《Java 面向对象程序设计 SSD3》

实验报告

项目名称 实验 2《一维数组》

专业班级 软件工程 2005 班

学　　号 8209200504

姓　　名 李均浩

实验成绩：

批阅教师：邝砾

2021 年 4 月 27 日

中南大学计算机学院实验报告

课程名称 Java 面向对象程序设计 SSD3

实验项目名称 实验 2《一维数组》

学生姓名 李均浩 **专业班级** 软件工程 2005 班 **学号** 8209200504

实验成绩 _____ **日期** 2021 年 4 月 27 日

实验学时: 2
每组人数: 1
实验类型: 1 (1: 基础性 2: 综合性 3: 设计性 4: 研究性)
实验要求: 1 (1: 必修 2: 选修 3: 其它)
实验类别: 2 (1: 基础 2: 专业基础 3: 专业 4: 其它)

一、实验目的

学习一维数组的用法、方法的定义和调用。

二、实验内容

1. (P236, 7.3) 编写程序，读取 1-100 之间的整数，然后计算每个数出现的次数。假定输入是以 0 结束的。以下是程序运行示例:

```
输入 1-100 之间的整数: 2  5  5  4  3  23  2  0 [回车]
2 出现 2 次
3 出现 1 次
4 出现 1 次
5 出现 2 次
23 出现 1 次
```

2. （P237，7.10）编写一个方法，求出数组中最小元素的下标。如果这样的元素个数大于 1，则返回最小下标。使用下面的方法头：

public static int indexofSmallestElement(double[] array)

编写测试程序，提示用户输入 10 个数字，调用这个方法，返回最小元素的下标，然后显示这个下标值。

3. （P236，7.5）编写程序， 读入 10 个数并显示互不相同的数（即一个数出现多次，但仅显示一次）。提示，读入一个数，如果它是一个新数，则将它存储在数组中，如果该数已经在数组中，则忽略它。输入之后，数组包含的都是不同的数。以下是运行示例：

输入 10 个整数：1　2　3　2　1　6　3　4　5　2

互不相同的数为：1　2　3　6　4　5

4.（P240，7.27）如果两个数组 list1 和 list2 内容相同，那么就说它们是相同的。使用下面的方法头编写一个方法，如果 list1 和 list2 是相同的，该方法就返回 true：

public static boolean equal(int[ ] list1, int[ ] list2)

编写一个测试程序，提示用户输入两个整数数列，然后显示它们两个是否相同。以下是运行示例。注意输入的第一个数字表示数列中元素的个数。

提示：可考虑使用 230-231 页 Arrays 类提供的方法进行组合调用

输入 list1：　5　2　5　6　6　1

输入 list2： 5 5 2 6 1 6 这两个数列是相同的

输入 list1： 5 5 5 6 6 1 输入 list2： 5 2 5 6 1 6 这两个数列是不同的

5. (**附加题 6.31 信用卡号的合法性，可选做**)信用卡号遵循下面的模式。一个信用卡号必须是 13-16 位的整数。它的开头必须是：
4，指 visa 卡
5，指 master 卡
37，指 American Express 卡
6，指 Discovery 卡

在 1954 年，IBM 的 Hans Luhn 提出一种算法，该算法可以验证信用卡号的有效性。这个算法在确定输入的卡号是否正确，或者这张信用卡是否能被正确扫描是非常有用的。该方法通常被称为 Luhn 检测或 Mod10 检测，描述如下（假设卡号是 4388576018402626）

（1） 从右至左对偶数位上的数字翻倍。如果数字翻倍后是一个两位数，那么就将这两位加在一起得到一位数。
2*2 =4
2*2=4
4*2=8
1*2=2
6*2=12（1+2=3）
5*2=10（1+0=1）
8*2=16（1+6=7）
4*2=8

