**信息科学与工程学院**

**2020－2021学年第一学期**

实 验 报 告

课程名称： Java编程技术

实验名称： 实验四 **(A)**

专 业 班 级 通信工程 二班

学 生 学 号 201800121050

学 生 姓 名 孟麟芝

实 验 时 间 2020年11月13日

实验报告

【实验目的】

1.继续熟练使用Java实现一些简单的算法。

2.掌握Java中类、对象等概念。

3.掌握GUI设计的基本方法。

4.掌握对数组、字符串等的操作。

【实验要求】

一、数组与字符串

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.编写一个程序，完成以下功能：

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。

二、类与对象

1. (1)创建一个Rectangle类，添加两个(private）属性width 与height。

(2)创建一个构造方法，选择width 与height为形式参数。

(3)在Rectangle 类中添加两个方法计算矩形的周长和面积。

2.创建三角形、梯形和圆形的类封装:分别定义三角形、梯形和圆形类，要求每个类中包含构造方法、求周长及面积的方法。最后在主方法main中为三角形、梯形和圆形类创建对象并打印输出各种图形的周长及面积。(一个文件中包含三个class和main)

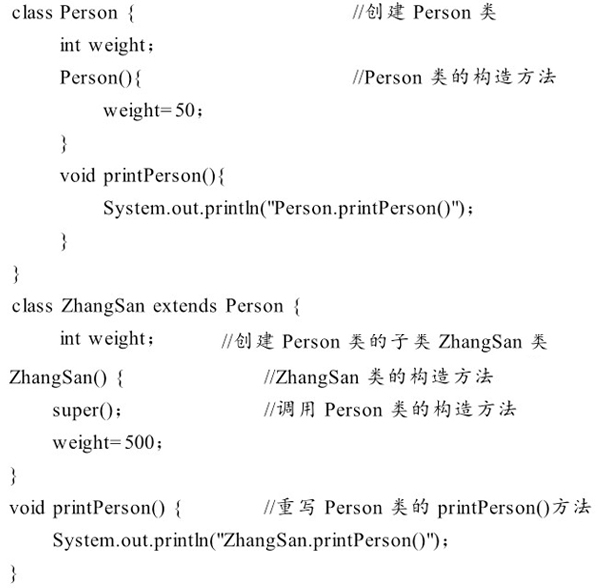
a) Trangle类具有类型为double 的三个边，以及周长、面积属性，Trangle类具有返回周长、面积以及修改三个边的功能。另外，Trangle类还具有一个boolean型的属性，该属性用来判断三个属能否构成一个三角形。

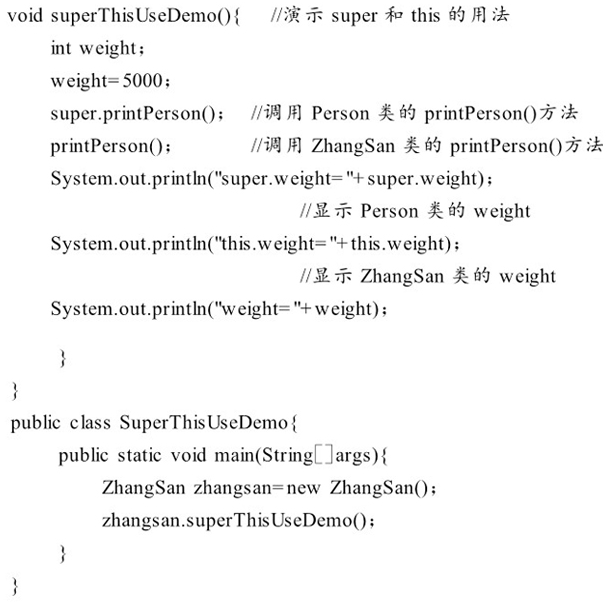
b) Lader类有类型double 的上底、下底、高、面积属性，具有返回面积的功能。

c) Circle类具有类型为double的半径、周长和面积属性，具有返回周长、面积的功能。

3.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。

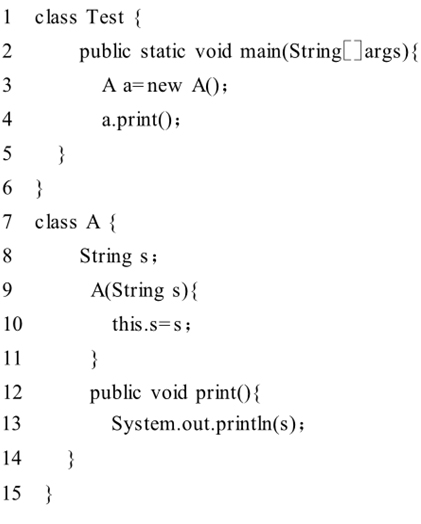
4.编译并运行下面的程序：





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。

5.下面的代码有什么错误，应该如何解决？



6.参考课本日历程序（但需修正），实现输出当前月份日历的程序（输出时要考虑上个月和下个月的天数情况），并将此程序改为万年历，可以任意输出指定月份的日历。

## 【数组与字符串】

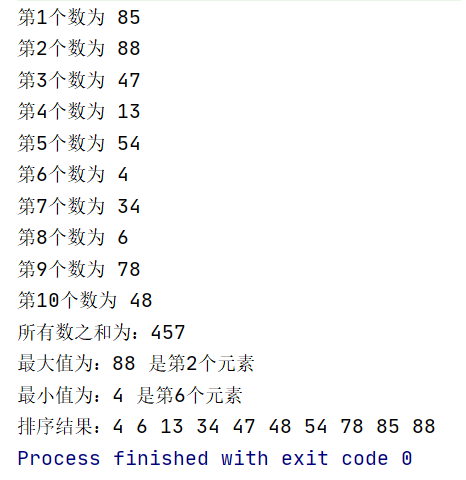
## 【第一个实验】

1.实验源代码

package arrays;  
  
