面向对象程序设计
作业报告

第一次

姓名

班级

学号

电话

Email

日期

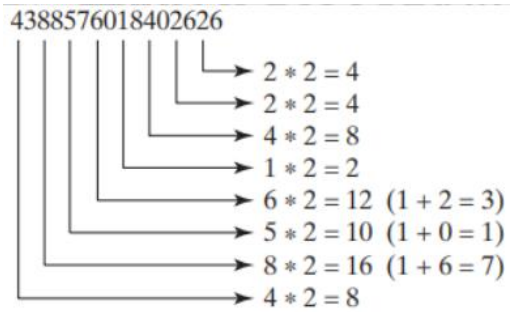
目录

题目 1：信用卡号码的校验

信用卡号码的编排遵循一定的规则，一般信用卡的号码长度是 13-16 位数字,并且首位数 字代表不同的卡类型(比如 4 代表 Visa 卡,5 代表 Master 卡等等)，另外它的所有数字使用 “Luhn 校验”算法（也称 mod 10 校验）完成检验。

1954 年，在 IBM 工作的 Hans Luhn 提出了用来验证信用卡号码的算法，该算法对卡片号 码输入是否正确起到了验证作用。随后这个算法被广泛应用于类似于身份识别码的验证。该算 法的具体执行步骤如下：（假如验证号码：4388576018402626）

1. 对数字序列从右到左，将所有的偶数位出现的数字翻倍计算，如果计算的结果是一个二 位数，那么就将二位数的每个数位的值进行加和运算。



2.将第 1 步中计算出的每个数字进行加和：4+4+8+2+3+1+7+8 = 37。

3.对数字序列从右到左，将所有的奇数位出现的数字进行加和：6+6+0+8+0+7+8+3=38。

4.将第 2 步和第 3 步的结果进行相加：37+38=75。

5.如果第 4 步计算的结果可以被 10 整除，那么这个卡号就是正确的，否则就是错误的。 因此上面例子中的卡号是一个非法的卡号。

编写一个程序，要求用户输入卡号（在程序中必须以读入整数代码的形式实现），输出用 户输入的卡号是否正确。

备注：

1. 信用卡的号码长度是 13-16 位，请注意存储卡号的整数类型；

2.如果不要求用户严格的输入 13-16 位数字，那么就需要在真正执行计算之前对用户输入 的数据合法性进行检查，请列举能想到的非法性输入，并将这些检查应用到程序实现中。

主程序设计:

1. 编写输入指令，运行代码时能让使用者输入一串数字（卡号）；
2. 编写方法，目的是检查输入卡号的合法性；
3. 通过调用方法检查卡号是否合法，并对其合法性进行输出。

检查卡号合法性的代码实现:

1. 编写 getSize（）方法以获取卡号的长度;

2. 编写 getPrefix（）方法以获取卡号前缀;

3. 编写 prefixMatched（）方法以确定卡号前缀是否为合法前缀；

4. 分别编写 calculateOfEvenPlace（）与 calculateOfOddPlace（）方法以计算卡号从右至左偶数位求和及从右至左奇数位求和的值；

5. 编写 isValid（）方法，检验卡号的合法性（若同时满足前缀匹配及奇偶数求和相加整体除十余零则合法）。

代码测试：

```
enter a credit card number:4388576018402626
4388576018402626 is invalid
```

```
enter a credit card number:4388576018410707
4388576018410707 is valid
```

总结：

通过在主程序中调用方法及在某一方法中调用其他方法（如本题代码中在 getPrefix（）中调用 getSize（），在 prefixMatched（）中调用 getPrefix（），在 calculateOfEvenPlace（）中调用 getSum（）等）可在简化代码的同时实现诸多功能，最后实现对输入卡号的合法性进行判断（通过检验可知代码输出的结果符合预期）。

题目 2：创建一个日期工具类 DateUtil

编写一个 DateUtil 类（其实是 library，只是用类这种语法将相关的函数聚集在一起），其为 与日期处理有关的工具类，具体包含的函数如下所列：

• static boolean isLeapYear(int year) o returns true if the given year is a leap year. A year is a leap year if it is divisible by 4 but not by 100, or it is divisible by 400.

• static boolean isValidDate(int year, int month, int day) o returns true if the given year, month and day constitute a given date. Assume that year is between 1 and 9999, month is between 1 (Jan) to 12 (Dec) and day shall be between 1 and 28|29|30|31 depending on the month and whether it is a leap year.

