

一.结构描述

1.1 总体情况

笔者设计了4个类来实现本次实验。分别是：

1.Show.java:用来与用户交互

2.TaxCalculator.java:接收用户输入计算个人所得税

3.TaxTable.java:保存个人所得税的有关信息，如个人所得税起征点，每级分割金额，每级税率

4.Main.java:主程序部分，用来载入Show.java

1.2 Show.java

屏幕输出

这部分代码用于与用户的交互，提示用户应该输入相关选择

```
1 // 构造函数，初始化税率表、税务计算器和输入扫描器
2 public Show() {
3     this.taxTable = new TaxTable();
4     this.taxCalculator = new TaxCalculator(taxTable);
5     this.scanner = new Scanner(System.in);
6     System.out.println("A simple calculator(By:w1z)"); // 输出欢迎信息
7 }
8
9 // 显示操作菜单
10 public void displayMenu() {
11     System.out.println("请选择操作: ");
12     System.out.println("1. 计算个人所得税");
13     System.out.println("2. 修改起征点");
14     System.out.println("3. 修改税率的分割金额");
15     System.out.println("4. 修改税率表");
16     System.out.println("5. 查看个人所得税表");
17     System.out.println("6. 退出");
18     displaySep(); // 显示分隔线
19 }
```

读取用户输入

这部分代码用来读取用户的选择，为了提高通用性，笔者将函数设置为接收一个正整数参数的值，通过这个可以接收1-n的用户输入，并且该部分设置了异常处理，对于小于1或大于n的数字都会判断非法，要求用户重新输入。

```
1 // 获取用户选择的操作编号
2 public int getUserChoice(int n) {
3     if (n <= 0)
4         return -1; // n非法
5     int choice = -1;
6     while (choice < 1 || choice > n) {
7         try {
8             System.out.printf("请输入1-%d的数字:", n);
9             choice = scanner.nextInt(); // 读取数字输入
```

```

10         if (choice < 1 || choice > n) { // 输入的数字非法
11             System.out.println("输入错误, 请重新输入。");
12         }
13     } catch (Exception e) { // 如果输入的不是数字
14         System.out.printf("输入错误, 请输入1-%d的数字。\\n", n);
15         scanner.nextLine();
16     }
17 }
18 return choice;
19 }

```

处理用户选择

这部分代码使用一个switch语句来对用户的输入选择进行不同情况的处理, 为了化代码, 将每个操作都封装成了一个函数(除了退出的6选择和非法情况的提示)。这些处理函数在下面说明。

```

1  // 处理用户选择的操作
2  public void handleChoice(int choice) {
3      switch (choice) {
4          case 1:
5              calculateTax();
6              break;
7          case 2:
8              updateThreshold();
9              break;
10         case 3:
11             updateSeparation();
12             break;
13         case 4:
14             updateTaxRate();
15             break;
16         case 5:
17             showTaxTable(); // 展示税率信息
18             break;
19         case 6:
20             System.out.println("再见! 欢迎下次使用!");
21             break;
22         default:
23             System.out.println("无效的选择!");
24             break;
25     }
26 }

```

计算个人所得税

部分代码要求用户输入收入, 并且在代码中包含了异常值的处理(要求用户重新输入)。在得到正确输入后调用 `taxCalculator.calculateTax()` 函数计算个人所得税

```

1  // 计算个人所得税
2  public void calculateTax() {
3      double income = -1;
4      while (income < 0) {
5          try {
6              System.out.println("请输入您的收入:");

```

```

7         income = scanner.nextDouble();
8         if (income < 0) {
9             System.out.println("收入不能为负数，请重新输入！");
10        }
11    } catch (Exception e) {
12        System.out.println("输入错误，请输入有效的收入！");
13        scanner.nextLine();
14    }
15 }
16 double tax = taxCalculator.calculateTax(income); // 计算个人所得税
17 System.out.printf("您应缴纳的个人所得税为：%.2f元\n", tax);
18 displaySep();
19 }

