

1. 求Fibonacci数列，1、2、3、5、8、13.....前10项数的和

- 源代码

```
package com.coderit1;

/**
 * @author coderit1
 */
public class Program01 {
    public static void main(String[] args) {
        /* 1. 求Fibonacci数列，1、2、3、5、8、13.....前10项数的和 */
        int[] array = new int[11];
        int sum = 0;
        int init = 2;

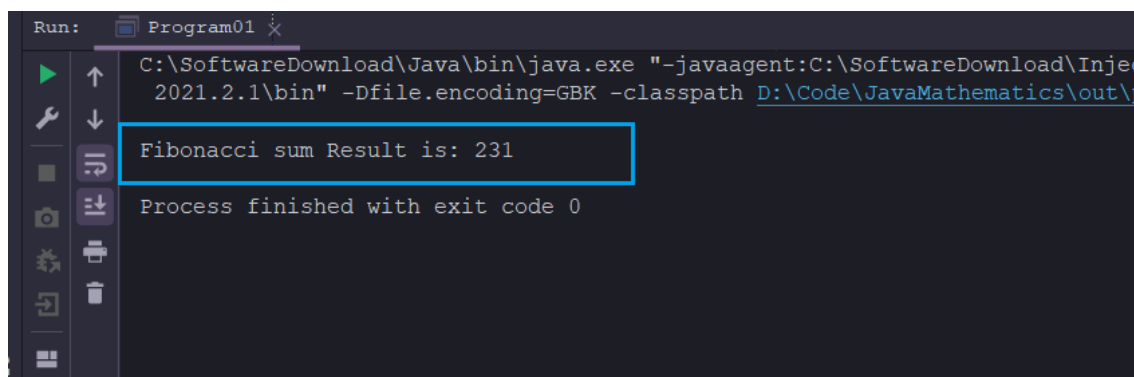
        array[0] = 1;
        array[1] = 2;

        // 遍历初始化
        for (int i = 1; i < array.length - init; i++) {
            array[i + 1] = array[i - 1] + array[i];
        }

        for (int i = 0; i < array.length - 1; i++) {
            sum += array[i];
        }

        System.out.println("Fibonacci sum Result is: " + sum);
    }
}
```

- 实验截图



2. 杨辉三角形（具有10行）的输出

- 源代码

```
package com.coderit1;

public class Program02 {
```

```

public static void main(String[] args) {

    int row = 10;
    int[][] yanghui = new int[row][row];

    for (int i = 0; i < row; i++) {
        for (int j = 0; j <= i; j++) {
            if (j == 0 || j == i) {
                yanghui[i][j] = 1;
            } else {
                yanghui[i][j] = yanghui[i - 1][j - 1] + yanghui[i - 1][j];
            }
            System.out.print(yanghui[i][j] + " ");
        }
        System.out.println();
    }
}

```

- 实验截图

```

Run: Program02
C:\SoftwareDownload\Java\bin\java.exe "-javaagent:C:\SoftwareDownload\InjectIDEA\IntelliJ ID
2021.2.1\bin" -Dfile.encoding=GBK -classpath D:\Code\JavaMathematics\out\production\JavaMat
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1

Process finished with exit code 0

```

3. 水仙花数的输出

- 源代码

```

package com.coderit1;

/**
 * @author coderit1
 */
public class Program03 {
    public static void main(String[] args) {
        /* 水仙花数 */
        int num1, num2, num3;

        for (int i = 100; i < 1000; i++) {
            // 1
            num1 = i / 100;
            // 5