• static int getDayOfWeek(int year, int month, int day): o returns the day of the week, where 0 for SUN, 1 for MON, ..., 6 for SAT, for the given date. Assume that the date is valid.

• static void printCalendar(int year, int month) o The calendar for the specified year/month is output on the screen according to the calendar output format.

• static void printCalendar(int year) o The Calendar for the specified year is output on the screen according to the calendar output format.

• static String formatDate(int year, int month, int day) o prints the given date in the format "xxxday d mmm yyyy", e.g., "Tuesday 14 Feb 2012". Assume that the given date is valid.

提示：

1.可以参看下面的步骤计算给定日期是星期中的哪一天：

1. Based on the first two digit of the year, get the number from the following "century" table.

1700-	1800-	1900-	2000-	2100-	2200-	2300-	2400-
4	2	0	6	4	2	0	6

Take note that the entries 4, 2, 0, 6 repeat.

2. Add to the last two digit of the year.

3. Add to "the last two digit of the year divide by 4, truncate the fractional part".

4. Add to the number obtained from the following month table:

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Non-Leap Year	0	3	3	6	1	4	6	2	5	0	3	5
Leap Year	6	2	same as above									

5. Add to the day.

6. The sum modulus 7 gives the day of the week, where 0 for SUN, 1 for MON, ..., 6 for SAT.

例如，根据上面的计算步骤，2023 年 10 月 1 日是星期日：（6+23+23/4+0+1）%7=0(星期日)。

2.日历的输出格式可以参考下面的格式：

MON	TUE	WED	THU	FRI	SAT	SUN
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

3. 当 DateUtil 类实现之后，需要编写一个程序 TestDateUtil.java 验证每个函数执行的正确 性，要求检查尽可能全面。（可以借助 Java API 中的 Calendar 类中的函数辅助验证）

设计：

一．编写类 DateUtil{}:

1. 编写 isLeapYear（）以判断输入年份是否为闰年:

（1）设输入年分为 N 年;

（2）若 N 除 4 余 0 且除 100 不余 0，或 N 除 400 余 0，则 N 年为闰年，否则 N 年为非闰年。

（3）若 N 为闰年，则返回 true，若 N 不是闰年，则返回 false

2.编写 isValidDate（）以判断输入的日期是否正确:

（1）首先检查输入的年份是否正确，设输入的年份为 N 年，若 N<1 或 N>9999，则直接返回 false，若 1<N<9999（及判定输入的年份正确），则继续验证;

（2）设输入的月份为 M 月，输入的日为 D 日，若 M==1&&3&&5&&7&&8&&10&&12 且 D<32 则返回 true，若 M==4&&6&&9&&11 且 D<31 则返回 true，若 M==2，则通过调用 isLeapYear（）判断 N 年是否为闰年，若为闰年且 D<30 则返回 true，若不是闰年且 D<29 则返回 true,其余所有情况均返回 false。

3.编写 getDayOfWeek（）以判断给定日期为周几

（1）设定 num1、num2、num3、num4、num5 以便后续计算；

(2)若给定年份 N 年除 400 的余数<100, 则令 num1=6; 若 100<N 年除 400 的余数<200, 则令 num1=4; 若 200<N 年除 400 的余数<300, 则令 num1=2； 若 300<N 年除 400 的余数<400，则令 num1=0；

（3）num2=给定年份 N 年的后两位数字（即 num2=N%100）

（4）num3=num2/4 舍弃小数部分

（5）先判断给定年份是否为闰年，并根据输入的月份的不同给 num4 赋予不同的值（若为闰年，输入月份 M 月==1 时 num4=6, M==2 时 num4=2，M==3 时 num4=3，M==4 时 num4=6，M==5 时 num4=1, M==6 时 num4=4, M==7 时 num4=6, M==8 时 num4=2, M==9 时 num4=5, M==10 时 num4=0, M==11 时 num4=3, M==12 时 num4=5；若不是闰年，输入月份 M 月==1 时 num4=0，M==2 时 num4=3, M==3 时 num4=3, M==4 时 num4=6, M==5 时 num4=1, M==6 时 num4=4，M==7 时 num4=6，M==8 时 num4=2，M==9 时 num4=5，M==10 时 num4=0，M==11 时 num4=3，M==12 时 num4=5）

