Java语言基础 Day02

1 数据交换程序(选做)

1.1 问题

实现两个变量间的数据交换，例如：有两个整数类型变量a和b，现需要使变量a的值和变量b的值进行交换。例如： a的原值为100，b的原值为200，交换后a的值为200，b的值为100。

1.2 方案

系统使用中间变量的方式来解决上述问题。

步骤1：把a变量的值赋给中间变量，代码如下所示:

```
01.     int temp = a;
```

步骤2：把b变量的值赋给a变量，代码如下所示:

```
01.     a = b;
```

步骤3：把中间变量的值赋给b变量，代码如下所示:

```
01.     b = temp;
```

最终实现了a变量的值和b变量的值的互换。如图-13所示：

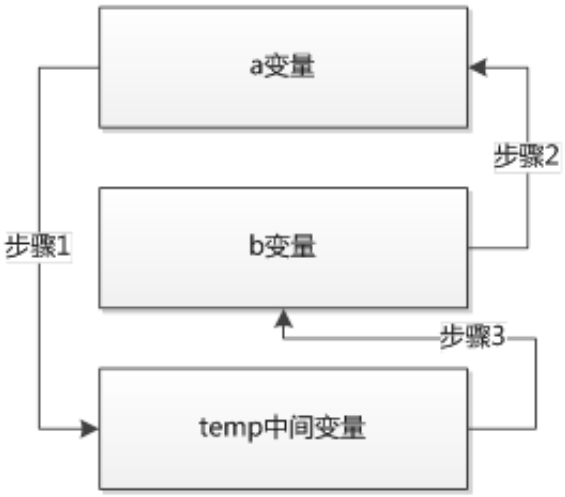


图-13

1.3 实现

系统代码实现如下：

```
01.     public class Swap {
02.         public static void main(String[] args) {
03.             int a = 100;
04.             int b = 200;
05.             System.out.println("a=" + a + ", b=" + b);
06.             int temp = a;
07.             a = b;
08.             b = temp;
09.             System.out.println("a=" + a + ", b=" + b);
10.         }
11.     }
```

隐藏

1.4 扩展

不使用中间变量来实现数据交换程序，即实现两个变量值的交换。可以先把a+b值赋给a，接着把a-b的值赋给b，最后把a-b的值在赋给a，请看如下代码：

```
01.     public class SwapExt {
02.         public static void main(String[] args) {
03.             int a = 100;
04.             int b = 200;
05.             System.out.println("a=" + a + ", b=" + b);
06.             a = a + b;
```

```
07.             b = a - b;
08.             a = a - b;
09.             System.out.println("a=" + a + ", b=" + b);
10.         }
11.     }
```

隐藏

2 计算自由落体运动中物体的位移

2.1 问题

该系统使用交互的方式计算自由落体运动中物体的位置。用户从控制台输入自由落体的时间t，系统计算经过时间t物体的自由落体位移。计算结果保留一位小数并且输出到控制台。系统交互过程如图-14所示：

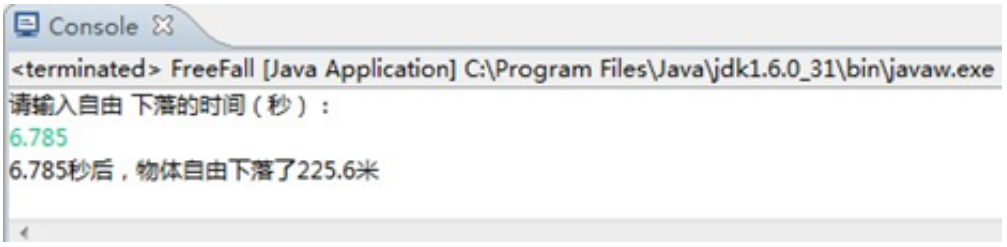


图-14

注：自由落体位移公式为： $s = 1/2 \times g \times t2$ ，其中：

s (位移（m）

t (时间（s）

g (重力加速度（9.8m/s2）

例如：从控制台接收到的时间t为10s，那么位移s=1/2 × 9.8 × 10 × 10=490.0

2.2 方案

系统使用double类型接收控制台输入的时间数据t。调用Scanner的nextDouble()方法接收用户从控制台输入的时间数据，该方法的返回值为double类型。代码如下所示：

```
01.     double t = scanner.nextDouble();
```

