目录

预备知识：通过键盘输入和文件输入

0.1 题目

请自行敲代码练习 Scanner 类接收键盘输入数据的模式。并通过 Java API 的查找发现 Scanner 与输入有关的其他函数，并加以验证练习。

0.2 结果展示

预备题较简单，完全为了熟悉 Scanner 的语法服务，代码附于附录。

```
Hi! yuan, the sum of 12 and 24.56 is 36.56
Enter an integer: 203
Enter a floating point number: 8.27
Enter your name: YangHao
Hi! YangHao, the sum of 203 and 8.27 is 211.27
```

图 1: Problem0

实验 1. UPC 码

1.1 题目

UPC 码由 12 个数字构成，其中最右侧的数字是校验位，我们用 d1 表示，那么 UPC 码经过下面的公式计算之后必须是 10 的倍数

$$(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11}) + 3\,(d_2 + d_4 + d_6 + d_8 + d_{10} + d_{12})\,.$$

比如，0-48500-00102 的校验位就是 8，因为经过上面的公式计算之后，只有数字 8 才可以得到 10 的倍数值 50

$$(8 + 0 + 0 + 0 + 5 + 4) + 3(2 + 1 + 0 + 0 + 8 + 0) = 50.$$

编写一个程序，通过命令行参数方式**输入 1 个 11 位的数字，输出一个完整的 12 位**的 UPC 码。注意：

1. 11 位数字可以包含前导零；
2. 11 位数字的大小已经使得数字的表示范围超过了 `int` 类型能够表示的大小了；
3. 如果不要求用户严格的输入 11 位的数字，那么就需要在真正执行计算之前对用户输入的参数合法性进行考虑，请问你能列举出有多少种不合法的输入出现呢？如果程序的健壮性足够好，那就必须能处理这些意外的输入。

1.2 题目分析

本题目共分为三部分：

1. **数据的输入**. 本题中数据是通过命令行输入的 11 位数字，也就是以 String 的形式存储在`args[0]`中，Java 中 String 类的丰富 method 可以帮助我们把 String 当作数组来处理。
2. **异常数据的检测**. 本题中可能出现三种异常数据：输入参数个数错误，长度不为 11，字符串有非数字符，校验位结果错误。
   - 输入参数个数错误可以通过`args.length`判断
   - 长度可以通过`args[0].length()`判断
   - 字符串的非字符数组在扫描数组过程中通过`Character.isDigit()`判断

- 校验位结果错误可以在计算完成后判断

3. **算法设计**. 观察 UPC 码的计算方法不难发现，UPC 码校验位的计算即用 50 减去三倍的偶位数和一倍的奇位数。将本题输入数据看作数组s，则s[0] $= d_2$，所以数组索引的奇偶和 UPC 码的奇偶相同。

4. **代码说明**. 依据前三条的思路构建代码如下

Listing 1: **Problem1.java**

```
1  public class Problem1 {
2      public static void main(String[] args) {
3          System.out.print("Input:");
4          for (String arg : args) System.out.print(" " + arg);
5          System.out.println();
6          if(args.length != 1){
7              System.out.println("ERROR! Please input 1 argument instead
                   of " + args.length + ".");
8              return;
9          }
10         if (args[0].length() != 11) {
11             System.out.println("ERROR! Please input a 11-bit number
                  instead of " + args[0].length() + " bits.");
12             return;
13         }
14         int result = 50;
15         char tmp;
16         for (int i = 0; i < 11; i++) {
17             if (Character.isDigit(tmp = args[0].charAt(i))) {
18                 if (i % 2 == 0) result -= 3 * Integer.parseInt(String.
                       valueOf(tmp));
19                 else result -= Integer.parseInt(String.valueOf(tmp));
20             } else {
21                 System.out.println("ERROR! The " + (i + 1) + "th input
                       bit is '" + tmp + "', it's not a digit.");
22                 return;
23             }
24         }
25         if (result > 9 || result < 0) {
```

```
26              System.out.println("ERROR! The UPC code is " + result + ",
                    it's invalid.");
27              return;
28          }
29          System.out.print(result + args[0]);
30      }
31  }
```

1.3 数据设计

本体主要目的为计算和检验异常数据，所以提供两个成功数据和几个对应于异常数据原因的数据得到验证即可。为了便于对比输入输出，将每一次的输出打印出来具体数据如下

Input	Output	Input	Output
04850000102	804850000102	04850000103	504850000103
04850000100	d_0 error(14)	04850000109	d_0 error(-13)
1	length error	012345678900	length error
0123456a789	not num	01?23456789	not num
04850000102 1	argu num error		

1.4 结果展示

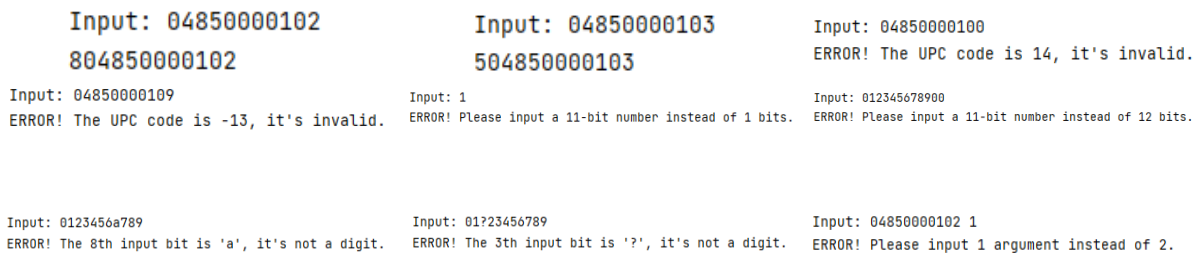


图 2: Output of Problem1

1.6 总结与收获

本题比较简单，总结收获如下：

- 对字符串的处理中可以使用丰富的 Java 库函数，让人大开眼界；
- 对程序健壮性的测试需要考虑很多可能，有多种违规输入，同时为了提高实用性也应该为不同的违规输入定制错误提醒；
- 由于违规输入较多，测试时如果采用 Junit 测试效果更好，可惜我被 Junit5 的配置折腾得快崩溃了，希望以后能再找机会学习。