（6）num5 为输入的天数 day（num5=day）

（7）返回的值为[（num1+num2+num3+num4+num5）%7]，其中返回值为 0、1、2……7 时分别代表当天为周日、周一、……周六。

4.编写 printCalendar（）实现输出给定年份的给定月份的日历：

（1）使用 System.out.println()输出"MON TUE WED THU FRI SAT SUN",以便对照后续输出的日期；

（2）调用 getDayOfWeek（）判断给定年份的给定月份的第一天为一周的第几天，根据结果通过输出空格的形式将第一天对应到对应的周几的下方，并用 for 循环 print 语句继续输出至当月的最后一天，其中每次输出结果到达了"SUN"的下方时，将 print 语句改为 println 语句以达到换行的效果。

5.编写 printCalendar（）以输出给定年份一整年的日历：

（1）运用 for 循环，调用上部分编写的 printCalendar（）函数，其中 month 的值依次为 1、2、3……12，即可获得一整年的日历。

6.编写 formatDate（）以输出给定格式的日期：

（1）调用 getDayOfWeek（）函数根据输出的值判断给定日期为周几；

（2）设字符串 date，date 的值根据（1）中的值确定，即（1）中值为 0 时 date 为"Sunday"、（1）中值为 1 时 date 为"Monday"……(1)中值为 6 时 date 为"Saturday"；

（3）根据输入的月份 M 月输出对应的月份的英文缩写，例如 M==1 时返回 date+" "+day+" Jan "+year 。

7.编写 TestDateUtil{}类以验证上述编程的正确性。

运行结果展示：

1.对 isLeapYear（）函数的展示：

```
//验证isLeapYear
System.out.println(DateUtil.isLeapYear(2023));
System.out.println(DateUtil.isLeapYear(2020));
System.out.println(DateUtil.isLeapYear(2000));
System.out.println(DateUtil.isLeapYear(1900));
```

```
false
true
true
false
```

2.对 isValidYear（）函数的展示：

```
//验证isValidDate
System.out.println(DateUtil.isValidDate( year: 2023, month: 10, day: 1));
System.out.println(DateUtil.isValidDate( year: 10000, month: 10, day: 1));
System.out.println(DateUtil.isValidDate( year: 2023, month: 13, day: 1));
System.out.println(DateUtil.isValidDate( year: 2023, month: 10, day: 41));
```

```
true
false
false
false
```

3.对 getDayOfWeek（）函数的展示：

```
//验证getDayOfWeek(0 for SUN,1 for MON,.....)
System.out.println(DateUtil.getDayOfWeek( year: 2023, month: 10, day: 1));
System.out.println(DateUtil.getDayOfWeek( year: 2020, month: 2, day: 29));
System.out.println(DateUtil.getDayOfWeek( year: 2050, month: 4, day: 25));
```

```
0
6
1
```

4.对 printCalendar（）函数的展示：

```
//验证printCalendar
DateUtil.printCalendar( year: 2023, month: 10);
DateUtil.printCalendar( year: 2023, month: 11);
DateUtil.printCalendar( year: 2023, month: 2);
```

```
MON TUE WED THU FRI SAT SUN
                        1
  2   3   4   5   6   7   8
  9  10  11  12  13  14  15
 16  17  18  19  20  21  22
 23  24  25  26  27  28  29
 30  31
MON TUE WED THU FRI SAT SUN
          1   2   3   4   5
  6   7   8   9  10  11  12
 13  14  15  16  17  18  19
 20  21  22  23  24  25  26
 27  28  29  30
MON TUE WED THU FRI SAT SUN
          1   2   3   4   5
  6   7   8   9  10  11  12
 13  14  15  16  17  18  19
 20  21  22  23  24  25  26
 27  28
```