系统根据自由落体的位移公式计算出位移值s，s的类型为double，代码如下所示：

```
01.     double s = 0.5 * g * t * t;
```

系统为了确保小数点后保留一位小数可以使用如下方法：

```
01.     s = Math.round(10 * s) / 10.0;
```

Math的round(double s)方法用于实现四舍五入的计算，返回值为long类型的数据。

需要注意的是round(double s)返回值为long类型，要除以double类型的数据后，其结果才会自动类型转换为double类型，因此被除数为double直接量10.0而不是int直接量10。

2.3 实现

系统代码实现如下：

```
01.     import java.util.Scanner;
02.     public class FreeFall {
03.         public static void main(String[] args) {
04.             Scanner scanner = new Scanner(System.in);
05.             double g = 9.80;
06.             System.out.println("请输入自由下落的时间（秒）：");
07.             double t = scanner.nextDouble();
08.             double s = 0.5 * g * t * t;
09.             s = Math.round(10 * s) / 10.0;
10.             System.out.println(t + "秒后，物体自由下落了" + s + "米");
11.         }
12.     }
```

隐藏

2.4 扩展(选做)

用户从控制台输入物体下落的距离（米），然后计算下落此距离所需要花费的时间（秒）。计算结果需要保留一位小数，并且将结果输出到控制台。系统交互过程如图-15所示：

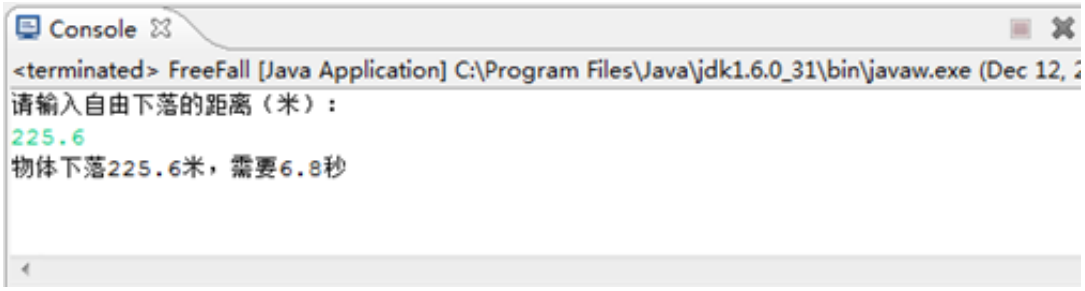


图-15

系统代码实现如下：

```
01.     import java.util.Scanner;
02.     public class FreeFallExt {
```

```
03.         public static void main(String[] args) {
04.             Scanner scanner = new Scanner(System.in);
05.             double g = 9.80;
06.             System.out.println("请输入自由下落的距离（米）：");
07.             double s = scanner.nextDouble();
08.             scanner.close();
09.             double t = Math.sqrt(2 * s / g);
10.             t = Math.round(10 * t) / 10.0;
11.             System.out.println("物体下落" + s + "米，需要" + t + "秒");
12.     }
```

隐藏

3 等额本息还款计算器（选做）

3.1 问题

系统使用交互的方式计算等额本息贷款的还款情况。用户从控制台输入贷款本金，贷款月利率和还款年数，系统根据上述信息计算每月还款金额，要求每月还款金额保留两位小数并且输出到控制台。系统交互过程如图-16所示：