（2） 将第一步得到的所有一位数相加。
4+4+8+2+3+1+7+8=37

（3） 将卡号里从右往左奇数位上所有数字相加。
6+6+0+8+0+7+8+3=38

（4） 将第二步和第三步得到的结果相加。

37+38=75

（5） 如果第四步得到的结果能被 10 整除，则卡号是合法的，否则是不合法的。

75%10 !=0

编写程序，提示用户输入一个 long 型整数的信用卡号码，显示这个数字是合法还是非法的。使用下面的方法设计程序：

```
/*Return true if the card number is valid*/
public static boolean isValid(long number)

/*Get the result from step 2*/
public static int sumOfDoubleEvenPlace(long number)

/*Return this number if it is a single digit, otherwise return the sum of the two digits*/
public static int getDigit(int number)

/*Return sum of odd place digits in number*/
public static int sumOfOddPlace(long number)

/*Return true if the digit d is a prefix for number*/
public static boolean prefixMatched(long number, int d)

/*Return the number of digits in d*/
public static int getSize(long d)

/*Return the first k number of digits from number. If the number of digits in number is less than k, return number*/
public static long getPrefix(long number, int k)
```

三、实验要求：

要求每个学生独立完成实验任务。

四、实验报告

1. **实验结果与分析**

第一题：

测试点(a)：0

(1)测试结果：

```
0
没有执行输入操作，未进行统计

进程已结束，退出代码为 0
```

(2)结果分析

用户直接输入了终止字符 0，按照预设提示用户未进行输入，符合预期。

测试点(b)：1 0

(1)测试结果：

```
1 0
1 出现的次数为：1

进程已结束，退出代码为 0
```

(2)结果分析

输入了 1 次数字 1，符合预期。

测试点(c)：1 2 2 3 4 88 8 8 2 2 0

(1)测试结果：

```
1 2 2 3 4 88 8 8 2 2 0
1 出现的次数为：1
2 出现的次数为：4
3 出现的次数为：1
4 出现的次数为：1
8 出现的次数为：2
88 出现的次数为：1

进程已结束，退出代码为 0
```

(2)结果分析

输入 1 次数字 1，4 次数字 2，1 次数字 3，1 次数字 4，2 次数字 8，1 次数字 88，符合预期。

测试点(d)：21 12 12 21 12 21 0
(1)测试结果：

```
21 12 12 21 12 21 0
12 出现的次数为：3
21 出现的次数为：3

进程已结束，退出代码为 0
```

(2)结果分析
输入了 3 次数字 12，3 次数字 21，符合预期。

测试点(e)：6 66 1 11 2 5 8 55 0
(1)测试结果：

```
6 66 1 11 2 5 8 55 0
1 出现的次数为：1
2 出现的次数为：1
5 出现的次数为：1
6 出现的次数为：1
8 出现的次数为：1
11 出现的次数为：1
55 出现的次数为：1
66 出现的次数为：1

进程已结束，退出代码为 0
```

(2)结果分析
输入了 3 次数字 12，3 次数字 21，符合预期。

测试点(f)：1 2 2 1 4512 0
(1)测试结果：

```
1 2 2 1 4512 0
请输入1-100之间的数字！超出范围的数字未统计！
1 出现的次数为：2
2 出现的次数为：2

进程已结束，退出代码为 0
```

(2)结果分析
输入了 2 次数字 1,2 次数字 2,最后输入的 4512 不在 1-100 的范围内，输出用户提示，不进行统计，符合预期。

测试点(g)：4 5 963 5 4 0

(1)测试结果：

```
4 5 963 5 4 0
请输入1-100之间的数字！超出范围的数字未统计！
4 出现的次数为：2
5 出现的次数为：2

进程已结束，退出代码为 0
```

(2)结果分析

输入了 2 次数字 4，2 次数字 5，中间输入的 963 不在 1-100 的范围内，输出用户提示，不进行统计，符合预期。

第二题：

测试点(a)：0 1 2 4 4 6 6 2 1 -4

(1)测试结果：

```
请输入十个数，以下将获取最小元素的最小下标：
0 1 2 4 4 6 6 2 1 -4
最小元素的最小下标为：9
进程已结束，退出代码为 0
```