实验 2. 数字转英语

2.1 题目

编写一个程序，从**命令行参数**中读取一个范围为 [-999999999, 999999999] 的整数，输出为这个整数转换成英语表示的等价形式。下面列出程序中可能需要用到的所有数字表示的英文单词：negative, zero, one, two, three, four, five, six, seven, eight, nine, ten, eleven,twelve, thirteen, fourteen, fifteen, sixteen, seventeen, eighteen, nineteen, twenty, thirty,forty, fifty, sixty, seventy, eighty, ninety, hundred, thousand, million。

注意：在转换过程中，尽可能使用**更大的数字单位**。比如 1500 这个数字应该的表示为 one thousand five hundred，而不应该是 fifteen hundred。

举例：数字 123419 转换成的英语表示为：one hundred twenty three thousand four hundred nineteen.

2.2 题目分析

1. **数据输入**. 注意到本题的输入范围在 Java 的`int`范围内，所以可以通过`Integer.parseInt`直接得到数字。

2. **数据处理**.

   - 可以参照英语的用语习惯，每三位划分，由数据范围可知最多划分成高中低三部分，分别用 million 和 thousand 分开，每一部分的打印逻辑都相同，可以总结为一个`void print_3digit(int a)`函数。
   - 为了避免负数可能导致的取余等计算问题，判断输入为负数并打印 negative 后即可。如果输入为 0 直接打印 zero 结束运行即可。
   - 特殊：如果输入为 0 直接打印 zero 结束运行即可

3.

2.3 代码展示

代码和注释如下所示，核心函数为`void print_3digit(int a)`(line26)，该函数将输入的三位数打印出来，这一过程又分为打印 hundred 和两位数打印`void print_2digit(int a)`(line35).

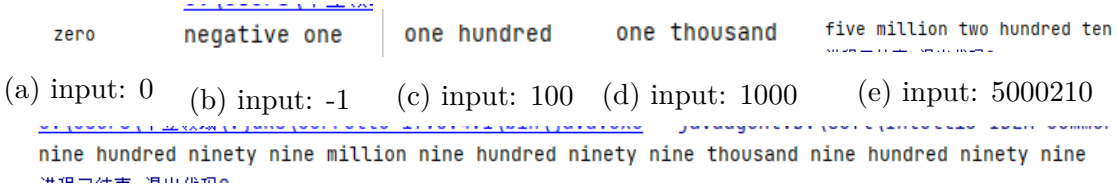
Listing 2: **Problem2.java**

```java
import static java.lang.Math.abs;

public class Problem2 {
    public static void main(String[] args) {
        int in = Integer.parseInt(args[0]);
        if (in == 0) {
            return;
        } else if (in < 0) {
            System.out.print("negative ");
            in = abs(in);
        }
        int low = in % 1000, middle = in / 1000 % 1000, high = in /
            1000000;
        if (high != 0) {
            print_3digit(high);
            System.out.print(" million ");
        }
        if (middle != 0) {
            print_3digit(middle);
            System.out.print(" thousand ");
        }
        print_3digit(low);
    }

    // print a(a 3-bit decimal number)'s literal form.
    public static void print_3digit(int a) {
        boolean mark = false;
        if (a / 100 != 0) {
            System.out.print(literal_digit(a / 100) + " hundred");
        }
        if ((a %= 100) != 0) print_2digit(a);
    }

    // print a(a 2-bit decimal number)'s literal form.
    public static void print_2digit(int a) {
```

```
35          if (a >= 20) {
36              System.out.print(switch (a / 10) {
37                  case 2 -> " twenty";
38                  case 3 -> " thirty";
39                  case 4 -> " forty";
40                  case 5 -> " fifty";
41                  case 6 -> " sixty";
42                  case 7 -> " seventy";
43                  case 8 -> " eighty";
44                  default -> " ninety";
45              });
46              if (a % 10 != 0) System.out.print(" " + literal_digit(a % 10));
47          } else if (a >= 10) {
48              System.out.print(switch (a % 10) {
49                  case 0 -> " ten";
50                  case 1 -> " eleven";
51                  case 2 -> " twelve";
52                  case 3 -> " thirteen";
53                  case 4 -> " fourteen";
54                  case 5 -> " fifteen";
55                  case 6 -> " sixteen";
56                  case 7 -> " seventeen";
57                  case 8 -> " eighteen";
58                  default -> " nineteen";
59              });
60          } else {
61              System.out.print(literal_digit(a));
62          }
63      }
64
65      // return a string which is a(digit)'s literal form.
66      public static String literal_digit(int a) {
67          return switch (a) {
68              case 1 -> "one";
69              case 2 -> "two";
70              case 3 -> "three";
```

```
71              case 4 -> "four";
72              case 5 -> "five";
73              case 6 -> "six";
74              case 7 -> "seven";
75              case 8 -> "eight";
76              default -> "nine";
77          };
78      }
79  }
```

2.4 结果展示



(a) input: 0 (b) input: -1 (c) input: 100 (d) input: 1000 (e) input: 5000210

(f) input: 999999999

图 3: Output of Problem2

2.5 总结与收获

- 程序一开始的设计过程并没有题目分析中写的那么有条理，函数的需要是在程序编写过程中感觉到的；
- idea 自动把`switch`的默认语法转换成一种更短更新颖易读的写法，之前从没接触过；
- **空格处理.** 我们已经将输出模块化了，虽然题目中没有要求，但如何在单词之间设计优雅的空格是一个值得思考的问题，具体设计过程会涉及很多条件处理。