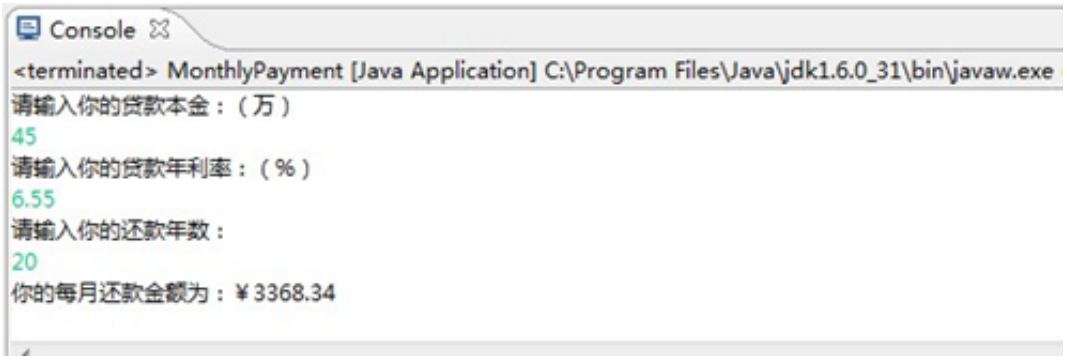


图-16

注：等额本息还款公式 ：payment = (p × r × (1 + r)m) / ((1 + r)m - 1)，

其中：

p (贷款本金（元）

r (贷款月利率

m (还款月数

3.2 方案

系统使用double类型定义贷款本金和贷款月利率，代码如下所示：

```
01.     double p = scanner.nextDouble() * 10000;
02.     double r = scanner.nextDouble() / 1200;
```

系统使用int类型定义还款年数，代码如下所示：

```
01.     int m = scanner.nextInt() * 12;
```

系统根据等额本息还款公式计算出还款情况，使用Math.pow（double a,double b）计算幂运算，该方法返回a的b次幂的值，代码如下所示：

```
01.     double payment = (p * r * Math.pow((1 + r), m))
02.                     / (Math.pow(1 + r, m) - 1);
```

系统使用Math.round()方法确保还款金额保留两位小数，代码如下所示：

```
01.     Math.round(payment * 100) / 100.0;
```

3.3 实现

系统代码实现如下：

```
01.     import java.util.Scanner;
02.     public class MonthlyPayment {
03.         public static void main(String[] args) {
04.             Scanner scanner = new Scanner(System.in);
05.             System.out.println("请输入你的贷款本金：（万）");
06.             double p = scanner.nextDouble() * 10000;
07.             System.out.println("请输入你的贷款年利率：（%）");
08.             double r = scanner.nextDouble() / 1200;
09.             System.out.println("请输入你的还款年数：");
10.             int m = scanner.nextInt() * 12;
11.             double payment = (p * r * Math.pow((1 + r), m))
12.                     / (Math.pow(1 + r, m) - 1);
13.             payment = Math.round(payment * 100) / 100.0;
14.             System.out.println("你的每月还款金额为：￥" + payment);
15.         }
16.     }
```

隐藏

3.4 扩展

计算使用等额本金方式还贷的情况。

用户从控制台输入贷款本金，贷款月利率和还款年数，系统根据上述信息计算第一个月和第二个月的还款金额，要求每月还款金额保留两位小数并且输出到控制台。系统交互过程如图-17所示：

