山东大学

信息科学与工程学院

2022－2023 学年第一学期

实 验 报 告

课程名称：　Java 编程技术

实验名称：　第四次实验（A）

专　业　班　级　通信工程三班

学　生　学　号　202000120202

学　生　姓　名　李鑫

实　验　时　间　2022 年 11 月 1 日

实验报告

【实验目的】

1. 强化使用 java 对数组与字符串的编程
2. 深化对类和对象的理解

【实验要求】

1. 随机生成一个 10 个元素组成一维数组，输出数组中的最大值、最小值、所有元素总和，以及最大值和最小值在该数组中的位置，并实现数组的排序。

2. 利用 arraycopy 和 copyof 方法分别实现两个数组的内容拷贝，并利用 foreach 语法显示两个数组元素的内容

3. 编写程序完成特定功能

1)声明一个名为 name 的 String 对象， 内容是“My name is NetworkCrazy”

2)打印字符串的长度

3)打印字符串的第一个字符

4)打印字符串的最后一个字符

5)打印子字符串 NetworkCrazy （substring()方法提取子字符串）

6)利用 lastIndexOf()方法搜索字符 'e' 最后一次出现的位置

4. 利用 StringBuffer 类实现在字符缓冲区中存放字符串 "Happy new year!"，
并通过 setCharAt 和 charAt 实现字符设置和替换，利用 insert 实现字符的插入，
利用 append 实现在字符串末尾添加相应的内容。

5. 创建矩形的类封装

6. 创建三角形、梯形和圆形的类封装

7. 编写一个程序，定义一个类，类有 3 个 plus ()方法，
分别实现两个数相加，三个数相加，四个数相加的程序。

8. 编译并运行给定的程序，观察分析运行结果，体会程序 super 和this 的用法，进一步 理解变量隐藏和方法重写的概念

9. 指出给定代码的错误并解决

10. 熟悉 Date 类和 Calendar 类的成员方法

【实验具体内容】

1. 随机生成一个 10 个元素组成一维数组，输出数组中的最大值、最小值、所有元素总 和，以及最大值和最小值在该数组中的位置，并实现数组的排序。

【实验代码】

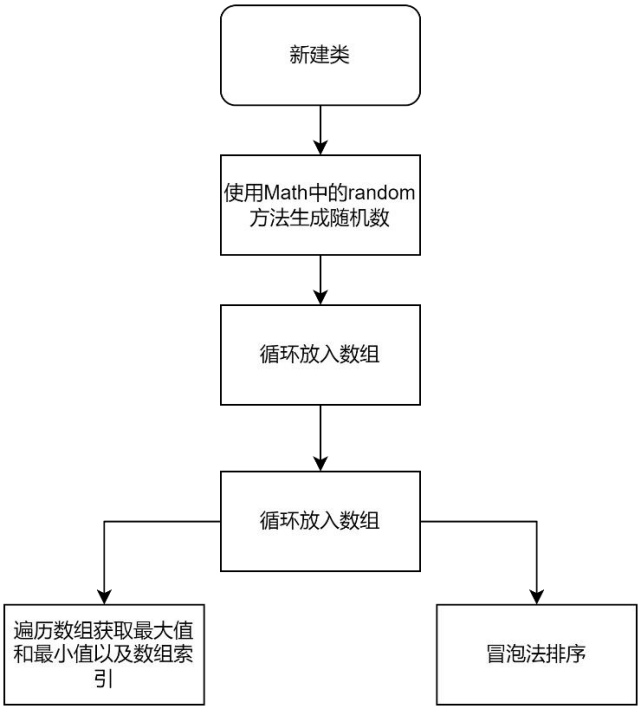
```
public class Homework4A1 {
    public static void main(String[] args) {
        // 新建数组，类型为整形，长度为 10
        int[] array = new int[10];
        // 声明变量 i,j 并初始化，i 主要用来作为循环变量，j 主要用来保存位置索引
        int i, j = 0;
        // 作为交换中间变量存在
        int temp;
        // 初始化累加变量
        int sum = 0;
        // 声明 max 和 min 后面用于保存最大值和最小值
        int max, min;
        // 屏幕提示
        System.out.println("随机生成的十个元素的数组为：");
        // 调用 length 方法控制循环遍历的次数
        for (i = 0; i < array.length; i++) {
            // Math.random 用于生成[0,1]的随机数，后面乘以 100 实现[0,100]随机数的生成、
            // Math.round 用于四舍五入
            // (int)强制转换类型,因为数组是整型数组
            array[i] = (int) Math.round(Math.random() * 100);
            // 输出每个元素
            System.out.println(array[i]);
```

```
            // 累加每个数组元素
            sum += array[i];
        }
        // 换行处理
        System.out.println("\n");
        // 输出最终累加值
        System.out.println("数组中所有值的和为: " + sum);
        // 初始化最大值为 array[0]
        max = array[0];
        // 循环比较数组最大值并记录其元素索引
        for (i = 0; i < array.length; i++) {
            if (max < array[i]) {
                max = array[i];
                j = i;
            }
        }
        // 输出数组最大值和位置索引
        System.out.println("数组的最大数为:  " + max + "    " + "在数
组中的位置为:  " + j);
        // 初始化最小值为 array[0]
        min = array[0];
        i = 0;
        // 循环比较数组最小值并记录其元素索引
        for (i = 0; i < array.length; i++) {
            if (min > array[i]) {
                min = array[i];
                j = i;
            }
        }
        // 输出数组最小值和位置索引
        System.out.println("数组的最小数为:  " + min + "    " + "在数
```

```
组中的位置为：  " + j);

        // 命令行提示交互信息

        System.out.print("数组升序排序为：  ");

        // 冒泡法进行元素排序，两层循环，第一循环从第一个元素到最后一个元素为一轮，第二个循环从第 i 个元素到最后一个元素为一轮
        for (i = 0; i < array.length; i++) {
            // j 从 i 开始，然后在 i 不变的小循环里每次加 1
            for (j = i; j < array.length; j++) {
                //j 每次加 1，然后让其对应的数组元素和 i 对应的数组元素比较要是 i 对应元素的值比 较大，那就让两者交换顺序。

                // 因为这里离需要的是升序排列，就要求数组后面的值要大于前面的值。这样每一轮小循环下来，最小的数就放在了最前面。
                // 之后大循环的 i 加 1， 再开始用下一个数与其后面的每一个数比较
                if (array[i] > array[j]) {
                    temp = array[i];
                    array[i] = array[j];
                    array[j] = temp;
                }
            }
            System.out.print(array[i] + "    ");
        }
        /*输出一个转义字符，实现换行*/
        System.out.println("\n");
        System.out.print("数组降序排序为：  ");

        // 冒泡法进行元素排序，两层循环，第一循环从第一个元素到最后一个元素为一轮，第二个循环从第 i 个元素到最后一个元素为一轮
        for (i = 0; i < array.length; i++) {
            //j 从 i 开始，然后在 i 不变的小循环里每次加 1
```

```
            for (j = i; j < array.length; j++) {
                //j 每次加 1，然后让其对应的数组元素和 i 对应的数组元素比较
要是 i 对应元素的值比较大，那就让两者交换顺序。
                //因为这里离需要的是升序排列，就要求数组后面的值要大 于前面
的值。这样每一轮小循环下来，最小的数就放在了最前面。
                //之后大循环的 i 加 1， 再开始用下一个数与其后面的每一个数
比较
                if (array[i]< array[j]) {
                    temp = array[i];
                    array[i] = array[j];
                    array[j] = temp;
                }
            }
            System.out.print(array[i] + "    ");
        }

    }

}
```

【实验流程图】



【实验结果截图】

```
"C:\Program Files\Eclipse Foundation\jdk-8.0.302.8-hotspot\bin\java.exe" ...
随机生成的十个元素的数组为:
80
71
88
22
69
48
63
37
27
64

数组中所有值的和为: 569
数组的最大数为: 88    在数组中的位置为: 2
数组的最小数为: 22    在数组中的位置为: 3
数组升序排序为: 22    27    37    48    63    64    69    71    80    88

数组降序排序为: 88    80    71    69    64    63    48    37    27    22
进程已结束,退出代码0
```

2. 利用 arraycopy 和 copyof 方法分别实现两个数组的内容拷贝，并利用 foreach 语法显示两个数组元素的内容

【涉及知识点】

1. copyof 方法是 Array 类的方法，所以在使用这个方法之前，要 import java.util.Arrays。它可以指定复制数字元素的个数，也可以指定起止位置。使用例子如下：

```
b1 = Arrays.copyOf(a1, 10);
```

2.以 arrayCopy(arr1, 2, arr2, 5, 10)为例，就实现了 arr1 数组从位置为 2 的元素开始，复制到 arr2 中位置是 5 的地方，并且复制的元素的个数是 10 个，2 为源操作位置，5 为目标操作位置，10 为操作长度。arr1 为源操作对象, arr2 为目的操作对象

