第一次实验报告

姓名：王毅 班级：计算机基地班 学号：320220939591

T1 2.3.1(1)

1．流程控制之顺序结构测试。应用程序是在操作平台的JVM上直接执行的程序。

- 在记事本（或任意一款文本编辑器）中输入以下代码并以Shiyan3_1_1.java存盘
- 用javac Shiyan3_1_1.java编译生成Shiyan3_1_1.class字节码文件
- 用java Shiyan3_1_1执行Shiyan3_1_1类，察看程序执行结果，并填空

```
public class Shiyan3_1_1 {
    public static void main(String[] args){
        System.out.println("我们已经知道地球平均半径6370.856千米");
        System.out.println("也知道公认的地球质量为5.98×10^24 kg");
        System.out.println("我们就可以使用数学公式计算地球的平均密度为：");
        double radius=6.370856E6;  //定义存储半径的浮点型变量，用科学计数法
        double mass=5.98E24; //定义存储质量的浮点型变量
        double volume=4*Math.PI*Math.pow(radius,3)/3;  //球的体积公式，注意数学公式在程序中的变换，用到了Math类中的常量和方法。
        double density=mass/volume; //计算平均密度
        System.out.print(density+"(千克/立方米)"); //输出数据
    }
}
```

解释public class Shiyan3_1_1的意思:public 表示这个类是共有的，class用来生命一个类，Shiyan3_1_1是文件和类的名字
请计算主方法中定义变量总共消耗了多少字节的内存:32字节
请解释程序的顺序结构:从main开始从上到下依次运行到结束

T2 2.3.1(2)

2. 命令行参数测试实验。执行程序时在命令行中输入的参数称为命令行参数，在Java中命令行参数是以字符串的形式传给main方法中的形参数组的。

- 用记事本输入以下代码并以paramtest.java存盘，用javac paramtest.java编译
- 用 java paramtest [姓名] [学号] 运行察看结果并填空，(注：输入自己的真实姓名和学号，输入时不要输入方括弧。)

```
public class Paramtest {
   public static void main(String args[]) {
      System.out.println("\n第一个参数是："+args[0]);
      System.out.println("\n第二个是："+args[1]);
   }
}
```

请解释(String args[])的含义:String args[] 是 main 方法的参数，它是一个字符串数组，用于从命令行接收参数。
请写出程序的执行结果：wy 320220939591

T3 3.3.1(1)

(1)理解抽象和封装。在第一个验证实验中，抽象一个简单的Person类，只拍象了视频讲解“人”的三个最基本的属性：年龄(age)、姓名(name)和性别(sex)。对它们进行了封装，并提供了相应的get方法和set方法，同时在类中也提供了两个构造方法，并给出了equals()和toString()方法。

- 用记事本或Ultraedit输入以下程序并以Person.java存盘。
- 用javac编译，用java执行，然后填空。

```
public class Person{
    private int age=0;
    private String name = "noname";
    private char sex = 'M';

    public Person(){}

    public Person(String n, int a,char s){
        name = n;
        if(a >= 0 && a < 140) age = a; //年龄有效性检查
        else age = 0;
        if(s == 'M') sex = s; //性别有效性检查
        else sex = 'F';
    }

    public void introduceme() {
        System.out.println("my name is: " + name + "\tmy age is: " + age);
        if(sex == 'M') System.out.println("I am man!");
        else System.out.println("I am woman!");
    }

    public String getName() { return name; }
    public void setName(String n) { name = n; }
    public int getAge() { return age; }
    public void setAge(int a){ //年龄有效性检查
        if(a >= 0 && a < 140) age = a;
        else age = 0;
    }

    public char getSex() { return sex; }
    public void setSex(char s){ //性别有效性检查
        if(s == 'M') sex = 'M';
        else sex = 'F';
    }

    public boolean equals(Person a) {
        if(this.name.equals(a.name) && this.age == a.age && this.sex == a.sex)
```

```
            return true;
        else
            return false;
    }

    public String toString(){
        return name + "," + sex + "," + age;
    }

    class PersonTest{
        public static void main(String args[]) {
            Person p1, p2;
            p1 = new Person("张三", 28, 'M');
            p2 = new Person();
            p2.setName("李四"); p2.setAge(38); p2.setSex('F');
            p1.introduceme();
            p2.introduceme();
        }
    }
}
```