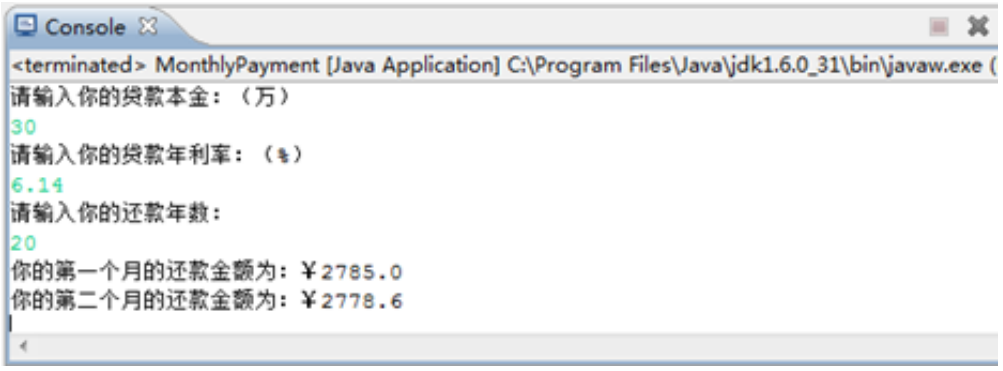


图-17

等额本金还款公式为 ：每月还款额=贷款本金/贷款期月数+（本金-已归还本金累计额）×月利率

比如，贷款金额（即贷款本金）为30万元，贷款年利率为 6.14%，还款年数为 20年，则计算方式如下：

每月本金:300000/240 = 1250

月利率：6.14%/12

首月还款：1250 + 300000 * 6.14%/12 = 2785

第 2 月还款： 1250 + (300000 - 1250) * 6.14%/12 = 2778.60

注：第一个月已归还的本金1250 在第二个月就不用再计算利息。

系统代码实现如下：

```
01.     import java.util.Scanner;
02.     public class MonthlyPaymentExt {
03.             Scanner scanner = new Scanner(System.in);
04.             System.out.println("请输入你的贷款本金：（万）");
05.             double p = scanner.nextDouble() * 10000;
06.             System.out.println("请输入你的贷款年利率：（%）");
07.             double r = scanner.nextDouble() / 1200;
08.             System.out.println("请输入你的还款年数：");
09.             int m = scanner.nextInt() * 12;
10.             scanner.close();
11.
12.             double firstPayment = p / m + p * r;
13.             firstPayment = Math.round(firstPayment * 100) / 100.0;
```

```
14.             System.out.println("你的第一个月的还款金额为：￥" +
15.                     firstPayment);
16.
17.             double secondPayment = p / m + (p - p / m) * r;
18.             secondPayment = Math.round(secondPayment * 100) / 100.0;
19.             System.out.println("你的第二个月的还款金额为：￥" +
20.                     secondPayment);
21.         }
22.     }
```

隐藏

4 计算牛郎星到织女星的距离（选做）

4.1 问题

计算牛郎星到织女星的距离。牛郎星到织女星的距离为16.4光年，如果一只喜鹊的长度是0.46米，计算一下牛郎织女真的要会面需要动用多少只喜鹊。控制台输出信息如图-18所示：

图-18

注：

光速：299792458米/秒

1光年=（1 * 365 * 24 * 60 * 60 * 光速 / 1000）公里

1公里=1千米

4.2 方案

使用double类型定义光年变量和喜鹊的长度变量，代码如下所示：

```
01.     double dLightYear = 16.4;
02.     double dMagpie = 0.46;
```

使用int类型定义光速变量，没有超过int的最大值，故使用int类型定义光速变量，代码如下所示：

```
01.     int speedOfLight = 299792458;
```

根据公式计算牛郎星到织女星的距离，使用long类型定义牛郎星到织女星的距离变量,代码如下所示：

```
01.     long d = (long) (dLightYear * 365 * 24 * 60 * 60 *
02.     speedOfLight / 1000);
```

根据公式计算出牛郎织女会面，搭起鹊桥需要的喜鹊只数，超过int类型的最大，但是小于long的最大值，故使用long类型定义需要喜鹊的只数变量，代码如下所示：

```
01.     long numberOfMagpie = (long) (d * 1000 / dMagpie);
```