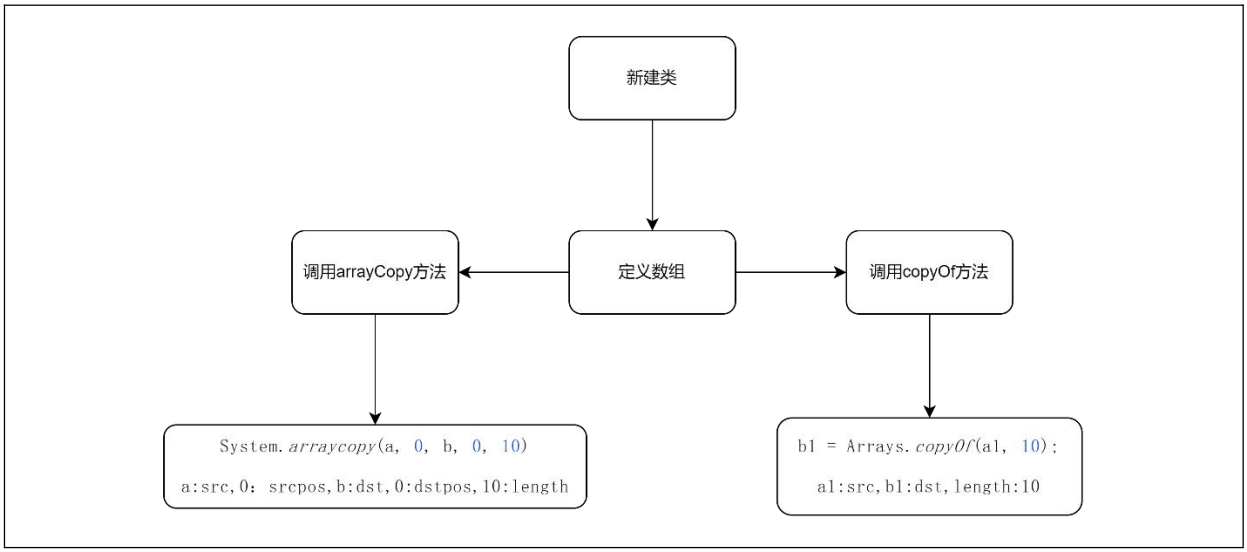
【实验代码】

```
import java.util.Arrays;

public class Homework4A2 {
    // 利用 arraycopy 和 copyof 方法分别实现两个数组的内容拷贝，并利用
foreach 语法显示两个数组元素的内容
```

```
    public static void main(String[] args) {
        //定义整形数组变量 a 、b 并对其进行初始化赋值
        int[] a = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};
        int[] b = {1489, 15, 89, 898, 48, 6798, 487, 8487, 965, 480};
        //直接调用 System 类的方法 arraycopy 进行复制，由前面的语法，实现的
是从数组 a 下标为 0 的元素开始，将 10 个数复制到数组 b 下标为 0 的位置
        System.arraycopy(a, 0, b, 0, 10);
        //foreach 输出数组 a
        System.out.println("数组 a: ");
        //使用 foreach 语句实现循环输出
        //foreach 语句的参数有三个，后两个变量用冒号隔开。第一个参数是指明
每个元素的类型，第二个参数是用来存放每一个元素的新变量的名字，第三个参量是要
实现遍历的数组的名字
        //数组 a 是一个整形数组，所以其中的每个元素都是整型变量，现在用 i 来
存放其中的每一个元素，并且进行输出
        for (int i : a) {
            System.out.print(i + " ");
        }
        System.out.println();
        /*使用 foreach 语句实现循环输出 b 数组。b 中的每一个元素都是整形*/
        System.out.println("数组 b : ");
        for (int i : b) {
            System.out.print(i + " ");
        }
        System.out.println();
        int[] a1 = {14,15,16,21,35,62,15,12,19,51};
        int[] b1 = {149, 15, 89, 88, 48, 679, 487, 8487, 965, 480};
        /*因为在最前面已经 inport 过这个类，所以在这里可以直接调用它的方法
实现将数组 a 中的 10 个元素复制到 b 中*/
        b1 = Arrays.copyOf(a1, 10);
        /*会用 foreach 语句循环输出 a 数组的每一个元素，a 数组的每一个元素
```

```
都是整形*/

        System.out.println("数组 a1: ");
        for (int i : a1) {
            System.out.print(i + " ");
        }
        System.out.println();
        /*使用 foreach 语句循环输出数组 b 的每一个元素，b 的每一个元素都是
整形*/

        System.out.println("数组 b1 : ");
        for (int i : b1) {
            System.out.print(i + " ");
        }
    }
}
```

【实验流程图】



【实验结果截图】

```
"C:\Program Files\Eclipse Foundation\jdk-8.0.302.8-hotspot\bin\java.exe" ...
数组 a:
1 2 3 4 5 6 7 8 9 0
数组 b :
1 2 3 4 5 6 7 8 9 0
数组 a1:
14 15 16 21 35 62 15 12 19 51
数组 b1 :
14 15 16 21 35 62 15 12 19 51
进程已结束,退出代码0
```

3. 编写程序完成特定功能

1) 声明一个名为 name 的 String 对象， 内容是“My name is NetworkCrazy”

2) 打印字符串的长度

3) 打印字符串的第一个字符

4) 打印字符串的最后一个字符

5) 打印子字符串 NetworkCrazy （substring()方法提取子字符串）

6) 利用 lastIndexOf()方法搜索字符 'e' 最后一次出现的位置

【涉及知识点】

substring()方法：属于 String 类，作用是返回字符串的子字符串。可以规定要返回的子字符串的起止位置（不返回结束位置的值）
lastIndexOf()方法：属于 String 类，作用是返回指定的字符最后出现的位置。执行时， 从指定的位置开始向前检索，若不指定位置，则默认从最后一个值开始往前检索。若是找到了，就返回一个整型数据代表其位置，要是没找到就返回- 1
length()方法：属于 String 类，作用是返回字符串的长度
charAt()方法：属于 String 类，作用是以字符串的形式返回指定位置的字符。注意字符串的下标从 0 开始。若是指定的位置不在字符串之内，则返回空字符串

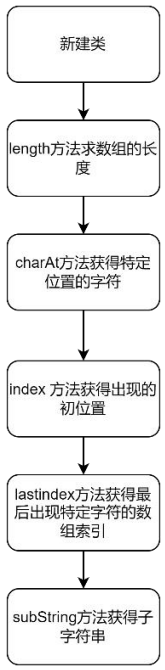
【实验代码】

```
public class Homework4A3 {
    public static void main(String[] args) {

        /*定字符串类的对象，并给它赋值*/

        String name = "My name is NetworkCrazy";
```

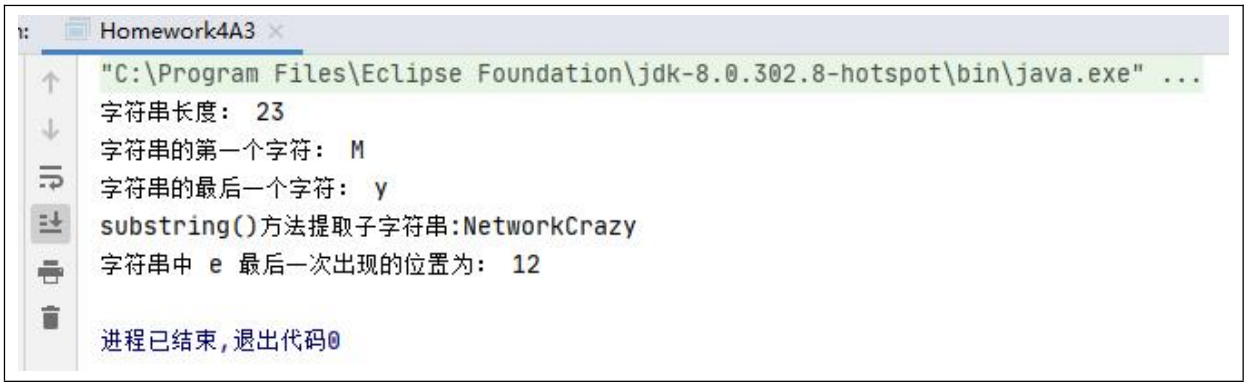
```
        /*调用 String 类的 length 方法获取字符串的长度*/
        /*并且调用 println 方法将其输出*/
        System.out.println("字符串长度： " + name.length());
        /*调用 String 类的 charAt 方法，返回指定位置的字符，此处的位置是开始*/
        System.out.println("字符串的第一个字符： " + name.charAt(0));
        /*调用 String 类的 length 方法获取字符串的长度,因为字符串的下标从 0 开始，所以长 度为 N 的字符串，其最后的下标是 N- 1*/
        /*然后调用 charAt()函数返回该位置的字符*/
        System.out.println("字符串的最后一个字符： " +
name.charAt(name.length() - 1));
        /*因为 substring 方法的参数是要返回的子字符串的起止位置，所以要先获取该子字符串 的起止位置*/
        /*定义一个 String 类的对象，他的值初始化为空，用作缓冲*/
        String buffer = " ";
        /*调用 String 的 indexOf 方法,获取 Network 在字符串中的位置*/
        int a = name.indexOf("Network");
        /*调用 String 的 lastindexOf 方法,获取 y 最后出现的位置*/
        int b = name.lastIndexOf("y");
        /*因为 subString 方法不返回结束位置的值，所以 b 的值应该加 1*/
        /*调用 subString 方法返回字字符串的，并且将该子字符串付给中间字符串*/
        buffer = name.substring(a, b + 1);
        /*调用 println 方法将字字符串输出*/
```

```
        System.out.println("substring()方法提取子字符串:" + buffer);
        /*使用 lastIndexOf 方法获取字符 e 在字符串中最后出现的位置,返回一个整型数据用于 代表该位置*/
        int m = name.lastIndexOf("e");
        System.out.println("字符串中 e 最后一次出现的位置为:  " + m);
    }
}
```

