第四章 声明和使用方法

1. 方法的概述

代码块

```
package com.powernode.method02;

public class Test01 {
    public static void main(String[] args) {
        /**
         * 打印3行5列的矩形
         */
        /*for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 5; j++) {
                System.out.print("*");
            }
            System.out.println();
        }*/
        method();
        //想再打印一次3行5列的矩形
        /*for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 5; j++) {
                System.out.print("*");
            }
            System.out.println();
        }*/
        method();
    }
    public static void method(){
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 5; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

1. 方法是什么：程序员为一段特定的代码，做了一个封装，为这个封装起了一个特定的名称，叫“方法”

2. 有什么用：
    a. 一次封装，可以多次使用，减少了代码的重复编写
    b. 结构比较清晰
    c. 方便程序的扩展和维护

2. 方法的语法

代码块

```
[修饰符]  返回类型 方法名称([形参列表]){
    方法体;
}
1.[修饰符]
  1.可以省略的
  2.暂定为public static
2.返回类型
  1.基本类型:byte,short ,int ,long,float ,double ,char,boolean
  2.引用类型:除了基本类型都是引用类型
  3.void:  无返回类型（不可以有返回值），仅仅让方法执行的时候可以使用void类型
3.方法名称：自定义，首单词小写，其余单词首字母大写，比如：getNameBySno
4.[形参列表]
  1.可以不写（什么时候写，什么时候不写）
  2.形参就是变量的声明
```

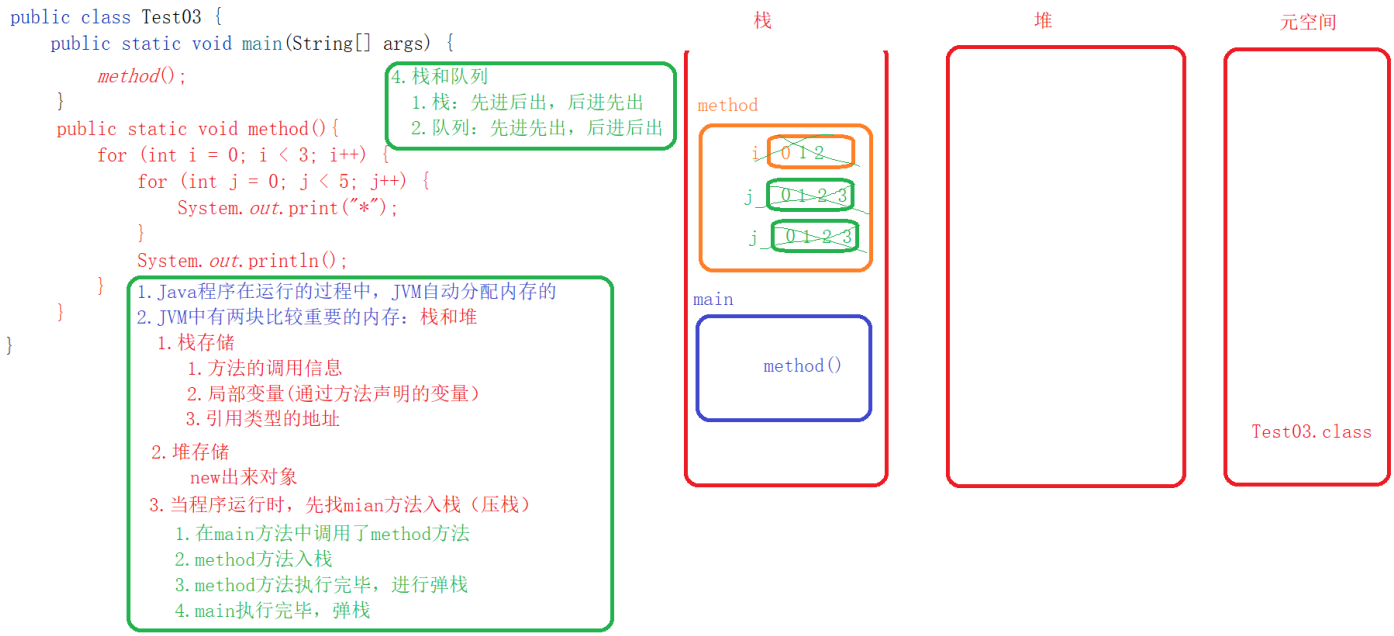
3. 方法的基本使用

代码块

