# 操作符

## 代码观察

**观察以下代码**

1. int a = 6--;

2. System.out.println(a);

3.

**问题：上⾯代码是否有错误，如果没有，程序结果是多少？**

* **答：**

--操作符不能用作常量和字面值故，错误在第一行，所以由于6是常量且没有储存位置故不能对其进行递减操作。

* + **修改后代码**

1. int a = 6;

2. a--；

3. System.out.println(a);

4.

## 代码分析

* **分析以下代码的运⾏结果【重点复习原码 反码 补码】**

****

* **答**

1. `System.out.println(5&9);`

* 运算过程：

  - `5` 的二进制表示为 `0101`，`9`的二进制表示为 `1001`。`&` 是按位与运算符，只有当两个对应的位都为 `1` 时，结果才为 `1`。因此，`5 & 9` 的结果为 `0001`，即十进制的 `1`。

* 结果：`1`

2. `System.out.println(5|9);`

* 运算过程：

  - `5` 的二进制表示为`0101`，`9`的二进制表示为 `1001`。`|` 是按位或运算符，只要有一个位为 `1`，结果就是 `1`。因此，`5 | 9` 的结果为 `1101`，即十进制的 `13`。

* 结果：`13`

3. `System.out.println(5^9);`

* 运算过程：

  - `5` 的二进制表示为 `0101`，`9` 的二进制表示为 `1001`。`^` 是按位异或运算符，相同为 `0`，不同为 1。因此，`5 ^ 9` 的结果为 `1100`，即十进制的 `12`。

* 结果：`12`

4. `System.out.println(~-5);`

* 运算过程：

  - `-5` 在 `Java` 中是以补码形式存储的，其二进制表示为 `11111111111111111111111111111011`。`~` 是按位取反运算符，将所有位翻转。因此，`~-5` 实际上会得到 `5` 的补码表示的前一位减一，即 `-6`。

* 结果：`-6`

5. `System.out.println(5<<2);`

* 运算过程：

   - `5` 的二进制表示为 `0101`，左移两位后为 `010100`，即十进制的 `20`。

* 结果：20

6. `System.out.println(-5<<2);`

* 运算过程：

  - `-5`的二进制表示为 `11111111111111111111111111111011`，左移两位后为 `1111111111111111111111111111101100`，即十进制的 `-20`。

* 结果：-20

7. `System.out.println(-5>>2);`

* 运算过程：

  - `-5` 的二进制表示为 `11111111111111111111111111111011`，右移两位后为 `11111111111111111111111111111110`，即十进制的 `-2`。

* 结果：`-2`

8. `System.out.println(-5>>>2);`

* 运算过程：

  - 无符号右移操作符` >>> `不考虑符号位，将 `11111111111111111111111111111011` 右移两位后为 `00111111111111111111111111111111`，即十进制的 `1073741822`。

* 结果：1073741822

9. `System.out.println(5>>2);`

* 运算过程：

  - `5` 的二进制表示为 `0101`，右移两位后为`0000`，即十进制的 `0`。

* 结果：`0`

10.  ` System.out.println(5>>34);`

* 运算过程：

  - 由于`5` 的二进制表示为 `0101`，而右移的位数大于其二进制长度，最终结果为 `0`。

* 结果：`0`

11.  ` System.out.println(97=='a');`

* 运算过程：

  - `97` 是字符 `'a'` 的 `ASCII` 值，所以 `97 == 'a'` 的结果为`true`。

* 结果：true

12.   `System.out.println(5.0==5);`

* 运算过程：

  - 浮点数 `5.0` 和整数 `5` 在比较时会自动类型转换，所以 `5.0 == 5` 的结果为 `true`。

* 结果：true

13.   `System.out.println(4>5 ^ 'c'>'a');`

* 运算过程：

  - 首先计算 `'c'>'a'`，这是 `true`，但 `true` 在 `Java` 中并不是 `1` 或 `0`，而是布尔类型的 `true`。然后计算` 4>5`，这显然是 `false`。然而，由于 `^` 是按位异或运算符，并不能直接应用于布尔值，所以此表达式在语法上是错误的。

* 结果：编译错误

14.   `System.out.println((int)(char)(byte)-1);`

* 运算过程：

  - 首先将 -1 转换为 byte 类型，然后转换为 char 类型，再转换为 int 类型。由于 -1 的二进制表示为全 1（11111111），经过这些转换后，结果仍然为 -1 的 int 表示，即 int 类型的 -1。

* 结果：-1

## 指定位置设置为1

## 指定位置设置为0

# 流程控制

## 代码分析

**分析下⾯⼏段代码，写出输出结果，最后运⾏程序进⾏验证**

**1）**

1. int b=5;

2. if(b>4)

3. System.out.println(b);

4. else

5. b--;

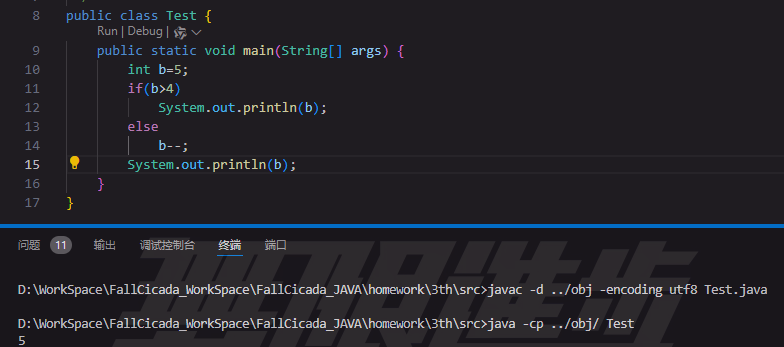
6. System.out.println(b)；

7.