4.3 实现

系统代码实现如下：

```
01.     public class AltairVega {
02.         public static void main(String[] args) {
03.             double dLightYear = 16.4; // 光年
04.             int speedOfLight = 299792458;
05.             long d = (long) (dLightYear * 365 * 24 * 60 * 60 *
06.     speedOfLight / 1000);
07.             System.out.println("牛郎星到织女星的距离是" +
08.     dLightYear + "光年,合" + d + "公里。");
09.             double dMagpie = 0.46;
10.             System.out.println("一只成年喜鹊的长度是" +
11.     dMagpie * 100 + "厘米。");
12.             long numberOfMagpie = (long) (d * 1000 / dMagpie);
13.             System.out.println("搭起鹊桥需要" + numberOfMagpie+"只喜鹊。");
14.         }
15.     }
```

隐藏

4.4 扩展

据称，一个人一生中要走的路加起来可以绕地球七十五圈，地球的平均半径为 6371.004 千米，而一个人走一步的步长大约是 60 厘米。请计算一个人一生中要走多少步？系统交互过程如图 - 19所示。

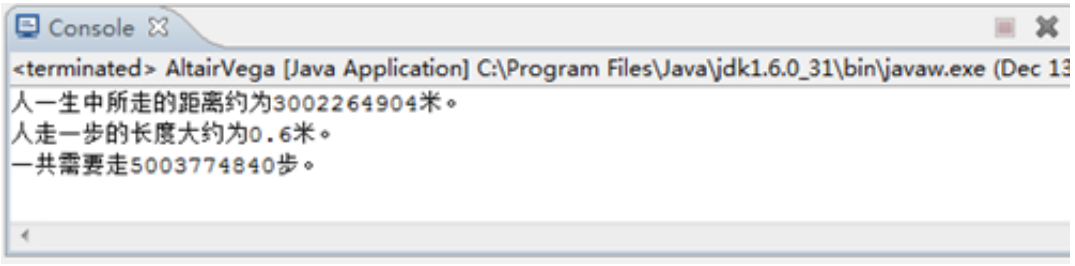


图-19

系统代码实现如下：

```
01.     public class AltairVegaExt {
02.         public static void main(String[] args) {
03.             int radius = 6371004; // 地球半径
04.             double girth = Math.PI * radius * 2;
05.             long totalLength = (long) (girth * 75);
06.             System.out.println("人一生中所走的距离约为" + totalLength +
07.                     "米。");
08.
09.             double stepLength = 0.6;
10.             System.out.println("人走一步的长度大约为" + stepLength + "米。");
11.
12.             long stepCount = (long) (girth * 75 / stepLength);
13.             System.out.println("一共需要走" + stepCount + "步。");
14.         }
15.     }
```

隐藏

5 符号函数程序

5.1 问题

使用交互的方式计算sgn (x)函数的值，用户在控制台输入x的值，系统计算出sgn(x)函数的值，要求使用三目运算符来实现。系统交互信息如图-20所示：

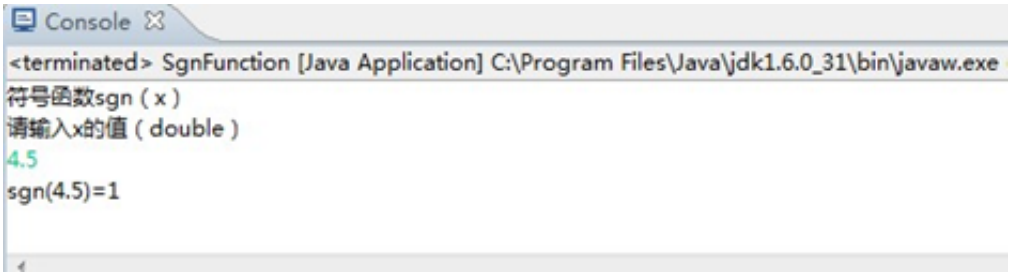


图-20

注：

当x>0时，sgn(x)=1；

当x=0时，sgn(x)=0；

当x<0时，sng(x)=-1。

5.2 方案

使用三目运算符计算符号函数sgn(x)的值，代码如下：

```
01.     int sgn = x > 0 ? 1 : (x < 0 ? -1 : 0);
```