【实验流程图】



【实验结果截图】



4. 利用 StringBuffer 类实现在字符缓冲区中存放字符串 "Happy new year!"，并通过 setCharAt 和 charAt 实现字符设置和替换，利用 insert 实现字符的插入，利用 append 实现在字符串末尾添加相应的内容。

【涉及知识点】

String 类是固定长度也不可修改，每次对 String 做修改，都是会重新产生一个新的字符串对象。而 StringBuffer 类是可变长度,可通过 append 方法修改

两者可以相互转化：对于 StringBuffer 转化为 String 可使用 String b=a.toString()

对于 String 转为 StringBuffer 可使用 StringBuffer b=new StringBuffer(string)

setCharAt()方法属于 StringBuffer 类，用于替换字符串中的某一个字符，其两个参数分别是要替换字符的位置和替换他的字符。

其常用的格式：setCharAt(int index,Char ch)

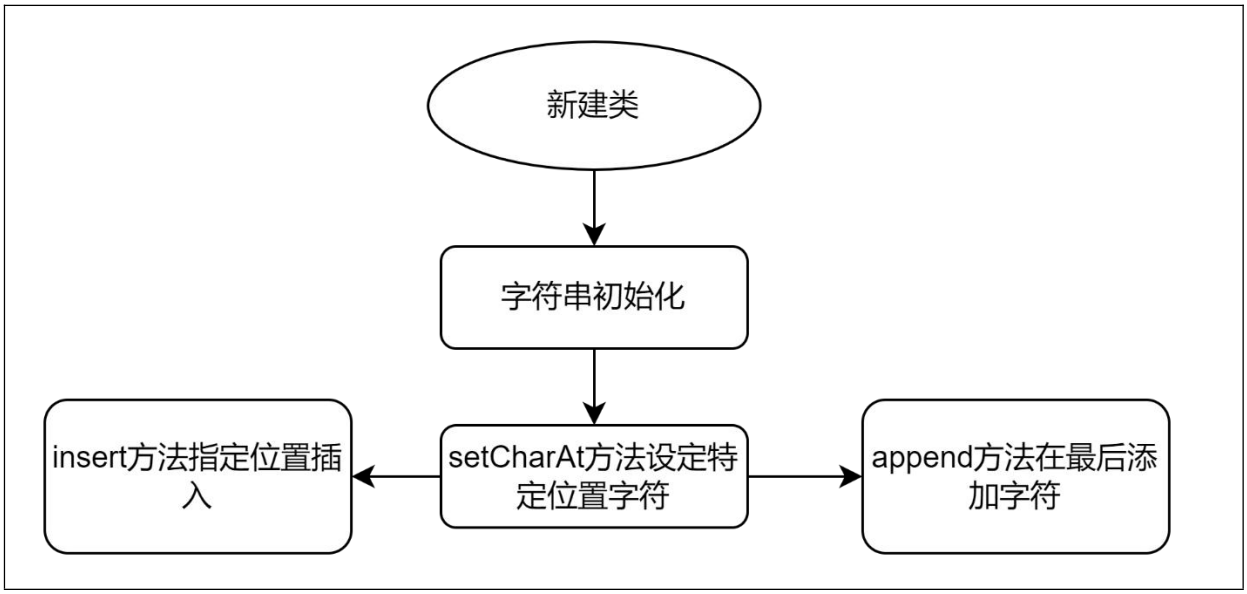
insert()方法：属于 StringBuffer 类，用于在指定位置插入指定的字符串，返回值是已 经修改过的字符串

【实验代码】

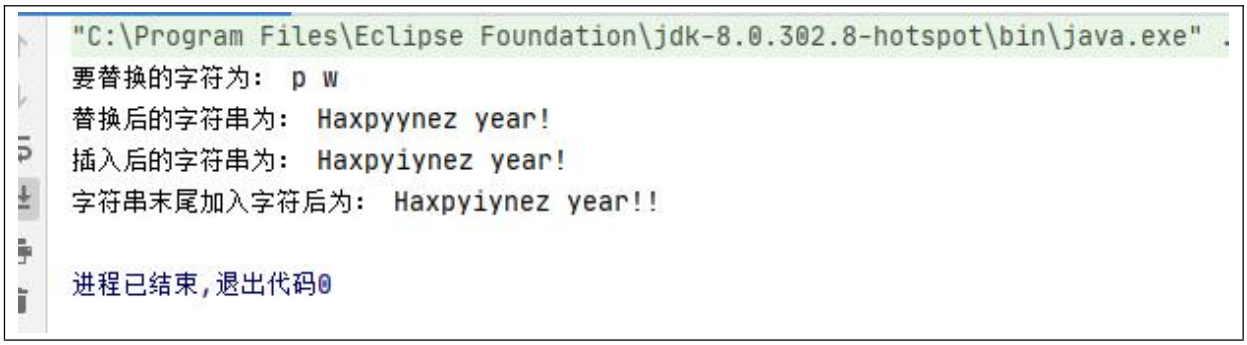
```
public class Homework4A4 {
    public static void main(String[] args) {
        /*定义 StringBuffer 类的对象，并且初始化为 Hppy new year! */
        StringBuffer A = new StringBuffer("Happy new year!");
        /*调用 StringBuffer 类的 toString 方法，将 StringBuffer 类的
对象转换成 String 类*/
        A.toString();
        System.out.print("要替换的字符为： ");
        /*定义三个新的字符型对象，并且使用 String 类的 charAt()方法获取指
定位置的元素*/
        char i = A.charAt(2);
        char j = A.charAt(5);
        char k = A.charAt(8);
        /*将字符串 A 中指定位置的元素输出*/
        System.out.print(i);
        System.out.print(j);
        System.out.print(k);
```

```
        /*输出一个转义字符以进行换行*/
        System.out.println();
        /*使用 setCharAt()方法在指定的位置更新值，指定的位置就是上面输出元素的位置*/
        A.setCharAt(2,'x');
        A.setCharAt(5,'y');
        A.setCharAt(8,'z');
        /*输出替换后的字符串*/
        System.out.print("替换后的字符串为： ");
        System.out.println(A);
        /*使用 insert 方法查在指定的位置插入字符*/
        A.insert(5,'i');
        System.out.print("插入后的字符串为： ");
        System.out.println(A);
        /*使用 append 函数在最后插入一个字符'!'*/
        A.append('!');
        System.out.print("字符串末尾加入字符后为： ");
        System.out.println(A);
    }
}
```

【实验流程图】



【实验运行截图】



5. 创建矩形的类封装

【涉及知识点】

成员类：即 Rectangle 这个类作为自己建立的类的成员变量，可以存放任意类型的数据
类中类，难免会出现成员名字相同的情况，这是局部变量就会屏蔽掉全局变量，要是想要继续使用全局变量，那就要使用 this 来代指成员类所在的类对象
构造函数与类同名，且他显化后原来的构造函数就没有了，要想让原来那个继续存在，就要声明一个与之同名但是无参的方法，以通过重载实现共存

