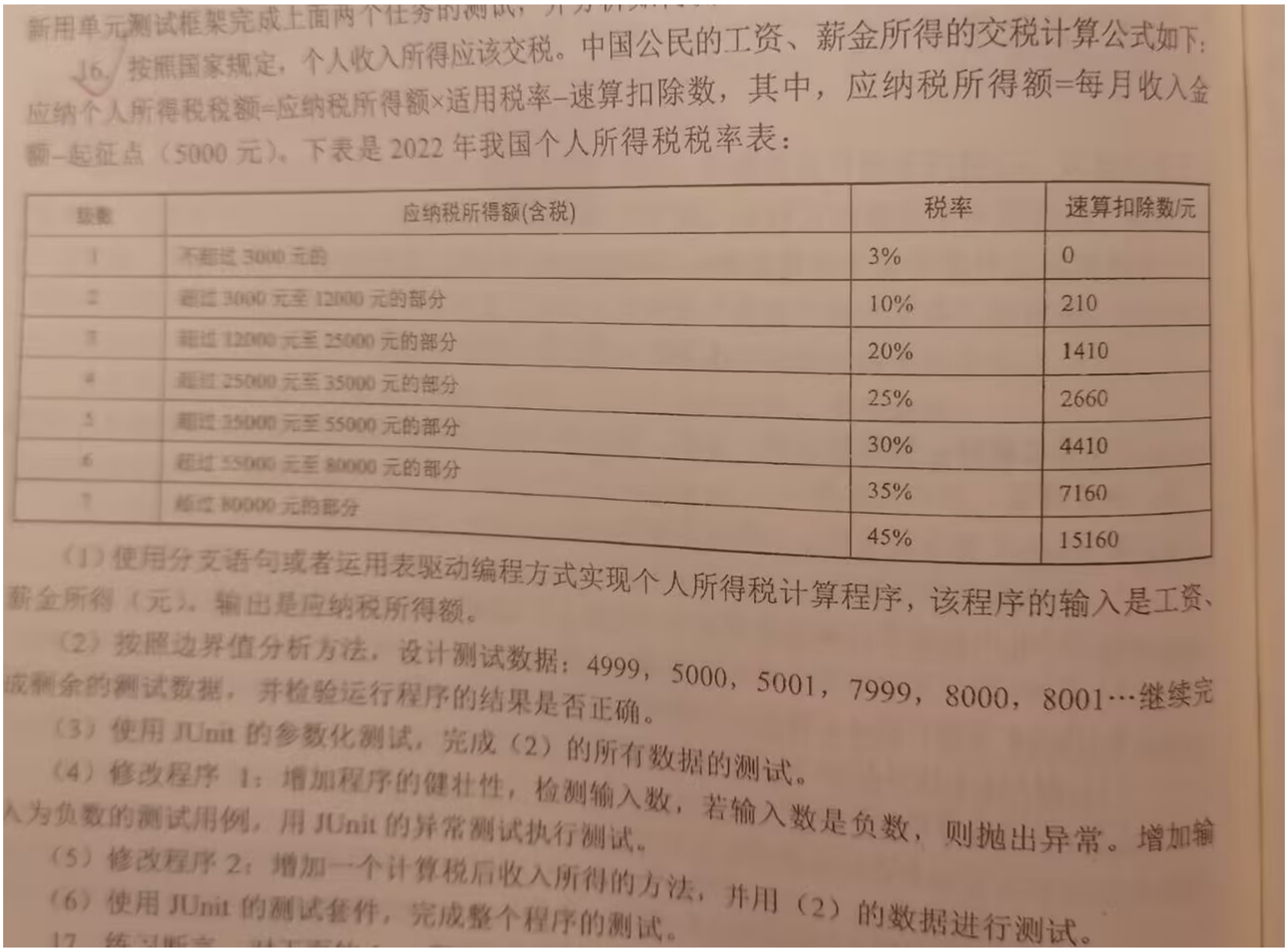
软件构造第4章作业



新用单元测试框架完成上面两个任务的测试，

16. 按照国家规定，个人收入所得应该交税。中国公民的工资、薪金所得的交税计算公式如下：应纳个人所得税税额=应纳税所得额×适用税率–速算扣除数，其中，应纳税所得额=每月收入金额–起征点（5000 元）。下表是 2022 年我国个人所得税税率表：

