方法

方法重载（overload）

一个类里面有多个同名的方法，但参数列表不同，这种机制叫方法重载

重载条件：同一个类中同名的方法，与修饰符和返回类型无关，与参数列表有关（参数类型，参数顺序（不同类型），参数个数）

```
package javaoopday02;

public class OverLoad {
    private String userName;
    private String userPwd;
    private String userNickname;

    public void register(String userName, String userPwd) {
        this.userName = userName;
        this.userPwd = userPwd;
    }

    public void register(String userName, int i) {
        this.userName = userName;
    }

    public void register(int i,String userName) {
        this.userName = userName;
    }

    public void register(String userName, String userPwd, String userNickname) {
        register(userName, userPwd);
        this.userNickname = userNickname;
    }

    //一个类要想打印它的属性信息，可以生成toString()方法
    @Override
    public String toString() {
        return "OverLoad [userName=" + userName + ", userPwd=" + userPwd + ",
userNickname=" + userNickname + "]";
    }

    public static void main(String[] args) {
        OverLoad overLoad = new OverLoad();
        overLoad.register("Trump", "123");
        System.out.println(overLoad);

        OverLoad overLoad2 = new OverLoad();
        overLoad2.register("Trump", "123", "懂王");
        System.out.println(overLoad2);
    }
}
```

传值调用和传引用调用

方法调用传基本类型，不会改变原来实参； 如果传的引用类型，会改变原来实参的值。

```
package javaooopday02;

public class Swap {

    //传值（基本类型）调用：方法里面改变参数的值，不影响调用方法的实参变量的值
    public void swap(int a, int b) {
        int temp = b;
        b = a;
        a = temp;
        System.out.println("方法里面：a=" + a + ", b=" + b);
    }

    //传引用调用：在方法里面修改参数的内容，会改变调用方法的实参
    public void swap(DataWrap dataWrap1) {
        int temp = dataWrap1.getA();
        dataWrap1.setA(dataWrap1.getB());
        dataWrap1.setB(temp);
    }

    public static void main(String[] args) {
        int m = 3;
        int n = 2;
        Swap swap = new Swap();
//      swap.swap(m, n);
//      System.out.println("原来的变量：m=" + m + ", n=" + n);

        DataWrap dataWrap1 = new DataWrap(3, 2);
        swap.swap(dataWrap1);
        System.out.println(dataWrap1);
    }

}
```

字符串常量池

String类型变量如果是用字面量赋值，这些值会在字符串常量池中保存。如果新定义一个变量，它的直接量的值已经在常量池存在，直接引用原来的字符串，不会新建一个。

用new 的方式给字符串赋值，完全等于创建对象，尽管两个字符串相同，也会创建出不同的对象

```
public static void main(String[] args) {
        //String类型如果是用字面量赋值，这些值会在字符串常量池中保存
        //如果新定义一个变量，它的直接量的值已经在常量池存在，直接引用原来的字符串，不会新建一
个
        String a = "Hello";
        String b = "Hello";

        //用new 的方式给字符串赋值，完全等于创建对象，尽管两个字符串相同，也会创建出不同的对象
        String c = new String("Hello");
        String d = new String("Hello");
        System.out.println(a==b);  //引用同一个字符串，所以地址相等
        System.out.println(c==d);
    }
```

递归方法调用

在方面里面会调用自己，递归一定要有退出机制

```
public class Sum100 {

    //递归调用  f(n) = f(n) + f(n-1) + f(n-2) + ... + f(1)
    //方法里面调用自己
    //一定要有结束条件
    public int sum(int n) {
        if(n==1) {
            return 1;
        }else {
            return n + sum(n-1);
        }
    }

    public static void main(String[] args) {
        Sum100 sum100 = new Sum100();
        System.out.println(sum100.sum(100));
    }
}
```

package和import

- package

package是包，用来组织类文件，等效于目录。也是避免类同名的一种机制。

多个包之间用点号分隔，每个包对应磁盘一个目录，点号后面的子包对应到子目录

每个类在第一行（非注释）语句写明包名： package xxx.xxx;（如果没有包名的类，它只能在“default package”(src目录)）

包名命名规则，一般公司的域名倒过来，比如 hqyj.com，包名: com.hqyj.projectname.modulename

- import （Ctrl + Shift + o，批量导入包）

当使用一个类，它跟当前类不在同一个包时，就需要使用import 语句导入包

import语句放在package之后，类定义之前，一个类可以有多条import语句；

可以用*通配符一次性导入包下的所有类

java.lang这个包是一些常用的核心类，所以这个包下的类不需要import就可以直接使用

初始化块

类加载时候就会执行，比构造器先执行, 跟在类中位置没有关系, 一个类里面可以有多个初始化代码块

初始化代码块可以用static修饰，static的初始化代码块只能执行一次，非static代码块每次new对象都会执行

```
//类加载时候就会执行，比构造器先执行，跟在类中位置没有关系
    {
        int n = 100;
        j = 88;
        System.out.println("初始化代码块: " + n);
    }


//一个类可以有个多个初始代码块
        static{
            System.out.println("第二个初始代码块");
        }
```

今日作业

1. 写一个类Login，有一个登录方法login，登录方式账号，密码和验证码；还可以用手机号和短信验证码登录。请写出重载的登录方法。
2. 写一个方法，参数是数组，在main方法里调用该方法，修改数组里的元素。在main方法里面重新打印数组元素，观察是否数组元素是否被修改
3. 用递归方式实现1到10的阶乘
4. 简述package和import的用法和作用
5. 写一个类，添加非静态初始化块和一个静态初始化块，添加构造方法。 创建两个对象，观察初始化块和构造方法的执行顺序。