Java语言基础 Day03

1 数字排序程序

1.1 问题

系统使用交互的方式给三个数字进行排序。例如：接收用户输入的三个整数a，b，c，a的原值是20，b的原值是5，c的原值是10，排序后a的值为5，b的值为10，c的值为20。系统交互情况如图-26所示：

图-26

1.2 方案

系统使用if语句实现对三个数的排序，首先判断a和b的大小情况并进行交换，代码如下：

```
01.     if (a > b) {
02.         temp = a;
03.         a = b;
04.         b = temp;
05.     }
```

其次判断a和c的大小情况，并进行交换，代码如下：

```
01.     if (a > c) {
02.         temp = a;
03.         a = c;
04.         c = temp;
05.     }
```

最后判断b和c的大小情况，并进行交换，代码如下：

```
    if (b > c) {
        temp = b;
        b = c;
        c = temp;
    }
```

1.3 实现

系统代码实现如下：

```
import java.util.Scanner;
public class MaxofThree {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("请依次输入三个整数：a，b，c（以空格隔开）");
        int a = scanner.nextInt();
        int b = scanner.nextInt();
        int c = scanner.nextInt();
        scanner.close();
        System.out.println("a=" + a + ", b=" + b + ", c=" + c);
        int temp = 0;
        if (a > b) {
            temp = a;
            a = b;
            b = temp;
        }
        if (a > c) {
            temp = a;
            a = c;
            c = temp;
        }
        if (b > c) {
            temp = b;
            b = c;
```

```
            c = temp;
        }
        System.out.println("a=" + a + ", b=" + b + ", c=" + c);
    }
}
```

隐藏

1.4 扩展

用户在控制台输入 3 个数值，需要找出这 3 个数值中的最小值，要求使用 if 语句来实现。系统交互信息如图-27所示：

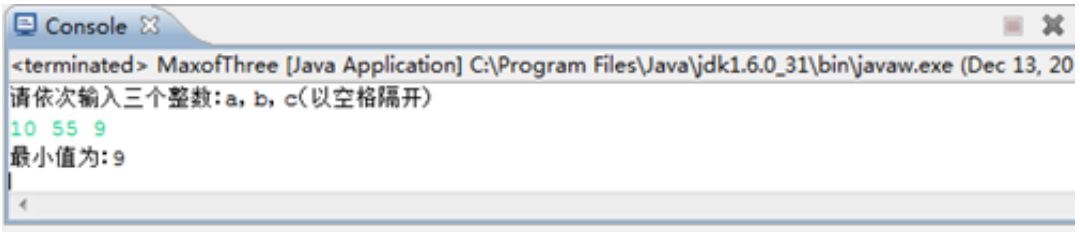

图-27

系统代码实现如下：

```
01.  import java.util.Scanner;
02.  public class MinofThree {
03.      public static void main(String[] args) {
04.          Scanner scanner = new Scanner(System.in);
05.          System.out.println("请依次输入三个整数：a，b，c（以空格隔开）");
06.          int a = scanner.nextInt();
07.          int b = scanner.nextInt();
08.          int c = scanner.nextInt();
09.          scanner.close();
10.
11.          int temp = a;
12.          if (temp > b) {
13.              temp = b;
14.          }
15.          if (temp > c) {
16.              temp = c;
17.          }
18.
19.          System.out.println("最小值为：" + temp);
20.      }
```

```
21.     }
```