import java.util.Arrays;  
import java.util.Random;  
  
public class SortArray {  
 public static void main(String[] args) {  
 int[] arr = new int[10];  
 int sum = 0;  
 Random randomNumber = new Random(1);  
 for (int i = 0; i < 10; i++) {  
 arr[i] = randomNumber.nextInt(100);  
 sum += arr[i];  
 System.*out*.println("第" + (i + 1) + "个数为 " + arr[i]);  
 }  
 System.*out*.println("所有数之和为：" + sum);  
 *getMaxAndMin*(arr);  
 Arrays.*sort*(arr);  
 System.*out*.print("排序结果：");  
 for (int i : arr) {  
 System.*out*.print(i + " ");  
 }  
 }  
  
 private static void getMaxAndMin(int[] array) {  
 int max = 0, maxPosition = 1, min = 2147483647, minPosition = 1, count = 0;  
 for (int i : array) {  
 count++;  
 if (i >= max) {  
 maxPosition = count;  
 max = i;  
 }  
 if (i <= min) {  
 min = i;  
 minPosition = count;  
 }  
 }  
 System.*out*.println("最大值为：" + max + " 是第" + maxPosition + "个元素");  
 System.*out*.println("最小值为：" + min + " 是第" + minPosition + "个元素");  
 }  
}

2.实验记录与说明分析

实验结果如下图所示，首先程序生成了10个随机数，然后求得了所有数之和，并找到了最大值与最小值及其所在位置。最后，程序实现了排序。



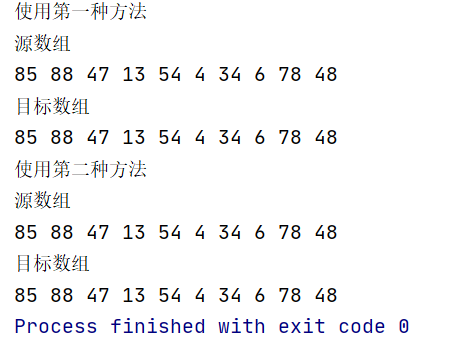
## 【第二个实验】

1.实验源代码

package arrays;  
  
import java.util.Arrays;  
import java.util.Random;  
  
public class CopyArray {  
 public static void main(String[] args) {  
 *//由于调用了该类中的非静态方法，需要先实例化* CopyArray copied = new CopyArray();  
 int[] sourceArray = new int[10];  
 int[] targetArray = new int[10];  
 Random randomNumber = new Random(1);  
 for (int i = 0; i < 10; i++) {  
 sourceArray[i] = randomNumber.nextInt(100);  
 }  
 *//两个方法互为重载* System.*out*.println("使用第一种方法");  
 copied.copyArray(sourceArray, targetArray);  
 System.*out*.println("\n使用第二种方法");  
 copied.copyArray(sourceArray);  
 }  
  
 private void copyArray(int[] source, int[] target) {  
 System.*arraycopy*(source, 0, target, 0, 10);  
 System.*out*.println("源数组");  
 for (int i : source) {  
 System.*out*.print(i + " ");  
 }  
 System.*out*.println("\n目标数组");  
 for (int i : target) {  
 System.*out*.print(i + " ");  
 }  
 }  
  
 public void copyArray(int[] source) {  
 int[] target = Arrays.*copyOf*(source, 10);  
 System.*out*.println("源数组");  
 for (int i : source) {  
 System.*out*.print(i + " ");  
 }  
 System.*out*.println("\n目标数组");  
 for (int i : target) {  
 System.*out*.print(i + " ");  
 }  
 }  
}

2.实验记录与说明分析

下面是程序运行的结果，可见使用两种参数进行复制，若参数设置合理则结果是一样的，两种方式各有优劣，arraycopy()方法需要指定源、目标数组及两个数组中的复制起始位置还有复制的长度，参数较多，比较灵活。



copyof()相对而言更加简洁一些，可以与创建数组写成一行。



这两种方法可以根据实际情况灵活选择。

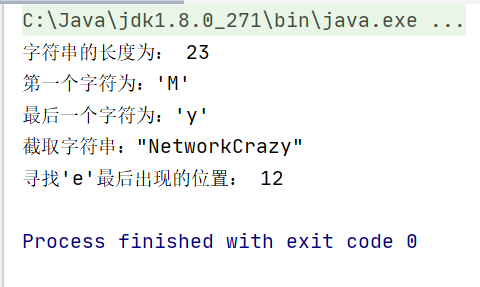
## 【第三个实验】

1.实验源代码

package arrays;  
  
public class OperateString {  
 public static void main(String[] args) {  
 String name = "My name is NetworkCrazy";  
 System.*out*.println("字符串的长度为： " + name.length());  
 System.*out*.println("第一个字符为：'" + name.charAt(0) + "'");  
 System.*out*.println("最后一个字符为：'" + name.charAt(name.length() - 1) + "'");  
 *//substring()索引时包括起始索引，不包括结尾索引* System.*out*.println("截取字符串：\"" + name.substring(11, 23) + "\"");  
 System.*out*.println("寻找'e'最后出现的位置： " + name.lastIndexOf("e"));  
 }  
}