```

更新个人所得税起征点

类似于上面的代码，对异常值输入进行了处理，在得到正常值后调用

`taxTable.setThreshold(threshold)` 来更新个人所得税起征点。

```

1 // 更新个人所得税起征点
2 public void updateThreshold() {
3     double threshold = -1;
4     while (threshold < 0) {
5         try {
6             System.out.print("请输入新的起征点：");
7             threshold = scanner.nextDouble(); // 读取输入
8             if (threshold < 0) { // 新的数据小于0，异常
9                 System.out.println("起征点不能为负数，请重新输入。");
10            }
11        } catch (Exception e) { // 输入的不是数字
12            System.out.println("输入错误，请输入有效的起征点。");
13            scanner.nextLine();
14        }
15    }
16    boolean ret = taxTable.setThreshold(threshold); // 更新个人所得税起征点
17    if (ret) { // 更新成功
18        System.out.printf("起征点已更新为%.2f\n.", threshold);
19    } else { // 程序可能出现编写错误
20        System.out.println("请检查程序是否编写错误!");
21    }
22    displaySep();
23 }

```

更新税率

这部分代码包括对输入税率的异常处理。并且允许用户修改单个税率或者所有税率`updateAllTaxRate()`设计用来修改所有税率，而`updateNthTaxRate()`用来处理修改单个税率。处理的方式也和上面类似(合法的税率应该在0-1之间)，在对输入值异常处理后得到合法的税率调用

`taxTable.setTaxRate(newTaxRate)`或`taxTable.setNthTaxRate(n,newTaxRate)`修改税率。因为在现实情况下，更高级的税率应该要比更低级的税率高，因此在`taxTable`的税率更新函数中要求税率更新后是逐级递增的。因此修改有可能失败，这时候会返回-1，程序可以根据这个返回值确定是否成功修改税率

```

1 // 修改单个税率
2 public void updateNthTaxRate() {

```

```

3      int taxRankNum = taxTable.getTaxRankNum();
4      double newTaxRate = -1;
5      int n = -1;
6      // 读取修改位置
7      while (n < 1 || n > taxRankNum) {
8          try {
9              System.out.printf("请输入要修改税率(1-%d)的位置: ", taxRankNum);
10             n = scanner.nextInt(); // 读取数字输入
11             if (n < 1 || n > taxRankNum) { // 输入的数字非法
12                 System.out.println("输入错误, 请重新输入。");
13             }
14         } catch (Exception e) { // 如果输入的不是数字
15             System.out.printf("输入错误, 请输入1-%d的数字。\\n", taxRankNum);
16             scanner.nextLine();
17         }
18     }
19     // 读取修改税率
20     while (newTaxRate <= 0 || newTaxRate > 1) {
21         try {
22             System.out.printf("请输入要修改的税率: ");
23             newTaxRate = scanner.nextDouble(); // 读取数字输入
24             if (newTaxRate <= 0 || newTaxRate > 1) { // 输入的数字非法
25                 System.out.println("输入错误, 请重新输入。");
26             }
27         } catch (Exception e) { // 如果输入的不是数字
28             System.out.printf("输入错误, 请输入0~1的浮点数。\\n");
29             scanner.nextLine();
30         }
31     }
32     int res = taxTable.setNthTaxRate(n, newTaxRate);
33     if (res != -1) {
34         System.out.printf("第%d级税率已经修改为%.2f\\n", n, newTaxRate);
35     } else {
36         System.out.println("请检查程序是否编写错误!");
37     }
38 }
39
40 // 更新所有税率
41 public void updateAllTaxRate() {
42     int taxRankNum = taxTable.getTaxRankNum(); // 获取税率级数
43     double[] newtaxRate = new double[taxRankNum]; // 存放税率的数组
44     System.out.printf("请输入%d个新税率, 你需要保证是单调上升的\\n",
45 taxRankNum);
46     int succeed = -1;
47     while (succeed == -1) {
48         int i = 0;
49         while (i < taxRankNum) {
50             try {
51                 newtaxRate[i] = scanner.nextDouble(); // 读取输入
52                 if (newtaxRate[i] <= 0 || newtaxRate[i] > 1) { // 输入小于
53 等于0或大于1, 异常
54                     System.out.println("税率不能为负数且不能大于1, 请重新输入。");
55                 } else {
56                     ++i;