```
package com.powernode.method02;

public class Test02 {
    public static void main(String[] args) {

        /**
         * 方法的调用:
         *   1.方法名称();
         *   2.main方法和自定义的方法，必须在同一个类
         *   3.定义的方法使用static修饰，且没有形参
         */
        System.out.println("------printRectangles方法即将执行------");
        printRectangles();
        System.out.println("------printRectangles方法执行完毕------");
    }
```

```java
    /**
     * [修饰符] 返回类型  方法名称([形参列表]){
     *     方法体;
     * }
     * 1.[修饰符] :  暂定为public static
     * 2.返回类型: void 仅仅让方法执行
     * 3.方法名称可以自定义
     * 4.[形参列表] 先不写
     * 5.注意:
     *   1.方法编写的位置
     *     1.写到类中，方法外面
     *     2.和main方法并列
     */
    public static void printRectangles(){
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 5; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

4. 方法调用的内存分析



代码块

```java
package com.powernode.method02;
```

```
2
3   public class Test03 {
4       public static void main(String[] args) {
5           /**
6            * 方法调用
7            */
8           method();
9       }
10      public static void method(){
11          for (int i = 0; i < 2; i++) {
12              for (int j = 0; j < 3; j++) {
13                  System.out.print("*");
14              }
15              System.out.println();
16          }
17      }
18
19  }
```

5. 在工作中方法的几种调用形式

5.1 无参数，无返回值

代码块

```
1   package com.powernode.method03;
2
3   public class Test01 {
4       public static void main(String[] args) {
5           //方法名称();
6           add();
7       }
8       //[修饰符] 返回类型 方法名称([形参类别]){方法体;}
9       //定义一个方法计算 3 + 2 的结果并输出
10      public static void add(){
11          int x = 2, y = 3;
12          int z = x + y;
13          System.out.println("z = " + z);
14      }
15  }
```

5.2 有参数，无返回值

```java
public class Test02 {
    public static void main(String[] args) {
        //方法名称([实参列表]);
        add( x: 10, y: 20);
    }
    //[修饰符] 返回类型 方法名称([形参类别]){方法体;}
    //定义一个方法:计算任意的两个整数相加
    //什么时候需要定义形参:方法内部,需要用到外界传入的数据,可以使用形参来接收
    public static void add(int x ,int y){//定义形参,用来接收两个整数
        int z = x + y;
        System.out.println("z = " + z);
    }
}
```

方法调用时注意事项:
1.个数要相等
2.实参的数据类型和形参数据类型匹配
3.实参使用来给形参赋值
4.按照数学的顺序赋值

代码块

```java
package com.powernode.method03;

public class Test02 {
    public static void main(String[] args) {
        //方法名称([实参列表]);
        add(10,20);
    }
    //[修饰符] 返回类型 方法名称([形参类别]){方法体;}
    //定义一个方法:计算任意的两个整数相加
    //什么时候需要定义形参:方法内部,需要用到外界传入的数据,可以使用形参来接收
    public static void add(int x ,int y){//定义形参,用来接收两个整数
        int z = x + y;
        System.out.println("z = " + z);
    }
}
```