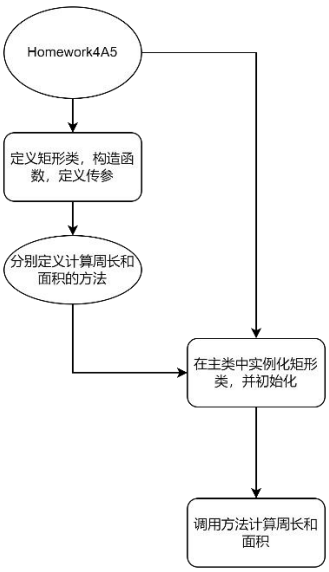
【实验代码】

```
public class Homework4A5 {
    public static void main(String[] args) {
        /*定义两个长整型变量，用来存放矩形的周长和面积*/
        double perimeter;
        double area;
        /*在栈中 new 一个空间，这个空间的大小就是一个 Rectangle 类的大小，将快内存的地址交给引用变量 R，这个 R 同时也是 Rectangle 类的对象*/
        Rectangle R = new Rectangle();
        /*调用 Rectangle 类的的 periodmeter 方法实现周长的求解*/
        perimeter = R.perimeter(2, 5);
        /*调用 Rectangle 类的 area 方法求解矩形的面积*/
        area = R.area(2, 5);
        System.out.println("矩形的周长为：  " + perimeter + "    "
```

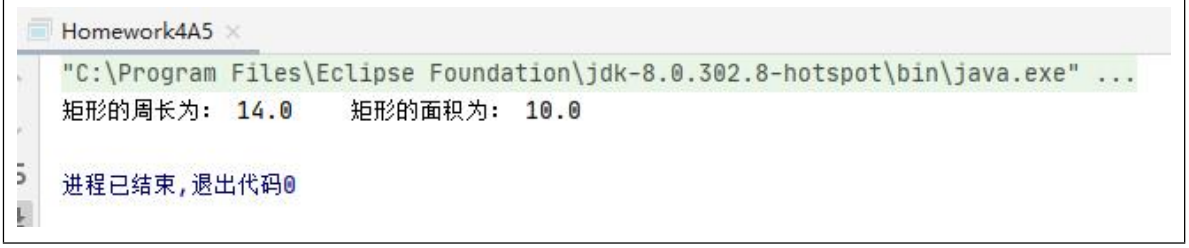
```
+ "矩形的面积为： " +
                area);
    }
}

/* Rectangle  类的定义，该类作为我自己的类的成员变量*/
class Rectangle {

    private double width;
    private double height;

    /*隐含的构造函数显化，因为后面要定义一个带有 width 和 height 的方法，只要那个函 数定义好，那么原来的和类同名的隐含的构造方法就没了，所以在这里声明他一下，但 是不需要参数，这就使得他和下面将要构造的方法名字相同但是参数不同，实现重载， 即让原来的隐式函数得以保存*/
    public Rectangle() {
    }


    /*构造一个 public 类型的方法，以 width 和 height 为形式参数*/
    public Rectangle(double width, double height) {
        /*在类成员中使用 this，就代表方法所属的对象，所以说这里的 this 就指代的前面自己 建立的类，也就是说将该方法中的参数赋给类中的成员变量*/
        this.width = width;
        this.height = height;
    }

    /*在成员类中定义 public 类型的方法，用于计算矩形的周长*/
    public double perimeter(double width, double height) {
        /*在 java 中函数的返回仍然使用 return*/
        return 2 * (width + height);
    }

    /*定义一个 public 的方法，用于计算矩形的面积*/
```

```
    public double area(double width, double height) {
        return width * height;
    }
}
```

【流程图】



【实验运行截图】



6. 创建三角形、梯形和圆形的类封装

【实验源代码】

```
public class Homework4A6 {

        public static void main(String[] args) {
            /*在内存空间中分别给三个类开辟一块内存空间，即对这三个类进行
实例化，每个类分 别构造一个对象并给该对象初始化*/
            Triangle T = new Triangle(7,8,9);
            Lader L = new Lader(4,7,2);
```

```
            Circle C = new Circle(2);
            /*三角形类的对象 T 调用三角形类 Triangle 的 perimeter
方法计算三角形的周长，调用 area 方法计算三角形的面积并输出*/
            System.out.println("三角形的周长为： " +
T.perimeter());
            System.out.println("三角形的面积为： " + T.area());
            /*三角形类的对象 T 调用三角形类 Triangle 的 modify 方法
修改三角形的三边*/
            T.modify(5, 6, 7);
            /*重新使用三角形类的对象 T 调用三角形类 Triangle 的
perimeter 方法计算三角形的周 长，调用 area 方法计算三角形的面积并输出
*/
            System.out.println("修改三边后三角形的周长为： " +
T.perimeter()); System.out.println("修改三边后三角形的面积为： " +
T.area());
            /*又因为在三角形类定义的三角形求周长和面积的方法中，首先就对
三条边进行了判断， 然后在进行计算，此处是为了验证判断语句的正确性*/
            T.modify(7, 3, 2);
            System.out.println("修改三边后不能构成一个三角形");
            /*不能够成三角形的话，两个方法都会报错*/
            System.out.println(T.area() + T.perimeter());
            /*梯形类的对象 L 调用三角形类 Lader 的 area 方法计算梯形
的面积并输出*/
            System.out.println("梯形的面积为： " + L.area());
            /*圆形类的对象 C 调用三角形类 Circle 的 perimeter 方法计算
```

```
圆形的周长，调用 area  方 法计算圆形的面积并输出*/
            System.out.println("圆形的周长为： " + C.perimeter());
            System.out.println("圆形的面积为： " + C.area());
        }
    }

    /*在我的类中定义三角形类作为成员类*/
    class Triangle{

        /*定义三个双精度浮点数用于存储三角形的三边长*/
        double a,b,c;
        /*定义布尔变量，用于判断三条边是否构成三角形*/
        boolean flag;
        /*与类名相同，还带括号，那这是构造方法，该构造方法的参数是三角形
的三条边*/
        public Triangle(double a,double b,double c) {
            this.a = a;
            this.b = b;
            this.c = c;
            /*首先判断输入的三条边是否构成三角形*/
            /*三角形的额判断条件是两边之和大于第三边*/
            if(a+b>c&&a+c>b&&c+b>a) {
                /*若是一个三角形，那么布尔型变量就为 1，表示正确*/
                flag = true;
            }
            else{
                flag = false;
            }
        }
        /*定义方法 perimeter()  ，该方法的返回值是双精度浮点数*/
```

```
        double perimeter() {
            /*首先根据 flag 的值判断是否能构成三角形*/
            if(flag) {
                /*在构造函数中，已经把构造函数的局部变量赋给了哥哥小明，
所以说这个小红也可 以使用哥哥小明的值*/
                return a + b + c;
            }


            /*若不能构成三角形的情况，就老实的输出不行*/
            else {
                System.out.println("三角形的周长为:此三边的值不能构
成一个三角形,无法 计算周长");
                return 0;
            }
        }
        /*定义方法 area，其返回值是双精度浮点数，用于计算三角形的面积*/
/*三角形面积的一般计算公式为
(p=(a+b+c)/2) ,S=sqrt[p(p-a)(p-b)(p-c)]*/
        double area() {
            if(flag) {
                /*计算面积是还是要根据 flag 首先判断是否能够成三角形*/
                double p = (a + b + c) / 2;
                /*调用 Math 类的 sqrt 方法计算平方根*/
                return Math.sqrt(p*(p-a)*(p-b)*(p-c));
            }
            /*遇到不满足的情况，就老实的输出不行即可*/
            else {
                System.out.println("三角形的面积为:此三边的值不能构
成一个三角形,无法 计算面积");
                return 0;
```

```
            }
        }
        /*定义方法，其返回值为 0，用于修改三角形的三边*/
        void modify(double a,double b,double c) {
            /*依旧是使用 this 来防止屏蔽掉哥哥小明，并且把调用函数获得
的形参（儿子小明）的值赋给哥哥小明*/
            this.a = a;
            this.b = b;

            this.c = c;
            /*只要三边的值发生了改变，那么就要重新判断一下是否还能构成三
角形*/
            if(a+b>c&&a+c>b&&c+b>a) {
                flag = true;
            }
            else {
                flag = false;

            }
        }
    }

    /*成员类*/
    class Lader{
        /*在这个类下定义了三个双精度浮点数，分别用来存放上底、下底和高*/
        double upper,lower;
        double height;
        /*定义该成员类的构造方法，其参数为梯形的上底下底和高*/
        public Lader(double upper,double lower,double height) {
            this.upper = upper;
            this.lower = lower;
            this.height = height;
        }
```

```
    /*定义该类的方法，其返回值是双精度浮点数，用来计算梯形的面积*/
    double area() {
        /*梯形的面积公式为（上底加下底）乘高除以二*/
        return (upper + lower)*height / 2;

    }
}

/*在定义圆形成员类*/
class Circle{
    double radius;
    /*定义构造方法*/
    public Circle(double radius) {
        /*使用 this 代指圆类，将构造方法生成的半径值交给哥哥小明*/
        this.radius = radius;
    }
    /*定义方法，该方法的返回值是双精度浮点数，用来计算圆的周长*/
    double perimeter() {
        return (2 * 3.14 * radius);
    }
    /*定义方法，该方法的返回值是单精度浮点数，用来计算圆的面积*/
    float area() {
        return (float) (3.14 * radius * radius);
    }
}
```