2.实验记录与说明分析

实验结果如下，可见每个字符串其实都是String类实例化出来的一个对象，其中含有许多属性和方法，丰富的字符串操作为后面GUI的制作提供了便利。

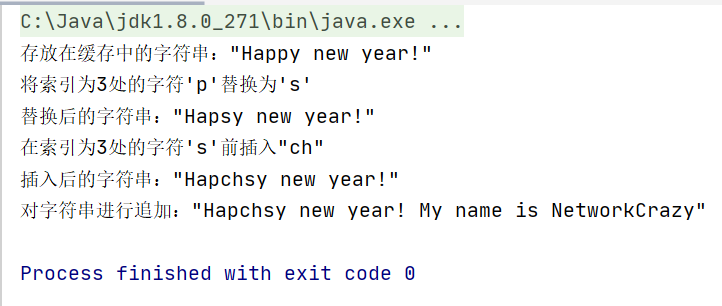


## 【第四个实验】

package arrays;  
  
public class TraInStringBuffer {  
 public static void main(String[] args) {  
 String name = "My name is NetworkCrazy";  
 StringBuffer stringInBuffer = new StringBuffer("Happy new year!");  
 System.*out*.println("存放在缓存中的字符串：\"" + stringInBuffer + "\"");  
 System.*out*.println("将索引为3处的字符'" + stringInBuffer.charAt(3) + "'替换为's'");  
 stringInBuffer.setCharAt(3, 's');  
 System.*out*.println("替换后的字符串：\"" + stringInBuffer + "\"");  
 System.*out*.println("在索引为3处的字符'" + stringInBuffer.charAt(3) + "'前插入\"ch\"");  
 stringInBuffer.insert(3, "ch");  
 System.*out*.println("插入后的字符串：\"" + stringInBuffer + "\"");  
 stringInBuffer.append(" ").append(name);  
 System.*out*.println("对字符串进行追加：\"" + stringInBuffer + "\"");  
 }  
}

2.实验记录与说明分析

下面是实验的结果。



本实验中用到了StringBuffer类，它与String类的区别在于，String的char数组值被声明为final，其值是不可变的，这就导致每次对String的操作都会生成新的String对象，不仅效率低下，而且浪费大量优先的内存空间。

而StringBuffer是可变类，和线程安全的字符串操作类，任何对它指向的字符串的操作都不会产生新的对象。每个StringBuffer对象都有一定的缓冲区容量，当字符串大小没有超过容量时，不会分配新的容量，当字符串大小超过容量时，会自动增加容量

另外还有一个常用的类是StringBuilder类，它的速度更快，但是线程不安全，其总结如下：

|  |  |  |
| --- | --- | --- |
| String | StringBuffer | StringBuilder |
| String的值是不可变的，这就导致每次对String的操作都会生成新的String对象，不仅效率低下，而且浪费大量优先的内存空间 | StringBuffer是可变类，和线程安全的字符串操作类，任何对它指向的字符串的操作都不会产生新的对象。每个StringBuffer对象都有一定的缓冲区容量，当字符串大小没有超过容量时，不会分配新的容量，当字符串大小超过容量时，会自动增加容量 | 可变类，速度更快 |
| 不可变 | 可变 | 可变 |
|  | 线程安全 | 线程不安全 |
|  | 多线程操作字符串 | 单线程操作字符串 |

## 【类与对象】

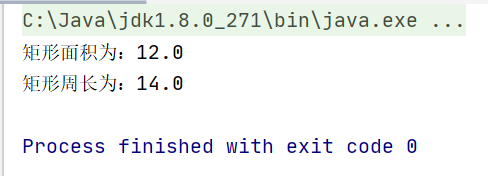
## 【第一个实验】

1.实验源代码

package classes.a;  
  
public class Rectangle {  
 private double width = 0;  
 private double height = 0;  
 double area = 0;  
 double perimeter = 0;  
  
 private Rectangle(double width, double height) {  
 this.width = width;  
 this.height = height;  
 }  
  
 private double getArea() {  
 this.area = this.height \* this.width;  
 return this.area;  
 }  
  
 private double getPerimeter() {  
 this.perimeter = (this.height + this.width) \* 2;  
 return this.perimeter;  
 }  
  
 public static void main(String[] args) {  
 Rectangle rect = new Rectangle(3, 4);  
 System.*out*.println("矩形面积为：" + rect.getArea());  
 System.*out*.println("矩形周长为：" + rect.getPerimeter());  
 }  
}

2.实验记录与说明分析

设置边长为3，4，实验结果如下，可见该类能实现相应功能。

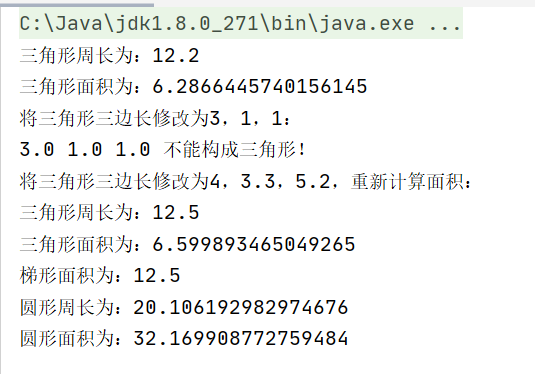


## 【第二个实验】

1.实验源代码

package classes.a;  
  
class Circle {  
 private double radius = 0;  
 double area = 0;  
 double perimeter = 0;  
  
 Circle(double radius) {  
 this.radius = radius;  
 }  
  
 double getArea() {  
 this.area = Math.*PI* \* radius \* radius;  
 return this.area;  
 }  
  
 double getPerimeter() {  
 this.perimeter = 2 \* Math.*PI* \* radius;  
 return this.perimeter;  
 }  
}  
  
class Lader {  
 private double up = 0;  
 private double bottom = 0;  
 private double height = 0;  
 double area = 0;  
  
 Lader(double up, double bottom, double height) {  
 this.up = up;  
 this.bottom = bottom;  
 this.height = height;  
 }  
  
 double getArea() {  
 this.area = (up + bottom) \* height / 2;  
 return this.area;  
 }  
}  
  
class Triangle {  
 boolean judge;  
 private double a = 0;  
 private double b = 0;  
 private double c = 0;  
 double area = 0;  
 double perimeter = 0;  
  