(2)结果分析

最小元素是-9，此相同元素最小索引于 9，输出符合预期。

测试点(b)：-777 -999 -999 -999 5 888 777 4545 -999 999

(1)测试结果：

```
请输入十个数，以下将获取最小元素的最小下标：
-777 -999 -999 -999 5 888 777 4545 -999 999
最小元素的最小下标为：1
进程已结束，退出代码为 0
```

(2)结果分析

最小元素是-999，此相同元素最小索引于 1，输出符合预期。

测试点(c)：54 45 45 5 0 0 5 45 0 45

(1)测试结果：

```
54 45 45 5 0 0 5 45 0 45
最小元素的最小下标为：4
进程已结束，退出代码为 0
```

(2)结果分析

最小元素是 0，此相同元素最小索引于 4，输出符合预期。

测试点(d):

(1)测试结果:

```
请输入十个数，以下将获取最小元素的最小下标：
1 1 1 1 1 1 1 1 1 1
最小元素的最小下标为：0
进程已结束，退出代码为 0
```

(2)结果分析

最小元素是1，此相同元素最小索引于0，输出符合预期。

测试点(e): 7 -7 7 7 -2 -7.2 7 1.2 -7.1 -7.0

(1)测试结果:

```
请输入十个数，以下将获取最小元素的最小下标：
7 -7 7 7 -2 -7.2 7 1.2 -7.1 -7.0
最小元素的最小下标为：5
进程已结束，退出代码为 0
```

(2)结果分析

最小元素是-7.2，此相同元素最小索引于5，输出符合预期。

测试点(f): -1.222 1 5.5 -1.222 4 1 1 1 -1.22 -1.222

(1)测试结果:

```
请输入十个数，以下将获取最小元素的最小下标：
-1.222 1 5.5 -1.222 4 1 1 1 -1.22 -1.222
最小元素的最小下标为：0
进程已结束，退出代码为 0
```

(2)结果分析

最小元素是-1.222，此相同元素最小索引于0，输出符合预期。

测试点(g): -1 -1.1 -1.11 -1.1111 -1.1111 -1.111 -1.111 -1.11 -1.1 -1.0

(1)测试结果:

```
请输入十个数，以下将获取最小元素的最小下标：
-1 -1.1 -1.11 -1.1111 -1.1111 -1.111 -1.111 -1.11 -1.1 -1.0
最小元素的最小下标为：3
进程已结束，退出代码为 0
```

(2)结果分析

最小元素是-1.1111，此相同元素最小索引于3，输出符合预期。

第三题：

测试点(a)：7 8 7 8 4 5 4 4 4 2

(1)测试结果：

```
请输入10个数，以下将不重复地显示输入的数：
7 8 7 8 4 5 4 4 4 2
结果如下：
7    8    4    5    2
进程已结束，退出代码为 0
```

(2)结果分析

重复的数有7，8，4，输出结果中未重复，符合预期。

测试点(b)：

(1)测试结果：1 1 1 1 1 1 1 1 1 1

```
请输入10个数，以下将不重复地显示输入的数：
1 1 1 1 1 1 1 1 1 1
结果如下：
1
进程已结束，退出代码为 0
```

(2)结果分析

重复的数只有1，符合预期。

测试点(c)：

(1)测试结果：11 1 11 11 1 111 0 0 0 1

```
请输入10个数，以下将不重复地显示输入的数：
11 1 11 11 1 111 0 0 0 1
结果如下：
11   1    111 0
进程已结束，退出代码为 0
```

(2)结果分析

重复的数有1，11,111，0，输出结果中未重复，符合预期。

测试点(d)：

(1)测试结果：-8 8 7 8 4 5 6 1 -4 10

```
请输入10个数，以下将不重复地显示输入的数：
-8 8 7 8 4 5 6 1 -4 10
结果如下：
-8   8    7    4    5    6    1    -4   10
进程已结束，退出代码为 0
```

