方法

一 内容回顾

程序结构之循环结构
循环结构包括两种：分别为 for while do-while

continue语句
1.作用：跳过本次循环，执行下一次循环（如果有多重循环，默认继续执行离自己最近的循环）提前终止本次循环
2.使用：只能在循环结构中使用
3.其它说明：使用Label标签改变继续执行的循环

break语句
1.作用：break语句用于终止某个语句块的执行
2.用法：如果是循环中，作用为跳出所在的循环，如果是在switch语句中，则为跳出所在的switch语句
3.其它说明：使用Lable标签实现跳出指定的循环

三种循环的比较
1. 对于同一问题，三种循环可相互替代
2. for循环功能强于while，do-while.但若不是明显地给出循环变量初终值(或修改条件),则可以用while 或do-while.以增强程序的结构化和可读性。
3. 要防止无限循环--死循环。

二 教学目标

1.掌握方法的使用
2.理解形参、实参的区别
3.掌握方法的重载
4.掌握方法的递归

三 教学导读

3.1. 为什么需要方法

我们从写第一个Java程序 HelloWorld 开始，就一直在提main方法（也叫函数），说main方法是Java程序的入口。main方法由JVM调用，我们自己写的代码都要写在main方法中，这样程序启动时，就能执行我们的代码。
我们回顾一下到目前为止我们知道的术语：
标识符 关键字 数据类型 进制 变量 常量 运算符 表达式 语句

我们编写程序一般从定义变量开始 比如 int a= 10;
然后通过运算符对变量进行各种操作组成了表达式 比如 a+5
各种表达式组合加上分号结尾，就有了语句 比如 int sum = a+5;
我们知道程序的执行就是从main方法第一条语句，执行到最后一条语句（当然中间有流程控制 选择 循环等）
如果我们把多条语句用大括号括起来，我们可以管它叫复合语句或语句块
语句块是做什么的呢?

一般来说任何一行代码（语句）都是完成某个小功能的，而多行代码（组成的语句块）肯定也是可以完成更复杂一些的功能

比如：我们昨天学的循环，一个打印九九乘法表的代码，求10000以内的完数等等

那问题来了，我们如何重复使用九九乘法表的整体代码呢?

我们目前的办法就是把这段代码整体复制，在需要使用的地方粘贴

但这样问题又来了，如果我复3份同样的代码在我程序中后，我发现这段代码逻辑有问题，需要修改，咋办?

我只能把三个地方都改一下，类似的问题太多

所以，我们需要方法

3.1. 方法的概念

Java的方法（Method）类似于其它语言的函数(Function)，指一段可以直接被另一段程序或代码引用的程序或代码。

一个较大的程序一般应分为若干个程序块，每一个模块用来实现一个特定的功能。所有的高级语言中都有子程序这个概念，用子程序实现模块的功能。

面向过程语言中，整个程序就是由函数（相互调用）组成的

面向对象语言中，方法（函数）是类的组成部份，整个程序是由很多类组成的

通俗讲，方法就是解决某件事情的办法，比如 我要上班，可以选择 步行，骑车，开车，公共交通，而每一个方式，在程序中就可能是一个方法。

3.3. 方法的组成要素

方法的组成要素：修饰符 返回值 方法名 参数 方法体 五个要素

3.4. 方法的补充说明

方法的优点:
1.使程序变得更简短清晰
2.有利于程序的维护（修改）
3.可以提高开发效率
4.可以提高代码的重用性
方法名的命名规则:
1.方法名必须以字母 下画线 $ 数字组成
2.不能以数字开头
3.方法名尽量由单词组成，如果有多个单词，第一个单词首字母小写，其它单词首字母大写
4.在同一个类中，方法名一般不能重名（方法重载除外）

四 教学内容

4.1. 方法的声明(会)

语法

```
访问权限修饰符 其它修饰符 返回值类型 方法名（参数列表）{

    方法体代码

    return 返回值;（如果返回值类型为 void 此行可省略）
}
//示例
public static void print(){

        System.out.println("我是打印方法");

    //return;因为返回值类型为 void 所以此行可省略
}
```

1.访问权限修饰符：public ，default【如果没有添加任何的访问权限修饰符，则默认为default，而default不需要显式的写出来】，目前使用的访问权限修饰符都和 main 方法保持一致，使用 public