**答:**

**声明整形变量b=5，如果b大于4则将b的值打印，反之，b做自减后打印b的值在控制终端打印5**

**运行结果：**

****

**2）**

1. }

2. Int b=5;

3. if(b>4)b--; System.out.println("b⼤于4");

4. else {

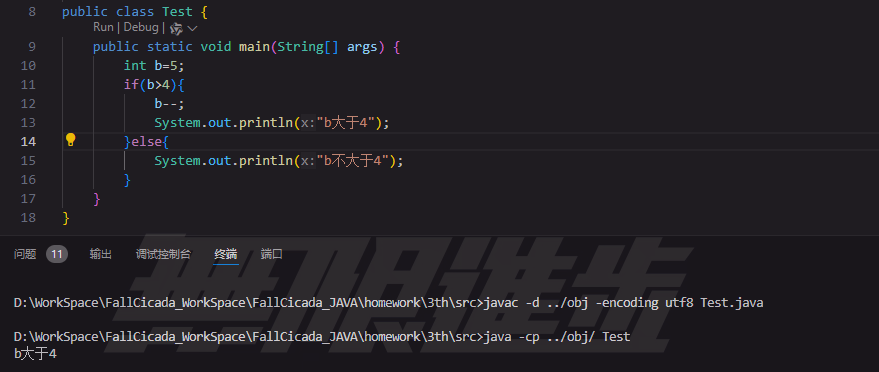
5. System.out.println("b不⼤于4");

6. }

7. }

8.

**答：**



3）

1. int age=45;

2. if(age>20) {

3. System.out.println("年轻⼈");

4. }else if(age>40){

5. System.out.println("中年⼈");

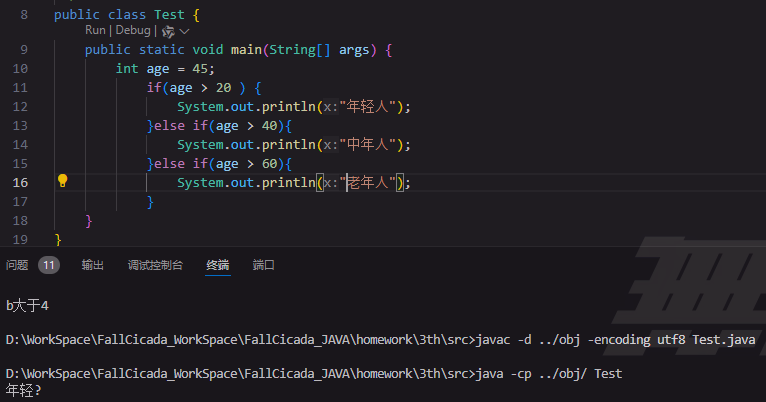
6. }else if(age>60){

7. System.out.println("⽼年⼈");

8. }

9.

**答：定义整形自变量age=20，判断如果大于20则输出年轻人，如果大于40则输出中年人，如果大于60输出老年人，题中age = 40 > 20所以输出年轻人**

****

**4）**

1. int a=80;

2. switch(a) {

3. case 90:{

4. System.out.println("优秀");

5. }

6. case 80:{

7. System.out.println("⼀般");

8. }

9. case 60:{

10. System.out.println("及格");

11. break;

12. }

13. case 50:{

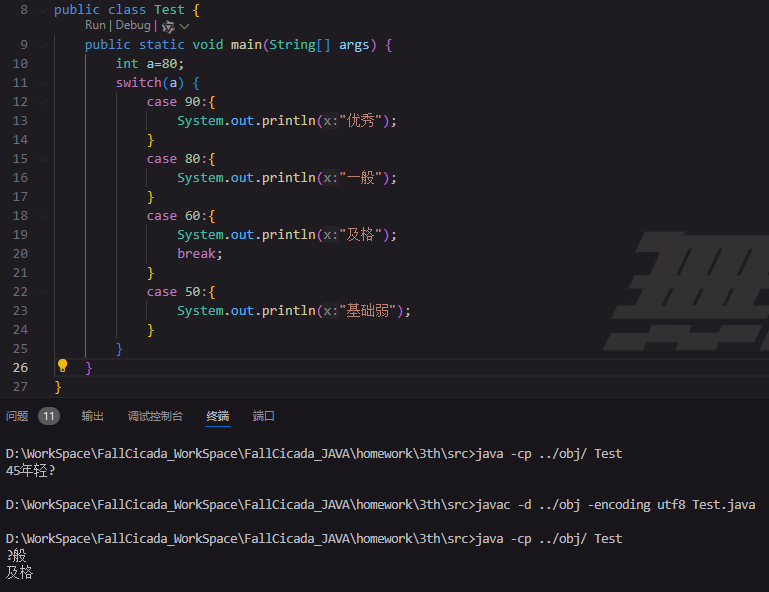
14. System.out.println("基础弱");

15. }

16. }

**答：**

**定义自变量a=80，如果a=90输出优秀，如果a=80输出一般，如果a=60输出60，由于case代码块中没有break所以会输出一般\n及格**

****

**5）**

1. int count=0;

2. while(count>0);

3. {

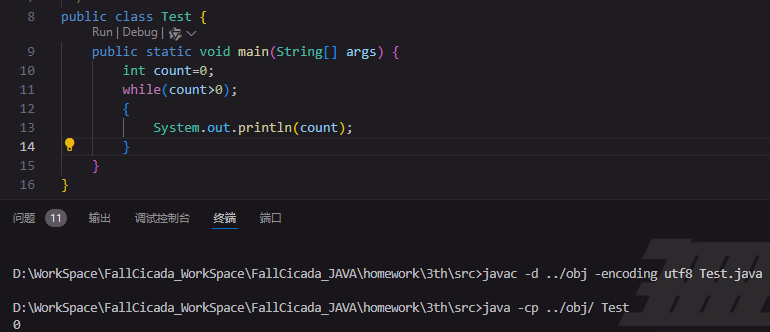
4. System.out.println(count);

5. }

6.