【运行结果截图】

```
Homework4A6
"C:\Program Files\Eclipse Foundation\jdk-8.0.302.8-hotspot\bin\java.exe" ...
三角形的周长为:  24.0
三角形的面积为:  26.832815729997478
修改三边后三角形的周长为:  18.0
修改三边后三角形的面积为:  14.696938456699069
修改三边后不能构成一个三角形
三角形的面积为:此三边的值不能构成一个三角形,无法 计算面积
三角形的周长为:此三边的值不能构成一个三角形,无法 计算周长
0.0
梯形的面积为:  11.0
圆形的周长为:  12.56
圆形的面积为:  12.56

进程已结束,退出代码0
```

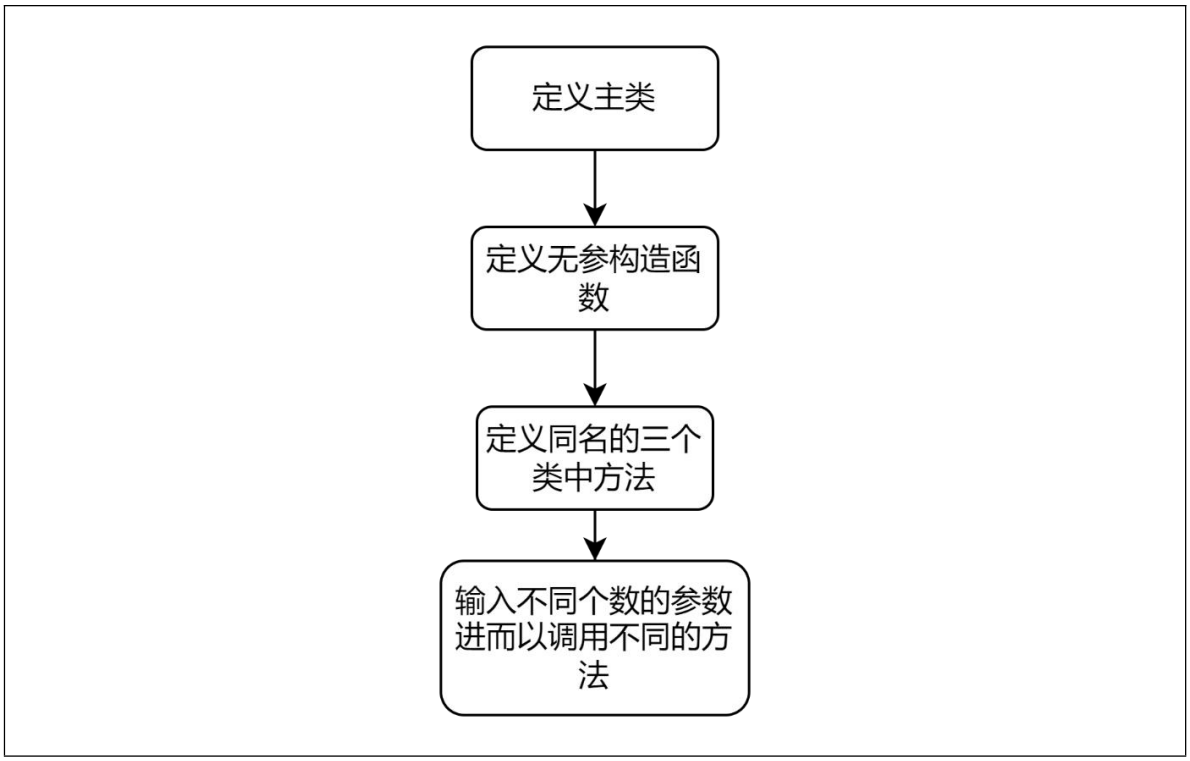
7. 编写一个程序，定义一个类，类有 3 个 plus ()方法， 分别实现两个数相加，三个数相加，四个数相加的程序。

【实验代码】

```
public class Homework4A7 {
    public static void main(String[] args) {
        /*因为要实现相加，所以要定义一个整型数据存放和，并且这个整型变量定义在主类中， 是最大的小明*/
        int sum;
        /*使用 new 在堆中新建一块内存区域，并生成引用变量*/
        Homework4A7 g = new Homework4A7 ();
        /*靠输入参数的不同调用三个同名的不同函数，分别实现两个数、三个数和四个 数的相加*/
        sum = g.plus(1,2);
        System.out.println("两个数的和:  " + sum);
        sum = g.plus(1,2,3);
        System.out.println("三个数的和:  " + sum);
        sum = g.plus(1,2,3,4);
        System.out.println("四个数的和:  " + sum);
    }
```

```
    /*定义方法，下面的三个方法只是参数不同，实现了重载*/

    /*第一个方法的返回值是整形，参数为两个整形变量，用于计算两个数的和*/
    int plus(int a,int b)
    {
        return a + b;
    }
    /*第二个方法的返回值是整形，参数为三个在整型变量，用于计算三个整数的
和*/
    int plus(int a,int b,int c)
    {
        return a + b + c;
    }
    /*第三个函数的返回值是整形，参数为四个整型变量，用于计算四个整形数据
的和*/
    int plus(int a,int b,int c,int d)
    {
        return a + b + c + d;

    }
}
```

【实验流程图】



【实验运行截图】

```
"C:\Program Files\Eclipse Foundation\jdk-8.0.302.8-hotspot\bin\java.exe" ...
两个数的和: 3
三个数的和: 6
四个数的和: 10

进程已结束,退出代码0
```

8. 编译并运行给定的程序， 观察分析运行结果， 体会程序 super 和 this 的用法， 进一步 理解变量隐藏和方法重写的概念

【实验代码】

```
class Person{
    /*该类只有一个成员变量--整型数据*/
    int weight;
    /*使用该类的构造方法构造一个小明出来*/
    Person(){
        weight=50;
    }
    /*定义该类的方法，其返回值为空*/
    void printPerson() {
```

```
        /*该方法就只有一个功能，就是输出，输出方法的名字（因为加了引号）
*/
        System.out.println("Person.printPerson()");


    }
}
/*定义 ZhangSan 类，并且延伸 Person 类，那么 ，ZhangSan 类就是
Person 类的子类*/
class ZhangSan extends Person{
    /*注意到，Person 类中也有一个小明叫 weight，那么在使用这个成员变量
的时候，父类 的 weigth 就被默认屏蔽了*/
    int weight;
    /*定义 ZhangSan 类的构造方法*/
    ZhangSan(){
        /*ZhangSan 类的构造方法，调用了父类的构造方法，因为在这里
super 指代父类，与类 同名还带括号的就是该类的构造方法*/
        super();
        /*ZhangSan 的构造方法构造了一个 ZhangSan 类对象 zhangsan 的
成员变量，并赋值 400*/
        weight=400;
    }
    /*定义该类的方法 ，其功能也是只有一个就是输出方法的名字（因为加了引
号） */ /*并且该方法的名称和签名和父类一样，因此在这里实现了对父类方法的
重写*/
    void printPerson() {
        System.out.println("ZhangSan.printPerson()");
    }
    /*定义该类的另一个方法，返回值为空，作用是显示 super 和 this 的区别
*/
```

```
    void superThisUseDemo() {
        /*在这个方法中又定义了一下 weight，并且和两个类的成员变量都同名，因为子类屏蔽 掉了父类的该成员变量，二该方法屏蔽掉了子类的改成员变量，所以下面使用 super 来 解除对父类屏蔽，用 this 解除对子类的屏蔽*/
        int weight;
        weight=1000;
        /*使用 super 指向父类，调用父类的方法*/
        super.printPerson();

        /*这里是调用自己的方法*/
        printPerson();
        /*输出父类的小明*/
        System.out.println("super.weight="+super.weight);
        /*输出方法所在类，即子类的小明*/
        System.out.println("this.weight="+this.weight);
        /*输出方法中的 weight*/
        System.out.println("weight="+weight);
    }
}
/*这里是 public 类，程序从他的 main 方法开始，尽管他放在了最后*/
public class Homework4A8{
    public static void main(String[] args) {
        /*因为上面的类 ZhangSan 默认权限，多以同一个包中的主类 JavaHomework4A8Hzh 可 以实例化他*/
        /*使用 new 方法实例化类，建立了一个对象 zhangsan*/
        ZhangSan zhangsan=new ZhangSan();
        /*使用该类的对象调用该类的方法，显示 super 和 this 的差异*/
        zhangsan.superThisUseDemo();
    }
}
```

【实验运行截图】

```
"C:\Program Files\Eclipse Foundation\jdk-8.0.302.8-hotspot\bin\java.exe" ...
Person.printPerson()
ZhangSan.printPerson()
super.weight=50
this.weight=400
weight=1000

进程已结束,退出代码0
```

super 用来指代被子类继承的父类，而 this 则指点成员类所属的对象

变量隐藏和方法重写发生在两个类之间，即存在继承的情况下。若子类和父类的成员变 量相同，则默认隐藏父类的成员变量，要想使用就得加一个 super 来引用。子类和父类 的方法同名且签名一致是，将会进行方法的重写，此时仍然是使用 super 来调用父类被 重写的方法。