实验 3. 长度 n 的子序列最大乘积

3.1 题目

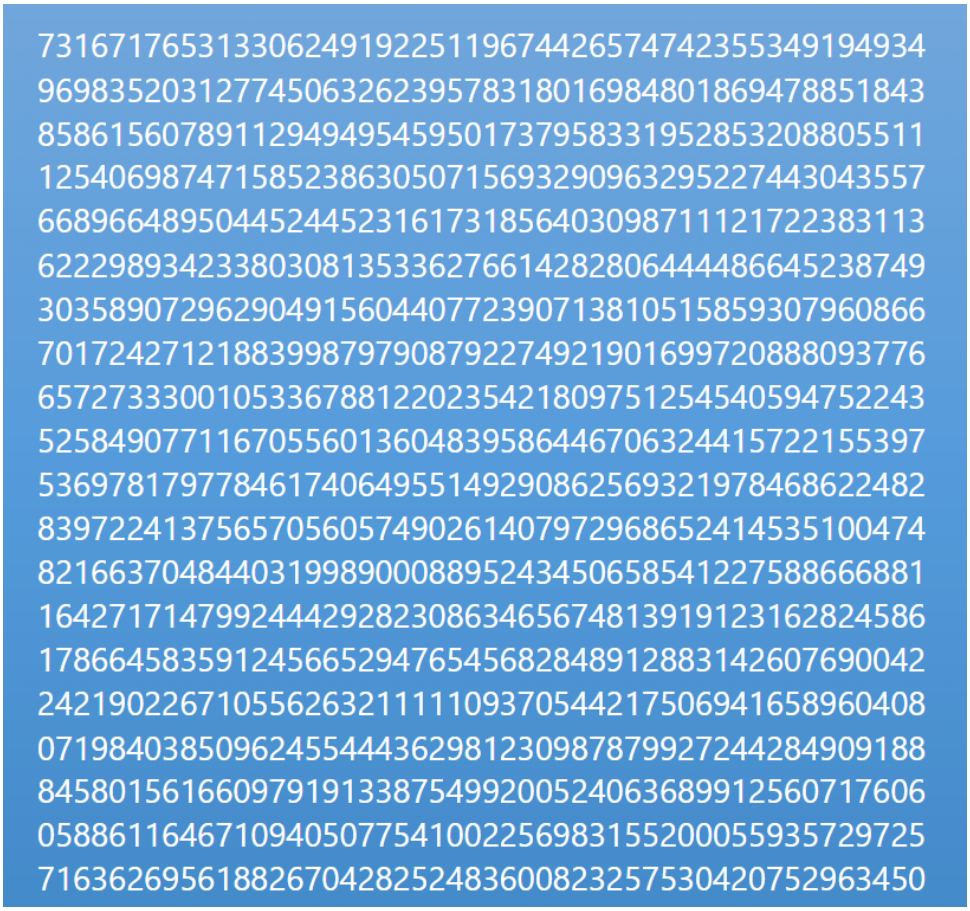
从文件中输入一个数字序列字符串，计算给定的长度n的子序列中的最大乘积值。

例如：如果输入“1027839564”，指定长度为 3 的最大子序列乘积值为 $270 = 9\times5\times6$；指定长度为 5 的最大子序列乘积值为 $7560 = 7\times8\times3\times9\times5$。

备注：

1. 数字序列字符串的最大长度maxLength的范围为：[1……1000]；
2. n的取值范围为 [1……maxLength-1]；
3. 程序要注意处理边界情况；
4. 程序的输入数据必须从文件中读取。

下图为一个长度为 1000 的字符串数字序列，在这个序列中，长度为 4 的最大子序列乘积为 $5832 = 9\times9\times8\times9$.



3.2 思路分析

题目本身很简单，解决和验证思路可以分为三部分

1. **数据输入**.
   - 题中要求从文件输入，我们将数据存放在 input 文件夹的 input3.txt 中；
   - 考虑到数据的长度，只能以字符串的形式输入，并通过 Problem1 中提到过的库函数以数组的方式处理数据。
2. **算法设计**. 只需要从第 [0……n-1] 位开始向后逐位扫描即可，但考虑到 n 位数相乘可能大于 int 甚至 long 类型的边界，综合考虑并查阅资料后，决定采用`Math`库里的`BigInteger`这个类. 下面对未采用的优化方案做出解释。
   - 高精度. 即用`String`模拟高位数乘法的过程。而实际上`BigInteger`类已经帮我们内部实现了这一点，而且配有丰富的函数和数值优化，这一定是比我们自己写更快更方便的。
   - 记忆组中的最小值，如果下一个比这个小就可以直接跳过。这一步看似优化了一步，实际上增加了比较的环节，从算法复杂性分析来看并没有优化。
3. **数据测试**. 本题思路简单，用题中所给数据验证即可

3.3 代码展示

为了便于验证，用一个循环来模拟不断查询新的最大子序列

Listing 3: **Problem3.java**

```
1  import java.io.File;
2  import java.io.FileNotFoundException;
3  import java.math.BigInteger;
4  import java.util.Scanner;
5
6  public class Problem3 {
7      public static void main(String[] args) throws FileNotFoundException {
8          Scanner ins = new Scanner(new File("./input/input3.txt"));
9          String s = ins.next();
10         while (true) {
11             System.out.print("Please input a length n, input 0 means quit\n
                   ");
```

```
12              Scanner inn = new Scanner(System.in);
13              int n = inn.nextInt();
14              if(n == 0) break;
15              else  System.out.println(SuperMaxMulti(s,n));
16              System.out.println();
17          }
18      }
19
20      /*
21      * use bigInteger settle String s's n length max multi result.
22       */
23       static String SuperMaxMulti(String s, int n){
24          BigInteger max = new BigInteger("0");
25          for(int i = 0;i < s.length() - n + 1;i++){
26              BigInteger tmp = new BigInteger("1");
27              for(int j = 0; j< n;j++) {
28                  tmp = tmp.multiply(BigInteger.valueOf((Long.parseLong(s.
                        substring(i + j, i + j + 1)))));
29              }
30              if(tmp.compareTo(max) > 0) max = tmp;
31          }
32          return String.valueOf(max);
33      }
34  }
```

3.4 结果展示

输入数据采用题目中的“1000 位数据”，并设置三种长度

```
Please input a length n, input 0 means quit
4
5832

Please input a length n, input 0 means quit
20
240789749760000

Please input a length n, input 0 means quit
100
0
```