**答：**

**定义整形自变量count=0如果count大于0则进入while循环体并持续输出count，反之单次输出count**

****

**6）**

1. for (int i = 0; i < 5; i++) {

2. System.out.print(i);

3. i\*=0.1;

4. }

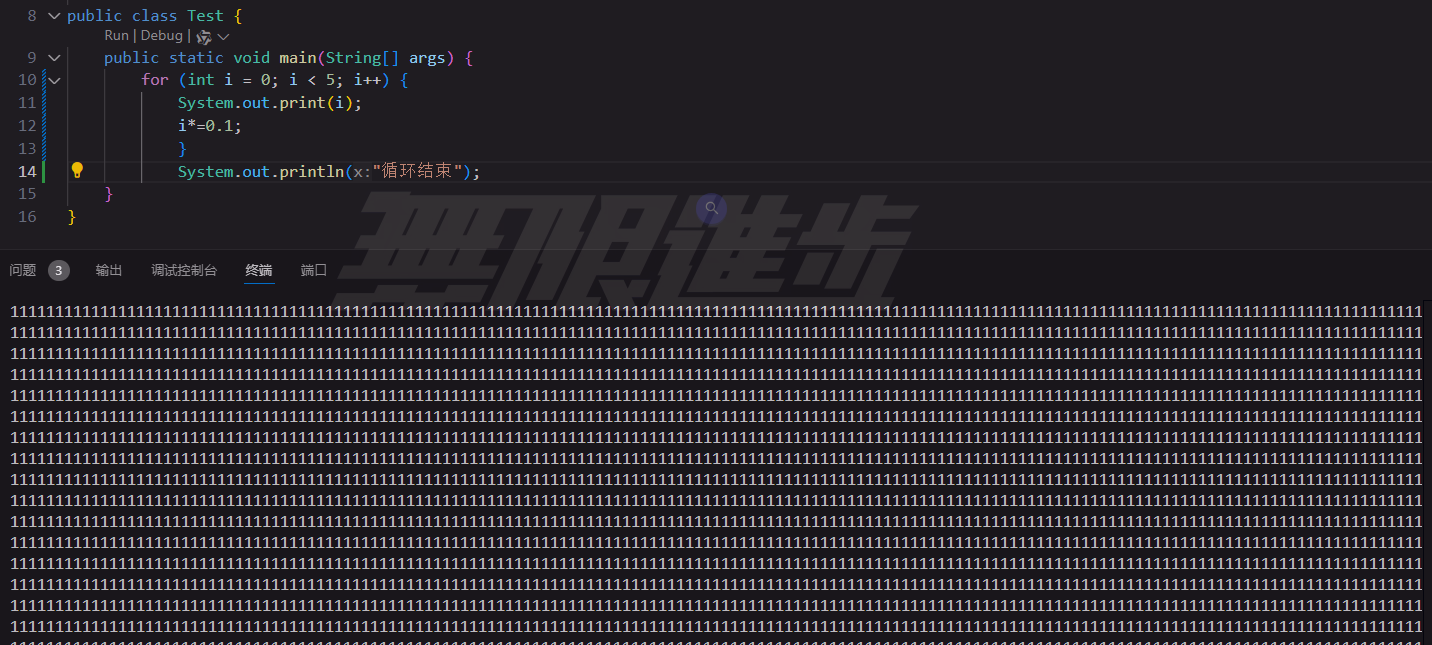
5. System.out.println("循环结束");

6.

7.

**答：**

开始for循环定义整形自变量i=0，然后打印自变量i，然后自变量i乘0.1重新赋值给i，由于是整形自变量所以为0再做自加为1，当满足i>=5时退出循环，由于当前循环无法满足，所以持续输出1.



**7）**

1. for(int i=0;i<3;i++) {

2.             System.out.println(i);

3.             if(i==1) {

4.                 continue;

5.              }

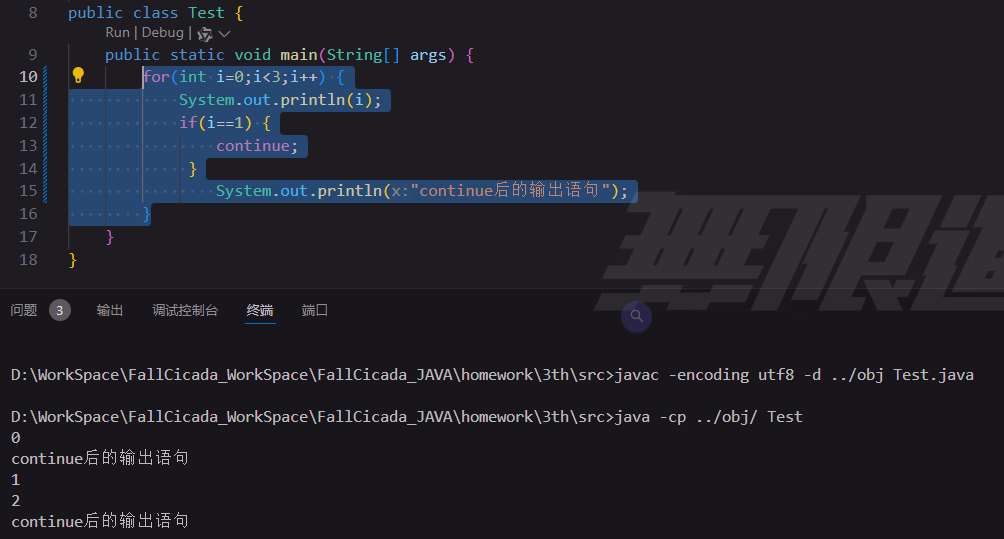
6.                 System.out.println("continue后的输出语句");

7.         }

8.