5.3 实现

系统代码实现如下：

```
01.     import java.util.Scanner;
02.     public class SgnFunction {
03.         public static void main(String[] args) {
04.             Scanner scanner = new Scanner(System.in);
05.             System.out.println("符号函数sgn（x）");
06.             System.out.println("请输入x的值（double）");
07.             double x = scanner.nextDouble();
08.             int sgn = x > 0 ? 1 : (x < 0 ? -1 : 0);
09.             System.out.println("sgn(" + x + ")=" + sgn);
10.         }
11.     }
```

隐藏

5.4 扩展（选做）

用户在控制台输入 3 个数值，需要找出这 3 个数值中的最大值，要求使用三目运算符来实现。系统交互信息如图-21所示：

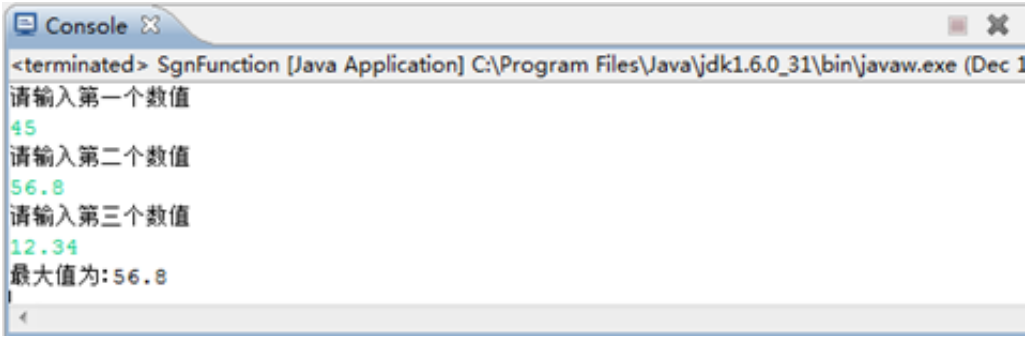


图-21

系统代码实现如下：

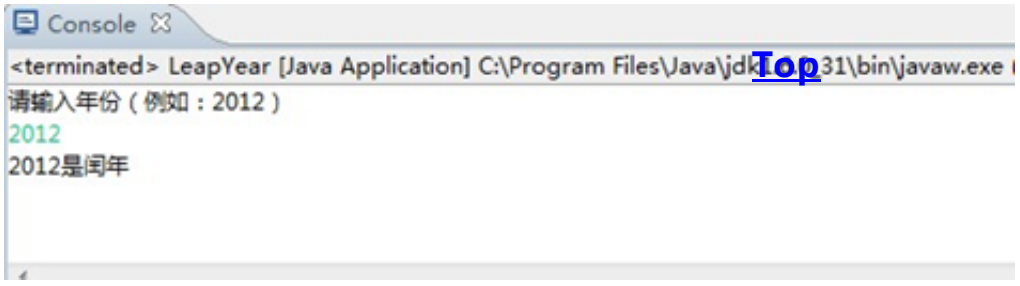
```
01.  import java.util.Scanner;
02.  public class SgnFunctionExt {
03.      public static void main(String[] args) {
04.          Scanner scanner = new Scanner(System.in);
05.          System.out.println("请输入第一个数值");
06.          double d1 = scanner.nextDouble();
07.          System.out.println("请输入第二个数值");
08.          double d2 = scanner.nextDouble();
09.          System.out.println("请输入第三个数值");
10.          double d3 = scanner.nextDouble();
11.          scanner.close();
12.
13.          double temp = d1 > d2 ? d1 : d2;
14.          double result = temp > d3 ? temp : d3;
15.          System.out.println("最大值为: " + result);
16.      }
17.  }
```