9. 指出给定代码的错误并解决

出现的问题：

使用了 String 类，但是没有在开头 import String 类所在的包
在主类中实例化 A 时，括号中没有参数，且类 A 中没有无参的构造函数，所以会实 例化失败

解决办法：

在代码文件开头 import String 类所在的包
在实例化的时候括号中输入一个字符串或在 A 类中重载一个无参的构造函数

【实验代码】

解决方案一

```
import java.lang.String;
public class Homework4A9_1 {
    public static void main(String[] args) {
        /*类 A 和主类不是包含关系也不是继承关系，而是定义在同一个包中，
并且 A 类无修饰符，是默认的权限，只要在一个包中的类都可以对其实例化*/
        A a = new A("I love programming");
        /*调用 A 类的方法，实现输出*/
        a.print();
    }
```

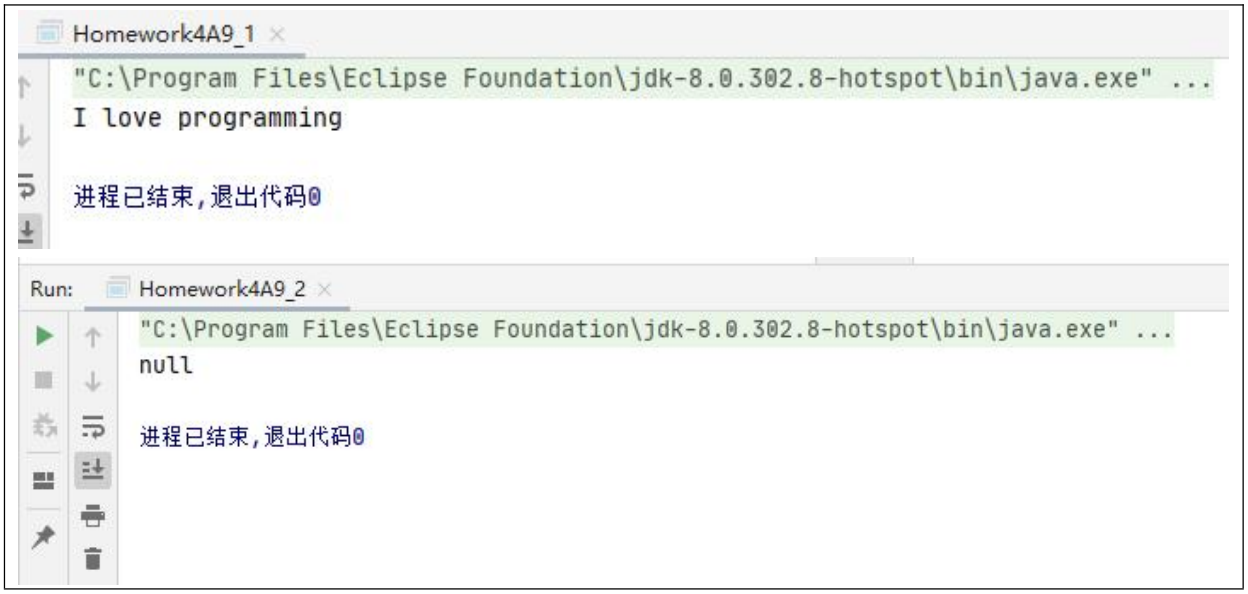
```
}
/*在一个包下定义一个与主类独立的类*/
class A{
    /*该类只有一个成员变量，且为字符串型*/
    String s;
    /*定义该类的构造函数，且该构造函数有参数，那么原来隐藏的无参的构造函数就自己 消失了*/
    A(String s){
        /*使用 this 指代该方法所在的对象即 A 类，此操作实现了小红中的参数传递出方法，交 给了哥哥小明*/
        this.s = s;
    }
    /*定义 A 类的方法，该方法仅仅用来输出字符串*/
    public void print() {
        System.out.println(s);
    }
}
```

解决方案二

```
public class Homework4A9_2 {
    public static void main(String[] args) {
        /*初始化的时候依旧保持参数为空*/
        B b = new B();
        b.print();
    }
}
class B {
    String s;
    /*重载一个无参的构造函数*/
    B() {
    }
    B(String s) {
        this.s = s;
    }
```

```
    public void print() {
        System.out.println(s);
    }
}
```

【运行结果】



10. 熟悉 Date 类和 Calendar 类的成员方法

Date 类

使用 Date 类代表当前系统时间：

常用于 Date date = new Date(); System.out.println(date);

即使用该类的构造方法创建的对象，代表了当前的时间，并且当前的时间可以直接输出 使用带参数的构造函数时，可以构造指定日期的 Date 类对象， Date 类中年份的参数应 该是实际需要代表的年份减去 1900，实际需要代表的月份减去 1 以后的值

Calendar 类：

Calender 类是一个抽象类， 需要使用 getInstance()方法创建实例

使用 Calendar 类代表当前的时间

Calendar c = Calendar.getInstance(); System.out.println(c)--注意返回的是一个对象

set 方法：设置年月 日参数。注意其参数的结构和 Date 类不一样。 Calendar 类中年份的 数值直接书写，月份的值为实际的月份值减 1，日期的值就是实际的日期值

若是只想要设设定某一个参数的值，这是 set 方法的参数变成两个，第一个表示要

设定 的参数

YEAR	年	MINUTE	分	DAY_OF_WEEK_IN_MONTH	某月中第几周
MONTH	月	SECOND/MILLISECOND	秒/毫秒	WEEK_OF_MONTH	日历式的第几周
DATE	日	DAY_OF_MONTH	和DATE一样	DAY_OF_YEAR	一年的第多少天
HOUR_OF_DAY	时	DAY_OF_WEEK	周几	WEEK_OF_YEAR	一年的第多少周

只需要设置一个值即可，其他的数值将会自动计算并修改

get 方法：用于获取，与 set 相反

add 方法：该方法的作用是在 Calendar 对象中的某个字段上增加或减少一定的数值，增 加是 amount 的值为正，减少是 amount 的值为负。

after(before)方法：用于判断当前日期的对象是都在 when 对象的后边（前边），其返回 值是一个布尔型变量

关于 Calendar 类中的星期

数组{1,2,3,4,5,6,7}分别表示{星期天、星期一、星期二、……、星期六}

所以获得了当天在本周是第一天之后，需要再减 1，才是星期几，所以星期天就是星期 0 了

【实验代码】

```
/*import 不同的包，加*表示声明这个包中所有的类*/
import java.util.*;
import java.io.*;
public class Homework10_1 {
    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance();
        /*Date 类封装了系统的日期和时间信息，Calendar 类则会根据系统日历

解释 Date 对象*/

        /*此处使用了 new 创建了一个 Date 的匿名类，并且将*/
        calendar.setTime(new Date());
```

