果:

```
Console   
<terminated> Random1000 [Java Application] C:\f  
第一个种子为1000的Random对象  
r1.nextBoolean():    true  
r1.nextInt():        1060493871  
r1.nextDouble():    0.574836350385667  
r1.nextGaussian():   -0.719106498075259  
-----
```

3.请看以下代码，你发现了有什么特殊之处吗？

```
public class MethodOverload {  
  
    public static void main(String[] args) {  
        System.out.println("The square of integer 7 is " + square(7));  
        System.out.println("\nThe square of double 7.5 is " + square(7.5));  
    }  
  
    public static int square(int x) {  
        return x * x;  
    }  
  
    public static double square(double y) {  
        return y * y;  
    }  
}
```

结果:

Console

<terminated> MethodOverload [Java Application]

The square of integer 7 is 49

The square of double 7.5 is 56.25

上述示例代码展示了Java的“方法重载（overload）”特性。计算 7^2 时，调用的是整型

```
public static int square(int x) {  
  
    return x * x;  
  
}
```

而计算 7.5^2 时，调用的是双精度类型

```
    public static double square(double y) {  
  
    return y * y;  
  
}
```

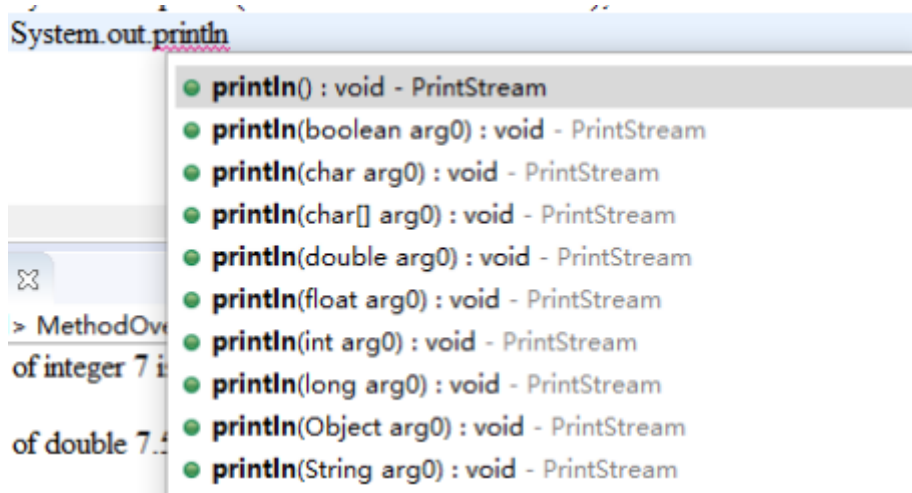
满足以下条件的两个或多个方法构成“重载”关系：

- （1）方法名相同；
- （2）参数类型不同，参数个数不同，或者是参数类型的顺序不同。

以上重载是参数类型不同。

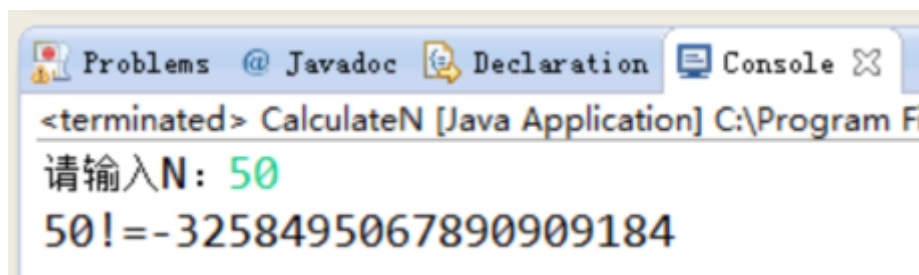
注意：方法的返回值不作为方法重载的判断条件。

4.查看一下JDK中System.out.println()方法，你发现了什么？



`System.out.println()`方法中实参表内可输入很多类型。

5. CalculateN 示例程序中的BUG，50! 出现负数



原因：由于计算机使用固定的位数来保存数值，因此，能处理的数值大小是有限的，当要处理的数值超过了这一范围时，计算机将会自动截断数值的二进制表示为它所能处理的最多位数，这将导致错误的处理结果。

6. 使用计算机计算组合数

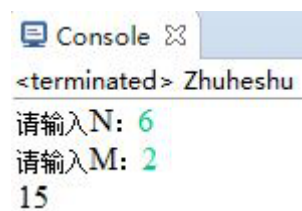
(1) 使用组合数公式利用 $n!$ 来计算

$$C_n^k = \frac{n!}{k!(n-k)!}$$

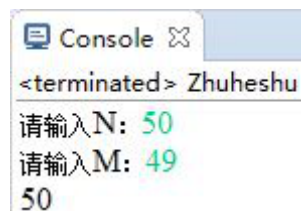
源代码:

```
import java.math.BigInteger;
import java.util.Scanner;
public class Zhuheshu {
    public static void main(String[] args) {
        System.out.print("请输入N: ");
        Scanner scanner=new Scanner(System.in);
        int n=scanner.nextInt();
        System.out.print("请输入M: ");
        Scanner scanner1=new Scanner(System.in);
        int k=scanner1.nextInt();
        System.out.println(ca(n).divide(ca(k).multiply(ca(n-k))));
    }
    public static BigInteger ca(int m) {
        if(m==1 || m==0){
            return BigInteger.valueOf(1);
        }
        return BigInteger.valueOf(m).multiply(ca((m-1)));
    }
}
```