2.其它修饰符：可以是 static final abstract 等等 也可以没有，在讲面向对象前我们都用 static

3.返回值类型：如果有返回值，需要用返回值的类型代替，如果没有返回值 需要用 void 代替

4.方法名：符合方法名的命名规则情况下，根据方法的功能，自行定义，最好能见名知义（看到名字就能明白方法的功能）

5.参数列表：如果方法所实现的功能中有未知项参与运算，就可以将未知项设置为参数

实际参数：实参，在方法外面定义，表示实际参与运算的值或者变量，作用为了给形参进行赋值

形式参数：形参，在方法中定义，用于接收实参的值，相当于是一个未被赋值的变量

形参数据类型 形参变量名称

形参 = 实参;

6.大括号：方法的实现，里面写方法的功能代码

7.return ： 将当前方法运行之后的结果进行返回，返回给当前方法的调用者。如果方法声明为void可省略，否则返回实际类型的变量

注意1：方法声明（并实现）仅仅是声明了这个方法，方法中的代码不会被执行的

注意2：方法声明的位置为类的内部，其它方法的外部

4.2. 方法的使用(调用)(会)

调用语法：

```
方法名称(实参列表);
//示例:
print();
```

注意：

a.实参的数量和类型必须和形参保持完全的一致，实现书写的顺序也必须和形参中的顺序保持完全一致

b.方法之间只能进行相互的调用，而不能在方法中声明方法，就目前而言声明的方法都和main方法是并列的

c.定义方法的时候，运算的结果会返回给调用者【在哪个方法中调用，运算的结果返回给哪个方法】

d.方法只有声明，没有调用.对当前程序没有任何作用，白写了.

e.方法没有声明，直接调用就会报错，不允许.

f.方法声明的位置为类的内部，其它方法外部的任何位置,没有顺序要求

g.同一个类中，方法名不能冲突（不能名字相同，只有方法重载情况除外）

h.方法中声明的变量都为局部变量。在哪个方法中声明的变量，就只能在哪个方法中使用，在方法外不能直接访问到

i.如果方法没有返回值，则方法调用相当于执行了某个功能，但没有直接结果返回给调用者，如果方法有返回值，则相当于执行了某个功能，并获得了一个结果（变量），对调用者而言，相当于接收了一个变量

示例代码

```java
//所有方法的声明需要在类的大括号内，其它方法的外部
public class DemoMethod {
    public static void main(String[] args) {
        //调用没有参数，没有返回值的方法
        test2();
    }
    //声明没有参数，没有返回值的方法 test2
    public static void test2(){
        System.out.println("test2()方法。。");
    }
}
```

4.3. 方法的参数(会)

a.形参，就是方法声明中的参数，在方法调用前为没有赋值的变量
b.实参,方法调用时，写在方法名后面小括号中的变量或常量
c.方法被调用时，用实参给形参赋值，这个过程叫传参
d.传参时需要注意的事项：实参的数量和类型必须和形参的数量和类型保持一致【相兼容的数据类型】

4.4. 方法的返回值(会)

1.在没有返回值的方法中使用 return 语句,要求 return单独成立一条语句，类似于break或者continue，后面不能跟任何的数值，直接跟一个分号，此时 return 语句作用为 结束整个方法的运行。
2.在有返回值的方法中使用 return 语句，要求 return后加加空格后跟着需要返回的变量和结尾的分号，此时return语句作用为 结束整个方法的运行，并将返回的变量传给方法的调用者.要求 返回值的实际变量类型需要与方法声明的返回值类型保持一致。
3.如果方法声明中有返回值，在方法体中使用了选择语句，如果需要有不同情况下的返回结果，那就都需要写 return 语句
4.无论在方法体的任何位置出现 return 语句，本次方法的调用都立即结束，返回到调用者。

4.5. 方法的随堂练习

练习1：没有参数，没有返回值的方法，方法的功能是九九乘法表的打印

```java
public class MethodDemo1 {

    //将原来的打印代码写到一个独立的方法中
    public static void print(){
        //九九乘法表 共有45个结果，所以需要循环45次 第一行输出一个结果，
        //第二行输出二个结果，...第九行输出九个结果
        //用来记录当前行号
        int row = 1;
        //用来记录当前列号（也就是当前行的第几个结果）
        int col = 1;
        for (int i = 0; i <45 ; i++) {
            //输出row行的第col个结果,不换行
            System.out.print(col +"*"+ row+"=" + (col*row));
            //同一行中多个结果之间的分隔符
            System.out.print("\t");
```

```
            //如果行号和列号相等，说明第row行已打印完成
            if(row ==col){
                //打印换行符
                System.out.println();
                //列号重置
                col =1;
                //行号加1
                row++;
            }
            else{
                //列号加1
                col++;
            }
        }
    }

    public static void main(String[] args) {
        //调用打印方法
        print();
    }

}
```