封装的意思是：封装是面向对象编程中的一个核心概念，主要指的是将对象的数据（属性）和操作这些数据的方法（行为）绑定在一起，形成一个独立的对象，并且对外隐藏对象的实现细节。
`p1 = new Person("张三", 28, 'M');`的意思是：构建一个Person实例，名字为张三，年龄为28，性别为男.
`p2.setName("陈红");`的用处是：将p2实例的名字换成陈红.

T4 3.3.1(2)

数组测试。数组是将相同类型的数据放在连续的存储区域，通过数组名和下标来使用每一个数组元素，注意Java中的数组名仅是一个引用变量名，并且Java认为数组是一个对象，有一个length属性用来表示数组元素个数，通过下标使用数组元素。请输入以下代码并以ArrayTest.java为文件名保存，然后编译运行，并回答问题。

```
public class ArrayTest{
    public static void main(String[]args){
        int[]a;
        Person[]b;
        a=new int[10];
        b=new Person[3];
        for(int i=0;i<10;i++){
            a[i]=(int)(100*Math.random());
        }
        b[0]=new Person("张三",28,'M');
        b[1]=new Person("李四",20,'M');
        b[2]=new Person();b[2].setName("葛优");
        b[2].setAge(46);b[2].setSex('F');
        for(int i=0;i<10;i++){
            System.out.println("a["+i+"]="+a[i]);
        }
```

```
        System.out.println(b[0]+"\n"+b[1]+"\n"+b[2]);
        System.out.println("a中元素个数:"+a.length);
        System.out.println("b中元素个数:"+b.length);
    }
}
```

Java语言中的数组和C语言中的数组在概念和使用上有若干关键的区别：

数组长度:
Java: 在Java中，数组是一个动态的对象，它有一个固定的长度属性。一旦初始化，Java数组的长度是不可变的。你可以通过数组的length属性来获取其大小。
C: 在C语言中，数组是静态的，没有内建的方式来直接获取数组的长度（除非使用特定的宏或者约定）。数组的大小在声明时必须是已知的或固定的，且在其生命周期内不会变化。

类型安全:
Java: Java是强类型语言，数组中的所有元素必须是声明时指定的数据类型，如果尝试存储错误类型的数据，会在编译时报错。
C: C语言是弱类型语言，对类型检查不是很严格。虽然C数组被定义为特定类型，但是由于指针和强制类型转换的使用，它可以容易地误用。

内存分配:
Java: 在Java中，所有的数组都是在堆上分配的，而且每个数组元素都会在创建时初始化为其数据类型的默认值。
C: C语言中的数组可以在栈（局部数组）或堆（使用malloc等动态分配）上创建。如果是在栈上创建，数组的元素初始值是未定义的；如果在堆上创建，则需要手动初始化。

数组越界:
Java: Java对数组越界提供了检查。如果尝试访问超出数组范围的元素，会抛出ArrayIndexOutOfBoundsException。
C: 在C语言中，数组越界不会抛出错误，因为C没有内建的越界检查。这可能导致未定义的行为，包括访问无效内存、程序崩溃等。

多维数组:
Java: 多维数组实际上是数组的数组，每一维都是一个独立的对象，并且每个子数组可以有不同的长度，这被称为不规则数组。
C: 多维数组是连续内存块的直接表示，并且每个维度的长度都必须是相同的。

数组传递给函数:
Java: 数组在Java中作为引用传递给方法，所以当数组作为参数传递时，传递的是数组的引用。
C: 在C语言中，数组名代表数组首元素的地址，所以当数组作为参数传递给函数时，实际上传递的是指向数组首元素的指针。

数组初始化:
Java: Java支持动态和静态初始化。可以在运行时动态地指定数组的大小，也可以直接使用花括号来初始化数组元素。
C: C语言支持静态初始化（使用花括号）和动态分配内存（例如使用malloc），但初始化方式不如Java灵活。