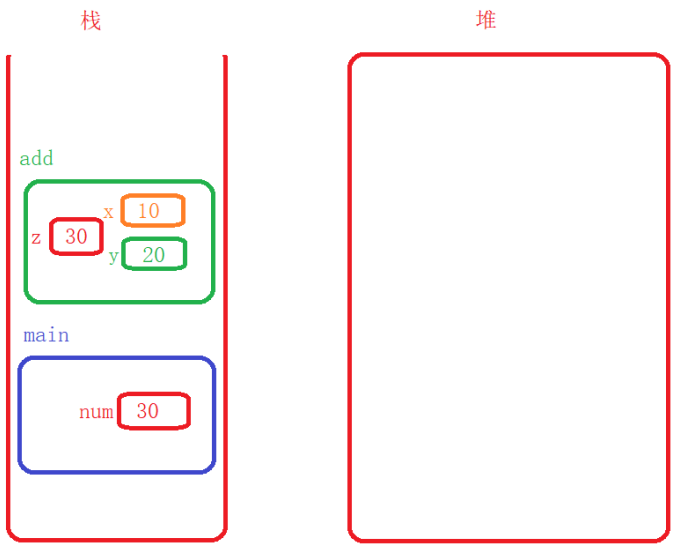
5.3 有参数,有返回值方法及内存分析

代码块

```java
package com.powernode.method03;

public class Test03 {
    public static void main(String[] args) {
        /**
         * 1.方法名称([实参列表])
         * 2.方法调用设计到形参和实参
         *   1.形参,定义方法时编写的参数,用来接收实参传递的值
         *   2.实参,给形参赋值
         */
        int num =  add(10,20);
```

```
12              System.out.println(num * num);
13          }
14          /**
15           * 1.声明一个方法，计算两个整数相加返回一个结果
16           * 2.拿到结果计算它的平方
17           * 3.方法声明
18           *   [修饰符] 返回类型 方法名称([形参列表]){
19           *       方法体;
20           *   }
21           *   1.[修饰符] :public staitc
22           *   2.返回类型
23           *       1.基本类型
24           *       2.引用类型
25           *       3.void ，仅仅让方法执行，不可以有返回值
26           *   3.返回类型注意事项
27           *       1.可以写成，返回值的【数据类型】
28           *       2.也可以写成，比返回值的【数据类型】 大的数据类型
29           *       3.总结：【返回类型】 >= 返回值的【数据类型】
30           *       4.通常情况下：会写成返回值的【数据类型】
31           *       5.没有返回值的情况，不可以写返回类型
32           *
33           */
34          public static int add(int x ,int y){
35              int z = x + y;
36              return z;//方法返回一个结果使用return，把z的值返回给调用者
37          }
38      }
```

```
package com.powernode.method03;

public class Test03 {
    public static void main(String[] args) {
        int num =  add(10,20);
        System.out.println(num * num);
    }

    public static int add(int x ,int y){
        int z = x + y;
        return z;
    }
}
```



5.4 无参数，有返回值

```java
package com.powernode.method03;

import com.sun.security.jgss.GSSUtil;

public class Test04 {
    public static void main(String[] args) {
        /*int x = square();
        System.out.println(x);*/
        System.out.println(square());
        //int num = method(); 方法没有返回值，不可以声明变量去接收
        //System.out.println(method());//无法解析方法'println(void)'
    }
    //计算3的平方并返回
    public static int square(){
        return 3 * 3;
    }
    public static void method(){
        System.out.println(3 * 3);
    }
}
```

5.5 方法调用的总结

1. 返回类型
    a. 返回类型的分类
        i. 基本类型
        ii. 引用类型
        iii. void
    b. 【返回类型】根据返回值的数据类型决定的
    c. void :仅仅让方法执行时，使用void
2. 数据类型关系链
    a. 返回值的【数据类型】 <= 【返回类型】
    b. 【返回类型】 <= 【接受类型】
3. 形参列表
    a. 方法内部如果需要用到外界传入的参数，就需要提供形参
    b. 形参：就是变量的声明
4. 实参和形参匹配
    a. 个数匹配