3.5 总结与收获

第一次看到本题时我想到了 3.2 中所写的一些思路，苦恼于怎么优化其时间复杂度以及优雅地写出来。并不知道也没有打算使用 BigInteger 但后来认真分析，查阅 JDK 的官方文档和 BigInteger 的实现原理后意识到 BigInteger 这个类的便捷性和性能远比自己写的好，

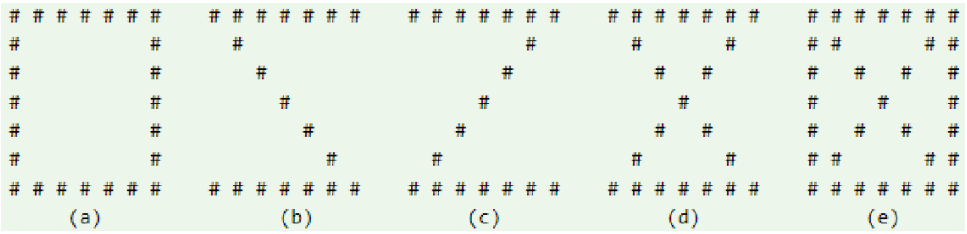
“如果你不知道为什么要优化，那这样的优化毫无意义 (not make sense at all)”

——Professor.Weaver (UC.Berkeley)

实验 4. 模式化打印图形

4.1 题目

编写一个程序，通过命令行参数的方式，接收两个参数。参数 1 指定打印的图形模式（分别为下图中的 $a\backslash b\backslash c\backslash d\backslash e$），参数 2 指定打印的图形的大小（一个非负整数），下面的模式图以大小 **7** 为例进行展示：



4.2 思路分析

1. **数据输入**. 数据以 String 的形式从命令行输入

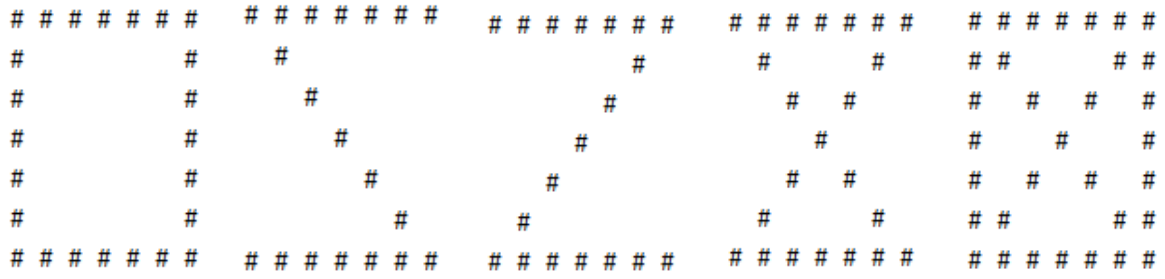
2. **程序思路**.

   - 可以把打印部分作为一个矩阵，如果矩阵中的某个元素行和列满足某种规律则打印`#`，否则打印 (空格). 通过观察可知，矩阵大小为`[n][2n-1]`. 我们可以不必真的开一个`int`或`boolean`二维数组的变量，而是直接打印出来。
   - 题目就提示这道题的思路，模块化打印: 分别对五种模式写五种打印函数，对输入的第一个参数`switch`.
     a. 第 0 列和第 n-1 列所有元素，第 0 行和第 n-1 行偶数索引的元素
     b. 行数等于列数/2，第 0 行，第 n-行的偶数索引的元素
     c. 行数等于列数/2，第 0 行，第 n-行的偶数索引的元素
     d. b 和 c 的结合
     e. a 和 d 的结合
   - 虽然五种模式的打印有一定的交叉，甚至可以通过两个叠加得到第三个，但如果只实现其中的几个函数意味着要对整个矩阵块扫描两次，而且要存储其状态，原先的思路是直接打印不存储。
   - **数据测试**. 打印图像题，采用题目数据通过即可。

4.3 代码与结果

4.2 中程序思路已经说的很清楚，代码中存在较多重复的，不再列出，具体附于附录。

运行结果如下



(a) input: a 7 (b) input: b 7 (c) input: c 7 (d) input: d 7 (e) input: e 7

图 4: Output of Problem4

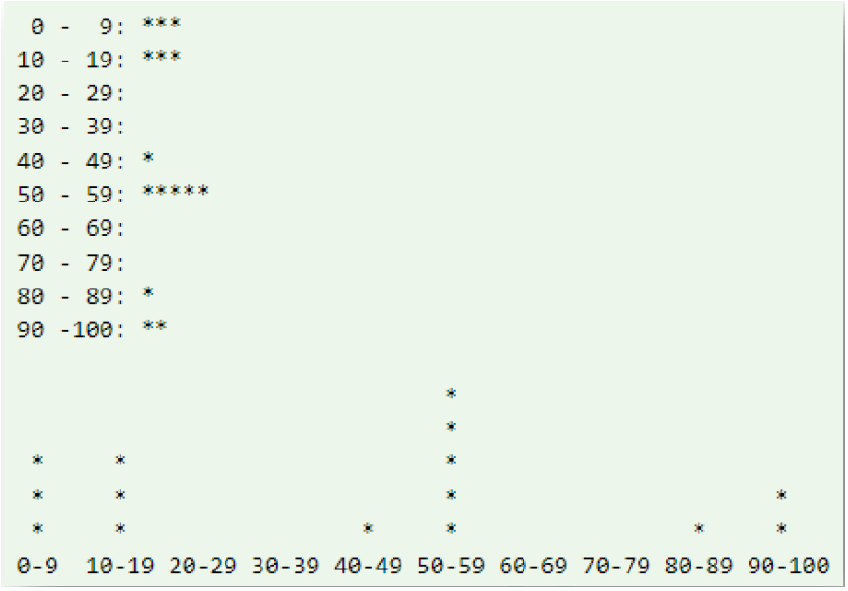
4.4 总结与收获

本题较为简单，但一开始输出时我以为矩阵是`[2n-1][2n-1]`，测试时才注意到，以后读题一定要更仔细一点，不要再因为这种间距不同导致理解产产生问题

实验 5. 直方统计图

5.1 题目

编写一个程序，从文件中读入不定个数的数据。每个数据都已空格或者回车分割，每个数的大小都在 [0..100] 之间，统计 10 个区间的数据的个数，并以两种方式将统计结果进行展示：水平直方图和垂直直方图。示例图如下：（测试数据自行准备）