 Triangle(double a, double b, double c) {  
 judgeTriangle(a, b, c);  
 if (this.judge) {  
 this.a = a;  
 this.b = b;  
 this.c = c;  
 }  
 }  
  
 private void judgeTriangle(double a, double b, double c) {  
 if (a + b < c || a + c < b || b + c < a) {  
 System.*out*.println(a + " " + b + " " + c + " " + "不能构成三角形！");  
 this.judge = false;  
 } else {  
 this.judge = true;  
 }  
 }  
  
 double getArea() {  
 double p = getPerimeter() / 2;  
 if (this.judge) {  
 this.area = Math.*sqrt*(p \* (p - this.a) \* (p - this.b) \* (p - this.c));  
 return this.area;  
 } else {  
 return 0;  
 }  
 }  
  
 double getPerimeter() {  
 if (this.judge) {  
 perimeter = this.a + this.b + this.c;  
 return this.perimeter;  
 } else {  
 return 0;  
 }  
 }  
  
 void changeSides(double a, double b, double c) {  
 judgeTriangle(a, b, c);  
 if (this.judge) {  
 this.a = a;  
 this.b = b;  
 this.c = c;  
 } else {  
 this.a = this.b = this.c = 0;  
 }  
 }  
}  
  
public class TestShapes {  
 public static void main(String[] args) {  
 Triangle tri = new Triangle(3, 4.2, 5);  
 Lader lad = new Lader(3, 2, 5);  
 Circle cir = new Circle(3.2);  
 System.*out*.println("三角形周长为：" + tri.getPerimeter());  
 System.*out*.println("三角形面积为：" + tri.getArea());  
 System.*out*.println("将三角形三边长修改为3，1，1：");  
 tri.changeSides(3, 1, 1);  
 System.*out*.println("将三角形三边长修改为4，3.3，5.2，重新计算面积：");  
 tri.changeSides(4, 3.3, 5.2);  
 System.*out*.println("三角形周长为：" + tri.getPerimeter());  
 System.*out*.println("三角形面积为：" + tri.getArea());  
 System.*out*.println("梯形面积为：" + lad.getArea());  
 System.*out*.println("圆形周长为：" + cir.getPerimeter());  
 System.*out*.println("圆形面积为：" + cir.getArea());  
 }  
}

2.实验记录与说明分析



三个类都具有相应的功能，上面的结果是一部分演示，经检验这些结果都是正确的。。

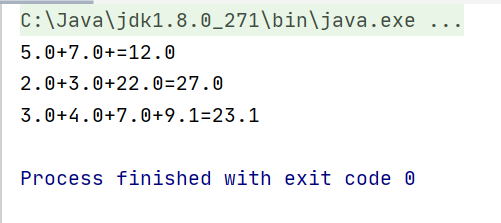
## 【第三个实验】

1.实验源代码

package classes.a;  
  
public class Plus {  
 public static void main(String[] args) {  
 Plus add = new Plus();  
 add.plus(5, 7);  
 add.plus(2, 3, 22);  
 add.plus(3, 4, 7, 9.1);  
 }  
  
 private void plus(double a, double b) {  
 System.*out*.println(a + "+" + b + "+" + "=" + (a + b));  
 }  
  
 private void plus(double a, double b, double c) {  
 System.*out*.println(a + "+" + b + "+" + c + "=" + (a + b + c));  
 }  
  
 private void plus(double a, double b, double c, double d) {  
 System.*out*.println(a + "+" + b + "+" + c + "+" + d + "=" + (a + b + c + d));  
 }  
}

2.实验记录与说明分析

这一个实验使用了方法的重载，重载要求方法名相同，方法参数不同，对方法的返回值没有要求。

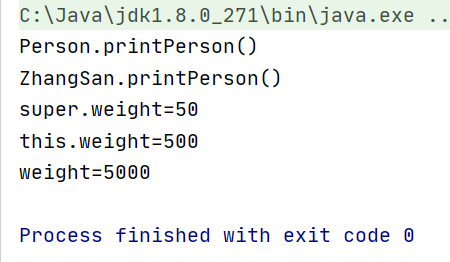


## 【第四个实验】

1.实验源代码

package classes.a;  
  
class Person {  
 int weight;  
  
 Person() {  
 weight = 50;  
 }  
  
 void printPerson() {  
 System.*out*.println("Person.printPerson()");  
 }  
}  
  
class ZhangSan extends Person {  
 int weight;  
  
 ZhangSan() {  
 *//这里其实不写也会自动调用父类的构造方法* super();  
 weight = 500;  
 }  
  
 void printPerson() {  
 System.*out*.println("ZhangSan.printPerson");  
 }  
  
 void superThisUseDemo() {  
 int weight;  
 weight = 5000;  
 super.printPerson();  
 printPerson();  
 *//这里的weight是Person类中的成员* System.*out*.println("super.weight=" + super.weight);  
 *//这里的weight是ZhangSan类中的成员* System.*out*.println("this.weight=" + this.weight);  
 *//这个weight是函数体内的局部变量* System.*out*.println("weight=" + weight);  
 }  
}  
  
public class SuperThisUseDemo {  
 public static void main(String[] args) {  
 ZhangSan zhangsan = new ZhangSan();  
 zhangsan.superThisUseDemo();  
 }  
}

2.实验记录与说明分析



从结果中可以看出，子类中可以通过super关键字调用父类中的成员变量和方法，this关键字则代表本类中的成员，而不加任何修饰则本类中的成员也有可能被同名的局部变量隐藏。

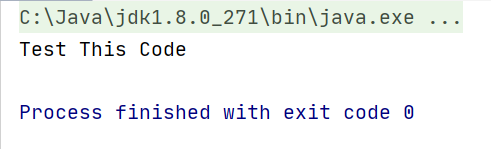
## 【第五个实验】

1.实验源代码

package classes.a;  
  
class Test {  
 public static void main(String[] args) {  
 *//这里源代码没有赋值，无法成功构造* A a = new A("Test This Code");  
 a.print();  
 }  
}  
  
class A {  
 String s;  
  