```

```

55         }
56     } catch (Exception e) { // 输入不是数字
57         System.out.println("输入错误，请输入有效的税率。");
58         scanner.nextLine();
59     }
60 }
61 succeed = taxTable.setTaxRate(newtaxRate);
62 if (succeed == -1) { // 输入的序列不是单调上升，被拒绝更新
63     System.out.println("您输入的税率序列并不是单调上升的，请重新输入!");
64 }
65 }
66 System.out.println("税率已经全部更新成功!");
67 }

```

更新分割金额

这部分处理和更新税率类似，只是对分割金额的异常判断稍微修改了下。并且要保证第一级分割金额保证为0(在笔者的实现中，第一级分割金额是从个人所得税起征点开始的，如果设置大于0，将会有一部分金额没有被计算个人所得税。因为个人所得税起征点的意义就是从该金额开始应该收税了，如果第一级分割金额不为0则违反了个人所得税起征点的实际意义)

```

1  // 修改分割金额
2  public void updateSeparation() {
3      System.out.println("请选择操作: ");
4      System.out.println("1. 修改单个分割金额");
5      System.out.println("2. 修改全部分割金额");
6      displaySep();
7      int choice = getUserChoice(2); // 读取用户选择
8      switch (choice) {
9          case 1:
10             updateNthSeparation(); // 修改单个分割金额
11             break;
12          case 2:
13             updateAllSeparation(); // 修改全部分割金额
14             break;
15          default:
16             System.out.println("无效的选择!");
17      }
18      displaySep();
19  }
20
21  // 修改单个分割金额
22  public void updateNthSeparation() {
23      int taxRankNum = taxTable.getTaxRankNum(); // 获取税率级数
24      double newSeparation = -1; // 新分割金额
25      int n = -1; // 修改的位置
26      // 读取修改位置
27      while (n <= 1 || n > taxRankNum) // 修改位置非法
28      {
29          try {
30              System.out.printf("请输入要修改金额(2-%d)的位置: ", taxRankNum);
31              n = scanner.nextInt(); // 读取数字输入
32              if (n < 1 || n > taxRankNum) { // 输入的数字非法
33                  System.out.println("输入错误，请重新输入。");

```

```

34         }
35         if (n == 1) {
36             System.out.println("禁止修改第一级分割金额(第一级分割金额固定
为)"); // 第一级分割金额固定为0
37         }
38     } catch (Exception e) { // 如果输入的不是数字
39         System.out.printf("输入错误, 请输入2-%d的数字。\\n", taxRankNum);
40         scanner.nextLine();
41     }
42 }
43 // 读取新的分割金额
44 while (newSeparation < 0) { // 分割金额小于0, 非法
45     try {
46         System.out.print("请输入分割金额: ");
47         newSeparation = scanner.nextDouble(); // 读取数字输入
48         if (newSeparation < 0) { // 输入的数字非法
49             System.out.println("输入错误, 请重新输入。");
50         }
51     } catch (Exception e) { // 如果输入的不是数字
52         System.out.printf("输入错误, 请输入正数。\\n");
53         scanner.nextLine();
54     }
55 }
56 int res = taxTable.setNthSeparation(n, newSeparation);
57 if (res != -1) {
58     System.out.printf("第%d级分割金额已经修改为%.2f\\n", n,
newSeparation);
59 } else {
60     System.out.println("请检查程序是否编写错误!");
61 }
62 }
63
64 // 修改全部分割金额
65 public void updateAllSeparation() {
66     int taxRankNum = taxTable.getTaxRankNum(); // 获取税率级数
67     double[] newSeparation = new double[taxRankNum]; // 存放分割金额的数组
68     System.out.printf("请输入%d个新分割金额, 你需要保证是单调上升的\\n",
taxRankNum);
69     int succeed = -1;
70     while (succeed == -1) {
71         int i = 0;
72         while (i < taxRankNum) {
73             try {
74                 newSeparation[i] = scanner.nextDouble(); // 读取输入
75                 if (newSeparation[i] < 0) { // 分割金额异常
76                     System.out.println("分割不能为负数, 请重新输入。");
77                 } else {
78                     if (i == 0 && newSeparation[i] != 0) {
79                         System.out.println("第一级分割金额必须为0, 请重新输
入。");
80                         continue;
81                     }
82                     ++i;
83                 }
84             } catch (Exception e) {