5.2 思路分析

本题可分为四个模块。

1. **数据输入**. 为了优雅方便地测试和生成数据，我们采用Random类批量生成并存储在input/input5.txt中

   - 最常用的是 Random 中的public int nextInt(int origin, int bound) 这一函数，该函数会返回一个 [origin, bound) 之间的整数，下文分别简称为上界和下界；
   - 题目未要求数据的个数。考虑到题目的相关要求，决定把以下界为 50，上界为 101 随机得到一个数据个数；
   - 每个数据以下界为 0,上界为 101 随机生成,用回车分割并通过BufferedWriter类（需要 throw IOException）写入到文件中。

2. **数据读取与统计**. 用一个长度为 10 的整型数组（stat）存储十个部分各自出现的频次。

   - 因为数据个数不确定，所以需要hasNext()确定。
   - 从文件中读取数据时可以读取每个数据后直接将数据/10，然后将对应的数组中的数+1。需要为 100 设置一个特例。

3. **水平直方图打印**. 对每一行只需在第 i 行：

   (a) 打印各自的前缀，第一行和最后一行设置一个特例；

   (b) 打印stat[i]个*并打印回车；

4. **垂直直方图打印**. 参考 Problem4 中以像素矩阵的角度打印。

   (a) 首先遍历数组得到最大值以确定这个矩阵的高度；

   (b) 不用把每个打印都看作一个像素，而是把每一部分看作像素，经过这样分割后只需观察每个像素内打印空格的规律；

   (c) 对每一行刚开始的时候打印一个空格作为特例，为了保证输出的正确性最后一次打印后即回车；

   (d) 最后一行需要输入每个部分的表示。

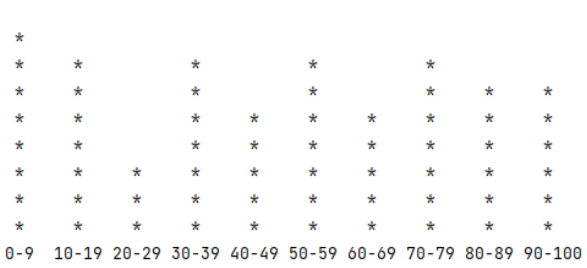
5.3 代码与测试

本题代码完全参照 5.2 中的思路，实现较为简单，具体可翻阅附录。
本题重在打印出的形式是否正确对应，所以随机打印即可。

5.4 运行结果

运行结果如下

```
 0 -  9: ********
10 - 19: *******
20 - 29: ***
30 - 39: *******
40 - 49: *****
50 - 59: *******
60 - 69: *****
70 - 79: *******
80 - 89: ******
90 -100: ******
```



5.5 总结与收获

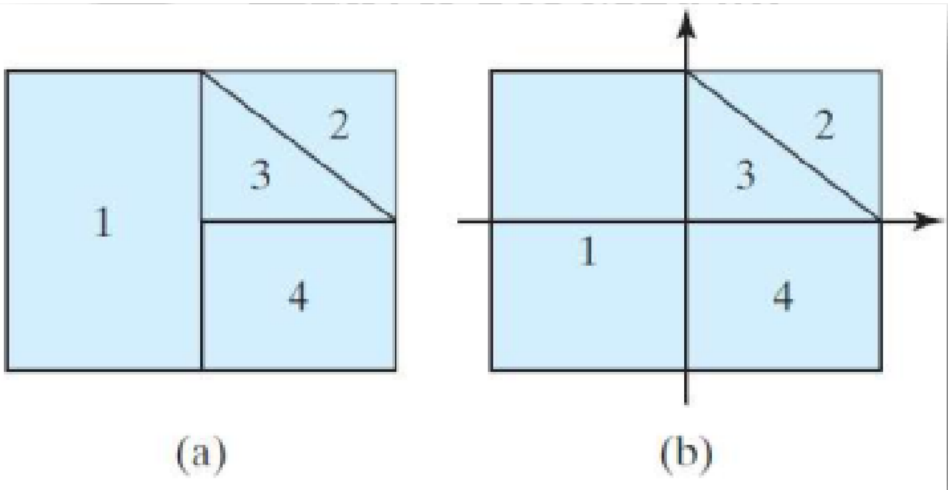
本题比较简单，只是有点繁琐，但需要考虑到设置范围的问题。其次需要从题目给定的样例中总结打印的规律，并把这个规律转换成打印时的一个个单元处理。

第一遍写代码时没有采用hasNext()，运行出现了问题，这说明我对文件输入依旧比较陌生；也没有采用即读即记录，而是打算先存储到 Vector中，以后写代码一定要下意识思考优化问题，但也不能看到什么都想到优化。

实验 6. 蒙特·卡洛方法模拟

6.1 题目

编写一个程序来模拟蒙特卡罗方法。具体的过程可以按照下面的方式来模拟：如下图所示（a）中的正方形，假设向正方形所代表的区域投掷飞镖，如果投掷 100000 次，那么飞镖落在奇数数字所对应区域的概率是多少呢？



提示：可以将（a）图放到如（b）所展示的坐标系中，程序随机的在整个区域中生成点，这样模拟投掷飞镖的行为。

6.2 思路分析

1. 如题目中提示所说，可以如 b 图所示建立坐标系，程序随机地在区域中生成点（即生成两个范围内的数作为横纵坐标）

2. 为了模拟更精确,随机生成应该采用`double nextDouble(double origin, double bound)`随机生成小数。代码中设正方形的范围为 $[-100, 100] \times [-100, 100]$。