```

```

        num2 = (i / 10) % 10;
        // 3
        num3 = (i % 10) % 10;

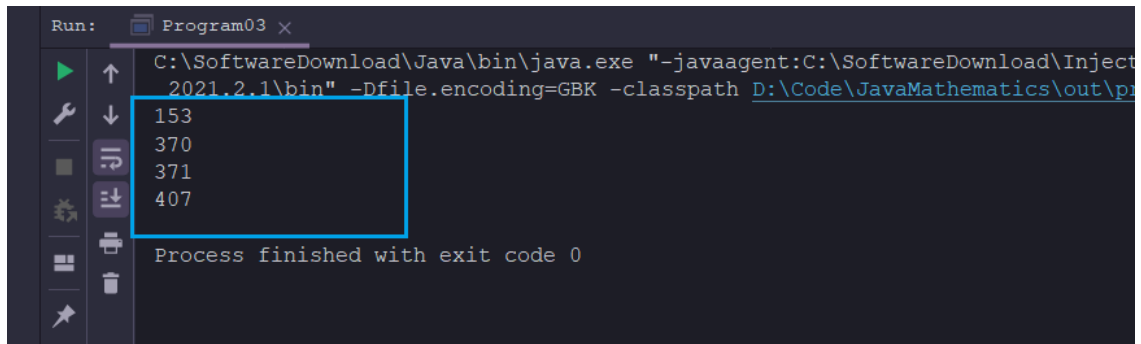
        if (i == (num1 * num1 * num1 + num2 * num2 * num2 + num3 * num3
* num3)) {
            System.out.println(i);
        }

    }

}
}

```

- 实验截图



4. 输出出现如下图形

- 源代码

```

package com.coderit1;

/**
 * @author coderit1
 */
public class Program04 {
    public static void main(String[] args) {
        /* 输出菱形 */
        /*
        @@@*
        @@***
        @*****
        @@***
        @@@*
        -----
        行数  空白  *数
        1    3    1
        2    2    3
        3    1    5
        4    2    3
        5    3    1
        -----
        */
    }
}

```

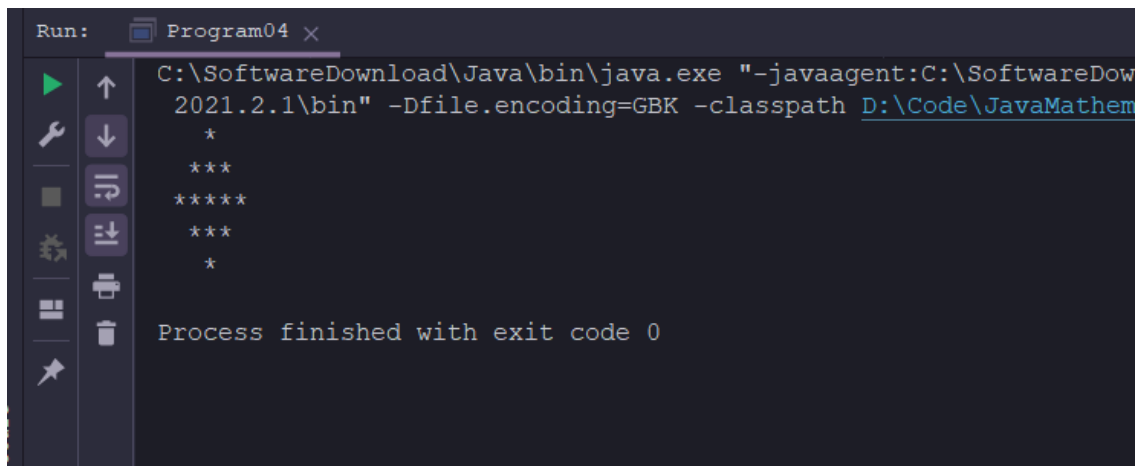
```

    for (int i = 0; i < 3; i++) {
        // 控制空格
        for (int j = 3; j > i; j--) {
            System.out.print(" ");
        }
        for (int k = 0; k < 2 * i + 1; k++) {
            System.out.print("*");
        }
        System.out.println();
    }

    for (int i = 2; i > 0; i--) {
        // 控制空格
        for (int j = 1; j <= 4 - i; j++) {
            System.out.print(" ");
        }
        for (int k = 1; k <= 2 * i - 1; k++) {
            System.out.print("*");
        }
        System.out.println();
    }
}
}