(2)结果分析

重复的数有8，结果中未重复输出，符合预期。

第四题：

测试点(a)：4 2 3 1 2　　4 2 2 1 3

(1)测试结果：

```
请输入第一个数组：
4 2 3 1 2
请输入第二个数组：
4 2 2 1 3
两数组相等

进程已结束，退出代码为 0
```

(2)结果分析

两数组内含元素相同，仅排序不同，符合预期。

测试点(b)：1 1　　1 1

(1)测试结果：

```
请输入第一个数组：
1 1
请输入第二个数组：
1 1
两数组相等

进程已结束，退出代码为 0
```

(2)结果分析

两数组仅含同一个元素，符合预期。

测试点(c)：7 4 5 6 1 44 8 9　　7 6 1 44 8 9 5 4

(1)测试结果：

```
请输入第一个数组：
7 4 5 6 1 44 8 9
请输入第二个数组：
7 6 1 44 8 9 5 4
两数组相等

进程已结束，退出代码为 0
```

(2)结果分析

两数组内含元素相同，仅排序不同，符合预期。

测试点(d)：6 4 5 1 2 3 0　　7 4 5 1 3 2 0 7

(1)测试结果：

```
请输入第一个数组：
6 4 5 1 2 3 0
请输入第二个数组：
7 4 5 1 3 2 0 7
两数组不相等

进程已结束，退出代码为 0
```

(2)结果分析

两数组元素个数不同，符合预期。

测试点(e)：5 4 1 2 3 8　　5 4 2 1 3 9

(1)测试结果：

```
请输入第一个数组：
5 4 1 2 3 8
请输入第二个数组：
5 4 2 1 3 9
两数组不相等

进程已结束，退出代码为 0
```

(2)结果分析

两数组元素个数相等，内含元素不完全相等，符合预期。

测试点(f)：6 4 8 7 9 5 4　　6 1 2 4 5 7 1

(1)测试结果：

```
请输入第一个数组：
4 1 1 1 1
请输入第二个数组：
4 2 2 2 2
两数组不相等

进程已结束，退出代码为 0
```

(2)结果分析

两数组元素个数相等，内含元素不完全相等，符合预期。

测试点(g)：41111　　42222

(1)测试结果：

```
请输入第一个数组：
6 4 8 7 9 5 4
请输入第二个数组：
6 1 2 4 5 7 1
两数组不相等

进程已结束，退出代码为 0
```

(2)结果分析

两数组元素个数相等，内含元素完全不相等，符合预期。。

第四题：

测试点(a)：4388576018402621

(1)测试结果：

```
================================================================================
Tips:此程序用于检验银行卡卡号的合法性，支持的种类有：Mastercard,Visa,American Express,Discovery
================================================================================
请输入您的银行卡号：4388576018402621
********************开始验证********************
Length = 16
************通过长度验证，进入前缀验证**************
prefix = 43
************通过前缀验证，进入Mod10验证************
70 % 10 = 0
您的卡号*符合*我们的验证规则！

进程已结束，退出代码为 0
```

(2)结果分析

卡号长度为 13 符合 13-16 的要求,方法内取得前 2 位为 43,比对后符合前缀 4 的要求,Mod10 检测结果为 0 符合规则,卡号合法。

测试点(b)：4388576018402626

(1)测试结果：

```
请输入您的银行卡号：4388576018402626
********************开始验证********************
Length = 16
************通过长度验证，进入前缀验证**************
prefix = 43
************通过前缀验证，进入Mod10验证************
75 % 10 = 5 ≠ 0
您的卡号*不符合*我们的验证规则！

进程已结束，退出代码为 0
```

(2)结果分析

卡号长度为 13 符合 13-16 的要求,方法内取得前 2 位为 43,比对后符合前缀 4 的要求，Mod10 检测结果为 5 不符合规则，卡号不合法。

测试点(c)：4848576018402626

(1)测试结果:

```
请输入您的银行卡号：4848576018402626
**********开始验证**********
Length = 16
**********通过长度验证，进入前缀验证**********
prefix = 48
**********通过前缀验证，进入Mod10验证**********
81 % 10 = 1 ≠ 0
您的卡号*不符合*我们的验证规则!
```