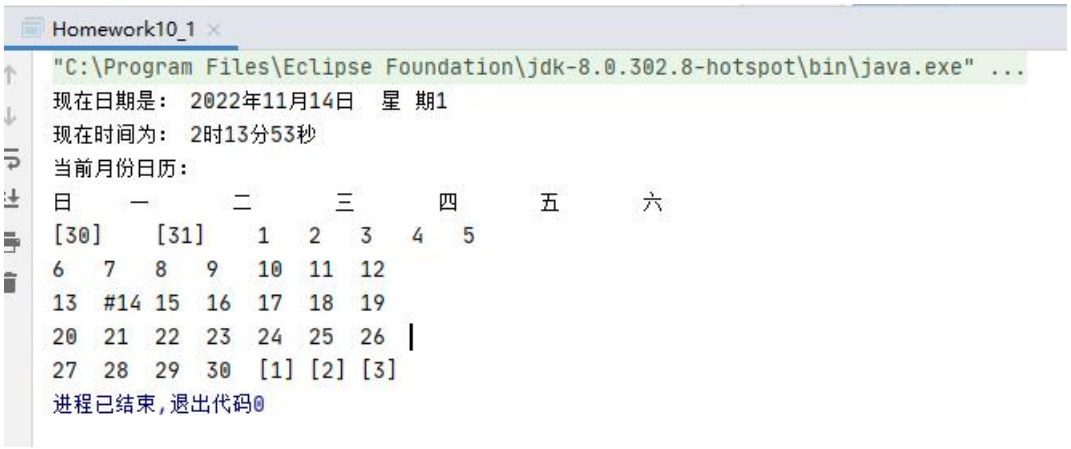
```
        /*使用 get 方法获取对象中年份的参数值，并将其存储在整型变量 year 中
*/

        int year = calendar.get(Calendar.YEAR);
        /*使用 get 方法获取月份，因为生成对象的月份值是实际值减 1，所以想得
到实际 值，就要加 1*/
        int month = calendar.get(Calendar.MONTH)+1;
        /*使用 get 方法获取本月中的第几天，并将其存储在整型变量 day 中*/
        int day = calendar.get(Calendar.DAY_OF_MONTH);
        /*获取星期几，这里的星期几，指的是一周中的第几天，因为周从周日开始，
所以周一其实是本周第二天*/
        int week = calendar.get(Calendar.DAY_OF_WEEK)- 1;
        /*获取时分秒*/
        int hour = calendar.get(Calendar.HOUR);
        int minute = calendar.get(Calendar.MINUTE);
        int second = calendar.get(Calendar.SECOND);


        /*使用 set 方法设置日期为 1，因为在这一天比较好设置前一个月和本月*/
        calendar.set(Calendar.DATE, 1);
        /*根据传入的参数代表的意思(年、月、周等) 查询当前 (年、月、周) 拥有
的最大 值。 如果是年就查询这一年中的天数， 如果月份就查询当前月中的天数， 如
果是周就查 询当前周的天数*/

        /*此处是查询当前月的最大天数*/
        int monthAllDay =
calendar.getActualMaximum( Calendar.DAY_OF_MONTH );
        /*获取当天的星期数，还是当天在当周的天数减 1*/
        int dayOfWeek = calendar.get(Calendar.DAY_OF_WEEK) - 1;
        /*调用方法实现输出当前的日期，因为前面把天数赋给了整型变量，所以就算
```

```
是改变了 日期，也可以使用该整型变量正确的输出日期*/
        System.out.println("现在日期是： " + year + "年" + month + "
月" + day + "日  星 期" + week);
        /*输出当前的时间*/
        System.out.println("现在时间为： " + hour + "时" + minute +
"分" + second + "秒 ");
                /*输出日历*/
                System.out.println("当前月份日历： ");
        /*输出表头，每个周从周日开始*/
        System.out.println("日 \t 一 \t 二 \t 三 \t 四 \t 五 \t
六");
        /*将本年的值还是作为本年*/
        calendar.set(Calendar.YEAR, year);
        /*month 中的值是本月实际值，减去 1 是上个月的实际值，再减去 1 就是
上个月的实际 值在 Calendar 中的存储值*/
        /*因为上个月的天数可能会根据年数而改变（比如二月），所以设置上个月
的时候还是 要保持年份不变的*/
        calendar.set(Calendar.MONTH, month-2);
        /*将月份设置为上个月之后，使用 getActualMaximum  方法获取上个月
的最大天数*/ /*这一步求出的是上个月的最大天数，也就是日历中本月一号前面的那
个【数字】*/
        int foreMaxDay =
calendar.getActualMaximum(Calendar.DAY_OF_MONTH);
        /* dayOfWeek  中存的是下个月 1 号是星期几*/
```

```
        int fdk = dayOfWeek;
              for (int k = 1; k <= dayOfWeek; k++) {
            /*本月 1 号是星期 N，那么上个月最后一天不就是星期 N- 1，所以是
fdk--*/
            fdk --;
            System.out.print("[" + (foreMaxDay-fdk) + "]" + "\t");
        }
        /*设置完上个月之后，再回到正确的月份，设置本月*/
        calendar.set(Calendar.MONTH, month- 1);
        /*使用 for 循环，输出本月的日期，且本月有几天，就循环多少次*/
        for (int i = 1; i <=monthAllDay; i++) {
            if( i == day ) {
                System.out.print("#");
            }
            /*每输出一个日期之后，就要空格*/
            System.out.print(i+"\t");
            int w =calendar.get( Calendar.DAY_OF_WEEK ) - 1;
            /*遇到周六就要换行*/
            if( w == 6 ) {
                System.out.println();
            }
            /*每循环一次，日期加 1*/
            calendar.add(Calendar.DATE, 1);
        }
             calendar.set(Calendar.MONTH, month);
        /*取下个月 1 号，将要补齐本月最后一天所在的星期*/
        calendar.set(Calendar.DATE, 1);
        /*获取下个月第一天是星期几*/
        int nextdayOfWeek = calendar.get(Calendar.DAY_OF_WEEK) - 1;
        /*下面是补齐本月最后一天所在星期*/
        /*若下个月第一天是星期 N ，本周的最后一天是星期六，所以需要补 6-N 天
*/
        for(int i = 1;i <= 6 - nextdayOfWeek + 1;i++) {
```

```
            /*直接输出 i 即可，因为补下个月是从 1 号开始补，而补上个月则是
从最后几天开始补 */
            System.out.print("[" + i + "]" + "\t");
        }
    }
}
```

【实验运行截图】



万年历

【实验代码】

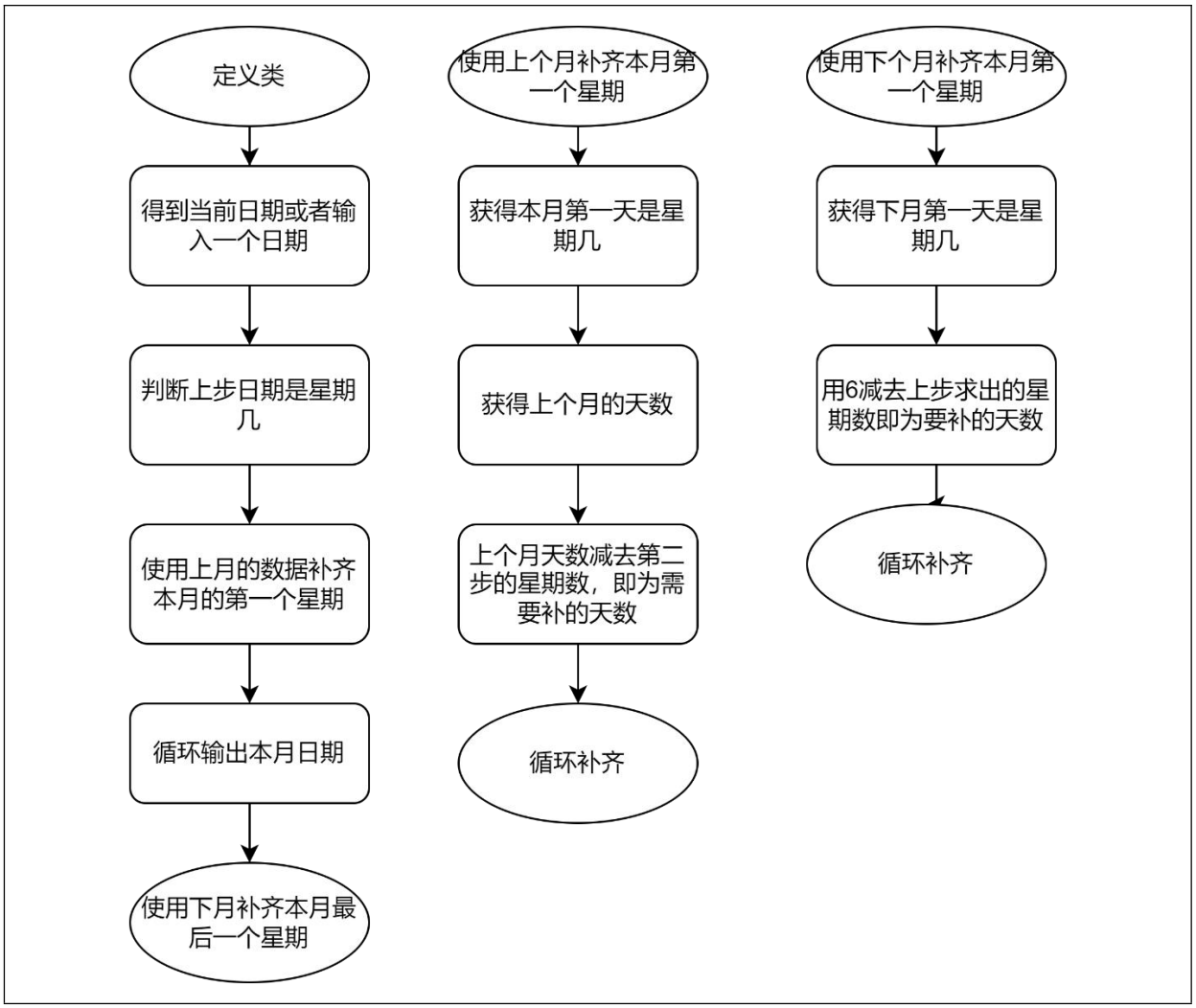
```
/*还是在开头声明一下将要用到的类*/
import java.util.Scanner;
import java.util.Calendar;
import java.util.Date;


public class Homework10_2{
    public static void main(String[] args) {
        /*因为 Calendar 类不能使用 new 实例化，只能使用 getInstance 方
法返回一个对象*/
        Calendar calendar = Calendar.getInstance();
        /*new 一个空间用于存放 Scanner 类的实例，这个实例的构造函数参数是
调用 System 类 in 方法得到的输入*/
        Scanner sc = new Scanner(System.in);
```

```
        System.out.print("请输入年份： ");
        /*调用 nextInt 方法得到整形数据，并将其存储在 y 中，y 就代表年份
*/
        int y = sc.nextInt();
        System.out.print("请输入月份： ");
        /*调用 nextInt 方法得到整形数据，并将其存在 m 中，m 就代表月份*/
        int m = sc.nextInt();
        /*使用 Calendar 类的 set 方法将年份设置成输入的年份*/
        calendar.set(Calendar.YEAR, y);
        /*使用 Calendar 类的 set 方法，将月份设置成 m- 1，这是因为存储值
是实际值减 1*/
        calendar.set(Calendar.MONTH, m- 1);
        /*然后下面的操作就和上个实验一样了，无非就是把用户输入得到日期当做当
前的 日期，然后使用上个月补齐第一周，再使用下个月补齐第二周即可*/
        /*前面已经设置好了年月，在设置日期的本月第一天*/
        calendar.set(Calendar.DATE, 1);
        /*获取本月有多少天*/
        int monthAllDay =
calendar.getActualMaximum( Calendar.DAY_OF_MONTH );
        /*获取今天是星期几（一星期中的第一天减去 1） */
        int dayOfWeek = calendar.get(Calendar.DAY_OF_WEEK) - 1;
        /*开始进行日历的输出*/
        System.out.println("当前月份日历： ");
        /*输出表头*/
        System.out.println("日 \t 一 \t 二 \t 三 \t 四 \t 五 \t
```