隐藏

2 个人所得税计算器

2.1 问题

计算个人所得税的缴纳情况。用户从控制台输入税前工资的金额，系统根据用户输入的工资金额计算应缴纳的税额，如图-28所示：

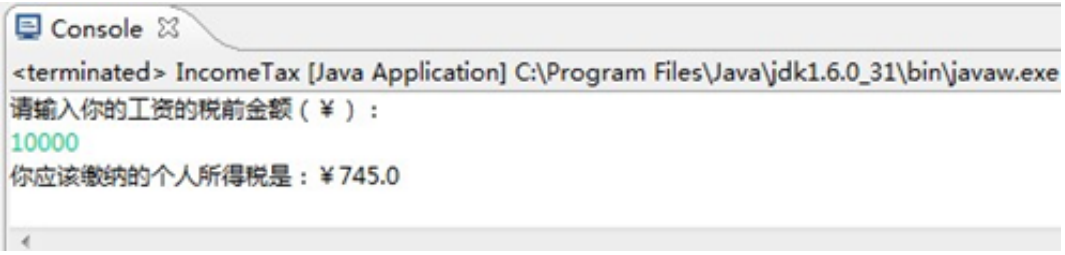

图-28

注：

工资个税的计算公式为：

应纳税额 =（工资薪金所得 - 扣除数）×适用税率 - 速算扣除数

全月应纳税所得额=月工资薪金所得 - 扣除数

2011年 9月1日起执行7级超额累进税率：扣除数为3500元。

个人所得税缴纳标准

全月应纳税所得额	税率	速算扣除数（元）
全月应纳税额不超过 1500 元	3%	0
全月应纳税额超过 1500 元至 4500 元	10%	105
全月应纳税额超过 4500 元至 9000 元	20%	555
全月应纳税额超过 9000 元至 35000 元	25%	1005
全月应纳税额超过 35000 元至 55000 元	30%	2755
全月应纳税额超过 55000 元至 80000 元	35%	5505
全月应纳税额超过 80000 元	45%	13505

2.2 方案

使用if语句的下列结构来完成该题目：

if(boolean表达式1){...}

else if(boolean表达式2){...}

else if(boolean表达式3){...}

.....