```

```

85         System.out.println("输入错误，请输入有效的分割金额。");
86         scanner.nextLine();
87     }
88 }
89 succeed = taxTable.setSeparation(newSeparation); // 修改所有分割金额
90 if (succeed == -1) { // 修改失败，说明不是单调上升的
91     System.out.println("您输入的分割金额序列并不是单调上升的，请重新输入!");
92 }
93 }
94 System.out.println("分割金额已经全部更新成功!");
95 }

```

展示个人所得税信息

这部分代码用于获取个人所得税的有关信息，通过taxTable.getThreshold(), taxTable.getSeparation()和taxTable.getTaxRate()可以获取个人所得税起征点，分割金额和各级税率。将它们逐个输出即可。

```

1  // 展示税率信息
2  public void showTaxTable() {
3      System.out.printf("当前个人所得税起征点为%.2f\n",
taxTable.getThreshold());
4      double[] separation = taxTable.getSeparation();
5      double[] taxRate = taxTable.getTaxRate();
6      for (int i = 0; i < taxRate.length; i++) {
7          if (i == taxRate.length - 1) {
8              // 最后一位特殊打印
9              System.out.printf("超过%.2f元的税率为%.2f\n", separation[i],
taxRate[i]);
10             } else {
11                 System.out.printf("超过%.2f到%.2f元税率为%.2f\n",
separation[i], separation[i + 1], taxRate[i]);
12             }
13         }
14         displaySep();
15     }

```

1.3 TaxCalculator.java

这部分代码用于计算个人所得税。核心思想是遍历每一个税收段，在每个税收段中增加的收入应该是本段的税率乘上本段可用的金额，而本段可用的金额要么是前后两端分割金额差，要么是剩余金额(剩余金额不足以跨过这一段，此时所有金额都计算了个人所得税)。

```

1  package hw1;
2
3  //税收计算器
4  public class TaxCalculator {
5      private final TaxTable taxTable;
6
7      public TaxCalculator(TaxTable taxTable) {
8          this.taxTable = taxTable;
9      }
10
11     //计算个人所得税

```

```

12     public double calculateTax(double income) {
13         double taxAbleIncome = income - taxTable.getThreshold();//减去起征点
14         if(taxAbleIncome <= 0) { //没有超过起征点，个人所得税为0
15             return 0;
16         }
17
18         double tax = 0; //个人所得税总和
19         double[] taxRate = taxTable.getTaxRate();//税率
20         double[] separation = taxTable.getSeparation();//分割金额
21
22         for(int i=0; i<taxRate.length; i++) {
23             double taxRateIncome = taxRate[i]; //当前一级税率
24             double taxRatePercentage = separation[i]; //当前一级分割金额
25
26             if(i==taxRate.length-1) { //最后一级税率直接乘即可
27                 tax += taxAbleIncome * taxRateIncome;
28                 break;
29             }
30             else {
31                 double nextTaxRatePercentage = separation[i+1]; //获取下一级分
割金额
32                 //这一级能够计算税的金额是剩余金额和两级分割金额差的最小值
33                 double incomeInThisRate = Math.min(taxAbleIncome,
nextTaxRatePercentage - taxRatePercentage);
34                 //计算本级应得的税收
35                 tax += incomeInThisRate * taxRateIncome;
36                 //更新剩余金额
37                 taxAbleIncome -= incomeInThisRate;
38
39                 if(taxAbleIncome <= 0) {
40                     //所有金额都计算过，退出
41                     break;
42                 }
43             }
44         }
45         return tax; //返回个人所得税
46     }
47 }