**答：**

开始for循环，定义整形自变量i=0，每次运行后i自增1，循环条件为i<3,进入循环后先打印i为0，和continue后的输出语句，i自加1，再次进入循环，打印i=1，满足if分支条件停止本次循环的后续执行，从头开始，i自加1，i=2后打印2并打印continue后的输出语句。



**7）**

1. for (int i = 0; i < 3; i++) {

2.             System.out.println(i);

3.             if (i == 1) {

4.                 return;

5.              }

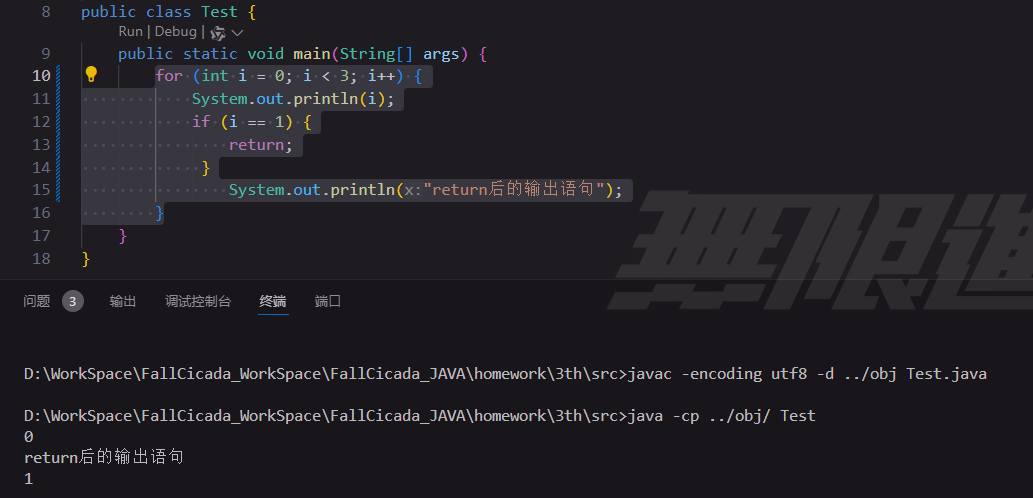
6.                 System.out.println("return后的输出语句");

7.         }

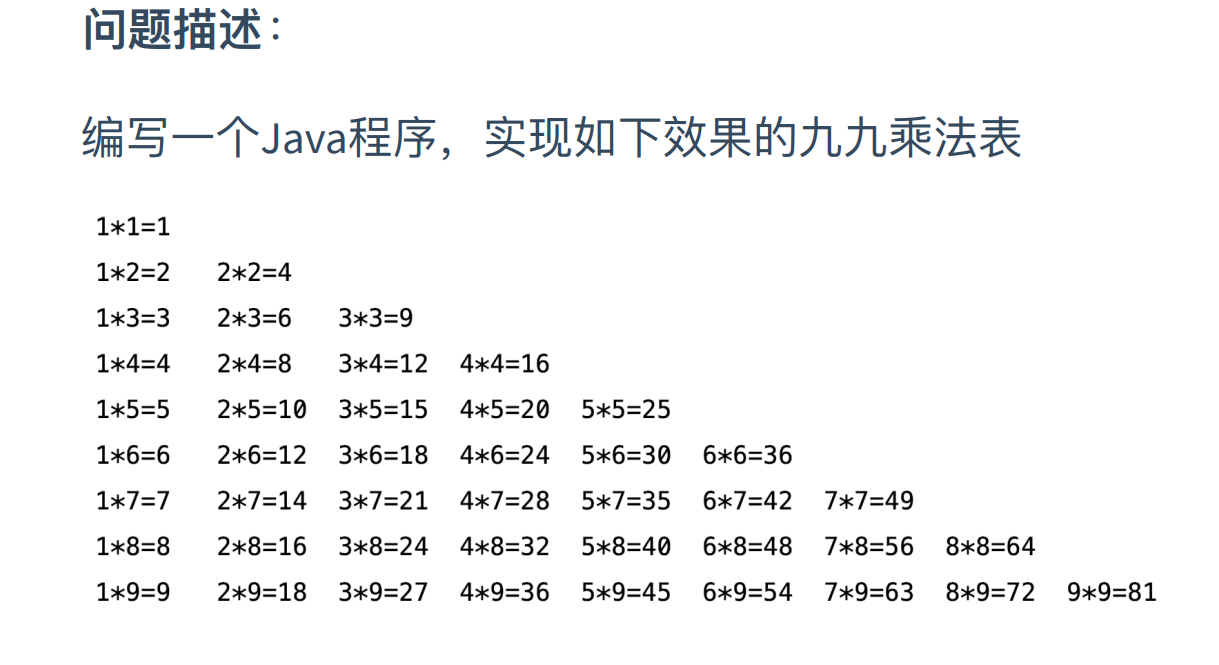
8.