 A(String s) {  
 this.s = s;  
 }  
  
 public void print() {  
 *//这里的s改为this.s更好* System.*out*.println(s);  
 }  
}

2.实验记录与说明分析

A类中由于显式的定义了一个含参构造方法，则编译器不会自动加入无参构造方法，程序中对A实例化时应加上参数，其类型为String类型，然后即可得到正确结果。



## 【第六个实验】

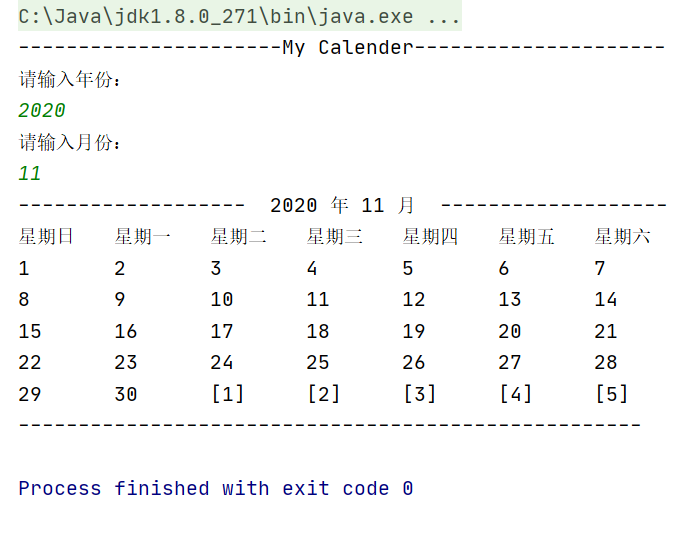
【控制台版本】

1.实验源代码

package application;  
  
import java.io.BufferedReader;  
import java.io.InputStreamReader;  
import java.util.Calendar;  
  
public class MyCalendar {  
 private void printCalendar(int year, int month) {  
 Calendar calendar = Calendar.*getInstance*();  
 *//参数需要月份从0开始记，故-1* calendar.set(year, month - 1, 1);  
 *//下面要寻找该月第一天是周几  
 //1：周日 2：周一 3：周二 4：周三 5：周四 6：周五 7：周六* int judgeFirstDay = calendar.get(Calendar.*DAY\_OF\_WEEK*);  
 *//获取该月天数* int daysOfThisMonth = calendar.getActualMaximum(Calendar.*DAY\_OF\_MONTH*);  
 *//获取该月最后一天是周几* calendar.set(year, month - 1, daysOfThisMonth);  
 int judgeLastDay = calendar.get(Calendar.*DAY\_OF\_WEEK*);  
 *//获取上月天数* calendar.set(year, month - 2, 1);  
 int daysOfLastMonth = calendar.getActualMaximum(Calendar.*DAY\_OF\_MONTH*);  
 *//下面作输出操作  
 //输出表头* System.*out*.println("------------------- " + year + " 年 " + month + " 月 " + " -------------------");  
 System.*out*.println("星期日\t星期一\t星期二\t星期三\t星期四\t星期五\t星期六");  
 *//输出上个月的后几天* for (int i = 1; i < judgeFirstDay; i++) {  
 System.*out*.print("[" + (daysOfLastMonth - judgeFirstDay + i) + "] ");  
 }  
 *//本月则依次输出即可* for (int i = 1; i <= daysOfThisMonth; i++) {  
 *//找到第i天* calendar.set(year, month - 1, i);  
 *//判断第i天是周几* int judgeDay = calendar.get(Calendar.*DAY\_OF\_WEEK*);  
 System.*out*.print(String.*format*("%-2d", i) + " ");  
 *//每到周六则要换行* if (judgeDay == 7) {  
 System.*out*.println();  
 }  
 }  
 *//下月的初几天也要输出* for (int i = 1; i < 7 - judgeLastDay + 1; i++) {  
 System.*out*.print("[" + i + "] ");  
 }  
 *//输出结尾* System.*out*.println("\n----------------------------------------------------");  
 }  
  
 public static void main(String[] args) throws Exception {  
 BufferedReader reader = new BufferedReader(new InputStreamReader(System.*in*));  
 MyCalendar calendar = new MyCalendar();  
 System.*out*.println("----------------------My Calender---------------------");  
 System.*out*.println("请输入年份：");  
 int year = Integer.*parseInt*(reader.readLine());  
 System.*out*.println("请输入月份：");  
 int month = Integer.*parseInt*(reader.readLine());  
 calendar.printCalendar(year, month);  
 }  
}

2.实验记录与说明分析

如下图所示，该程序实现了万年历的功能，且在展示某月日历时可以同时展示出上个月的后几天和下个月的前几天。



【GUI版本】

1.第一个实验源代码

package application;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.util.Calendar;  
  
public class GUICalendar extends JFrame {  
 private JButton lastMonth = new JButton("上个月");  
 private JButton nextMonth = new JButton("下个月");  
 private JButton check = new JButton("查询");  
 private JLabel yearLabel = new JLabel(" 年");  
 private JLabel monthLabel = new JLabel(" 月");  
 private JTextArea getYear = new JTextArea();  
 private JTextArea getMonth = new JTextArea();  
 private JTextArea showArea = new JTextArea(10, 50);  
 private JLabel weekLabel = new JLabel("星期日 星期一 星期二 星期三 星期四 星期五 星期六 ");  
 private int year = 2020;  
 private int month = 1;  
  