3. 每次生成后需要判断该点是否在奇数数字，即 1、3 区域，通过简单的平面几何相关知识即可确定。如果在就将用来记录次数的变量 $+1$。

4. 随机生成 100000 次结束后输出记录次数的变量/100000 的值，此处需要注意的是需要强制转换为 double 否则会自动转为整型。

6.3 代码与结果展示

代码如下

Listing 4: **Problem6.java**

```java
import java.util.Random;

public class Problem6 {
    public static void main(String[] args) {
        Random r = new Random();
        int num = 0;
        for (int i = 0; i < 100000; i++) {
            double x = r.nextDouble(-100,100);
            double y = r.nextDouble(-100, 100);
            if (x < 0) num++;
            else if (x > 0 && y > 0 && x + y < 100) num++;
        }
        System.out.print((double) num / 100000);
    }
}
```

结果如下

```
0.62677
```

6.3 总结与收获

本题看似简单但输出有一个大坑，即强制转换，很久没使用我几乎已经遗忘了这个性质

本题的名字看似唬人，但题目难度很低，不要被吓到，仔细阅读要求即可

附录代码与运行结果汇总

实验 0. 通过键盘输入和文件输入

Listing 5: **Problem0.java**

```java
import java.io.File;
import java.util.Scanner;
import java.io.FileNotFoundException;

public class Problem0 {
    public static void main(String[] args) throws FileNotFoundException {
        inputscanner();
        keystroke();
    }

    public static void keystroke() {
        int num1;
        double num2;
        String name;
        double sum;

        Scanner in = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        num1 = in.nextInt();
        System.out.print("Enter a floating point number: ");
        num2 = in.nextDouble();
        System.out.print("Enter your name: ");
        name = in.next();
        System.out.printf("Hi! %s, the sum of %d and %.2f is %.2f \n", name
            , num1, num2, (num1 + num2));
    }

    public static void inputscanner() throws FileNotFoundException {
        int num1;
```

```
29          double num2;
30          String name;
31          double sum;
32          Scanner in = new Scanner(new File("./input/input0.txt"));
33          num1 = in.nextInt();
34          num2 = in.nextDouble();
35          name = in.next();
36          System.out.printf("Hi! %s, the sum of %d and %.2f is %.2f\n", name,
                num1, num2, (num1 + num2));
37          in.close();
38      }
39  }
```

```
Hi! yuan, the sum of 12 and 24.56 is 36.56
Enter an integer: 203
Enter a floating point number: 8.27
Enter your name: YangHao
Hi! YangHao, the sum of 203 and 8.27 is 211.27
```

图 5: 实验 0 运行结果

实验 1. UPC 码

Listing 6: **Problem1.java**

```
1  public class Problem1 {
2      public static void main(String[] args) {
3          System.out.print("Input:");
4          for (String arg : args) System.out.print(" " + arg);
5          System.out.println();
6          if(args.length != 1){
7              System.out.println("ERROR! Please input 1 argument instead of "
                   + args.length + ".");
8              return;
9          }
10         if (args[0].length() != 11) {
11             System.out.println("ERROR! Please input a 11-bit number instead
                   of " + args[0].length() + " bits.");
12             return;
13         }
14         int result = 50;
15         char tmp;
16         for (int i = 0; i < 11; i++) {
17             if (Character.isDigit(tmp = args[0].charAt(i))) {
18                 if (i % 2 == 0) result -= 3 * Integer.parseInt(String.
                       valueOf(tmp));
19                 else result -= Integer.parseInt(String.valueOf(tmp));
20             } else {
21                 System.out.println("ERROR! The " + (i + 1) + "th input bit
                       is '" + tmp + "', it's not a digit.");
22                 return;
23             }
24         }
25         if (result > 9 || result < 0) {
26             System.out.println("ERROR! The UPC code is " + result + ", it's
                   invalid.");
27             return;
28         }
```

```
29          System.out.print(result + args[0]);
30      }
31  }
```