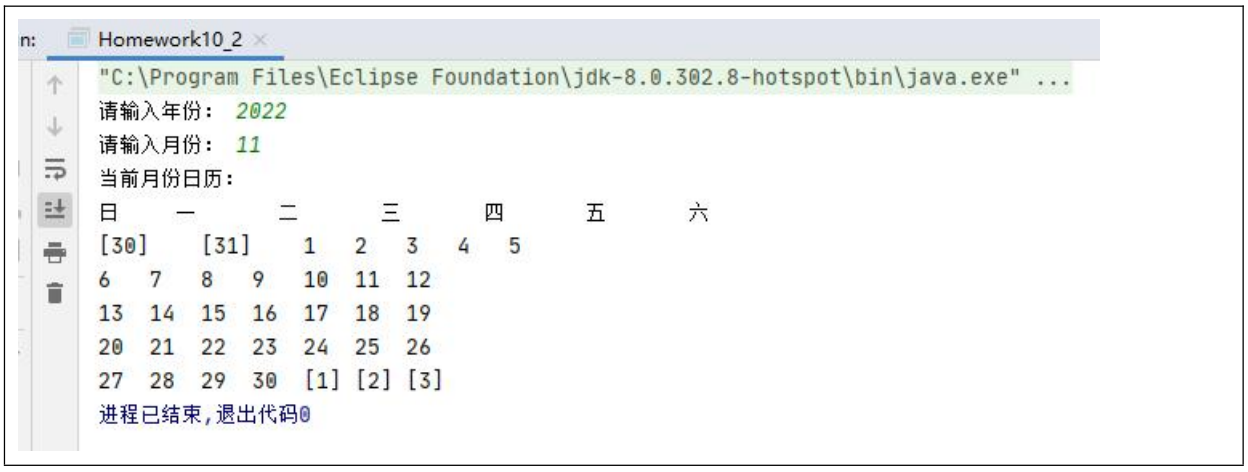
```
六");

        /*使用 set 方法保留年份不变，将月份调整为本月的上个月 */
        calendar.set(Calendar.YEAR, y);
        calendar.set(Calendar.MONTH, m-2);
        /*使用 getActualMaximum  方法获得上月最大天数，也就是本月 1 号之
前的那个数字*/
        int foreMaxDay =
calendar.getActualMaximum(Calendar.DAY_OF_MONTH);
        /*记录下本月第一天是星期几*/
        int fdk = dayOfWeek;




        /*使用 for 循环补齐本月 1 号所在星期。假设本月 1 号是星期 N，那么
他在所在星期 是第 N+1 天，所以前面需要使用上个月补齐 N 天*/
        for (int k = 1; k <= dayOfWeek; k++) {
            fdk --;
            /*输出是顺序输出，所以从星期零开始补，星期令的日期就是上个月最大
的日期减去需 要补的日期数，然后 fdk 减 1 ，得到后一天*/
            System.out.print("[" + (foreMaxDay-fdk) + "]" + "\t");


        }
        /*再输出原来的月，即将实际月份减去 1 即可，再使用 set 函数即可设置
*/
        calendar.set(Calendar.MONTH, m- 1);
        /*当月有几天，就循环几次，将本月的所有日期按顺序输出*/
        for (int i = 1; i <=monthAllDay; i++) {
            System.out.print(i+"\t");
            /*在这里不需要加个#代表当前的天，因为一开始想要查询的是某个月
```

```
的，而不是具体到 某一天*/
            /*另外，还是需要使用一个变量存放当前是星期几，以便在输出完星期六
之后换行*/
            int w =calendar.get( Calendar.DAY_OF_WEEK ) - 1;
            if( w == 6 ) {
                System.out.println();
            }
            /*每循环一次，就将日期加 1 ，用于通过星期判断是否换行*/
            calendar.add(Calendar.DATE, 1);
        }
        /*本月输出完之后，就要再考虑下个月了，下个月即是 m+1，在类中的存放
值就是 m*/
        calendar.set(Calendar.MONTH, m);
        /*将当前的日期设置成 1 号，因为下个月就是前几天来补齐*/
        calendar.set(Calendar.DATE, 1);


        /*获取下个月 1 号的星期数*/
        int nextdayOfWeek = calendar.get(Calendar.DAY_OF_WEEK) - 1;
        /*因为每一个星期最大是星期六，所以就用 6 减去下月 1 号的星期数，即
可得到需要 补齐的天数*/
        if(nextdayOfWeek != 0) {
            for(int i = 1;i <= 6 - nextdayOfWeek + 1;i++) {
                System.out.print("[" + i + "]" + "\t");
            }
        }
    }
}
```

【实验流程图】



【运行结果截图】



【实验反思】

本次实验是大学以来经历的最多的一次实验，一共花费了一周时间去做、查、和同学讨论，通过实践越发觉得 Java 和 C#以及 python 三门语言中的共通之处以及面向对象编程的优越性和强大性，本次实验的任务主要是熟悉 Java 中的字符串操作、数组操作、

以及类的有关操作，虽然我没有做过 Java 有关的项目，只是有一些小型的作业作为练手。但是这学期我的 C#作业上深刻体会到了这些基础知识的重要性，比如我们在部署深度学习的 Onnx 模型的时候就是通过初始化四维数组，然后遍历数组得到，还有在显示病人 X 光影响分割结果的同时也需要传出病人的身份证号、姓名、性别以及 Vgg 分类模型的结果，就用到了字符串分割的知识，还有在 python 的深度学习的 pipeline 代码中也学习到了一些实用的类操作，比如注册器、装饰器等语法。

通过从本次实验我对面向对象编程又有了更新一层的认识：

当解决一个问题的时候，面向对象会把事物抽象成对象的概念，就是说这个问题里面有哪些对象，然后给对象赋一些属性和方法，然后让每个对象去执行自己的方法，问题得到解决。易维护、易复用、易扩展，由于面向对象有封装、继承、多态性的特性，可以设计出低耦合的系统，使系统 更加灵活、更加易于维护。

也了解到了面向对象编程的三大基本特性：封装，继承，多态

封装，就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。一个类就是一个封装了数据以及操作这些数据的代码的逻辑实体。在一个对象内部，某些代码或某些数据可以是私有的，不能被外界访问。通过这种方式，对象对内部数据提供了不同级别的保护，以防止程序中无关的部分意外的改变或错误的使用了对象的私有部分。

继承，指可以让某个类型的对象获得另一个类型的对象的属性的方法。它支持按级分类的概念。继承是指这样一种能力：它可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展。 通过继承创建的新类称为“子类”或“派生类”，被继承的类称为“基类”、“父类”或“超类”。继承的过程，就是从一般到特殊的过程。要实现继承，可以通过 “继承”（Inheritance）和“组合”（Composition）来实现。继承概念的实现方式有二类：实现继承与接口继承。实现继承是指直接使用父类的属性和方法而无需额外编码的能力；接口继承是指仅使用属性和方法的名称、但是子类必须提供实现的能力。

多态，是指一个类实例的相同方法在不同情形有不同表现形式。多态机制使具有不同内部结构的对象可以共享相同的外部接口。这意味着，虽然针对不同对象的具体操作不同，但通过一个公共的类，它们（那些操作）可以通过相同的方式予以调用。