b. 数据类型匹配（直接可以赋值，不用强制转换）

c. 【实参数据类型】 <= 【形参数据类型】

5. void返回类型，不可以接收

6. continue，break和return

代码块

```
package com.powernode.method04;

public class Test01 {
    public static void main(String[] args) {
        /**
         * 1.continue,break,return
         *    1.continue:
         *      1.含义：继续，结束当前当次循环，继续下一次循环
         *      2.在哪里使用：循环
         *    2.break
         *      1.含义：终止或者中断
         *      2.在哪里使用：
         *          1.分支语句：表示结束分支语句
         *          2.循环：结束循环
         *    3.return
         *      1.含义：返回
         *      2.在方法中使用
         *        1.返回一个结果给调用者，并结束方法
         *        2.不返回任何信息结束方法
         * 2.三个关键字后都不可以写任何语句，因为没有机会执行
         *
         */
        testContinue();//结束一次循环
        testBreak();//结束整个循环
        testReturn();//结束整个方法
    }

    public static void testContinue() {
        //输出【1-6】之间2的倍数
        for (int i = 1; i <= 6; i++) {
            if (i % 2 != 0) {
                continue;//结束当前当次循环，继续下一次循环
            }
            System.out.println("i = " + i);
        }
        System.out.println("----------------");
    }
```

```
38      public static void testBreak() {
39          //输出【1-6】之间2的倍数
40          for (int i = 1; i <= 6; i++) {
41              if (i % 2 != 0) {
42                  break;//结束循环
43              }
44              System.out.println("i = " + i);
45          }
46          System.out.println("----------------");
47      }
48      public static void testReturn() {
49          //输出【1-6】之间2的倍数
50          for (int i = 1; i <= 6; i++) {
51              if (i % 2 != 0) {
52                  return;//结束方法
53              }
54              System.out.println("i = " + i);
55          }
56          System.out.println("----------------");
57      }
58  }
```

7. 方法重载（overload）

7.1 方法重载及其好处

代码块

```
1   package com.powernode.method04;
2
3   public class Test02 {
4       public static void main(String[] args) {
5           /**
6            * 1.方法重载（会背）
7            *    1.在同一个类中
8            *    2.方法名称相同
9            *    3.参数列表不同
10           *        1.个数不同
11           *        2.参数类型不同
12           *        3.顺序不同【数据类型顺序】
13           *  2.有什么好处：
14           *    1.方便方法声明（名称统一，不用起过多的名称）
15           *    2.方法他人,调用时不用记忆那么多的名称
16           */
17          add(2,3);
```

```
18          add(2.3,2.2);
19          add(2,3,4);
20          //典型的方法重载
21          System.out.println("abc");
22          System.out.println(123);
23          System.out.println(1.2);
24      }
25      public static void add(int x,int y){
26          System.out.println(x +y);
27      }
28
29      public static void add(int x,float y){
30          System.out.println(x +y);
31      }
32      public static void add(float x,int y){
33          System.out.println(x +y);
34      }
35
36
37      public static void add(double x,double y){
38          System.out.println(x +y);
39      }
40
41      public static void add(int x,int y,int z){
42          System.out.println(x + y + z);
43      }
44  }
```

7.2 方法的匹配规则

代码块

```
1   package com.powernode.method04;
2
3   public class Test03 {
4       public static void main(String[] args) {
5           /**
6            * 1.方法调用：其实就是实参给形参赋值：
7            * 2.匹配规则：
8            *      1.数据类型：精确匹配
9            *      2.数据类型：从    小   到     大
10           */
11          add(2,3);
12      }
13     /* public static void add(int x ,int y){
14          System.out.println("Test03.add(int x ,int y)");
```

```
    }*/
    public static void add(float x ,float y){
        System.out.println("Test03.add(float x ,float y)");
    }
    public static void add(double x ,double y){
        System.out.println("Test03.add(double x ,double y)");
    }

    /**
     * 1.如下代码注释掉，报错
     * 2.java对add方法的调用不明确
     *    add(float x ,float y) 和 add (int x ,double y)
     *    jvm分不清应该调用谁
     */
    /*public static void add(int x ,float y){
        System.out.println("Test03.add(int x ,float y)");
    }*/
    public static void add(int x ,double y){
        System.out.println("Test03.add(int x ,double y)");
    }



}
```

