

JAVA 实验报告

本文是 Java 实验的实验报告。

一、数字金字塔

题目描述：输入一个正整数 n ($n < 16$)，输出一个如图的数字金字塔（下图是当 $n=7$ 的输出）。不考虑输入错误的情形。要求使用 `Scanner` 作为输入，`System.out.print` 作为输出。

解题思路：先由用户输入行数 n 。

对于第 i 行， $1 \leq i \leq n$ ，其行前空格数为 $n-i$ 。接着从 i 输出至 1 再输出至 i 。

源代码：

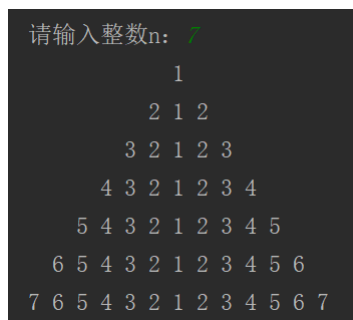
```
import java.util.Scanner;

public class Test {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int n;

        System.out.print("请输入整数 n: ");
        n = input.nextInt();

        for(int i = 1; i <= n; ++i) {
            int j;
            for(j = 1; j <= n - i; ++j) {
                System.out.print(" ");
            }
            for(j = i; j >= 1; --j) {
                System.out.print(j + " ");
            }
            for(j = 2; j <= i; ++j) {
                System.out.print(j + " ");
            }
            System.out.print("\n");
        }
    }
}
```

运行截图



```
请输入整数n: 7
      1
     2 1 2
    3 2 1 2 3
   4 3 2 1 2 3 4
  5 4 3 2 1 2 3 4 5
 6 5 4 3 2 1 2 3 4 5 6
7 6 5 4 3 2 1 2 3 4 5 6 7
```

二、PI 的近似值

题目描述：使用下式计算 π 的近似值并显示。其中 i 的值由用户输入。要求使用 `JOptionPane.showInputDialog` 作为输入，`JOptionPane.showMessageDialog` 作为输出。

解题思路：`JOptionPane` 是一种对话框的便捷使用形式，通过 `JOptionPane.showInputDialog` 可以读取输入数据，但是读入数据只能是 `String` 类型。故另使用函数转化为整数类型。

对于第 i 次计算，都将这一项前的符号同上次颠倒，然后计算这一项的值，其中分母

为 $2i-1$ ，然后加到总和上，最后再乘以 4。

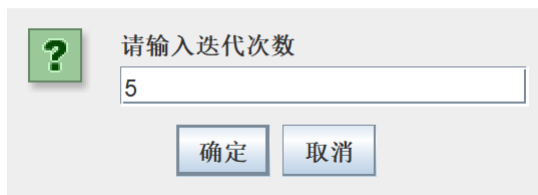
源代码：

```
import javax.swing.JOptionPane;

public class Test {
    public static void main(String[] args) {
        int number = 0;
        String numString = null;

        numString = JOptionPane.showInputDialog("请输入迭代次数");
        number = Integer.parseInt(numString);

        double pi = 0, term = 0, sign = 1;
        for(int i = 1; i <= number; ++i){
            term = sign * 1 / (2*i - 1);
            sign = -1;
            pi += term;
        }
        pi *= 4;
        JOptionPane.showMessageDialog(null, "pi = " + pi);
    }
}
```



三、回文素数

题目描述：回文素数指某一个数既是回文数、又是素数，例如 2，3，5，7，11，101，131，…编程找出前 100 个回文素数，并且要求按照每行 10 个的格式输出。输出方式任选。