5.对 printCalendar（）函数的展示:

```
//验证printCalendar
DateUtil.printCalendar( year: 2023);
DateUtil.printCalendar( year: 2020);
```

```
Month1
MON TUE WED THU FRI SAT SUN
                        1
  2   3   4   5   6   7   8
  9  10  11  12  13  14  15
 16  17  18  19  20  21  22
 23  24  25  26  27  28  29
 30  31
Month2
MON TUE WED THU FRI SAT SUN
          1   2   3   4   5
  6   7   8   9  10  11  12
 13  14  15  16  17  18  19
 20  21  22  23  24  25  26
 27  28
Month3
MON TUE WED THU FRI SAT SUN
          1   2   3   4   5
  6   7   8   9  10  11  12
 13  14  15  16  17  18  19
 20  21  22  23  24  25  26
 27  28  29  30  31
Month4
MON TUE WED THU FRI SAT SUN
                    1   2
  3   4   5   6   7   8   9
 10  11  12  13  14  15  16
 17  18  19  20  21  22  23
 24  25  26  27  28  29  30

Month5
MON TUE WED THU FRI SAT SUN
  1   2   3   4   5   6   7
  8   9  10  11  12  13  14
 15  16  17  18  19  20  21
 22  23  24  25  26  27  28
 29  30  31
```

```
Month6
MON TUE WED THU FRI SAT SUN
                1   2   3   4
  5   6   7   8   9  10  11
 12  13  14  15  16  17  18
 19  20  21  22  23  24  25
 26  27  28  29  30
Month7
MON TUE WED THU FRI SAT SUN
                    1   2
  3   4   5   6   7   8   9
 10  11  12  13  14  15  16
 17  18  19  20  21  22  23
 24  25  26  27  28  29  30
 31
Month8
MON TUE WED THU FRI SAT SUN
      1   2   3   4   5   6
  7   8   9  10  11  12  13
 14  15  16  17  18  19  20
 21  22  23  24  25  26  27
 28  29  30  31
Month9
MON TUE WED THU FRI SAT SUN
                1   2   3
  4   5   6   7   8   9  10
 11  12  13  14  15  16  17
 18  19  20  21  22  23  24
 25  26  27  28  29  30
Month10
MON TUE WED THU FRI SAT SUN
                        1
  2   3   4   5   6   7   8
  9  10  11  12  13  14  15
 16  17  18  19  20  21  22
 23  24  25  26  27  28  29
 30  31
```

```
Month11
MON TUE WED THU FRI SAT SUN
          1   2   3   4   5
  6   7   8   9  10  11  12
 13  14  15  16  17  18  19
 20  21  22  23  24  25  26
 27  28  29  30
Month12
MON TUE WED THU FRI SAT SUN
                  1   2   3
  4   5   6   7   8   9  10
 11  12  13  14  15  16  17
 18  19  20  21  22  23  24
 25  26  27  28  29  30  31


Month1
MON TUE WED THU FRI SAT SUN
          1   2   3   4   5
  6   7   8   9  10  11  12
 13  14  15  16  17  18  19
 20  21  22  23  24  25  26
 27  28  29  30  31
Month2
MON TUE WED THU FRI SAT SUN
                      1   2
  3   4   5   6   7   8   9
 10  11  12  13  14  15  16
 17  18  19  20  21  22  23
 24  25  26  27  28  29
Month3
MON TUE WED THU FRI SAT SUN
                          1
  2   3   4   5   6   7   8
  9  10  11  12  13  14  15
 16  17  18  19  20  21  22
 23  24  25  26  27  28  29
 30  31
```

```
Month4
MON TUE WED THU FRI SAT SUN
          1   2   3   4   5
  6   7   8   9  10  11  12
 13  14  15  16  17  18  19
 20  21  22  23  24  25  26
 27  28  29  30
Month5
MON TUE WED THU FRI SAT SUN
                  1   2   3
  4   5   6   7   8   9  10
 11  12  13  14  15  16  17
 18  19  20  21  22  23  24
 25  26  27  28  29  30  31

Month6
MON TUE WED THU FRI SAT SUN
  1   2   3   4   5   6   7
  8   9  10  11  12  13  14
 15  16  17  18  19  20  21
 22  23  24  25  26  27  28
 29  30
Month7
MON TUE WED THU FRI SAT SUN
          1   2   3   4   5
  6   7   8   9  10  11  12
 13  14  15  16  17  18  19
 20  21  22  23  24  25  26
 27  28  29  30  31
Month8
MON TUE WED THU FRI SAT SUN
                      1   2
  3   4   5   6   7   8   9
 10  11  12  13  14  15  16
 17  18  19  20  21  22  23
 24  25  26  27  28  29  30
 31
```

```
Month9
MON TUE WED THU FRI SAT SUN
      1   2   3   4   5   6
  7   8   9  10  11  12  13
 14  15  16  17  18  19  20
 21  22  23  24  25  26  27
 28  29  30
Month10
MON TUE WED THU FRI SAT SUN
              1   2   3   4
  5   6   7   8   9  10  11
 12  13  14  15  16  17  18
 19  20  21  22  23  24  25
 26  27  28  29  30  31
Month11
MON TUE WED THU FRI SAT SUN
                          1
  2   3   4   5   6   7   8
  9  10  11  12  13  14  15
 16  17  18  19  20  21  22
 23  24  25  26  27  28  29
 30
Month12
MON TUE WED THU FRI SAT SUN
      1   2   3   4   5   6
  7   8   9  10  11  12  13
 14  15  16  17  18  19  20
 21  22  23  24  25  26  27
 28  29  30  31
```