练习2：没有返回值，有一个参数的方法，方法功能为计算指定整数的阶乘

```
public class MethodDemo2 {

    //计算指定数字 numbert的阶乘
    public  static  void factorial(int number){
        int sum = 1;
        int i = 1;
        while(i<=number){
            sum=sum*i;
            i++;
        }
        System.out.println("数字 "+number +" 的阶乘为:"+sum);
    }

    public static void main(String[] args) {
        //声明实参变量
        int number = 8;
        //调用方法，将实参的值传给形参
        factorial(number);
    }

}
```

练习3：有返回值，有一个参数的方法，方法功能为判断指定的整数是否为质数

```
public class MethodDemo3 {
```

```
    //判断指定的数字 number是否为质数，是返回真，不是返回假
    public  static  boolean checkPrime(int number){
        //用来记录是否是质数的布尔变量 true 就是质数
        boolean prime = true;
        //循环判断当前数字是否能被1和它本身外的数字整除
        for(int i=2; i<number;i++){
            //如果能被整除，说明不是质数
            if(number%i==0){
                //设置标质不是质数
                prime=false;
                //跳出循环，也就是只要发生过整除，后续就没必要再判断了
                break;
            }
        }
        return prime;
    }

    public static void main(String[] args) {
        //调用打印方法

        //声明实参变量
        int number = 11;
        //调用方法，将实参的值传给形参，并接收返回值
        boolean prime = checkPrime(number);
                //根据结果，给出结论字符串
        String info = prime?"是质数":"不是质数";
                //输出最后的结果
        System.out.println("数字 " + number +" "+info);
    }

}
```

练习4：有返回值，有两个参数的方法，方法的功能为 计算指定数字的n次方的值

```
public class MethodDemo4 {

    //计算指定数字 number的n次方的值并返回
    public static int power(int number , int n){
        int sum = 1;
        while (n>0){
            sum *= number;
            n--;
        }
        return sum;
    }
    public static void main(String[] args) {

        //声明实参变量
        int number = 2;
        int n = 10;
        //调用方法，将实参的值传给形参

        int sum = power(number,n);
```

```
        System.out.println("数字 " + number + " "+ n +"次方的结果是:"+sum);
    }

}
```

练习5：方法内调用其它方法，方法功能 把一个数分解质因数，传入需要分解的数，返回分解的结果

```
public class MethodDemo5 {

    //判断指定的数字 number是否为质数，是返回真，不是返回假
    public  static  boolean checkPrime(int number){
        //用来记录是否是质数的布尔变量 true 就是质数
        boolean prime = true;
        //循环判断当前数字是否能被1和它本身外的数字整除
        for(int i=2; i<number;i++){
            //如果能被整除，说明不是质数
            if(number%i==0){
                //设置标质不是质数
                prime=false;
                //跳出循环，也就是只要发生过整除，后续就没必要再判断了
                break;
            }
        }
        return prime;
    }
    //将指定的整数进行质因数分解，将结果以字符串返回，比如传入90，返回90=2 * 3 * 3 * 5
    public static String primeFactorization(int number){

        String  result = "";
        int middleNumber = number;
        for(;;){
            //内层循环，目的是找到当前数middleNumber的最小因子
            for(int j=2;j<middleNumber;j++){
                if(middleNumber%j==0){//说明找到了最小因子j
                    result= result+j+"*";//最小因子保存
                    middleNumber = middleNumber/j; //把当前数用最小因子分解，准备下次分解
                    break;//跳出循环
                }
            }

            if(checkPrime(middleNumber)){//如果middleNumber是质数
                result =  number+ "="  + result + middleNumber;
                break;
            }
        }
        //返回最后的结果
        return result;
    }
    public static void main(String[] args) {


        //声明实参变量
```

```
        int number = 90;
        //调用方法，将实参的值传给形参
        String result = primeFactorization(number);

        System.out.println("数字 " + number + " 分解质因数后的结果为:"+ result);
    }

}
```

4.6. 方法的内存展示(会)

4.6.1 java的内存分区

java将内存分成了5块儿,分别是堆区,栈区,方法区,本地方法区,寄存器
栈区:里面存放数据的特点是:先进后出,我们主要将加载时的局部变量和函数放在栈区,数据的特点是使用完立刻释放
堆区:存放的是实体(对象和数组),实体可以同时存放多个值,实体里面的变量如果不赋值,会有默认值.整型数据默认值是0,boolean---false

了解:
方法区:程序运行中的二进制文件等(比如:.class)
本地方法区:存放外界引入的c,c++等的内容
寄存器:也可以称为计数器.