 public GUICalendar() {  
 Container container = this.getContentPane();  
 *//设置窗体的标题* setTitle("Wesley's Calendar");  
 *//设置窗体的显示位置及大小* setBounds(100, 100, 1000, 250);  
 *//设置窗体关闭按钮的动作为退出* setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 *//放置两个Panel，布局方式为GridLayout* BoxLayout panelLayout = new BoxLayout(container, BoxLayout.*Y\_AXIS*);  
 container.setLayout(panelLayout);  
 JPanel chooseDate = new JPanel();  
 JPanel showCalendar = new JPanel();  
 container.add(chooseDate);  
 container.add(Box.*createVerticalStrut*(20));  
 container.add(showCalendar);  
 *//在chooseDate中放置控件  
 //创建网格布局管理器对象* GridLayout chooseLayout = new GridLayout(1, 6);  
 chooseLayout.setHgap(20);  
 *//设置容器采用网格布局管理器* chooseDate.setLayout(chooseLayout);  
 lastMonth.setFont(new Font("黑体", Font.*BOLD*, 20));  
 chooseDate.add(lastMonth);  
 getYear.setFont(new Font("黑体", Font.*BOLD*, 32));  
 chooseDate.add(getYear);  
 yearLabel.setFont(new Font("黑体", Font.*BOLD*, 28));  
 chooseDate.add(yearLabel);  
 getMonth.setFont(new Font("黑体", Font.*BOLD*, 32));  
 chooseDate.add(getMonth);  
 monthLabel.setFont(new Font("黑体", Font.*BOLD*, 28));  
 chooseDate.add(monthLabel);  
 nextMonth.setFont(new Font("黑体", Font.*BOLD*, 20));  
 chooseDate.add(nextMonth);  
 *//在日历显示区域showCalendar放置控件* FlowLayout showLayout = new FlowLayout(FlowLayout.*CENTER*, 0, 10);  
 *//设置容器采用FlowLayout* showCalendar.setLayout(showLayout);  
 weekLabel.setFont(new Font("黑体", Font.*BOLD*, 16));  
 showCalendar.add(weekLabel);  
 check.setFont(new Font("黑体", Font.*BOLD*, 20));  
 showCalendar.add(check);  
 showArea.setEditable(false);  
 showArea.setOpaque(false);  
 showArea.setBorder(null);  
 showArea.setFont(new Font("黑体", Font.*BOLD*, 12));  
 showCalendar.add(showArea);  
 *//下面加入监听* this.addListener();  
 }  
  
 private void addListener() {  
 check.addActionListener(actionEvent -> {  
 *//若未填写直接查询，则会直接输出程序内置的月份* year = Integer.*parseInt*(getYear.getText());  
 month = Integer.*parseInt*(getMonth.getText());  
 printCalendar(year, month);  
 });  
 nextMonth.addActionListener(actionEvent -> {  
 if (month == 12) {  
 year++;  
 month = 1;  
 } else {  
 month++;  
 }  
 printCalendar(year, month);  
 getYear.setText(String.*valueOf*(year));  
 getMonth.setText(String.*valueOf*(month));  
 });  
 lastMonth.addActionListener(actionEvent -> {  
 if (month == 1) {  
 year--;  
 month = 12;  
 } else {  
 month--;  
 }  
 printCalendar(year, month);  
 getYear.setText(String.*valueOf*(year));  
 getMonth.setText(String.*valueOf*(month));  
 });  
 }  
  
 private void printCalendar(int year, int month) {  
 StringBuffer stringCalendar = new StringBuffer("");  
 Calendar calendar = Calendar.*getInstance*();  
 *//参数需要月份从0开始记，故-1* calendar.set(year, month - 1, 1);  
 *//下面要寻找该月第一天是周几  
 //1：周日 2：周一 3：周二 4：周三 5：周四 6：周五 7：周六* int judgeFirstDay = calendar.get(Calendar.*DAY\_OF\_WEEK*);  
 *//获取该月天数* int daysOfThisMonth = calendar.getActualMaximum(Calendar.*DAY\_OF\_MONTH*);  
 *//获取该月最后一天是周几* calendar.set(year, month - 1, daysOfThisMonth);  
 int judgeLastDay = calendar.get(Calendar.*DAY\_OF\_WEEK*);  
 *//获取上月天数* calendar.set(year, month - 2, 1);  
 int daysOfLastMonth = calendar.getActualMaximum(Calendar.*DAY\_OF\_MONTH*);  
 *//下面作输出操作  
 //输出上个月的后几天* for (int i = 1; i < judgeFirstDay; i++) {  
 stringCalendar.append(" [" + (daysOfLastMonth - judgeFirstDay + i) + "] ");  
 }  
 *//本月则依次输出即可* for (int i = 1; i <= daysOfThisMonth; i++) {  
 *//找到第i天* calendar.set(year, month - 1, i);  
 *//判断第i天是周几* int judgeDay = calendar.get(Calendar.*DAY\_OF\_WEEK*);  
 stringCalendar.append(" " + String.*format*("%-4d", i) + " ");  
 *//每到周六则要换行* if (judgeDay == 7) {  
 stringCalendar.append("\n");  
 }  
 }  
 *//下月的初几天也要输出* for (int i = 1; i < 7 - judgeLastDay + 1; i++) {  
 stringCalendar.append(" [" + i + "] ");  
 }  
 showArea.setText(String.*valueOf*(stringCalendar));  
 }  
  
 public static void main(String[] args) throws Exception {  
 GUICalendar frame = new GUICalendar();  
 frame.setVisible(true);  
 frame.setResizable(false);  
 UIManager.*setLookAndFeel*(UIManager.*getSystemLookAndFeelClassName*());  
 }  
}