```

- 实验截图



5. 从键盘输入两个数a、b求出其最大公约数和最小公倍数

- 源代码

```

package com.coderit1;

import java.util.Scanner;

/**
 * @author coderit1
 */
public class Program05 {
    public static void main(String[] args) {
        /*

```

```

* 最大公约数：两个数可以共同约去的最大一个数字
* 最小公倍数：两个数各自的 1 2 3... 倍数之后,最小的相等的一个数字,就是最小公
倍数
* 最大公约数与最小公倍数之间的关系,两个数字的乘积 / 最大公约数 = 最小公倍数
* Eg. 12 和 6 的最大公约数 6
* 计算方式:
*      12: 1 2 3 4 6 12
*      6:  1 2 3 6
*      12和6都可以被 6 整除
* 辗转相减法: 12和6: (12-6)->(6,6)-> 0 所以最大公约数 6 【始终使用最大的数
减去最小的数】
*
*
* 公倍数计算方式:
* 12 和 4:
*      12: 12*1=12 12*2=24 12*3=36 12*4 ...
*      4:  4*1=4  4*2=8  4*3=12 4*4 ...
* 最小公倍数: 12
* 最小公倍数: 两个数各自的 1 2 3 ... 倍数之后,最小的相等的一个数字,就是最小
公倍数
* 最大公约数与最小公倍数之间的关系: 两个数的乘积/最大公约数 = 最小公倍数
* */

Scanner scanner = new Scanner(System.in);

System.out.println("输入两个整数: ");
int num1 = scanner.nextInt();
int num2 = scanner.nextInt();

/* 辗转相减法 存储 num1和 num2 */
int tempX = num1;
int tempY = num2;

while (tempX != tempY) {
    /* 辗转相减法是使用大数减去小数 */
    if (tempX > tempY) {
        /* 始终改变的是大数 */
        tempX = tempX - tempY;
    } else {
        /* 始终改变的是大数 */
        tempY = tempY - tempX;
    }
}

/* 最大公约数的结果: 跳出 while 循环就是 tempX == tempY */
int maxCommonDivisor = tempX;

/* 最大公约数与最小公倍数之间的关系: 两个数的乘积/最大公约数 = 最小公倍数 */
int minCommonMultiple = (num1 * num2) / maxCommonDivisor;

System.out.println(num1 + "和" + num2 + "的最大公约数是: " +
maxCommonDivisor);
System.out.println("最小公倍数: " + minCommonMultiple);
}
}

```

```
C:\SoftwareDownload\Java\bin\java.exe "-javaagent:C:\SoftwareDownload\InjectIDEA\In
2021.2.1\bin" -Dfile.encoding=GBK -classpath D:\Code\JavaMathematics\out\productio
输入两个整数:
12 6
12和6的最大公约数是: 6
最小公倍数: 12

Process finished with exit code 0
```

6. 输出100以内的所有素数

- 源代码

```
package com.coderit1;

/**
 * @author coderit1
 */
public class Program06 {
    public static void main(String[] args) {
        /* 100 以内的素数:
         * 质数(素数)是指在大于 1 的自然数中,除了 1 和 它本身以外不再有其他因数的自然
         数
         * */
        for (int i = 1; i < 100; i++) {
            boolean flag = true;
            for (int j = 2; j < i; j++) {
                if (i % j == 0) {
                    flag = false;
                    break;
                }
            }
            if (flag) {
                System.out.println(i);
            }
        }
    }
}
```

- 实验截图

```
Program06
C:\SoftwareDownload\Java\bin\java.exe "-javaagent:C:\SoftwareDownload\InjectIDEA\IntelliJ IDEA 2021.2.1\lib\
2021.2.1\bin" -Dfile.encoding=GBK -classpath D:\Code\JavaMathematics\out\production\JavaMathematics com.coc
1 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
Process finished with exit code 0
```