```

1.4 TaxTable.java

初始化

这部分代码用于对本类的初始化，初始化初始化的数据从代码中可以看出


```

1 private int rankNum;//税率分界点个数
2 private double threshold;// 个人所得税起点
3 private double[] taxRate;// 税率
4 private double[] separation;// 每一级税率的分割金额
5
6 // 初始化函数
7 public TaxTable() {
8     threshold = 5000;
9     rankNum = 7;
10    separation = new double[] { 0, 3000, 12000, 25000, 35000, 55000,
11    80000 };
12    taxRate = new double[] { 0.03, 0.1, 0.2, 0.25, 0.3, 0.35, 0.45 };
13 }

```

获取和更新个人所得税

这部分代码用于获取和更新个人所得税。在修改个人所得税的输入部分，其实已经保证了输入的合理性，但是为了分离两部分的异常处理，因此笔者在这部分代码仍然保留了异常处理的代码。

```

1 // 获取个人所得税起点
2 public double getThreshold() {
3     return this.threshold;
4 }
5
6 // 设置个人所得税起征点
7 public boolean setThreshold(double newThreshold) {
8     // 小于0，非法输入
9     if (newThreshold < 0)
10        return false;
11    this.threshold = newThreshold;
12    return true;
13 }

```

剩余部分

考虑到篇幅以及这次任务的难度，笔者觉得剩余部分简单写写就好，不辛苦助教了。

剩余部分的功能包括获取税率级数 `rankNum`、分割金额 `separation`、税率 `TaxTable` 以及他们的设置。理论上不需要对这些部分的修改再做异常处理，因为在输入的时候已经确保了输入是合法的。但是笔者认为输入和实现部分代码的关联性相对较小，因此在具体的实现部分仍然保留了异常处理，如果出现错误则返回-1，部分函数是返回false。

1.5 Main.java

这部分是主程序部分，用来载入Show.java与用户交互，调用Show.java的有改观函数来实现与用户的交互，并判断何时终止程序

```

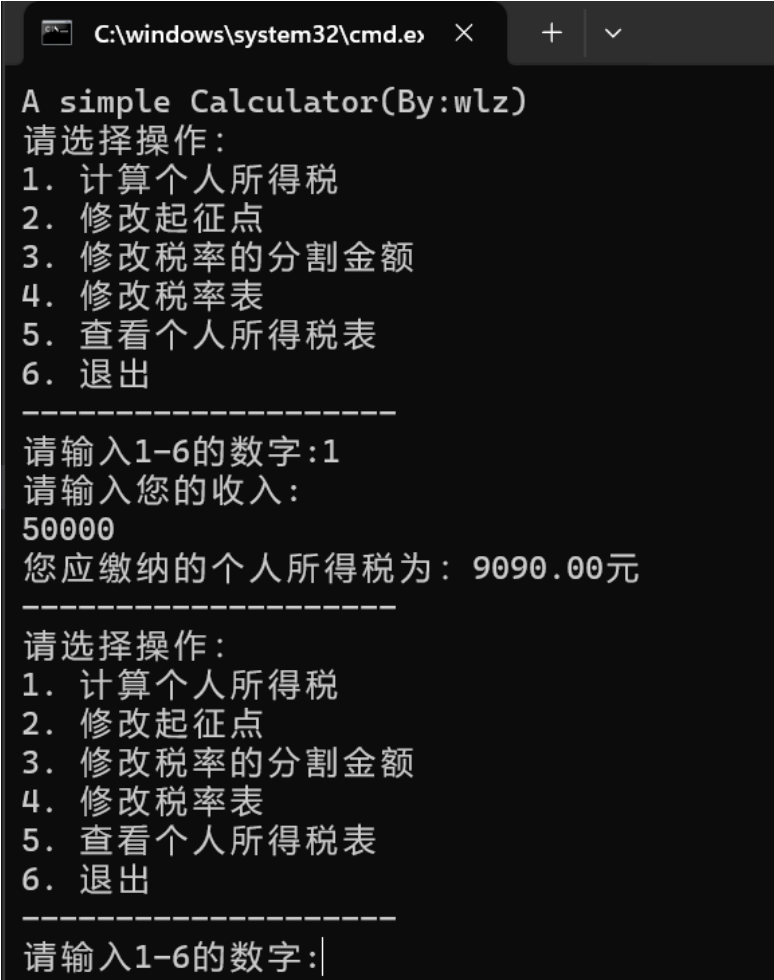
1 package hw1;
2
3 public class Main {
4     public static void main(String[] args) {
5         Show show = new Show();
6         int choice;
7         do{
8             show.displayMenu();//打印菜单
9             choice = show.getUserChoice(6);//获取用户的输入
10            show.handleChoice(choice);//处理用户输入
11        }while(choice != 6);
12    }
13 }