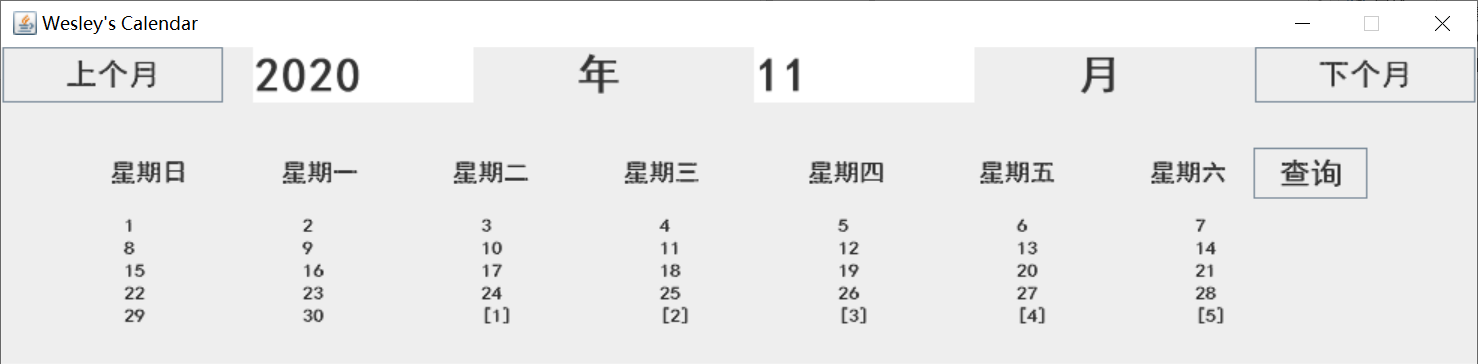
2.第二个实验源代码

package applications;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
import java.util.Calendar;  
  
public class PerpetualCalendar extends JFrame {  
 private final JTextField getMonth = new JTextField(4);  
 private final JTextField getYear = new JTextField(4);  
 private final JLabel yearLabel = new JLabel("年");  
 private final JLabel monthLabel = new JLabel("月");  
 private final JButton lastMonth = new JButton("上个月");  
 private final JButton nextMonth = new JButton("下个月");  
 private final JButton check = new JButton("查 询");  
 private final JButton[] dayButton = new JButton[42];  
 private final String[] week = {"周日", "周一", "周二", "周三", "周四", "周五", "周六"};  
 private final JButton[] weekButton = new JButton[7];  
 private int year = 2020;  
 private int month = 1;  
  
 public PerpetualCalendar() {  
 *// 设置标题* setTitle("Wesley's Calendar");  
 setSize(440, 260);  
 setLocation(500, 300);  
 init();  
 }  
  
 private void init() {  
 *//设置查询部分* yearLabel.setFont(new Font("宋体", Font.*BOLD*, 14));  
 monthLabel.setFont(new Font("宋体", Font.*BOLD*, 14));  
 check.setFont(new Font("宋体", Font.*BOLD*, 14));  
 getYear.setFont(new Font("Times new Roman", Font.*BOLD*, 16));  
 getMonth.setFont(new Font("Times new Roman", Font.*BOLD*, 16));  
 lastMonth.setFont(new Font("宋体", Font.*BOLD*, 14));  
 nextMonth.setFont(new Font("宋体", Font.*BOLD*, 14));  
 JPanel chooseDate = new JPanel();  
 chooseDate.add(lastMonth);  
 chooseDate.add(getYear);  
 chooseDate.add(yearLabel);  
 chooseDate.add(getMonth);  
 chooseDate.add(monthLabel);  
 chooseDate.add(check);  
 chooseDate.add(nextMonth);  
 *//设置日期显示部分* JPanel datesPanel = new JPanel();  
 datesPanel.setLayout(new GridLayout(7, 7, 0, 0));  
 for (int i = 0; i < 7; i++) {  
 weekButton[i] = new JButton(" ");  
 weekButton[i].setText(week[i]);  
 datesPanel.add(weekButton[i]);  
 weekButton[i].setBorder(null);  
 }  
 *//添加日期* for (int i = 0; i < 42; i++) {  
 dayButton[i] = new JButton(" ");  
 datesPanel.add(dayButton[i]);  
 dayButton[i].setBackground(Color.*WHITE*);  
 }  
 *//设置日期显示* JPanel showArea = new JPanel();  
 *//边界布局管理器* showArea.setLayout(new BorderLayout());  
 showArea.add(datesPanel, BorderLayout.*SOUTH*);  
 showArea.add(chooseDate, BorderLayout.*NORTH*);  
 getContentPane().add(showArea);  
 *//加入监听* lastMonth.addActionListener(e -> {  
 if (month == 1) {  
 year--;  
 month = 12;  
 } else {  
 month--;  
 }  
 printCalendar(year, month);  
 getYear.setText(String.*valueOf*(year));  
 getMonth.setText(String.*valueOf*(month));  
 });  
 nextMonth.addActionListener(e -> {  
 if (month == 12) {  
 year++;  
 month = 1;  
 } else {  
 month++;  
 }  
 printCalendar(year, month);  
 getYear.setText(String.*valueOf*(year));  
 getMonth.setText(String.*valueOf*(month));  
 });  
 check.addActionListener(e -> {  
 if (e.getSource() == check) {  
 *// 如果点击确定按钮就调用setDay()重新方法绘制按钮* this.year = Integer.*parseInt*(getYear.getText());  
 this.month = Integer.*parseInt*(getMonth.getText());  
 printCalendar(year, month);  
 }  
 });  
 getYear.addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyReleased(KeyEvent e) {  
 super.keyReleased(e);  
 if (e.getKeyCode() == 10) {  
 year = Integer.*parseInt*(getYear.getText());  
 month = Integer.*parseInt*(getMonth.getText());  
 printCalendar(year, month);  
 }  
 }  
 });  
 getMonth.addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyReleased(KeyEvent e) {  
 super.keyReleased(e);  
 if (e.getKeyCode() == 10) {  
 year = Integer.*parseInt*(getYear.getText());  
 month = Integer.*parseInt*(getMonth.getText());  
 printCalendar(year, month);  
 }  
 }  
 });  
 }  
  