隐藏

6 闰年判断程序

6.1 问题

判断某年是否为闰年。根据用户在控制台输入的年份值，判断该年是否为闰年。系统交互情况如图-22所示：



Top

图-22

6.2 方案

使用数学运算符取余运算（%），关系运算符等于（==）和不等于（！=），逻辑运算符逻辑与（&&）和逻辑或（||），来判断某年是否为闰年，判断的结果为boolean类型的值，如果为闰年boolean类型的值为true，否则为false，代码如下所示：

```
01.  boolean isLeapYear = (year % 4 == 0 && year % 100 != 0)
02.                  || year % 400 == 0;
```

使用三目运算符，获取是否为闰年的String类型信息，输出到控制台，代码如下所示：

```
01.     String msg = isLeapYear ? year + "是闰年" : year + "不是闰年";
```

6.3 实现

系统代码实现如下：

```
01.     import java.util.Scanner;
02.     public class LeapYear {
03.         public static void main(String[] args) {
04.             Scanner scanner = new Scanner(System.in);
05.             System.out.println("请输入年份（例如：2012）");
06.             int year = scanner.nextInt();
07.             boolean isLeapYear = (year % 4 == 0 && year % 100 != 0)
08.                     || year % 400 == 0;
09.             String msg = isLeapYear ? year + "是闰年" : year + "不是闰年";
10.             System.out.println(msg);
11.         }
12.     }
```

隐藏

6.4 扩展（选做）

计算某年某月的天数。由用户在控制台输入年份和月份值，然后计算年该月的天数，并输出在控制台。系统交互情况如图-23所示：

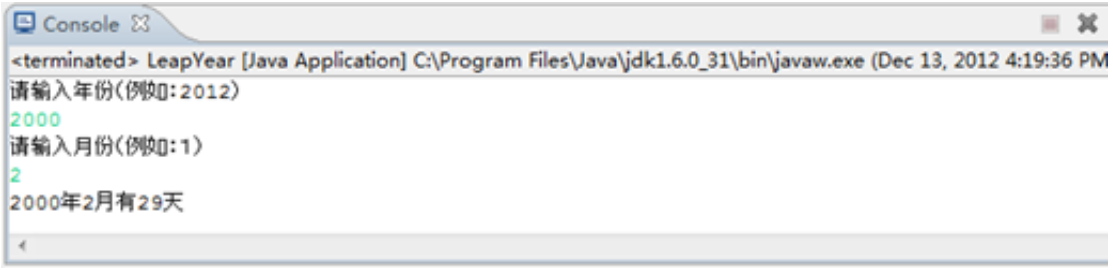


图-23

注：大月有31天，小月有30天，闰年的2月有29天，非闰年的2月有28天。

系统代码实现如下：

```
01.     import java.util.Scanner;
```

```
02.     public class LeapYear {
03.         public static void main(String[] args) {
04.             Scanner scanner = new Scanner(System.in);
05.             System.out.println("请输入年份（例如：2012）");
06.             int year = scanner.nextInt();
07.             System.out.println("请输入月份（例如： 1 ）");
08.             int month = scanner.nextInt();
09.             scanner.close();
10.
11.             // 判断是否是闰年
12.             boolean isLeapYear = (year % 4 == 0 && year % 100 != 0)
13.                     || year % 400 == 0;
14.
15.             // 判断大月和小月，2月除外
16.             boolean isLittleMonth = month == 4 || month == 6 || month == 9
17.                     || month == 11;
18.             boolean isLargeMonth = month == 1 || month == 3 || month == 5
19.                     || month == 7 || month == 8 || month == 10
20.                     || month == 12;
21.             // 计算天数
22.             int days = isLittleMonth ? 30 : (isLargeMonth ? 31 :
23.                     (isLeapYear ? 29: 28));
24.             // 输出
25.             System.out.println(year + "年" + month + "月有" + days + "天");
26.     }
```

隐藏