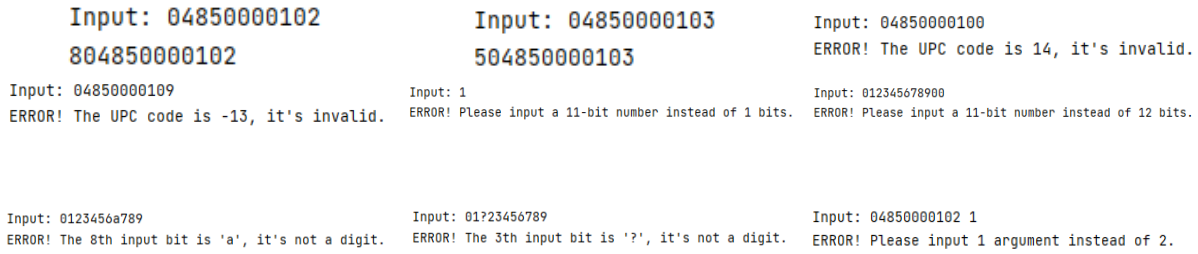


图 6: Output of Problem1

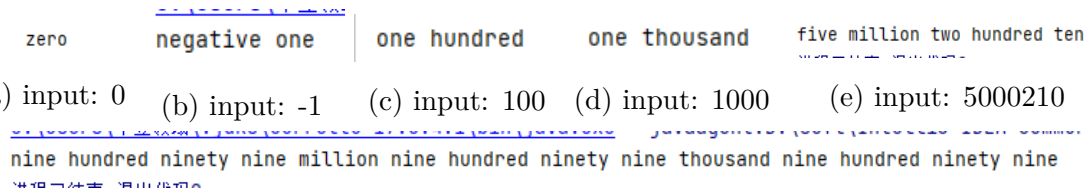
实验 2. 数字转英语

Listing 7: **Problem2.java**

```
1  import static java.lang.Math.abs;
2
3  public class Problem2 {
4      public static void main(String[] args) {
5          int in = Integer.parseInt(args[0]);
6          if (in == 0) {
7              return;
8          } else if (in < 0) {
9              System.out.print("negative ");
10             in = abs(in);
11         }
12         int low = in % 1000, middle = in / 1000 % 1000, high = in /
               1000000;
13         if (high != 0) {
14             print_3digit(high);
15             System.out.print(" million ");
16         }
17         if (middle != 0) {
18             print_3digit(middle);
19             System.out.print(" thousand ");
20         }
21         print_3digit(low);
22     }
23
24     // print a(a 3-bit decimal number)'s literal form.
25     public static void print_3digit(int a) {
26         boolean mark = false;
27         if (a / 100 != 0) {
28             System.out.print(literal_digit(a / 100) + " hundred");
29         }
30         if ((a %= 100) != 0) print_2digit(a);
31     }
32
```

```
33      // print a(a 2-bit decimal number)'s literal form.
34      public static void print_2digit(int a) {
35          if (a >= 20) {
36              System.out.print(switch (a / 10) {
37                  case 2 -> " twenty";
38                  case 3 -> " thirty";
39                  case 4 -> " forty";
40                  case 5 -> " fifty";
41                  case 6 -> " sixty";
42                  case 7 -> " seventy";
43                  case 8 -> " eighty";
44                  default -> " ninety";
45              });
46              if (a % 10 != 0) System.out.print(" " + literal_digit(a % 10));
47          } else if (a >= 10) {
48              System.out.print(switch (a % 10) {
49                  case 0 -> " ten";
50                  case 1 -> " eleven";
51                  case 2 -> " twelve";
52                  case 3 -> " thirteen";
53                  case 4 -> " fourteen";
54                  case 5 -> " fifteen";
55                  case 6 -> " sixteen";
56                  case 7 -> " seventeen";
57                  case 8 -> " eighteen";
58                  default -> " nineteen";
59              });
60          } else {
61              System.out.print(literal_digit(a));
62          }
63      }
64
65      // return a string which is a(digit)'s literal form.
66      public static String literal_digit(int a) {
67          return switch (a) {
68              case 1 -> "one";
```

```
69              case 2 -> "two";
70              case 3 -> "three";
71              case 4 -> "four";
72              case 5 -> "five";
73              case 6 -> "six";
74              case 7 -> "seven";
75              case 8 -> "eight";
76              default -> "nine";
77          };
78      }
79  }
```



(a) input: 0 (b) input: -1 (c) input: 100 (d) input: 1000 (e) input: 5000210

(f) input: 999999999

图 7: Output of Problem2

实验 3. 长度 n 的子序列最大乘积

Listing 8: **Problem3.java**

```
1  import java.io.File;
2  import java.io.FileNotFoundException;
3  import java.math.BigInteger;
4  import java.util.Scanner;
5
6  public class Problem3 {
7      public static void main(String[] args) throws FileNotFoundException {
8          Scanner ins = new Scanner(new File("./input/input3.txt"));
9          String s = ins.next();
10         while (true) {
11             System.out.print("Please input a length n, input 0 means quit\n
                   ");
12             Scanner inn = new Scanner(System.in);
13             int n = inn.nextInt();
14             if(n == 0) break;
15             else  System.out.println(SuperMaxMulti(s,n));
16             System.out.println();
17         }
18     }
19
20     /*
21     * use bigInteger settle String s's n length max multi result.
22      */
23      static String SuperMaxMulti(String s, int n){
24         BigInteger max = new BigInteger("0");
25         for(int i = 0;i < s.length() - n + 1;i++){
26             BigInteger tmp = new BigInteger("1");
27             for(int j = 0; j< n;j++) {
28                 tmp = tmp.multiply(BigInteger.valueOf((Long.parseLong(s.
                       substring(i + j, i + j + 1)))));
29             }
30             if(tmp.compareTo(max) > 0) max = tmp;
31         }
```

```
32          return String.valueOf(max);
33      }
34  }
```

```
Please input a length n, input 0 means quit
4
5832

Please input a length n, input 0 means quit
20
240789749760000

Please input a length n, input 0 means quit
100
0
```