结果:

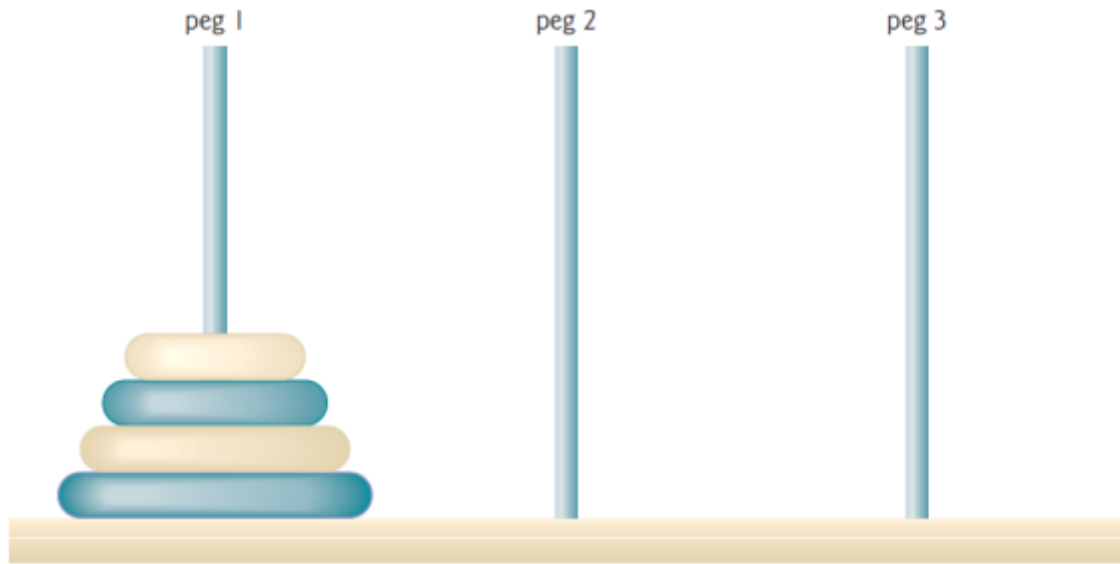


```
<terminated> Zhuheshu
请输入N: 6
请输入M: 2
15
```



```
<terminated> Zhuheshu
请输入N: 50
请输入M: 49
50
```

7.递归编程解决汉诺塔问题。用Java实现



源程序：

```
//用递归方式编程解决汉诺塔问题
//王荣荣 2016/10/16
import java.util.Scanner;
public class Hannuota {
    public static void main(String[] args){
        System.out.print("请输入盘子的个数：");
        Scanner scanner=new Scanner(System.in);
        int disks=scanner.nextInt();//盘子的个数
        final int source=1;//盘子的初始位置第一根柱子上
        final int desk=3;//盘子的最终位置第三根柱子上
        final int spare=2;//临时存放盘子的位置第二根柱子上
        move(disks, source, desk, spare);
    }
    public static void move(int disks, int source, int desk, int spare){
        if(disks==1)
            System.out.println(source+"->" +desk);
        else{
            move(disks-1, source, spare, desk);
            System.out.println(source+"->" +desk);
            move(disks-1, spare, desk, source);
        }
    }
}
```

结果:

```
Console
<terminated> Hannuota
请输入盘子的个数: 3
1->3
1->2
3->2
1->3
2->1
2->3
1->3
```

8.使用递归方式判断某个字串是否是回文 (palindrome)

```
//王荣荣2016/10/15
import java.util.Scanner;
public class Huiwen {
    public static void main(String[] args){
        String str="";
        System.out.println("请输入一个字符串:");
        Scanner in=new Scanner(System.in);
        str=in.nextLine();
        StringBuffer sb=new StringBuffer(str);
        sb.reverse();
        int n=0;
        for(int i=0;i<str.length();i++){
            if(str.charAt(i)==sb.charAt(i))
                n++;
        }
        if(n==str.length())
            System.out.println(str+"是回文!");
        else System.out.println(str+"不是回文!");
    }
}
```

结果:

Console

<terminated> Huiwen [Java .

请输入一个字符串:

jdkkj

jdkkj是回文!

Console

<terminated> Huiwen [Java ,

请输入一个字符串:

12321

12321是回文! |

Console

<terminated> Huiwen [Java /

请输入一个字符串:

fghjk

fghjk不是回文! |