else{...}

如图-29所示：（图中T表示true，F表示false，taxIncome表示全月应纳税所得额，tax表示应纳税额）

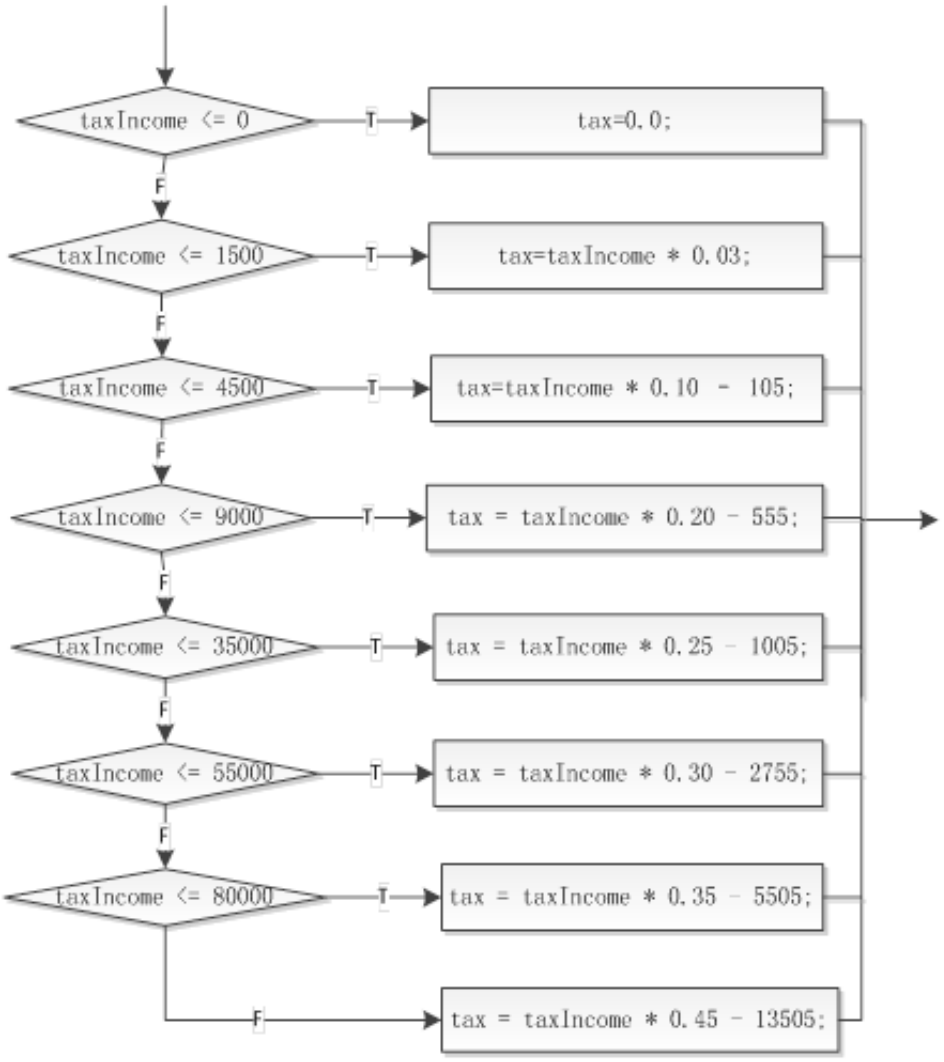

图-29

2.3 实现

系统代码实现如下：

```
01.     public class IncomeTax {
02.         public static void main(String[] args) {
03.             Scanner scanner = new Scanner(System.in);
04.             System.out.println("请输入你的工资的税前金额（￥）：");
05.             double salary = scanner.nextDouble();
06.             double tax = 0.0;
07.             double taxIncome = salary - 3500;
08.             if (taxIncome <= 0) {
09.                 tax = 0.0;
10.             } else if (taxIncome <= 1500) {
11.                 tax = taxIncome * 0.03;
12.             } else if (taxIncome <= 4500) {
```

```
13.                 tax = taxIncome * 0.10 - 105;
14.             } else if (taxIncome <= 9000) {
15.                 tax = taxIncome * 0.20 - 555;
16.             } else if (taxIncome <= 35000) {
17.                 tax = taxIncome * 0.25 - 1005;
18.             } else if (taxIncome <= 55000) {
19.                 tax = taxIncome * 0.30 - 2755;
20.             } else if (taxIncome <= 80000) {
21.                 tax = taxIncome * 0.35 - 5505;
22.             } else {
23.                 tax = taxIncome * 0.45 - 13505;
24.             }
25.             System.out.println("你应该缴纳的个人所得税是：￥" + tax);
26.         }
27.     }
```

隐藏

2.4 扩展

计算个人工资收入的保险缴纳情况以及个人所得税。用户从控制台输入税前工资的金额，系统根据用户输入的工资金额计算应缴纳的各项保险的总金额，并计算应该缴纳的税额。系统交互过程如图-30所示：