图 8: Output of Problem3

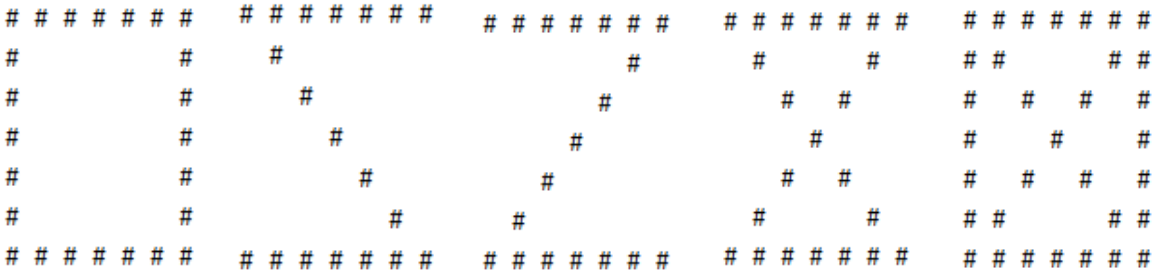
实验 4. 模式化打印图形

Listing 9: **Problem4.java**

```java
public class Problem4 {
    public static void main(String[] args) {
        int m = Integer.parseInt(args[1]);
        if (m == 0) return;
        switch (args[0]) {
            case "a" -> print_a(m);
            case "b" -> print_b(m);
            case "c" -> print_c(m);
            case "d" -> print_d(m);
            case "e" -> print_e(m);
            default -> System.out.print("The 1st argument is incorrect.");
        }
    }

    /*
     * print figure in a form with n*n width.
     */
    public static void print_a(int n) {
        int m = 2 * n - 1;
        boolean[][] pixel = new boolean[n][m];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                if (j == 0 || j == m - 1) System.out.print("#");
                else if ((i == 0 || i == n - 1) && j % 2 == 0) System.out.
                    print("#");
                else System.out.print(" ");
            }
            System.out.println();
        }
    }

    /*
     * print figure in b form with n*n width.
```

```
33      */
34     public static void print_b(int n) {
35         int m = 2 * n - 1;
36         boolean[][] pixel = new boolean[n][m];
37         for (int i = 0; i < n; i++) {
38             for (int j = 0; j < m; j++) {
39                 if ((i == 0 || i == n - 1 || j / 2 == i) && (j % 2 == 0))
40                     System.out.print("#");
41                 else System.out.print(" ");
42             }
43             System.out.println();
44         }
45     }
46 
47     /*
48      * print figure in c form with n*n width.
49      */
50     public static void print_c(int n) {
51         int m = 2 * n - 1;
52         boolean[][] pixel = new boolean[n][m];
53         for (int i = 0; i < n; i++) {
54             for (int j = 0; j < m; j++) {
55                 if ((i == 0 || i == n - 1 || (m - j) / 2 == i) && (j % 2 ==
                        0))
56                     System.out.print("#");
57                 else System.out.print(" ");
58             }
59             System.out.println();
60         }
61     }
62 
63     /*
64      * print figure in d form with n*n width.
65      */
66     public static void print_d(int n) {
67         int m = 2 * n - 1;
```

```
68          boolean[][] pixel = new boolean[n][m];
69          for (int i = 0; i < n; i++) {
70              for (int j = 0; j < m; j++) {
71                  if ((i == 0 || i == n - 1 || (m - j) / 2 == i || j / 2 == i
                        ) && (j % 2 == 0))
72                      System.out.print("#");
73                  else System.out.print(" ");
74              }
75              System.out.println();
76          }
77      }
78
79      /*
80       * print figure in e form with n*n width.
81       */
82      public static void print_e(int n) {
83          int m = 2 * n - 1;
84          boolean[][] pixel = new boolean[n][m];
85          for (int i = 0; i < n; i++) {
86              for (int j = 0; j < m; j++) {
87                  if (j == 0 || j == m - 1) System.out.print("#");
88                  else if ((i == 0 || i == n - 1 || (m - j) / 2 == i || j / 2
                            == i) && (j % 2 == 0))
89                      System.out.print("#");
90                  else System.out.print(" ");
91              }
92              System.out.println();
93          }
94      }
95  }
```



(a) input: a 7 (b) input: b 7 (c) input: c 7 (d) input: d 7 (e) input: e 7

图 9: Output of Problem4

实验 5. 直方统计图

Listing 10: **Problem5.java**

```java
import java.io.*;
import java.util.*;


public class Problem5 {
    public static void main(String[] args) throws IOException {
        GenerateInput();
        Scanner in = new Scanner(new File("./input/input5.txt"));
        int[] stat = new int[10];
        while (in.hasNext()) {
            int tmp = in.nextInt() / 10;
            if (tmp == 10) tmp -= 1;
            stat[tmp]++;
        }
        Print_Hori(stat);
        System.out.println();
        Print_Vert(stat);
    }

    /*
     * Generate needed data into input5.txt
     */
    public static void GenerateInput() throws IOException {
        BufferedWriter out = new BufferedWriter(new FileWriter("./input/
            input5.txt"));
        Random r = new Random();
        for (int i = 0; i < r.nextInt(50, 101); i++)
            out.write(r.nextInt(101) + "\n");
        out.close();
    }

    /*
     * print src (length is 10) which represents 10 segments' times in
```

```
        Horizontal form.
33      */
34      public static void Print_Hori(int[] src) {
35          for (int i = 0; i < 10; i++) {
36              if (i == 0) System.out.print(" 0 -  9: ");
37              else if (i == 9) System.out.print("90 -100: ");
38              else System.out.print(i * 10 + " - " + (i * 10 + 9) + ": ");
39              for (int j = 0; j < src[i]; j++) System.out.print("*");
40              System.out.println();
41          }
42      }
43
44      /*
45       * print src (length is 10) which represents 10 segments' times in
            Vertical form.
46       */
47      public static void Print_Vert(int[] src) {
48          int max = 0;
49          for (int i = 0; i < 10; i++) if (src[i] > max) max = src[i];
50          for (int i = 0; i < max; i++) {
51              System.out.print(" ");
52              for (int j = 0; j < 10; j++) {
53                  if (max - i <= src[j]) System.out.print("*");
54                  else System.out.print(" ");
55                  if (j != 9) System.out.print("     ");
56                  else System.out.println();
57              }
58          }
59          for (int k = 0; k < 10; k++) {
60              if (k == 0) System.out.print("0-9 ");
61              else if (k == 9) System.out.print(" 90-100");
62              else System.out.print(" " + k * 10 + "-" + (k * 10 + 9));
63          }
64      }
65  }
```

```
 0 -  9: ********
10 - 19: *******
20 - 29: ***
30 - 39: *******
40 - 49: *****
50 - 59: *******
60 - 69: *****
70 - 79: *******
80 - 89: ******
90 -100: ******

 *
 *    *          *          *          *
 *    *          *          *          *    *    *
 *    *          *    *     *    *     *    *    *
 *    *          *    *     *    *     *    *    *
 *    *    *     *    *     *    *     *    *    *
 *    *    *     *    *     *    *     *    *    *
 *    *    *     *    *     *    *     *    *    *
0-9  10-19 20-29 30-39 40-49 50-59 60-69 70-79 80-89 90-100
```

图 10: Output of Problem5

实验 6. 直方统计图

Listing 11: **Problem6.java**

```java
import java.util.Random;

public class Problem6 {
    public static void main(String[] args) {
        Random r = new Random();
        int num = 0;
        for (int i = 0; i < 100000; i++) {
            double x = r.nextDouble(-100,100);
            double y = r.nextDouble(-100, 100);
            if (x < 0) num++;
            else if (x > 0 && y > 0 && x + y < 100) num++;
        }
        System.out.print((double) num / 100000);
    }
}
```

```
0.62677
```

图 11: Output of Problem6

参考文献

A. 解题内容参考

a. Java® Platform, Standard Edition & Java Development Kit Version 17 API Specification [DB/OL]
https://docs.oracle.com/en/java/javase/17/docs/api/index.html, 2022.

B. LaTeX 代码参考

b. 【LaTeX】自用简洁模板（六）：学校作业

c. LaTeX 里「添加程序代码」的完美解决方案