7. 有1、2、3、4个数字，能组成多少个互不相同且无重复数字的三位数？都是多少

- 源代码

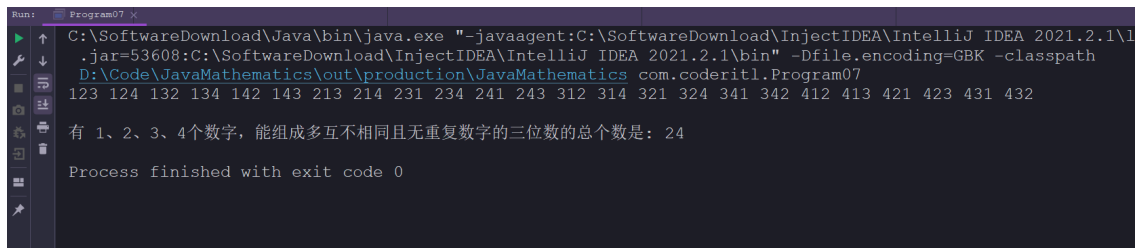
```
package com.coderitl;

/**
 * @author coderitl
 */
public class Program07 {
    public static void main(String[] args) {
        /* 有 1、2、3、4个数字，能组成多少个互不相同且无重复数字的三位数？都是多少 */
        int number = 0, count = 0;
        int init = 5;

        for (int a = 1; a < init; a++) {
            for (int b = 1; b < init; b++) {
                for (int c = 1; c < init; c++) {
                    /* 123 = 1 * 100 + 2 * 10 + 3 */
                    number = a * 100 + b * 10 + c;
                    /* 条件限制：互不相同且无重复数字 */
                    if (a != b && a != c && b != c) {
                        System.out.print(number + " ");
                        count++;
                    }
                }
            }
        }

        System.out.println();
        System.out.println("有 1、2、3、4个数字，能组成多互不相同且无重复数字的三位数  
的总个数是： " + count);
    }
}
```

- 实验截图



```
Run: Program07
C:\SoftwareDownload\Java\bin\java.exe "-javaagent:C:\SoftwareDownload\InjectIDEA\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=53608:C:\SoftwareDownload\InjectIDEA\IntelliJ IDEA 2021.2.1\bin" -Dfile.encoding=GBK -classpath D:\Code\JavaMathematics\out\production\JavaMathematics com.coderitl.Program07
123 124 132 134 142 143 213 214 231 234 241 243 312 314 321 324 341 342 412 413 421 423 431 432
有 1、2、3、4个数字，能组成多互不相同且无重复数字的三位数的总个数是： 24
Process finished with exit code 0
```

8. 有一分数序列：2/1，3/2，5/3，8/5，13/8，21/13...求出这个数列的前20项之和

- 源代码

```
package com.coderitl;
```

```

/**
 * @author ccoderitl
 */

public class Program08 {
    public static void main(String[] args) {
        /* 有一分数序列：2/1,3/2,5/3,8/5,13/8,21/13...求出这个数列的前 20 项之和
        */
        /*
        * 2 3 5 8 13 21 a 分子
        * 1 2 3 5 8 13 b 分母
        * -----
        */

        int init = 20;

        double a = 1,
               b = 1,
               c = 0,
               sum = 0;

        for (int i = 0; i < init; i++) {
            c = a + b;
            a = b;
            b = c;
            sum += b / a;
        }

        System.out.println(sum);
    }
}

```

- 实验截图

```

C:\SoftwareDownload\Java\bin\java.exe "-javaagent:C:\SoftwareDownload\Inject
.jar=56947:C:\SoftwareDownload\InjectIDEA\IntelliJ IDEA 2021.2.1\bin" -Dfi
D:\Code\JavaMathematics\out\production\JavaMathematics com.coderitl.Progra

这个数列的前 20 项之和： 32.66026079864164

Process finished with exit code 0