**答：**

开始执行循环定义整形自变量i = 0，循环后i自加1，循环条件为i<3，i=0，进入循环打印i值，为0，并打印return后的输出语句，i自加1，再次进入循环，打印1，满足if分支条件进入if语句，执行return结束方法。



## 九九乘法表



**答：**

**代码：**

1. public class Test {

2.     public static void main(String[] args) {

3.         int i,j;

4.         for (i = 1; i <= 9; i++) {

5.             for(j = 1; j <= i; j++){

6.                 System.out.print(j+"\*"+i+"=" + i\*i+"\t");

7.                 if (i == j) {

8.                 System.out.println("");

9.                  }

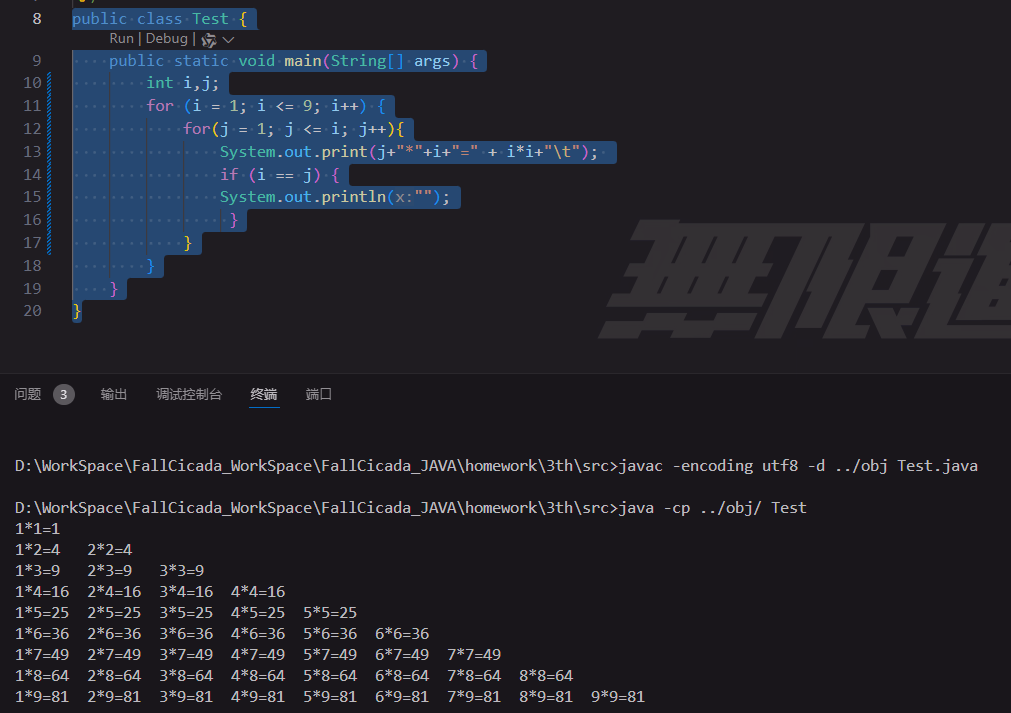
10.             }

11.         }

12.     }

13. }

**运行结果：**

****

## 素数问题



1. public class Test {

2.

3.     public static void main(String[] args) {

4.         int count = 0;

5.

6.         // 遍历101到200之间的所有数

7.         for (int i = 101; i <= 200; i++) {

8.             if (isPrime(i)) {

9.                 count++;

10.                 System.out.println(i);

11.             }

12.         }

13.

14.         // 输出素数的数量

15.         System.out.println("101到200之间共有 " + count + " 个素数");

16.     }

17.

18.     /\*\*

19.      \* 判断一个数是否为素数

20.      \* 素数定义为大于1且仅能被1或自身整除的数

21.      \*/

22.     private static boolean isPrime(int number) {

23.         // 数字小于等于1时，不是素数

24.         if (number <= 1) {

25.             return false;

26.         }

27.         // 遍历从2到number的平方根的所有数，检查是否能整除number

28.         for (int i = 2; i <= Math.sqrt(number); i++) {

29.             // 如果number能被i整除，则number不是素数

30.             if (number % i == 0) {

31.                 return false;

32.             }

33.         }

34.         // 如果没有找到可以整除number的数，那么number是素数

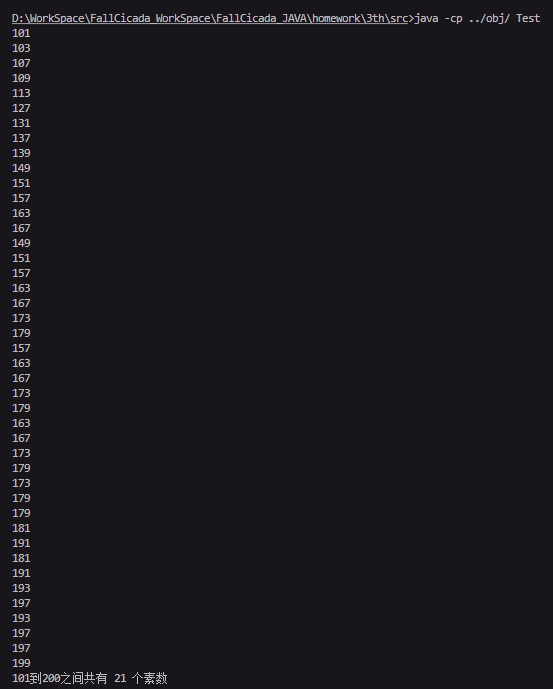
35.         return true;