```

二、结果测试

2.1 计算个人所得税

点击 start.bat , 输入1,50000



```

C:\windows\system32\cmd.exe
A simple Calculator(By:wlz)
请选择操作：
1. 计算个人所得税
2. 修改起征点
3. 修改税率的分割金额
4. 修改税率表
5. 查看个人所得税表
6. 退出
-----
请输入1-6的数字:1
请输入您的收入：
50000
您应缴纳的个人所得税为：9090.00元
-----
请选择操作：
1. 计算个人所得税
2. 修改起征点
3. 修改税率的分割金额
4. 修改税率表
5. 查看个人所得税表
6. 退出
-----
请输入1-6的数字:|

```

结果与样例一样

2.2 查看税表信息

输入5，会显示当前税表

```
A simple Calculator(By:wlz)
请选择操作：
1. 计算个人所得税
2. 修改起征点
3. 修改税率的分割金额
4. 修改税率表
5. 查看个人所得税表
6. 退出
-----
请输入1-6的数字:5
当前个人所得税起征点为5000.00
超过0.00到3000.00元税率为0.03
超过3000.00到12000.00元税率为0.10
超过12000.00到25000.00元税率为0.20
超过25000.00到35000.00元税率为0.25
超过35000.00到55000.00元税率为0.30
超过55000.00到80000.00元税率为0.35
超过80000.00元的税率为0.45
-----
请选择操作：
```

2.3 修改起征点

输入2修改起征点，再输入6000，最后输入5查看税表信息

```
3. 修改税率的分割金额
4. 修改税率表
5. 查看个人所得税表
6. 退出
-----
请输入1-6的数字:2
请输入新的起征点: 6000
起征点已更新为6000.00
。-----
请选择操作:
1. 计算个人所得税
2. 修改起征点
3. 修改税率的分割金额
4. 修改税率表
5. 查看个人所得税表
6. 退出
-----
请输入1-6的数字:5
当前个人所得税起征点为6000.00
超过0.00到3000.00元税率为0.03
超过3000.00到12000.00元税率为0.10
超过12000.00到25000.00元税率为0.20
超过25000.00到35000.00元税率为0.25
超过35000.00到55000.00元税率为0.30
超过55000.00到80000.00元税率为0.35
超过80000.00元的税率为0.45
-----
```