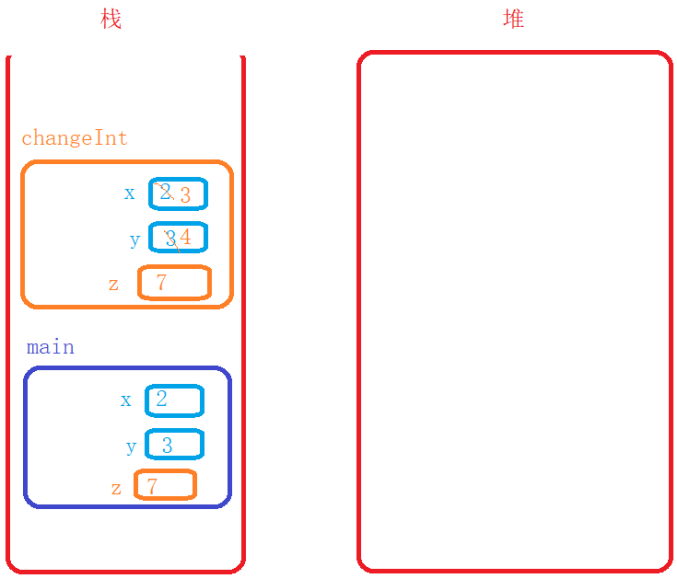
8. 方法中的基本类型值传递

代码块

```
package com.powernode.method04;

public class Test04 {
    public static void main(String[] args) {
       int x = 2, y = 3;
        System.out.println("x = " + x);//2
        System.out.println("y = " + y);//3
        int z = changeInt(x, y);
        System.out.println("x = " + x);//2
        System.out.println("y = " + y);//3
        System.out.println("z = " + z);//7
    }

    public static int changeInt(int x, int y) {
        x ++;
```

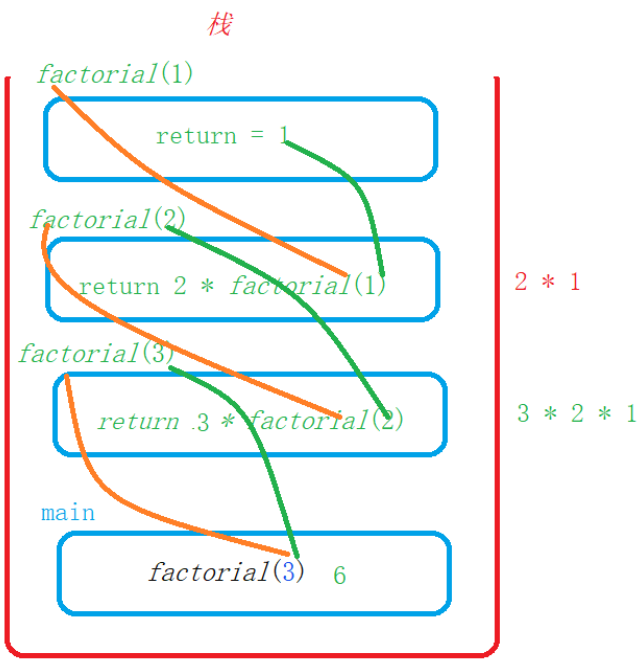
```
16            y ++;
17            int z = x + y;
18            System.out.println("x = " + x);//3
19            System.out.println("y = " + y);//4
20            System.out.println("z = " + z);//7
21            return z;
22        }
23
24
25    }
```

```
public class Test04 {
    public static void main(String[] args) {
       int x = 2, y = 3;
        System.out.println("x = " + x);//2
        System.out.println("y = " + y);//3
        int z = changeInt(x, y);
        System.out.println("x = " + x);//2
        System.out.println("y = " + y);//3
        System.out.println("z = " + z);//7
    }

    public static int changeInt(int x, int y) {
        x ++;
        y ++;
        int z = x + y;
        System.out.println("x = " + x);//3
        System.out.println("y = " + y);//4
        System.out.println("z = " + z);//7
        return z;
    }
}
```



9. 方法递归

```
package com.powernode.method05;

public class Test01 {
    public static void main(String[] args) {
        int factorial = factorial(3);
        System.out.println("factorial = " + factorial);
    }
    public static int factorial(int x){
        if (x == 1 || x == 0) {
            return 1;
        }
        return x * factorial(x - 1);
    }

}
```