36.     }

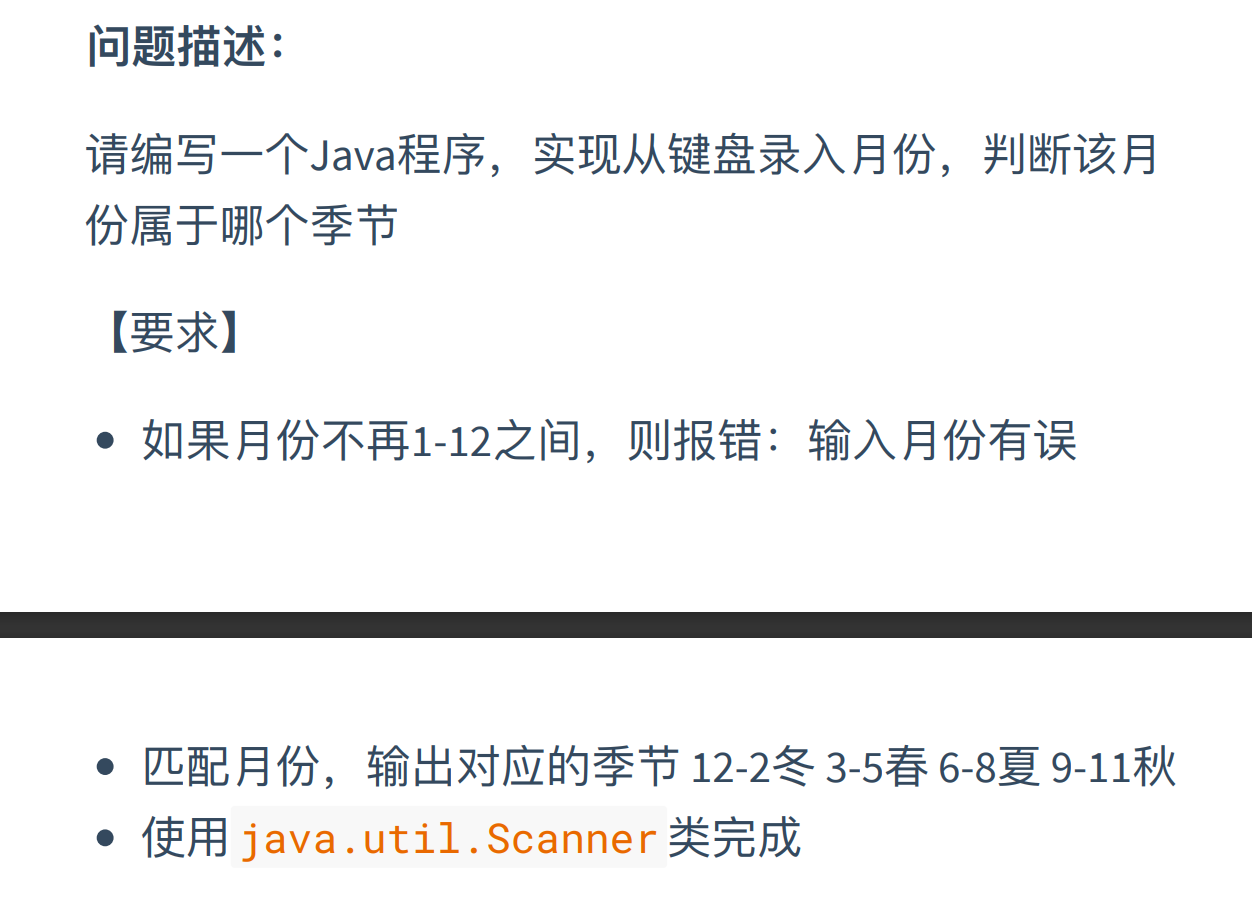
37. }

38.

**运行结果：**



## 月份计算



**答：**

1. import java.util.Scanner;

2.

3. public class Test {

4.     public static void main(String[] args) {

5.         Scanner scanner = new Scanner(System.in);

6.         System.out.println("请输入月份：");

7.         int month = scanner.nextInt();