2.4 修改分割金额

输入3，再输入1选择修改单个分割金额，再输入3选择第三级，把第三级分割金额从12000改成15000，输入5验证

```
-----
请输入1-6的数字:3
请选择操作:
1. 修改单个分割金额
2. 修改全部分割金额
-----
请输入1-2的数字:1
请输入要修改金额(2-7)的位置: 3
请输入分割金额: 15000
第3级分割金额已经修改为15000.00
-----
请选择操作:
1. 计算个人所得税
2. 修改起征点
3. 修改税率的分割金额
4. 修改税率表
5. 查看个人所得税表
6. 退出
-----
请输入1-6的数字:5
当前个人所得税起征点为6000.00
超过0.00到3000.00元税率为0.03
超过3000.00到15000.00元税率为0.10
超过15000.00到25000.00元税率为0.20
超过25000.00到35000.00元税率为0.25
超过35000.00到55000.00元税率为0.30
超过55000.00到80000.00元税率为0.35
超过80000.00元的税率为0.45
```

2.5 修改税率

输入4修改第三级税率为0.18，再输入5打印税表验证

```
-----  
请选择操作：  
1. 计算个人所得税  
2. 修改起征点  
3. 修改税率的分割金额  
4. 修改税率表  
5. 查看个人所得税表  
6. 退出  
-----
```

```
请输入1-6的数字:5  
当前个人所得税起征点为6000.00  
超过0.00到3000.00元税率为0.03  
超过3000.00到15000.00元税率为0.10  
超过15000.00到25000.00元税率为0.18  
超过25000.00到35000.00元税率为0.25  
超过35000.00到55000.00元税率为0.30  
超过55000.00到80000.00元税率为0.35  
超过80000.00元的税率为0.45  
-----
```

2.6 再一次计算个人所得税

输入1，再输入金额88000，应缴税金额为：

$$1. 88000 - 6000 = 82000$$

$$2. 3000 * 0.03 + 12000 * 0.1 + 10000 * 0.18 + 10000 * 0.25 + 20000 * 0.3 + 25000 * 0.35 + 2000 * 0.45 = 21240$$

与程序计算出的结果一致

请选择操作：

1. 计算个人所得税
 2. 修改起征点
 3. 修改税率的分割金额
 4. 修改税率表
 5. 查看个人所得税表
 6. 退出
-

请输入1-6的数字：1

请输入您的收入：

88000

您应缴纳的个人所得税为： 21240.00元

2.7 修改所有分割金额

输入3，选择2，再输入所有分割金额，分割金额一共7个，按序输入：0,1000,2000,3000,4000,5000,6000，最后输入5验证

第一级分割金额必须为0，请重新输入。

0

1000

2000

3000

4000

5000

6000

分割金额已经全部更新成功！

请选择操作：

1. 计算个人所得税
2. 修改起征点
3. 修改税率的分割金额
4. 修改税率表
5. 查看个人所得税表
6. 退出

请输入1-6的数字：5

当前个人所得税起征点为6000.00

超过0.00到1000.00元税率为0.03

超过1000.00到2000.00元税率为0.10

超过2000.00到3000.00元税率为0.18

超过3000.00到4000.00元税率为0.25

超过4000.00到5000.00元税率为0.30

超过5000.00到6000.00元税率为0.35

超过6000.00元的税率为0.45

修改成功

2.8 修改所有税率

输入4，选择2，再依次输入税率0.05,0.1,0.2,0.3,0.4,0.5,0.6，最后输入5验证


```
请输入7个新税率，你需要保证是单调上升的
0.05
0.1
0.2
0.3
0.4
0.5
0.6
税率已经全部更新成功！
-----
请选择操作：
1. 计算个人所得税
2. 修改起征点
3. 修改税率的分割金额
4. 修改税率表
5. 查看个人所得税表
6. 退出
-----
请输入1-6的数字：5
当前个人所得税起征点为6000.00
超过0.00到1000.00元税率为0.05
超过1000.00到2000.00元税率为0.10
超过2000.00到3000.00元税率为0.20
超过3000.00到4000.00元税率为0.30
超过4000.00到5000.00元税率为0.40
超过5000.00到6000.00元税率为0.50
超过6000.00元的税率为0.60
-----
```