级数	应纳税所得额(含税)	税率	速算扣除数/元
1	不超过 3000 元的	3%	0
2	超过 3000 元至 12000 元的部分	10%	210
3	超过 12000 元至 25000 元的部分	20%	1410
4	超过 25000 元至 35000 元的部分	25%	2660
5	超过 35000 元至 55000 元的部分	30%	4410
6	超过 55000 元至 80000 元的部分	35%	7160
7	超过 80000 元的部分	45%	15160

（1）使用分支语句或者运用表驱动编程方式实现个人所得税计算程序，该程序的输入是工资、薪金所得（元），输出是应纳税所得额。

（2）按照边界值分析方法，设计测试数据：4999，5000，5001，7999，8000，8001…继续完成剩余的测试数据，并检验运行程序的结果是否正确。

（3）使用 JUnit 的参数化测试，完成（2）的所有数据的测试。

（4）修改程序 1：增加程序的健壮性，检测输入数，若输入数是负数，则抛出异常。增加输入为负数的测试用例，用 JUnit 的异常测试执行测试。

（5）修改程序 2：增加一个计算税后收入所得的方法，并用（2）的数据进行测试。

（6）使用 JUnit 的测试套件，完成整个程序的测试。

1. 写程序

1. 第一题

```java
public class IncomeTaxCalculator {
    public static double calculateTax(double salary) {
        double taxableIncome = salary - 5000;
        if (taxableIncome <= 0) {
            return 0;
        }
        double tax = 0;
        if (taxableIncome <= 3000) {
            tax = taxableIncome * 0.03 - 0;
        } else if (taxableIncome <= 12000) {
            tax = taxableIncome * 0.1 - 210;
        } else if (taxableIncome <= 25000) {
            tax = taxableIncome * 0.2 - 1410;
        } else if (taxableIncome <= 35000) {
            tax = taxableIncome * 0.25 - 2660;
        } else if (taxableIncome <= 55000) {
            tax = taxableIncome * 0.3 - 4410;
        } else if (taxableIncome <= 80000) {
            tax = taxableIncome * 0.35 - 7160;
        } else {
            tax = taxableIncome * 0.45 - 15160;
        }
        return tax;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("请输入工资: ");
        double salary = scanner.nextDouble();
        double tax = calculateTax(salary);
        double taxableIncome = salary - 5000;
        System.out.println("应纳税所得额: " + (taxableIncome > 0 ? taxableIncome : 0) + "元");
        System.out.println("应纳个人所得税税额: " + tax + "元");
        scanner.close();
    }
}
```

2. 测试

工资(元)	**应纳税所得额=工资-5000**	**应纳税额(元)**	**税后收入=工资-税额(元)**
4999	免税	免税	4999.00
5000	免税	免税	5000.00
5001	1	1*0.03－0=0.03	5000.97
7999	2999	2999*0.03-0=89.97	7909.03
8000	3000	3000*0.03-0=90.00	7910.00
8001	3001	3001*0.10-210=90.10	7910.90
16999	11999	11999*0.10-210=989.90	16009.10
17000	12000	12000*0.10-210=990.00	16010.00
17001	12001	12001*0.20-1410=990.20	16010.80

30000	25000	25000*0.20-1410=3590.00	26410.00
30001	25001	25001*0.25-2660=3590.25	26410.75
40000	35000	35000*0.25-2660=6090.00	33910.00
40001	35001	35001*0.30-4410=6090.30	33910.70
60000	55000	55000*0.30-4410=12090.00	47910.00
60001	55001	55001*0.35-7160=12090.35	47910.65
85000	80000	80000*0.35-7160=20840.00	64160.00
85001	80001	80001*0.45-15160=20840.45	64160.55