解题思路：从 2 开始一一判断该数是否同时为回文数和素数，并存储在一个数组中。

判断回文数：将数字转化为字符串，从两边向中间扫描，若遇到不符合要求者返回 false。

对于素数，则检查其是否存在位于 2 至 \sqrt{n} 的因子，若有则为合数，反之为素数。

输出使用双层循环，j 控制输出数组中元素，i 负责控制格式。

源代码：

```
import java.util.Scanner;

public class Test {
    public static boolean isPla(int num){
        String strNum = null;
        strNum = String.valueOf(num);
        int lengthN = strNum.length();
        for(int i = 0; i < lengthN/2; ++i){
            if(strNum.charAt(i) != strNum.charAt(lengthN-i-1))
                return false;
        }
        return true;
    }
}
```

```

    }

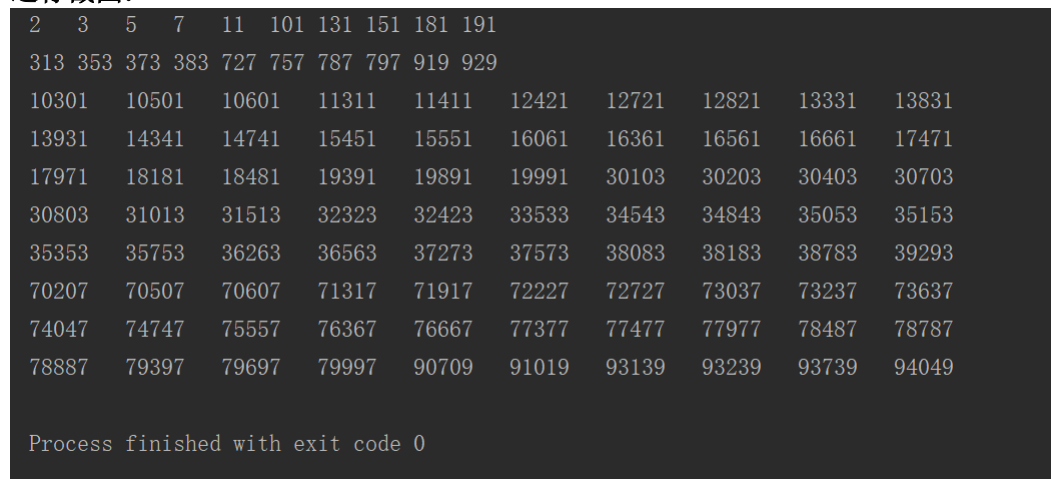
    public static boolean isPrime(int num){
        int i = 0;
        boolean isPri = true;
        for(i = 2; i*i <= num; ++i){
            if(num % i == 0){
                isPri = false;
                break;
            }
        }
        return isPri;
    }

    public static void main(String[] args) {
        int[] plaPrime = new int[101];
        int count = 0, num = 2;

        while(count < 100){
            if(isPrime(num) && isPla(num)){
                plaPrime[count++] = num;
            }
            ++num;
        }
        for(int i = 0; i < 10; ++i){
            for(int j = 0; j < 10; ++j)
                System.out.print(plaPrime[10*i+j]+"\\t");
            System.out.print("\\n");
        }
    }
}

```

运行截图:



```

2 3 5 7 11 101 131 151 181 191
313 353 373 383 727 757 787 797 919 929
10301 10501 10601 11311 11411 12421 12721 12821 13331 13831
13931 14341 14741 15451 15551 16061 16361 16561 16661 17471
17971 18181 18481 19391 19891 19991 30103 30203 30403 30703
30803 31013 31513 32323 32423 33533 34543 34843 35053 35153
35353 35753 36263 36563 37273 37573 38083 38183 38783 39293
70207 70507 70607 71317 71917 72227 72727 73037 73237 73637
74047 74747 75557 76367 76667 77377 77477 77977 78487 78787
78887 79397 79697 79997 90709 91019 93139 93239 93739 94049

Process finished with exit code 0

```

```
74047 74747 75557 76367 76667 77377 77477 77977 78487 78787
78887 79397 79697 79997 90709 91019 93139 93239 93739 94049
```

四、数组合并

题目描述:

写一个合并数组的方法，将两个已排序的数组合并成一个有序的大数组。方法的原型如下：

```
public static int[] merge(int[] list1, int[] list2)
```

要求提供一个测试的 main 函数，让用户输入两个数组的大小及元素值，最后调用上述方法并输出合并结果。

输入输出方式任选。

解题思路：从两个表的头开始扫描，如果 list1[i]<list2[j]，则放入 list1[i]，否则放入 list2[j]。假如扫描完某一个表后还有一个表没有扫描完毕，则将未归并的元素一一复制进来即可。

源代码:

```
import java.util.Scanner;

public class Test {
    public static int[] merge(int[] list1, int[] list2) {
        int[] mergeArr = new int[list1.length + list2.length];
        int i = 0, j = 0, length = 0;

        while (i < list1.length && j < list2.length) {
            if (list1[i] < list2[j]) {
                mergeArr[length++] = list1[i];
                i++;
            } else {
                mergeArr[length++] = list2[j];
                j++;
            }
        }
        while(i < list1.length) {
            mergeArr[length++] = list1[i];
            i++;
        }
        while(j < list2.length) {
            mergeArr[length++] = list2[j];
            j++;
        }
        return mergeArr;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int length1 = 0, length2 = 0;
        length1 = input.nextInt();
        int list1[] = new int[length1];
        for(int i = 0; i < length1; ++i){
            list1[i] = input.nextInt();
        }
        length2 = input.nextInt();
        int[] list2 = new int[length2];
        for(int i = 0; i < length2; ++i){
```

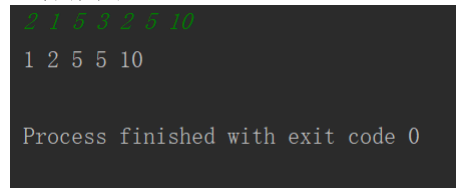
```

        list2[i] = input.nextInt();
    }

    int[] list = new int[length1 + length2];
    list = merge(list1, list2);
    for(int i = 0; i < list.length; ++i){
        System.out.print(list[i] + " ");
    }
    System.out.print("\n");
}
}

```

运行截图：



```

2 1 5 3 2 5 10
1 2 5 5 10

Process finished with exit code 0

```

心得体会：

通过本次实验，我熟悉了 Java 的基本编程语句，进一步巩固了 Java 的基础。Java 与 C/C++ 语言有着很大的不同，需要特别注意。

在输出时，如第一题不可用 `println`，而应该使用 `print`。`println` 在输出括号内内容后将自动换行，造成输出格式错误。

数组结构中，可以使用输入的 `int` 型数据定义数组长度，而 C/C++ 语言不行。

对于 Java 字符串 `String` 类型，不可像 C/C++ 中直接用 `[]` 直接访问或提取元素，需使用 `charAt()` 函数。