2.9 最后一次计算个人所得税

这一次计算初始金额为20000的个人所得税：

1. $20000 - 6000 = 14000$

2. $1000 * 0.05 + 1000 * 0.1 + 1000 * 0.2 + 1000 * 0.3 + 1000 * 0.4 + 1000 * 0.5 + 8000 * 0.6 = 6350$

与程序计算的结果一致

```
-----
请选择操作：
1. 计算个人所得税
2. 修改起征点
3. 修改税率的分割金额
4. 修改税率表
5. 查看个人所得税表
6. 退出
-----
请输入1-6的数字：1
请输入您的收入：
20000
您应缴纳的个人所得税为： 6350.00元
-----
```

至此，笔者可以认为程序的逻辑是正确的，衔接是紧密无误的。接下来就要测试程序的健壮性。

三、异常检测

3.1 负数起征点

当笔者输入了-100的起征点时，程序检测到并提醒重新输入，再输入200后修改成功。

```
。
请选择操作：
1. 计算个人所得税
2. 修改起征点
3. 修改税率的分割金额
4. 修改税率表
5. 查看个人所得税表
6. 退出
-----
请输入1-6的数字：2
请输入新的起征点： -100
起征点不能为负数，请重新输入。
请输入新的起征点： 200
起征点已更新为200.00
```

3.2 错误的分割金额

支持检测两种错误，一种是不满足分割金额单调上升，一种是金额为负数

```
-----  
请输入1-6的数字:3  
请选择操作:  
1. 修改单个分割金额  
2. 修改全部分割金额  
-----  
请输入1-2的数字:1  
请输入要修改金额(2-7)的位置: 5  
请输入分割金额: 500  
修改失败! 分割金额不满足单调上升条件!  
-----
```

```
-----  
请输入1-6的数字:3  
请选择操作:  
1. 修改单个分割金额  
2. 修改全部分割金额  
-----  
请输入1-2的数字:4  
输入错误, 请重新输入。  
请输入1-2的数字:-23123  
输入错误, 请重新输入。  
请输入1-2的数字:-33  
输入错误, 请重新输入。  
请输入1-2的数字:1  
请输入要修改金额(2-7)的位置: 5  
请输入分割金额: -3543  
输入错误, 请重新输入。  
请输入分割金额: |
```

3.3 错误的税率

支持检测两种错误，一种是不满足税率单调上升（有钱人肯定交的税更多啊），一种是金额为负数或 ≥ 1 的情况

```
-----  
请输入1-6的数字:4  
请选择操作:  
1. 修改单个税率  
2. 修改全部税率  
-----  
请输入1-2的数字:1  
请输入要修改税率(1-7)的位置: 4  
请输入要修改的税率: 1.5  
输入错误, 请重新输入。  
请输入要修改的税率: -0.5  
输入错误, 请重新输入。  
请输入要修改的税率: |
```

```

-----
请输入1-6的数字:4
请选择操作:
1. 修改单个税率
2. 修改全部税率
-----
请输入1-2的数字:1
请输入要修改税率(1-7)的位置: 5
请输入要修改的税率: 0.15
修改失败! 税率不满足单调上升条件!
-----

```

3.4 选择错误

输入的数字不在选择范围之内

```

-----
请选择操作:
1. 计算个人所得税
2. 修改起征点
3. 修改税率的分割金额
4. 修改税率表
5. 查看个人所得税表
6. 退出
-----
请输入1-6的数字:0
输入错误, 请重新输入。
请输入1-6的数字:7
输入错误, 请重新输入。
请输入1-6的数字:2132
输入错误, 请重新输入。
请输入1-6的数字:43.54
输入错误, 请输入1-6的数字。
请输入1-6的数字:

```