8.         // 验证输入的月份是否有效

9.         if (month < 1 || month > 12) {

10.             System.out.println("输入的月份有误");

11.         } else {

12.             if (month == 12 || (month >= 1 && month <= 2)) {

13.                 System.out.println(month + "月，是冬季");

14.             } else if (month >= 3 && month <= 5) {

15.                 System.out.println(month + "月，是春季");

16.             } else if (month >= 6 && month <= 8) {

17.                 System.out.println(month + "月，是夏季");

18.             } else if (month >= 9 && month <= 11) {

19.                 System.out.println(month + "月，是秋季");

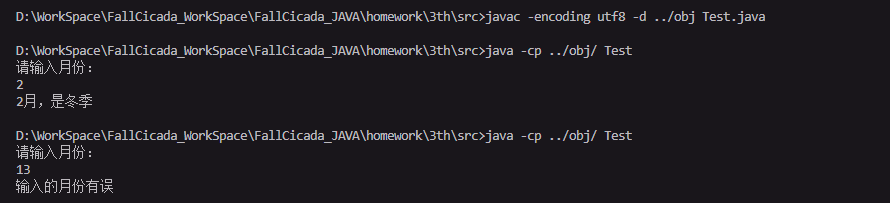
20.             }

21.         }

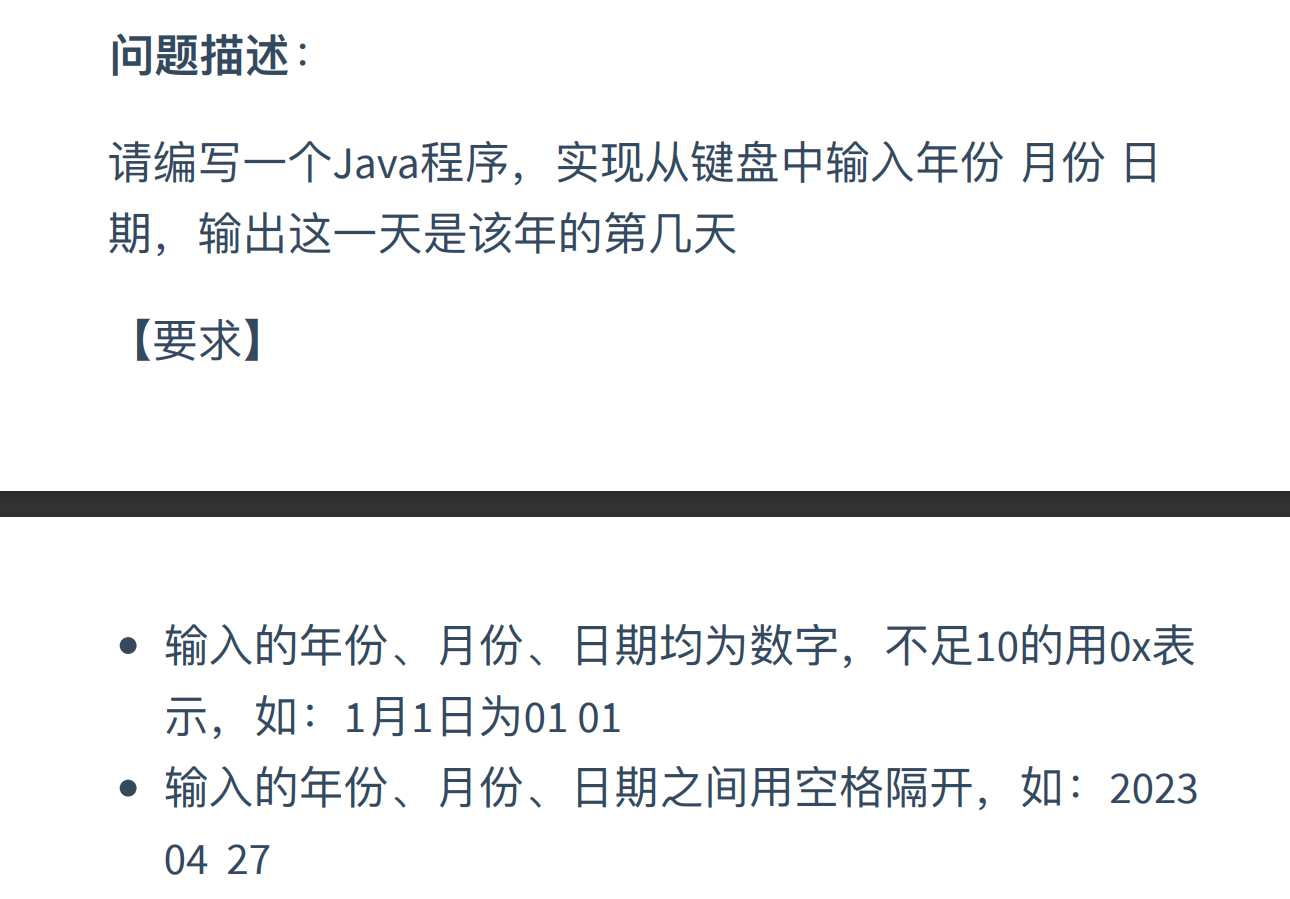
22.     }

23. }

24.

**运行结果：  
**

## 天数计算



**答：**

1. import java.util.Scanner;

2.

3. public class Test {

4.     public static void main(String[] args) {

5.         Scanner scanner = new Scanner(System.in);

6.         // 读取一行输入

7.         System.out.println("请输入年份、月份和日期，使用空格分隔: ");

8.         String input = scanner.nextLine();

9.         // 分割输入

10.         // 使用正则表达式 "\\s+" 将输入字符串按一个或多个空格分割成字符串数组

11.         String[] parts = input.split("\\s+");

12.         int year = Integer.parseInt(parts[0]);//提取年份

13.         int month = Integer.parseInt(parts[1]);//提取月份

14.         int day = Integer.parseInt(parts[2]);//提取日期

15.         // 计算这一天是该年的第几天

16.         int dayOfYear = calculateDayOfYear(year, month, day);

17.         // 输出结果

18.         System.out.println("这一天是该年的第 " + dayOfYear + " 天。");

19.         scanner.close();

20.     }

21.     private static int calculateDayOfYear(int year, int month, int day) {

22.         int[] daysInMonth = {0, 31, isLeapYear(year) ? 29 : 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

23.         int dayOfYear = day;

24.         // 逐月累加天数

25.         for (int i = 1; i < month; i++) {

26.             dayOfYear += daysInMonth[i];

27.         }

28.         return dayOfYear;

29.     }

30.     private static boolean isLeapYear(int year) {

31.         // 判断是否为闰年

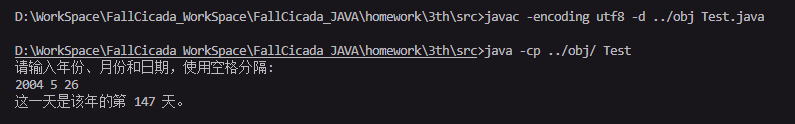
32.         return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

33.     }

34. }

35.

**运行结果：**

****

## 完全数计算

**答：**

1. import java.util.Scanner;

2.

3. public class Test {

4.     public static void main(String[] args) {

5.         Scanner scanner = new Scanner(System.in);

6.         System.out.println("请输入一个整数：");

7.         int num = scanner.nextInt();

8.         // 判断num是否为完全数

9.         if (isPerfectNumber(num)) {

10.             System.out.println("是完全数");

11.         } else {

12.             System.out.println("不是完全数");

13.         }

14.     }

15.