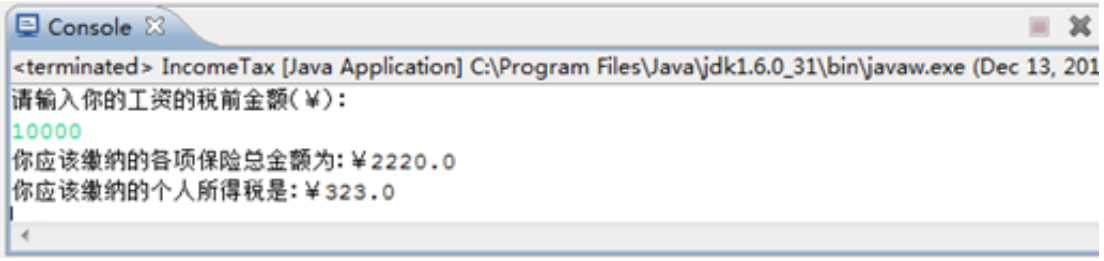

图-30

注：

工资个税的计算公式为：

应纳税额=（工资薪金所得 - 各项保险金额 - 扣除数）×适用税率 - 速算扣除数

全月应纳税所得额=月工资薪金所得 - 各项保险金额 - 扣除数

其中，各项保险金额的缴纳比例如下：

养老保险：月工资薪金所得 × 8%

医疗保险：月工资薪金所得 × 2%

失业保险：月工资薪金所得 × 0.2%

公积金：月工资薪金所得 × 12%

系统代码实现如下：

```java
public class InsuranceTax {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入你的工资的税前金额（￥）：");
        double salary = scanner.nextDouble();
        scanner.close();
        // 计算各项保险金额
        double endowmentInsurance = salary * 0.08;
        double medicalInsurance = salary * 0.02;
        double unemploymentInsurance = salary * 0.002;
        double accumulationFund = salary * 0.12;
        double insurance = endowmentInsurance + medicalInsurance
                + unemploymentInsurance + accumulationFund;
        System.out.println("你应该缴纳的各项保险总金额为：￥" +
                insurance);
        // 计算个税
        double tax = 0.0;
        double taxIncome = salary - insurance - 3500;
        if (taxIncome <= 0) {
            tax = 0.0;
        } else if (taxIncome <= 1500) {
            tax = taxIncome * 0.03;
        } else if (taxIncome <= 4500) {
            tax = taxIncome * 0.10 - 105;
        } else if (taxIncome <= 9000) {
            tax = taxIncome * 0.20 - 555;
        } else if (taxIncome <= 35000) {
            tax = taxIncome * 0.25 - 1005;
        } else if (taxIncome <= 5500) {
            tax = taxIncome * 0.30 - 2755;
        } else if (taxIncome <= 80000) {
            tax = taxIncome * 0.35 - 5505;
        } else {
            tax = taxIncome * 0.45 - 13505;
        }
        System.out.println("你应该缴纳的个人所得税是：￥" + tax);
    }
}
```

隐藏

3 命令解析器

3.1 问题

命令解析器。有如下功能供用户选择：显示全部记录 ，查询登录记录 ，退出。当用户在控制台输入1，用户选择的功能为显示全部记录；输入2，用户选择的功能为查询登录记录；输入0，用户选择的功能为退出。系统交互情况如图-31所示：

图-31

3.2 方案

系统使用while(true)循环实现用户重复的进行选择功能，当while(true)循环遇到break时，循环结束，使用if语句判断用户所选择的功能类型。代码如下

```
01.     while (true) {
02.         System.out.println("
03.     请选择功能: 1.显示全部记录  2.查询登录记录  0.退出");
04.         command = scanner.next();
05.         if ("1".equals(command)) {
06.             System.out.println("显示全部记录");
07.         } else if ("2".equals(command)) {
08.             System.out.println("查询登录记录");
09.         } else if ("0".equals(command)) {
10.             System.out.println("欢迎使用");
11.             break;
12.         }
13.     }
```

3.3 实现

系统代码实现如下：

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

```
02.     public class CommandByIf {
03.         public static void main(String[] args) {
04.             Scanner scanner = new Scanner(System.in);
05.             String command = null;
06.             while (true) {
07.                 System.out.println("
08.     请选择功能: 1.显示全部记录  2.查询登录记录  0.退出");
09.                 command = scanner.next();
10.                 if ("1".equals(command)) {
11.                     System.out.println("显示全部记录");
12.                 } else if ("2".equals(command)) {
13.                     System.out.println("查询登录记录");
14.                 } else if ("0".equals(command)) {
15.                     System.out.println("欢迎使用");
16.                     break;
17.                 }
18.             }
19.             scanner.close();
20.         }
21.     }
```

隐藏

3.4 扩展

使用switch...case语句实现命令解析器程序。

命令解析器程序提供如下功能供用户选择：“显示全部记录”，“查询登录记录”和“退出”。当用户在控制台输入1，则表示选择“显示全部记录”功能，程序在界面输出文本“显示全部记录”；用户输入2，则表示选择“查询登录记录”功能，程序在界面输出文本“查询登录记录”；用户输入0，则表示选择“退出”功能“，程序在界面输出文本”欢迎使用“，且程序结束。