堆区中的数据会在某个时刻被释放-通过垃圾回收机制.
垃圾回收机制是通过一个线程控制的,由于这个线程的等级比较低,所以不会立刻执行,数据就不会立刻释放.

4.6.2 方法在内存中的工作原理

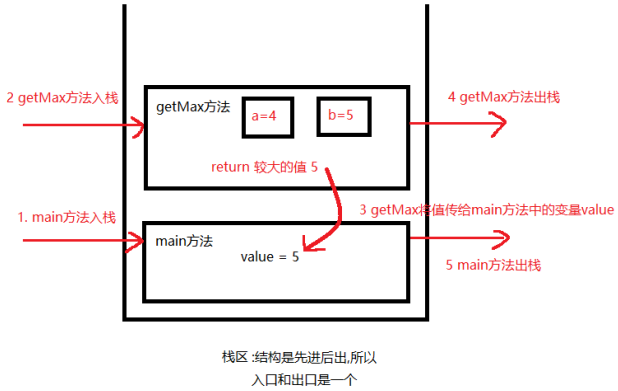
- 示例代码

```
public class Demo5 {
    //实例:求两个数的最大值
    public static void main(String[] args) {
        int value = getMax(4,5);
        System.out.println(value);
    }

    public  static int getMax(int a ,int b) {
        if (a>b) {
            return a;
        }else {
            return b;
        }
    }
}
```

- 内存展示图

```
public class Demo5 {
        //实例:求两个数的最大值
        public static void main(String[] args) {
                int value = getMax(4,5);
                System.out.println(value);
        }

        public  static int getMax(int a ,int b) {
                if (a>b) {
                        return a;
                }else {
                        return b;
                }
        }
}
```

注意:为了更加清晰的理解概念,这里只展示了方法在栈区中的使用



栈区 :结构是先进后出,所以
入口和出口是一个

- 执行过程说明

1.程序开始执行后会先找到程序的入口main方法,main方法入栈,并执行main中的代码,给局部变量value开辟空间,执行getMax方法
2.getMax方法入栈,在方法内部会进行运算,求出a和b的最大值.
3.getMax方法会将return后面接收到最大值通过方法的返回值传到main方法,并赋值给变量value
4.当getMax执行完(可以是执行return或者执行方法的右大括号),会执行出栈操作
5.当main执行完,执行出栈操作,到此程序执行完成.

总结:通过内存展示我们可以更清晰的了解方法原理.

4.7. 方法的重载(会)

什么是方法的重载
定义:同一个类中，方法名字相同，参数列表不同，就叫方法重载
说明:
1. 参数列表的不同包括，参数个数不同，参数数据类型不同，参数顺序不同
2. 方法的重载与方法的修饰符和返回值没有任何关系

概念:一个类中的，一个功能方法的多种体现形式（有不同的方法体）。
举例:
1、人类，有吃的功能：eat()
```
eat(食物);
eat(药);
eat(口香糖);
```

2、求和的功能:
```
getSum(int i,int j);
getSum(double d1, double d2);
```
3、水:
常温：液态

0度以下：固态

100度以上：气态

就是同一个功能的方法，因为参数的不同，调用的具体的方法也不同。
如何判定多个方法是否是重载的？衡量标准，要同时满足以下三条：

A：必须同一个类中。
B：方法名称必须一致。
C：参数列表必须不同。（顺序，个数，类型）

和static，public，返回值，void等等都没有关系。
优点：

1、简化了开发的压力
2、简化了记忆的压力
3、调用方法更方便，更简洁，又满足了不同的情况

基本原理：

当方法名称一致时，通过形式参数列表的不同来选择要执行的方法。

4.8. 方法重载的随堂练习

```
//演示方法的重载
/*
在同一个类中，如果满足以下的条件，则称为这几个方法之间彼此重载
        a.方法名相同
        b.参数不同【数量不同或者类型不同】
        c.访问权限修饰符和返回值类型没有影响
*/
class OverloadingDemo
{
  public static void show() {
        System.out.println("无参无返回值的show");
    }
    //1.改变参数
    public static void show(int a) {
        System.out.println("int的show");
    }

    public static void show(String a) {
        System.out.println("String的show");
    }

    public static void show(String a,int b) {
        System.out.println("String  int的show");
    }

    //2.改变返回值:返回值对方法的重载没有任何影响
    //只改变返回值类型，其他都不改变，则对于编译器而言，则认为是同一个方法
    /*
    public static String show() {
        System.out.println("String返回值的show");
```

```
        return "abc";
    }
    */

    //3.访问权限修饰符
    //只改变访问权限修饰符，其他都不改变，则对于编译器而言，则认为是同一个方法
    /*
    static void show() {
        System.out.println("show");
    }
    */
    public static void main(String[] args)
    {
        //对于重载函数而言，具体调用的是哪个函数，取决于所传的参数
        show("10");
        show("10",10);
    }
}
```