16.     static boolean isPerfectNumber(int num) {

17.         int sum = 0;

18.         // 遍历从1到num/2的所有数字，累加所有能整除num的数

19.         for (int i = 1; i <= num / 2; i++) {

20.             if (num % i == 0) {

21.                 sum += i;

22.             }

23.         }

24.         // 如果累加和等于原数，则为完全数

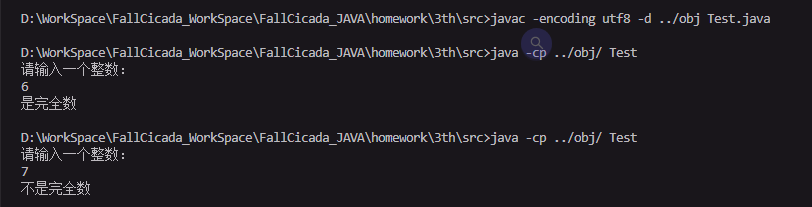
25.         return sum == num;

26.     }

27. }

28.

**运行结果：**

****

## 图形输出



**答：**

1. public class Test {

2.     public static void main(String[] args) {

3.         int height = 4; // 图形的高度

4.

5.         // 输出上半部分（包括中间行）

6.         for (int i = 1; i <= height; i++) {

7.             // 输出空格

8.             for (int j = 1; j <= height - i; j++) {

9.                 System.out.print(" ");

10.             }

11.             // 输出星号

12.             for (int j = 1; j <= 2 \* i - 1; j++) {

13.                 System.out.print("\*");

14.             }

15.             System.out.println();

16.         }

17.

18.         // 输出下半部分

19.         for (int i = height - 1; i >= 1; i--) {

20.             // 输出空格

21.             for (int j = 1; j <= height - i; j++) {

22.                 System.out.print(" ");

23.             }

24.             // 输出星号

25.             for (int j = 1; j <= 2 \* i - 1; j++) {

26.                 System.out.print("\*");

27.             }

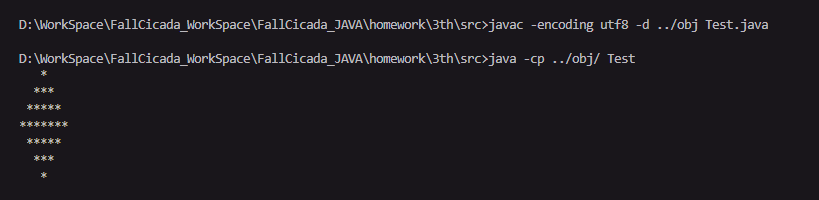
28.             System.out.println();

29.         }

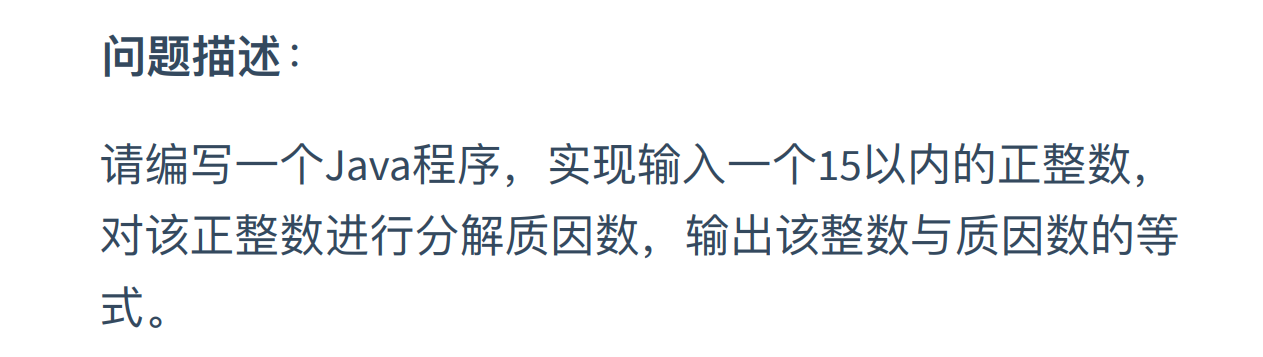
30.     }

31. }

32.

****

## 分解质因数



**答：**

1. import java.util.Scanner;

2.

3. public class Test {

4.     public static void main(String[] args) {

5.         Scanner scanner = new Scanner(System.in);

6.         int num = scanner.nextInt();

7.         // 判断是否符合计算范围要求

8.         if (num > 0 && num < 15) {

9.             // 进行分解质因数计算

10.             StringBuilder factors = new StringBuilder();

11.             printPrimeFactors(num, factors);

12.             // 输出num与质因数的等式

13.             System.out.println(num + " = " + factors.toString());

14.         } else {

15.             System.out.println("输入数字不符合要求");

16.         }

17.     }

18.     private static void printPrimeFactors(int num, StringBuilder factors) {

19.         // 临时变量，用于在质因数分解过程中逐步除以因数

20.         int tempNum = num;

21.         // 从2开始遍历到num，寻找所有的质因数

22.         for (int i = 2; i <= num; i++) {

23.             // 当tempNum能被当前因数i整除时，进入循环

24.             while (tempNum % i == 0) {

25.                 // 将当前因数追加到结果字符串中

26.                 factors.append(i);

27.                 // 如果tempNum除以i的商大于1，说明还有未分解的因数，追加分隔符

28.                 if (tempNum / i > 1) {

29.                     factors.append(" × ");

30.                 }

31.                 // 更新tempNum为tempNum除以i的商，继续分解

32.                 tempNum /= i;

33.             }

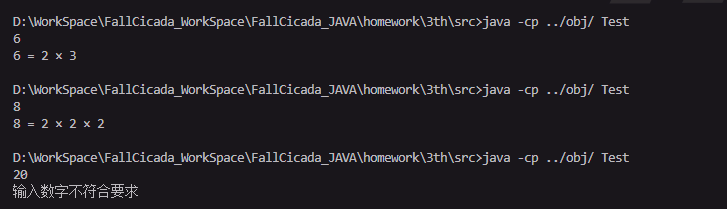
34.         }

35.     }

36. }

37.

**运行结果：**

****