代码块

```java
package com.powernode.method05;

public class Test01 {
    public static void main(String[] args) {
        /**
         * 1.需求：3!
         *   3! = 3 * 2 * 1
         *   0! = 1
         *   1!= 1
         * 2.递归:
         *   1.方法内部调用方法自己
         *   2.优点：代码简洁
         *   3.缺点：耗费内存
         * 3.递归可以理解为循环的一种替代形式
         */
        int factorial = factorial(3);
        System.out.println("factorial = " + factorial);
    }
    public static int factorial(int x){
        if (x == 1 || x == 0) {
            return 1;
        }
        return x * factorial(x - 1);
    }

}
```

10. StackOverflowError

代码块

```java
package com.powernode.method05;

public class Test02 {
    public static void main(String[] args) {
        method02();
    }
    public static void method01(){
        int i = 0;
        while(true){
            System.out.println(i++);
        }
```

```
    }
    //.StackOverflowError堆栈溢出错误
    public static void method02(){
        int i = 0;
        System.out.println(i++);
        method02();
    }
}
```

11. 夸类调用（*高内聚，低耦合*）

代码块

```
package com.powernode.method05;

public class Test03 {
    public static void main(String[] args) {
        /**
         * 1.软件的开发原则：高内聚，低耦合
         *    1.高内聚：方法，类或者模块处理的业务功能单一性比较高，称为高内聚
         *    2.低耦合：类与类之间，模块与模块之间的关系低，称为低耦合
         */
        /**
         * 跨类调用
         *   1.类名称.方法名称([实参列表])
         *   2.要求被调用的方法必须时static修饰的
         */
        int max = MyMath.max(2, 3);
        System.out.println(max);
        int add = MyMath.add(2, 3);
        System.out.println(add);
        insert();
        String s = queryAll();
        System.out.println("s = " + s);

    }
    //做业务的方法：增,删,改,查...
    public static void insert(){
        System.out.println("Test03.insert");
    }
    public static String queryAll(){
        return "查询结果";
    }
    //.....

}
```

作业

1. 声明一个printCube方法（无参，无返回值），在方法中输出1-5的立方，在main方法中调用该方法。
2. 声明一个method方法（无参方法，有返回值），在method方法中，除打印一个10x8的矩形外，再计算该矩形的面积，并将其作为方法返回值，在main方法中调用该方法，接收返回的面积值并打印。
3. 声明一个method方法（有参，有返回值），在method方法提供m和n两个参数，方法中打印一个m*n的矩形，并计算该矩形的面积， 将其作为方法返回值
4. （基本类型的值传递）
    a. 定义Test04类，在Test04类中声明一个静态的方法add(int x,int y)
    b. 在add方法中分别修改参数x和y的值之后，计算两个数相加并把结果返回;
    c. 在该类的main方法中，定义两个变量（int x ,int y）并赋值，
    d. 调用add方法（传x和y的值），接收返回值并打印，
    e. 在main方法中输出x和y的值，验证是否被改变。
5. （方法重载）
   1.定义一个Test05类，声明两个重载的div方法
      1.第一个方法有两个double参数，计算两数相除并返回
      2.第二个方法有两个float参数，计算两数相除并返回
   2.在main方法中调用两个重载div方法并打印输出结果
6. （跨类调用）
    a. 定义一个MyMath类,
        i. 声明一个方法计算两数最大值
        ii. 再声明一个方法计算两数最小值
    b. 定义一个Test06类，再main方法中调用，计算最大值和最小值的方法
7. 复习
    a. if语句
    b. switch语句
    c. 循环
        i. while

ii. do-while

iii. for

d. 双层循环

测试

1.输入用户名、密码，如果正确(用户名：18610241888,密码：123)，输出登录成功，

如果密码错误，输出“登录失败\n账号或密码错误，请重新输入”

2.使用for循环输出九九乘法

3.定义一个计算器类实现（加、减、乘、除）

12.