内存回收:
Java: Java有垃圾回收机制，不需要手动释放数组占用的内存。
C: 在C语言中，如果在堆上分配了内存，需要手动使用free函数释放。

试解释`b=new Person[3];`语句和`b[0]=new Person("张三", 28, 'M');`语句的作用，以及它们之间的区别和关系。
区别：第一个是构建了一个数组b，b里面存的是Person类，后面一句是构造b[0]。
关系：正常逻辑，没有特殊关系。

T5 3.3.1(3)

Java方法的参数传递用法。Java方法的参数传递是传值操作。对于基本数据类型(如int、char类型)变量作为参数，方法内对参数的操作实质是对参数复制变量的操作，不会改变原变量的值；对于引用类型（如数组、字符串）变量作为参数，方法内对该变量的操作是对引用变量所指向对象的操作，会改变原对象的数据。下面示例演示了方法参数调用的传递情况。

```
public class MethodParameter{
    public static void main(String[]args){
        int a=6;
        char[]str=new char[]{'H','e','l','l','o'};
        StringBuffer sb=new StringBuffer("TOM");
        changeAddr(str,sb);
        System.out.println(str);
        System.out.println(sb.toString());
        changeValue(a,str,sb);
        System.out.println(a);
        System.out.println(str);
        System.out.println(sb.toString());
    }

    private static void changeAddr(char[]c,StringBuffer sb){
        c = new char[]{'Y','e','l','l','o'};
        sb = new StringBuffer("SawYer");
    }

    private static void changeValue(int a,char[]c,StringBuffer sb){
        a = 8;
        c[0] = 'Y';
        sb.append("SawYa");
    }
}
```

结果

```
Hello
TOM
6
Yello
TOMSawYa
```

T6 3.3.3(2)

设计矩阵类Matrix，实现矩阵的加减法运算以及equals()方法和toString()方法。

```
public class Matrix {
    private int[][] data;
    private int rows;
    private int cols;

    // 构造方法
    public Matrix(int rows, int cols) {
        this.rows = rows;
        this.cols = cols;
        this.data = new int[rows][cols];
    }

    // 通过二维数组构造矩阵
    public Matrix(int[][] data) {
        if (data != null && data.length != 0) {
            this.rows = data.length;
            this.cols = data[0].length;
            this.data = new int[rows][cols];
            for (int i = 0; i < rows; i++) {
                System.arraycopy(data[i], 0, this.data[i], 0, cols);
            }
        }
    }

    // 加法
    public Matrix add(Matrix other) {
        if (this.rows != other.rows || this.cols != other.cols) {
            throw new IllegalArgumentException("Matrix dimensions do not match.");
        }
        Matrix result = new Matrix(rows, cols);
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result.data[i][j] = this.data[i][j] + other.data[i][j];
            }
        }
        return result;
    }

    // 减法
    public Matrix subtract(Matrix other) {
        if (this.rows != other.rows || this.cols != other.cols) {
            throw new IllegalArgumentException("Matrix dimensions do not match.");
        }
        Matrix result = new Matrix(rows, cols);
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result.data[i][j] = this.data[i][j] - other.data[i][j];
            }
```

```
        }
        return result;
    }

    // equals方法
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (!(obj instanceof Matrix)) return false;
        Matrix other = (Matrix) obj;
        if (this.rows != other.rows || this.cols != other.cols) return false;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (this.data[i][j] != other.data[i][j]) return false;
            }
        }
        return true;
    }

    // toString方法
    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        for (int[] row : data) {
            for (int element : row) {
                sb.append(element).append(" ");
            }
            sb.append("\n");
        }
        return sb.toString();
    }

    public static void main(String args[]){
        int[][] data1 = {{1, 2, 3}, {4, 5, 6}};
        int[][] data2 = {{7, 8, 9}, {10, 11, 12}};
        Matrix m1 = new Matrix(data1);
        Matrix m2 = new Matrix(data2);
        System.out.println(m1.add(m2));
        System.out.println(m1.subtract(m2));
    }
}
```