系统实现代码如下：

```
01.     import java.util.Scanner;
02.     public class CommandBySwitch {
03.         public static void main(String[] args) {
04.             Scanner scanner = new Scanner(System.in);
05.             int command = 0;
06.             $1: while (true) {
07.                 System.out.println("
08.     请选择功能: 1.显示全部记录  2.查询登录记录  0.退出");
```

```
09.                 command = scanner.nextInt();
10.                 switch (command) {
11.                 case 1:
12.                     System.out.println("显示全部记录");
13.                     break;
14.                 case 2:
15.                     System.out.println("查询登录记录");
16.                     break;
17.                 case 0:
18.                     System.out.println("欢迎使用");
19.                     break $1;
20.                     default :
21.                 }
22.             }
23.             scanner.close();
24.         }
25.     }
```

隐藏

4 累加、阶乘及求 □算法（求□选做）

4.1 问题

累加、阶乘和求 □ 的算法。先在控制台输出从0累加到10000的和；然后在控制台输出1到20的阶乘；最后求出程序有效精度范围内的 □ 的值，并在界面上输出。控制台输出情况如图-32所示：

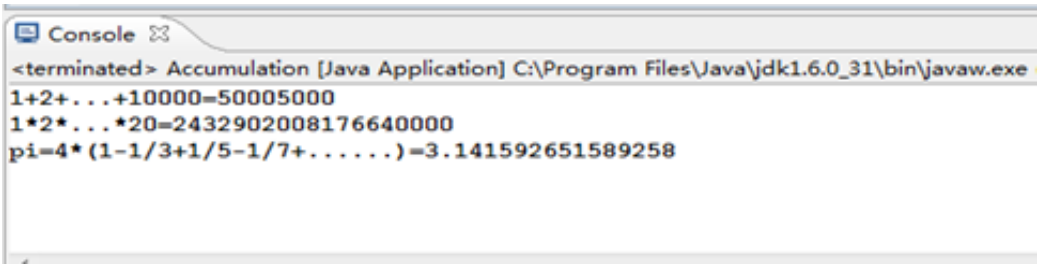

图-32

注：圆周率 π 的计算公式为：π/4=1-1/3+1/5-1/7+1/9-1/11+……。

4.2 方案

系统使用for循环计算从0累加到10000的和。代码如下

```
01.     int sum = 0;
02.     for (int i = 0; i <= 10000; i++) {
03.         sum += i;
```

```
    }
```

系统使用for循环计算1到20的阶乘。代码如下：

```
    long f = 1;
    for (int i = 2; i <= 20; i++) {
        f *= i;
    }
```

为尽量精确的计算 π 的值，可以循环 10 亿次。代码如下：

```
    double pi = 0;
    int flag = -1;
    for (int i = 1; i <= 1000000000; i += 2) {
        flag *= -1;
        pi += flag * 1.0 / i;
    }
```

4.3 实现

系统代码实现如下：

```
public class Accumulation {
    public static void main(String[] args) {
        // 累加
        int sum = 0;
        for (int i = 0; i <= 10000; i++) {
            sum += i;
        }
        System.out.println("1+2+...+10000=" + sum);

        // 阶乘
        long f = 1;
        for (int i = 2; i <= 20; i++) {
            f *= i;
        }
```

```
15.             System.out.println("1*2*...*20=" + f);
16.
17.             // π
18.             double pi = 0;
19.             int flag = -1;
20.             for (int i = 1; i <= 1000000000; i += 2) {
21.                 flag *= -1;
22.                 pi += flag * 1.0 / i;
23.             }
24.             System.out.println("pi=4*(1-1/3+1/5-1/7+.....)=" + 4 * pi);
25.         }
26.     }
```

4.4 扩展(选做)