3. JUnit

代码块

```java
@RunWith(Parameterized.class)
public class IncomeTaxCalculatorParameterizedTest {
    private double salary;
    private double expectedTax;
    private double expectedAfterTax;

    public IncomeTaxCalculatorParameterizedTest(double salary, double expectedTax, double expectedAfterTax) {
        this.salary = salary;
        this.expectedTax = expectedTax;
        this.expectedAfterTax = expectedAfterTax;
    }

    @Parameterized.Parameters(name = "{index}: salary={0}, expectedTax={1}, expectedAfterTax={2}")
    public static Collection<Object[]> data() {
        return Arrays.asList(new Object[][]{
                {4999, 0.00, 4999.00},
                {5000, 0.00, 5000.00},
                {5001, 0.03, 5000.97},
                {7999, 89.97, 7909.03},
                {8000, 90.00, 7910.00},
                {8001, 90.10, 7910.90},
                {16999, 989.90, 16009.10},
                {17000, 990.00, 16010.00},
                {17001, 990.20, 16010.80},
                {30000, 3590.00, 26410.00},
                {30001, 3590.25, 26410.75},
                {40000, 6090.00, 33910.00},
                {40001, 6090.30, 33910.70},
                {60000, 12090.00, 47910.00},
                {60001, 12090.35, 47910.65},
                {85000, 20840.00, 64160.00},
                {85001, 20840.45, 64160.55}
        });
    }

    @Test
    public void testCalculateTax() {
        double actualTax = IncomeTaxCalculator.calculateTax(salary);
        assertEquals("工资=" + salary + "的税额不正确", expectedTax, actualTax, 0.01);
    }

    @Test
    public void testAfterTaxIncome() {
        double actualTax = IncomeTaxCalculator.calculateTax(salary);
        double actualAfterTax = salary - actualTax;
        assertEquals("工资=" + salary + "的税后收入不正确", expectedAfterTax, actualAfterTax, 0.01);
    }
}

```

4. 修改程序1

增加负数校验，抛出异常

```
public class IncomeTaxCalculator {
    public static double calculateTax(double salary) {
        // 新增：校验输入为负数时抛出异常
        if (salary < 0) {
            throw new IllegalArgumentException("工资不能为负数");
        }

        double taxableIncome = salary - 5000;
        if (taxableIncome <= 0) {
            return 0;
        }

        ...
    }
}
```

异常测试类

```java
public class IncomeTaxCalculatorExceptionTest {
    @Test(expected = IllegalArgumentException.class)
    public void testNegativeSalary() {
        IncomeTaxCalculator.calculateTax(-1000);
    }
}
```

5. 修改程序2

新增计算税后收入的方法

```java
public class IncomeTaxCalculator {
    //原方法不变

    //新增
    public static double calculateAfterTaxIncome(double salary) {
        double tax = calculateTax(salary);
        return salary - tax;
    }

    // main方法补充输出税后收入:
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("请输入工资: ");
        double salary = scanner.nextDouble();
        try {
            double tax = calculateTax(salary);
            double taxableIncome = Math.max(salary - 5000, 0);
            double afterTax = calculateAfterTaxIncome(salary);
            System.out.println("应纳税所得额: " + taxableIncome + "元");
            System.out.println("应纳个人所得税税额: " + tax + "元");
            System.out.println("税后收入: " + afterTax + "元");
        } catch (IllegalArgumentException e) {
            System.out.println("错误: " + e.getMessage());
        }
        scanner.close();
    }
}
```

测试

```java
@Test
public void testAfterTaxIncome() {
    double actualAfterTax =
IncomeTaxCalculator.calculateAfterTaxIncome(salary);
    assertEquals("工资=" + salary + "的税后收入不正确", expectedAfterTax,
actualAfterTax, 0.01);
}
```

6. 测试

代码块

```java
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({
    IncomeTaxCalculatorParameterizedTest.class,
    IncomeTaxCalculatorExceptionTest.class
})
public class IncomeTaxCalculatorTestSuite {
    ...
}
```