方法的定义及使用

很多情况下是有可能需要重复执行一些代码的。方法在主类中定义，并且由主方法直接调用

1.方法的基本定义
本次方法是定义在主类之中并且由主方法直接调用的，所以方法的定义语法如下：
public static 返回值类型 方法名称([参数类型 变量,...]) {
//该方法主要执行的代码
[return [返回值] ;]
}

基本上可以定义数据类型，引用数据类型在方法之中可以返回数据的处理，如果要返回数据可以使用return（返回数据类型于方法的返回值类型相同，如果不返回数据，则该方法可以使用void进行声明）

范例：定义一个无参无返回值的方法
```
public class JavaStudy {
    public static void main(String mage[]) {
        printFANGFA() ; //执行方法调用
        printFANGFA() ; //执行方法调用
    }
    public static void printFANGFA() { //该方法包含了一行代码
        System.out.println("(**********") ;
        System.out.println("(*     我好帅啊，我为什么这么帅气  *") ;
        System.out.println("(**********") ;
    }
}
```

关于方法名称与变量定义命名要求：
方法名称定义的时候，首字母小写，而后每个字母大写
在进行变量名称定义的时候，也和方法名称一样要求首字母大写，而后每个字母小写。

方法的本质就是为了开发者可以重复调用，并且所有的程序一定都是通过主方法开始执行的。

范例：定义一个有参数有返回值的方法
```
public class JavaStudy {
    public static void main(String mage[]) {
        String result = get (20.0) ;
        System.out.println(result) ;
        System.out.println(get(1.0)) ; //返回值可以直接输出
    }
    public static String  get (double money) {
        if (money >= 10.0) {
            return "给你带了一份快餐，找零：" + (money - 10.0) ;
        } else {
                return "对不起。您的余额不足，请先充值，或者您选择简陋的吃下" ;
        }
    }
}
```

在进行方法定义的时候，如果方法的返回值类型位void，可以利用return来结束调用

范例：使用return结束方法

```
public class JavaStudy {
   public static void main(String mage[]) {
      sale(3) ;
      sale(-3) ;
   }
   public static void  sale (int money) {
      if (money <= 0) { //余额不足
         return ;
      }
      for (int x = 1 ; x  <=money ; x ++) {
         System.out.println("王见见开始笑了" + x + "次") ;
         }
      }
   }
```

总结：方法就是一个可以被调用的代码块而已，方法的代码内容建议不要太多。精简即可

2.方法重载

方法重载是一个重要概念，当方法名称相同，参数不同的时候就就称为方法重载

范例：采用方法重载进行定义

```
public class JavaStudy {
   public static void main(String mage[]) {
      int resultA = sum (10,20) ; //调用两个int参数方法
      int resultB = sum (10,20,30) ;
      double resultC = sum (10.2,20.3) ;
      System.out.println("加法执行结果：" + resultA) ;
      System.out.println("加法执行结果：" + resultB) ;
      System.out.println("加法执行结果：" + resultC) ;
   }
   public static int sum(int x,int y) {
      return x + y ;
   }
   public static int sum(int x,int y,int z) {
      return x + y + z ;
   }
   public static double sum(double x,double y) {
      return x + y ;
   }
}
```

该方法可以接收两个Int变量、三个Int变量。同一个方法名称，但是可以通过不同的参数或类型实现不同方法体的调用，这样就是方法重载的定义。

（方法的重载与方法的返回值类型没有任何关系，但他只跟函数有关系）

基本开发原则：

只要是方法重载建议其返回值类型相同

范例：观察一个程序代码

```
public class JavaStudy {
   public static void main(String mage[])  {
      System.out.println("1") ;
      System.out.println("1.1") ;
      System.out.println(true) ;
```

```
        System.out.printl("hello,world!") ;
    }
}
```

所有的输出操作支持各种数据类型所以：System.out.println(我本身就是个方法程序) ;

3.方法的递归调用

方法的递归调用就是一个方法自己调用自己，利用整个可以解决一些重复且麻烦的问题，

在进行递归调用的时候有以下问题：

一定要设置方法递归调用的结束条件；

每一次调用的过程之中一定要修改参数条件

范例：实现 1 ~ 100的累加

```
 public class JavaStudy {
    public static void main(String mage[]) {
        System.out.println(sum(100)) ;
    }
    public static int sum(int num) { //执行累加
        if (num == 1) { //不累加了
            return 1 ;
        }
        return num + sum(num - 1) ; //递归调用
    }
}
```

代码过程：

sum()：sum过程一共由两个方法发起，一个是主方法，一个是sum自己。return 100 + sum(99) ；

当第二次执行sum()、sum()递归调用：return 99 + sum(98) ；

.....

第九十九次执行sum()、sum()递归调用：return 2 + sum(1) ;

第一百次执行sum()、sum()递归调用return 1 ;

整体形势：return 100 + 99 + 98 + …….. +2 + 1

递归调用虽然很爽，但是会有很少的情况下出现有递归的情况（如果处理不当还会造成溢出（貌似写这玩意没溢出过的都不算是个优秀的程序员 > _< ！ ））

范例：计算 ”1 ！+ 2 ！+ 3! + 4! + 5! …. + 90!"

```
public class JavaStudy {
    public static void main(String mage[]) {
        System.out.println(sum(90)) ;
    }
    public static double sum(int num) {
        if (num == 1 ) {
            return 1 ;
        }
        return fan(num) + sum(num - 1) ;
    }
    public static double fan(int num) { //执行累加
        if (num == 2) { //不累加了
            return 1 ;
        }
        return num * fan(num - 1) ; //递归调用
    }
}
```

有一部分递归是可以通过循环完成的，但是使用递归可能会比循环；逻辑简单点。。

在进行阶乘计算的时候必须考虑数据类型。不能使用Int或是long，仅能够使用double