 private void printCalendar(int year, int month) {  
 Calendar calendar = Calendar.*getInstance*();  
 calendar.set(year, month - 1, 1);  
 int daysOfThis = calendar.getActualMaximum(Calendar.*DATE*);  
 int firstDayOfThis = calendar.get(Calendar.*DAY\_OF\_WEEK*);  
 calendar.set(year, month - 2, 1);  
 int daysOfLast = calendar.getActualMaximum(Calendar.*DATE*);  
 for (int i = 0; i < firstDayOfThis - 1; i++) {  
 dayButton[i].setText("[" + (daysOfLast - firstDayOfThis + i + 2) + "]");  
 }  
 for (int i = 0; i < daysOfThis; i++) {  
 dayButton[firstDayOfThis - 1 + i].setText(String.*valueOf*(i + 1));  
 }  
 for (int i = firstDayOfThis - 1 + daysOfThis; i < dayButton.length; i++) {  
 dayButton[i].setText("[" + (i - firstDayOfThis - daysOfThis + 2) + "]");  
 }  
 }  
  
 public static void main(String[] args) throws Exception {  
 PerpetualCalendar frame = new PerpetualCalendar();  
 UIManager.*setLookAndFeel*(UIManager.*getSystemLookAndFeelClassName*());  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setVisible(true);  
 frame.setResizable(false);  
 }  
}

3.实验记录与说明分析

运行后得到了如下的界面，输入一个年月即可查询，还可以通过点击“上个月”和“下个月”按钮查询。

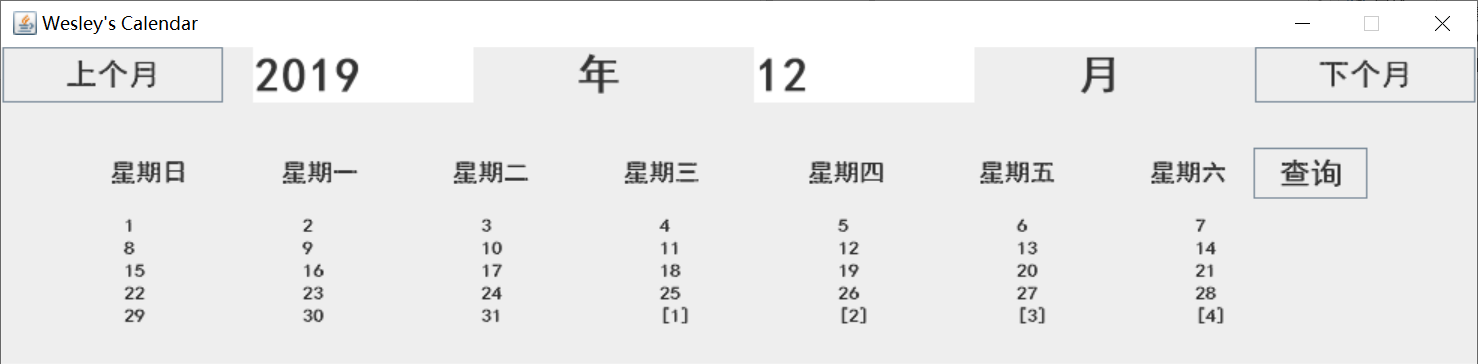




若未输入任何年月即点击下个月或上个月，程序将会利用内部存储的初始年月，即2020年1月，则直接点击下个月时会查询到2020年2月的日历。



同理，点击上个月将查询到2019年12月的日历。



该界面的缺点在于布局方式选择并不好，甚至有些“居中”是通过空格来实现的，所以通过固定窗口大小，保证了布局不会乱掉。

将界面做一下修改，使用按钮存放输出的日期，可以明显改善界面的美观程度，如下所示：



其功能与上个版本完全一致，只是界面更加规整了。

另外值得一提的是Java中ActionListener的使用，在加入监听时使用到了匿名类的概念，即我们需要从接口或者父类继承来一个类，重写其方法，并只需要得到这个匿名类的一个对象，若这个对象的类型非常明确，可使用lambda表达式来代替匿名类的声明，如：

lastMonth.addActionListener(actionEvent -> {  
 if (month == 1) {  
 year--;  
 month = 12;  
 } else {  
 month--;  
 }  
 printCalendar(year, month);  
 getYear.setText(String.*valueOf*(year));  
 getMonth.setText(String.*valueOf*(month));  
 });  
 可以大大简化程序，提高程序可读性。

## 【实验心得】

本次实验对Java编程中，变量的作用域、方法重载、构造函数的相关问题有了更深的了解，它们与C#中的规则基本一致（或者说大多数面向对象编程的思路都是相通的），今后面对这类问题不应当再想当然了。

本次实验中对字符串和数组的操作对GUI的制作十分有帮助，基本了解了StringBuffer，StringBuilder，String三个类的异同。

万年历的制作体现了高级编程语言的灵活性，在Java中，许多功能组件已经有了现成的“轮子”，利用好它们，可以让我们真正把目光着眼于对程序的设计和对更高级功能的实现上。若在Python中想实现万年历，则已经有了现成的模块（calendar），只需调用即可。

GUI的制作显得困难重重，Java中的GUI搭建不是可视化的，几个布局容器各有优势的同时也各有劣势，这一点上，甚至不如VisualBasic方便直观，每个控件也需要自主定义、声明，在哪里声明、如何实现，都可以自由选择，显得“过于灵活”，一时无从下手。

但当东拼西凑，修修补补制作完一个粗糙的万年历的GUI后，能明显感到跨过了一个“门槛”，Java GUI的编写又显得不是那么难了，在搭建其他GUI的时候，明显得心应手了许多。

面向对象编程相对于面向过程编程，其学习过程是更加有趣的，也希望自己能够在未来的使用中，真正掌握好这类语言。