```

9. 已知 $S=1-1/2+1/3-...+1/n-1/(n+1)$ ，利用while循环编程求解n=100时的S的值

- 源代码

```

package com.coderitl;

public class Program09 {
    public static void main(String[] args) {

```

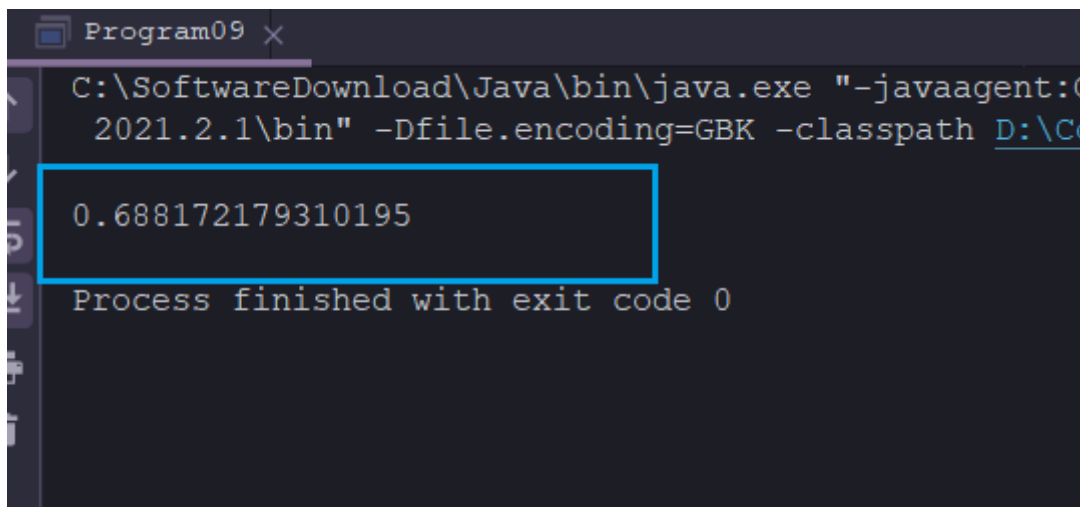


```

/* 已知:  $s = 1 - 1/2 + 1/3 - 1/4 + \dots + 1/(n-1) - 1/n$ , 编写程序求解  $n = 100$  时的S值 */
double s = 1;
int i = 2;
while (i <= 100) {
    if (i % 2 == 0) {
        s = s - 1 / (double) i;
    } else {
        s = s + 1 / (double) i;
    }
    i++;
}
System.out.println(s);
}
}

```

- 实验截图



10. 例如 $6=1+2+3$.编程找出1000以内的所有完数

- 源代码

```

package com.coderit1;

/**
 * @author coderit1
 */

public class Program10 {
    public static void main(String[] args) {
        /*
         * 一个数如果恰好等于它的因子之和, 这个数就称为 完数 例如  $6 = 1 + 2 + 3$ . 编程
         找出 1000 以内的所有完数
         */
        int i, j;
        int init = 1000;

        for (i = 1; i < init; i++) {

            int sum = 0;

```

```

        for (j = 1; j <= i / 2; j++) {
            if (i % j == 0) {
                sum = sum + j;
            }
        }

        if (sum == i) {

            for (j = 1; j < i / 2; j++) {
                if (i % j == 0) {
                    System.out.print(j + "\t");
                }
            }
            System.out.println("is: " + i);
        }
    }
}

```

- 实验截图

```

Program10 x
C:\SoftwareDownload\Java\bin\java.exe "-javaag
2021.2.1\lib\idea_rt.jar=62280:C:\SoftwareDov
-Dfile.encoding=GBK -classpath D:\Code\JavaMa
com.coderitl.Program10
1 2 is: 6
1 2 4 7 is: 28
1 2 4 8 16 31 62 124 is: 496
Process finished with exit code 0

```