4.9. 方法的递归(会)

定义：在一个方法内，调用方法本身，称为方法的递归(注意和重载的区别)
说明：方法递归包含了一种隐式的循环，会重复执行某段代码，但是这种重复不需要使用循环语句来进行控制

4.10. 方法递归的随堂练习

练习1：使用递归计算1到数字n的和

```
public class recursionDemo
{
    //计算 1到数字n的和
    public static int sum(int n){

        //数字1的和为1，直接返回
        if(n == 1){
            return 1;
        }
        //数字2及以上的数字和为 当前数字本身加上它前面所有数字的和
        return sum(n-1)+n;

    }

    public static void main(String[] args)
    {
        int number = 5;

        int sum = sum(number);

        System.out.println("数字1到" + number+"的和为:" + sum);
```

```
    }
}
```

五 实战应用

5.1 实战案例之

```

```

5.2 实战案例之

```

```

5.3 实战案例之

```

```

六 教学总结

6.1 课程重点

```
1.掌握方法的语法结构,包括参数,返回值.
2.掌握方法的使用
3.掌握重载的原理以及使用
4.掌握递归的原理以及使用
```

6.2 课程难点

```
1.递归的实现
```

七 课后作业

1. 设计方法，计算两个日期之间相差多少天。两个日期的年月日由参数控制。

```
// 设计方法，计算两个日期之间相差多少天。两个日期的年月日由参数控制。
static int getDelta(int fromYear, int fromMonth, int fromDay, int toYear, int toMonth, int
```

```
toDay) {
    // 1、起始那一天，是fromYear的第几天
    // 2、终止那一天，是toYear的第几天
    // 3、计算 fromYear的1月1日 ~ toYear的1月1日相差多少天
    // 4、  3 + 2 - 1

    int fromDays = getDays(fromYear, fromMonth, fromDay);
    int toDays = getDays(toYear, toMonth, toDay);

    int days = 0;
    for (int y = fromYear; y < toYear; y++) {
        days += check(y) ? 366 : 365;
    }

    return days + toDays - fromDays;
}


/**
 * 计算一个日期是当年的第几天
 * @param year
 * @param month
 * @param day
 * @return
 */
static int getDays(int year, int month, int day) {
    int days = day;

    for (int m = 1; m < month; m++) {
        if (m == 1 || m == 3 || m == 5 || m == 7 || m == 8 || m == 10 || m == 12) {
            days += 31;
        }
        else if (m == 4 || m == 6 || m == 9 || m == 11) {
            days += 30;
        }
        else if (m == 2) {
            days += check(year) ? 29 : 28;
        }
    }

    return days;
}

/**
 * 验证一个年份是不是一个闰年
 * @param year
 * @return
 */
static boolean check(int year) {
    return year % 4 == 0 && year % 100 != 0 || year % 400 == 0;
}
```

2. 开发一个标题为“FlipFlop”的游戏应用程序。它从 1 计数到 100，遇到 3 的倍数就替换为单词“Flip”，5 的倍数就替换为单词“Flop”，既为 3 的倍数又为 5 的倍数则替换为单词“FlipFlop”

```
public static void flipFlop() {
    for (int i = 1; i <= 100; i++) {
        if (i % 3 == 0 && i % 5 == 0) {
            System.out.println("FlipFlop");
        }
        else if (i % 3 == 0) {
            System.out.println("Flip");
        }
        else if (i % 5 == 0) {
            System.out.println("Flop");
        }
        else {
            System.out.println(i);
        }
    }
}
```

3. 两个自然数X，Y相除，商3余10，被除数、除数、商、余数的和是163，求被除数、除数。

```
public static void test() {
    for (int x = 0; x < 163; x++) {
        for (int y = 1; y < 163; y++) {
            // 求商
            int s = x / y;
            // 求余
            int l = x % y;

            if (s == 3 && l == 10 && (x + y + s + l == 163)) {
                System.out.println("x = " + x + ", y = " + y);
            }
        }
    }
}
```

八 解决方案

8.1. 应用场景

8.2. 核心面试题

1.方法的传参过程是如何工作的

在调用方法的使用，在方面参数中写入实参，jvm运行时，会把实参赋值给形参。

2.return关键字的用法有哪些，举例说明

```
return;  //结束方法 ,可以省略

return  sum;  //返回结果 sum , 结束方法
```

3.什么是方法的重载？举例说明

同一个类中，方法名相同，方法参数列表不同