由用户在控制台输入一个整数，然后实现如下功能：

1.计算从 1 到该整数之间所有偶数的累加和，并在控制台输出计算结果；

2.求出该整数的所有约数，即可以整除该整数的数值（如：10可以被1、2、5、10整除，这些数字是10的约数），并逐一在控制台输出；

系统交互过程如图-33所示：

图-33

系统代码实现如下：

```
01.     public class AccumulationExt {
02.         public static void main(String[] args) {
03.             Scanner scanner = new Scanner(System.in);
04.             System.out.println("请输入整数（例如：100）");
05.             int n = scanner.nextInt();
```

```
06.             scanner.close();
07.
08.             //累加偶数
09.             int sum = 0;
10.             for (int i = 2; i <= n; i = i + 2) {
11.                 sum += i;
12.             }
13.             System.out.println("1到" + n  + "之间所有的偶数的累加和为："
14.                     + sum);
15.             //整除因子
16.             System.out.println(n + "的约数有：");
17.             for (int i = 1; i <= n; i++) {
18.                 if(n % i == 0)
19.                     System.out.println(i);
20.             }
21.         }
22.     }
```

隐藏

5 数列求和

5.1 问题

有数列如：9，99，999，…，9999999999。需要用程序计算此数列的结果，并在控制台输出结果。系统交互情况如图-34所示：

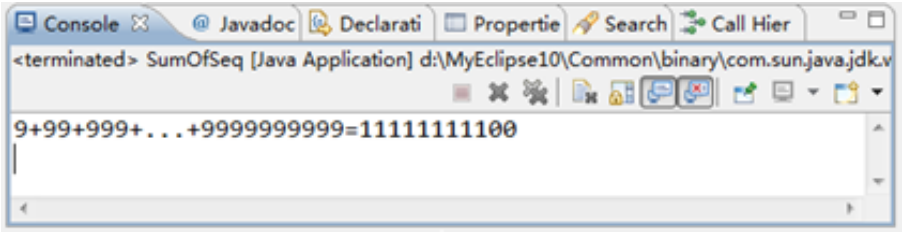

图-34

5.2 方案

此数列的规律为：下一个数值等于前一个数乘以10并加上9。因此，先需要声明用以表示数列的第一个数, 数值较大因此需要使用long类型，代码如：

```
01.     long nine = 9;
```

此数列的和的数值较大，因此，需要使用 long 类型的变量来存储结果。代码如：

```
01.     long result = nine;
```

然后使用for循环计算数列的和。代码如下

```
01.     for (int i = 2; i <= 10; i++) {
02.         nine = nine * 10 + 9;
03.         result += nine;
04.     }
```

5.3 实现

系统代码实现如下：

```
01.     public class SumOfSeq {
02.         public static void main(String[] args) {
03.             // 数列求和
04.             long nine = 9;
05.             long result = nine;
06.             for (int i = 2; i <= 10; i++) {
07.                 nine = nine * 10 + 9;
08.                 result += nine;
09.             }
10.             System.out.println("9+99+999+...+9999999999=" + result);
11.         }
12.     }
```

隐藏

5.4 扩展

计算调和数列的和并将结果输出到控制台。系统交互过程如图-35所示：

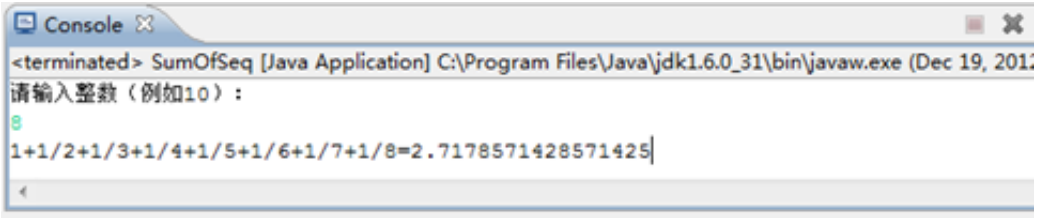

图-35

注：调和数列是指，由自然数的倒数组成的数列，如：1 + 1/2 + 1/3 + … 1/n。

系统代码实现如下：

```java
public class SumOfSeqExt {
    public static void main(String[] args) {
        // 调和数列求和
        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入整数（例如10）：");
        int n = scanner.nextInt();

        double result = 1;
        System.out.print("1+");
        for (int i = 2; i < n; i++) {
            result += 1.0 / i;
            System.out.print("1/" + i + "+");
        }
        result += 1.0 / n;

        System.out.print("1/" + n + "=" + result);
    }
}
```

Top

隐藏