(2)结果分析

卡号长度为 16 符合 13-16 的要求,方法内取得前 2 位为 48,比对后符合前缀 4 的要求，Mod10 检测结果为 1 不符合规则，卡号不合法。

测试点(d)：375156478945124

(1)测试结果:

```
请输入您的银行卡号：375156478945124
**********开始验证**********
Length = 15
**********通过长度验证，进入前缀验证**********
prefix = 37
**********通过前缀验证，进入Mod10验证**********
63 % 10 = 3 ≠ 0
您的卡号*不符合*我们的验证规则!

进程已结束，退出代码为 0
```

(2)结果分析

卡号长度为 15 符合 13-16 的要求,方法内取得前 2 位为 37,比对后符合前缀 37 的要求,Mod10 检测结果为 3 不符合规则，卡号不合法。

测试点(e)：3645662124542121

(1)测试结果：

```
请输入您的银行卡号：3645662124542121
**********开始验证**********
Length = 16
**********通过长度验证，进入前缀验证**********
prefix = 36
您的卡号*不符合*我们的验证规则！

进程已结束，退出代码为 0
```

(2)结果分析

卡号长度为 16 符合 13-16 的要求,方法内取得前 2 位为 36，比对后未找到符合前缀，不再进行 Mod10 检测，卡号不合法。

测试点(f)：56512354612365485 45

(1)测试结果：

```
请输入您的银行卡号：5651235461236548545
**********开始验证**********
Length = 19
您的卡号*不符合*我们的验证规则！

进程已结束，退出代码为 0
```

(2)结果分析

卡号长度为 19，不符合 13-16 的要求，不再进行卡号前缀检测，也不再进行 Mod10 检测，卡号不合法。

2. 心得体会

第一题：

本题主要编写了一段冒泡排序代码，将输入的数组整体进行排序，随后从头开始遍历数组，若前后两元素相等，该元素的出现次数自增。在这题的练习中，巩固了数组的输入输出以及数组排序的知识点。

第二题：

程序采取了从数组末尾向前遍历的方式，在寻找最小下标这一条件下似乎要更加顺应思维。这一题主要的点就在于数组的遍历过程中对最值的相互比对和重新替换赋值。这一题巩固了数组的输入以及数组的遍历，求最值，循环语

句这些知识点。

第三题：

本题主要的想法是：将每次输入的数先暂存至一个变量中，随后对当前的数组进行遍历，若数组中存在相同的变量则直接进行下一次迭代，否则就将暂存变量赋值到数组的下一项中。关键点在于对循环、逻辑条件的设置和数组的遍历。本题巩固了数组的输入、遍历、循环这些知识点。

第四题：

本题主要的想法是：编写一段冒泡排序程序，将输入的两个数组升序排列，随后从头开始遍历两个数组，并依次比对同一位置上的元素是否相同，一旦在相同索引位置上遇到两个元素不相同的情况，立即返回 false。关键点在于如何对元素相同但序列不同的两个数组进行比较，对数组进行排序再依次比对就比较方便。本题巩固了数组排序，数组输入，数组遍历，方法的声明、创建、调用、参数的传递这些知识点。

第五题：

本题编写了 6 个方法，分别实现获取卡号前缀、获取卡号位数、获取卡号偶数位经过特定处理后的结果、获取卡号技术位经过特定处理后的结果、获取单位数字、总判断卡号是否合法。本题关键点在于将各个功能分隔开，通过分模块的方式来实现程序，在于对方法之间的调用、值的传递以及返回。本题初步利用了自顶向下、逐步求精的思想，体会到了分模块设计的好处。

【附源程序】

```
import java.lang.reflect.Array;
import java.util.Arrays;
import java.util.Scanner;

public class Exp_2_1 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);


System.out.println("================================================
");
        System.out.println("请输入数字，以'0'结束输入。以下将统计各数字的出
现次数：");

System.out.println("================================================
");
        byte[] list = new byte[1000];
        int listSize = 0;

        //将数字输入到数组
        for (int i = 0; ; i++) {
            int temp;
            temp = input.nextInt();
            if (i == 0 && temp == 0) {
                System.out.println("没有执行输入操作，未进行统计");
                return;
            } else if (temp == 0)
                break;
            if (temp >= 0 && temp <= 100) {
                list[i] = (byte) temp;
                listSize++;
            }
            else{
                System.out.println("请输入 1-100 之间的数字！超出范围的数字
未统计！");
                i--;
            }
        }

        //对数组进行排序
        boolean changed = true;
        int temp;
        do {
            changed = false;
```

```
            for (int j = 0; j < listSize - 1; j++)
                if (list[j] > list[j + 1]) {
                    temp = list[j];
                    list[j] = list[j + 1];
                    list[j + 1] = (byte) temp;
                    changed = true;
                }
        } while (changed);

        int i = 0;
        int appearTimes = 1;
        while (i != listSize) {
            if (list[i] == list[i + 1]) {
                appearTimes++;
            } else {
                System.out.print(list[i] + " 出现的次数为：" + appearTimes
+ "\n");
                appearTimes = 1;
            }
            i++;
        }
    }
}
```

源代码 1 Exp_2_1.java

```
import java.util.Scanner;

public class Exp_2_2 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        double[] list = new double[1000];
        System.out.println("请输入十个数，以下将获取最小元素的最小下标：");
        //输入至数组 list
        for (int i = 0; i < 10; i++) {
            list[i] = input.nextDouble();
        }

        //调用方法
        int minIndex = indexOfSmallestElement(list);
        //输出结果
        System.out.print("最小元素的最小下标为：" + minIndex);
    }
```

```
    public static int indexOfSmallestElement(double[] array) {
        double minElement = array[9];
        int minIndex = 9;
        //从数组的末尾向前遍历
        for (int i = 8; i >= 0; i--) {
            if (array[i] <= minElement && i < minIndex) {
                minElement = array[i];
                minIndex = i;
            }
        }
        return minIndex;
    }
}
```

源代码 2 Exp_2_2.java

```
import java.util.Scanner;

public class Exp_2_3 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        int[] list = new int[1000];
        int temp;
        int index = 0;

        System.out.println("请输入 10 个数，以下将不重复地显示输入的数：");
        //输入至数组
        for (int i = 0; i < 10; i++) {
            temp = input.nextInt();
            if (index == 0) {
                list[0] = temp;
                index++;
                continue;
            }
            boolean isSame = false;
            for (int j = 0; j < index; j++) {
                if (list[j] == temp) {
                    isSame = true;
                    break;
                }
            }
            if (!isSame) {
                list[index] = temp;
```

```
                index++;
            }
        }
        System.out.println("结果如下：");
        for (int i = 0; i < index; i++) {
            System.out.print(list[i] + "\t");
        }
    }
}
```

源代码 3 Exp_2_3.java

```
import java.util.Scanner;

public class Exp_2_4 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        //建立两个数组
        int[] list_1 = new int[1000];
        int[] list_2 = new int[1000];

        //输入第一个数组
        System.out.println("请输入第一个数组：");
        int listSize_1;
        listSize_1 = input.nextInt();
        for (int i = 0; i < listSize_1; i++) {
            list_1[i] = input.nextInt();
        }
        bubbleSort(list_1, listSize_1);

        //输入第二个数组
        System.out.println("请输入第二个数组：");
        int listSize_2;
        listSize_2 = input.nextInt();
        for (int i = 0; i < listSize_2; i++) {
            list_2[i] = input.nextInt();
        }
        bubbleSort(list_2, listSize_2);

        boolean isSame = equal(list_1, list_2, listSize_1, listSize_2);
        if (isSame)
            System.out.println("两数组相等");
        else
```

```
            System.out.println("两数组不相等");
        }

        public static boolean equal(int[] list_1, int[] list_2, int
listSize_1, int listSize_2) {
            if (listSize_1 != listSize_2)
                return false;
            int i = 0;
            while (i <= listSize_1) {
                if (list_1[i] != list_2[i])
                    return false;
                i++;
            }
            return true;
        }

        public static void bubbleSort(int[] list, int listSize) {
            //sort
            boolean changed = true;
            int temp;
            do {
                changed = false;
                for (int j = 0; j < listSize - 1; j++)
                    if (list[j] > list[j + 1]) {
                        temp = list[j];
                        list[j] = list[j + 1];
                        list[j + 1] = temp;
                        changed = true;
                    }
            } while (changed);
        }
}
```

源代码 4 Exp_2_4.java

```
import java.util.Scanner;

public class Exp_2_5 {
    public static void main(String[] args) {

System.out.println("==============================================
==========================================");
        System.out.println("Tips:此程序用于检验银行卡卡号的合法性，支持的种
类有：Mastercard,Visa,American Express,Discovery");
```

```
System.out.println("==========================================
========================================");

        Scanner input = new Scanner(System.in);

        long cardNumber;
        System.out.print("请输入您的银行卡号：");
        cardNumber = input.nextLong();

        if (isValid(cardNumber))
            System.out.println("您的卡号*符合*我们的验证规则！");
        else
            System.out.println("您的卡号*不符合*我们的验证规则！");
    }

    /*Return true if the card number is valid*/
    public static boolean isValid(long number) {
        System.out.println("********************开始验证
********************");
        System.out.println("Length = " + getSize(number));
        if (!(getSize(number) >= 13 && getSize(number) <= 16))
            return false;

        System.out.println("************通过长度验证，进入前缀验证
*************");

        long prefix = getPrefix(number, 2);
        System.out.println("prefix = " + prefix);
        if (prefix / 10 != 4 && prefix / 10 != 5 && prefix / 10 != 6
&& prefix != 37)
            return false;

        System.out.println("************通过前缀验证，进入 Mod10 验证
************");
        int evenSum = sumOfDoubleEvenPlace(number);
        int oddSum = sumOfOddPlace(number);
        int sum = evenSum + oddSum;

        if (sum % 10 == 0) {
            System.out.println(sum + " % 10 = 0");
            return true;
        } else {
```

```
            System.out.println(sum + " % 10 = " + sum % 10 + " ≠ 0");
            return false;
        }
    }

    /*Get the result from step 2*/
    public static int sumOfDoubleEvenPlace(long number) {
        String cardNumber = String.valueOf(number);

        int evenSum = 0;
        for (int i = cardNumber.length() - 2; i >= 0; i -= 2) {
            int temp = ((int) (cardNumber.charAt(i)) - '0') * 2;
            evenSum += getDigit(temp);
        }
        return evenSum;
    }

    /*Return sum of odd place digits in number*/
    public static int sumOfOddPlace(long number) {
        String cardNumber = String.valueOf(number);

        int oddSum = 0;
        for (int i = cardNumber.length() - 1; i >= 0; i -= 2) {
            int temp = ((int) (cardNumber.charAt(i)) - '0');
            oddSum += temp;
        }
        return oddSum;
    }

    /*Return this number if it is a single digit, otherwise return the
sum of the two digits*/
    public static int getDigit(int number) {
        if (number < 10)
            return number;
        else {
            return number / 10 + (number - number / 10 * 10);
        }
    }

    /*Return the number of digits in d*/
    public static int getSize(long d) {
        String cardNumber = String.valueOf(d);
        return cardNumber.length();
    }
```

```
    /*Return the first k number of digits from number. If the number
of digits in number is less than k, return number*/
    public static long getPrefix(long number, int k) {
        String s = String.valueOf(number);
        return Integer.parseInt((s.substring(0, k)));
    }
}
```

源代码 5 Exp_2_5.java