6.对 formatDate（）函数的展示：

```
//验证formatDate
System.out.println(DateUtil.formatDate( year: 2023, month: 10, day: 1));
System.out.println(DateUtil.formatDate( year: 2020, month: 1, day: 10));
System.out.println(DateUtil.formatDate( year: 1900, month: 2, day: 28));
```

```
Sunday 1 Oct 2023
Friday 10 Jan 2020
Wednesday 28 Feb 1900
```

总结：

通过查阅资料、查看题目提示等，学习并掌握了：

（1）判断给定年份是否为闰年的方法；

（2）判断给定年份、月份、日数是否正确的方法;

（3）判断给定年份、月份、日数为一周中的哪一天的方法;

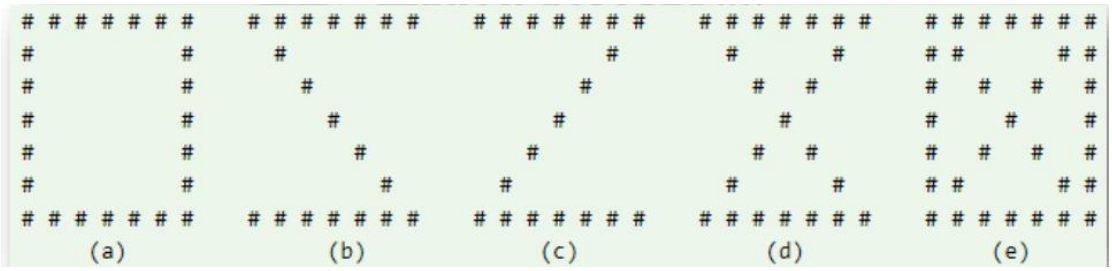
（4）打印给定年份的给定月份的日历的方法;

（5）打印给定年份整年的日历的方法;

（6）以给定格式输出对应年份、月份、日数的周几（英文）、日数（数字）、月份（英文缩写）、年份（数字）的方法。

题目 3：模式化打印图形

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\b\c\d\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：



设计：

1. 调用 Scanner 函数，让使用者输入“a”、“b”、“c”、“d”、“e”中的一个字母；

2. 调用 Scanner 函数，让使用者输入其对图形大小的预期，如输入“7”则第一行有 7 个“#”组成；

3. 使用 for 循环，轮流输出空格和“#”作为图形的第一行，大小为输入的 size 大小；

4. 使用 if 语句，用 equals（）方法检测输入的字母，并调用对应的图形的函数，输出第二行至第 size-1 行的图形；

5. 使用 for 循环，轮流输出空格和“#”作为图形的第 size 行，大小为输入的 size 大小；

6. 分别编写函数以输出上述五个图形中的任意一种：

a：

（1）设整数变量 size 为图形的大小；

（2）使用 for 循环，通过向左右两个“#”中间添加适当空格的方法使其与例题图形类似，以此输出第二至第 size-1 行的图形；

b:

（1）设整数变量 size 为图形的大小;

（2）使用 for 循环，根据行数的增加而增加每行第一个“#”前空格的数量，输出第二行至第 size-1 行的图形;

c:

（1）设整数变量 size 为图形的大小;

（2）使用 for 循环，根据行数的增加而减少每行第一个“#”前空格的数量，输出第二行至第 size-1 行的图形;

d:

（1）设整数变量 size 为图形的大小;

（2）使用 for 循环，输出第二至第 size-1 行的图形，对每行再使用 for 循环，使其循环输出空格或“#”，使用 if 语句判定输出“#”还是空格，使每行“#”出现在正确的位置;

e:

（1）设整数变量 size 为图形的大小;

（2）使用 d 中对第二至第 size-1 行的图形的方法，只需将循环中的第一和最后一个空格改为“#”即可;

运行结果展示:

1. 输出 3 行“a”:

```
Please enter 'a' or 'b' or 'c' or 'd' or 'e':
a
Please enter a number to determine the figure size:
3
# # #
#   #
# # #
```

2. 输出 5 行“b”:

```
Please enter 'a' or 'b' or 'c' or 'd' or 'e':
b
Please enter a number to determine the figure size:
5
# # # # #
  #
    #
      #
# # # # #
```

3.输出 7 行“c”:

```
Please enter 'a' or 'b' or 'c' or 'd' or 'e':
c
Please enter a number to determine the figure size:
7
# # # # # # #
          #
        #
      #
    #
  #
# # # # # # #
```

4.输出 9 行“d”:

```
Please enter 'a' or 'b' or 'c' or 'd' or 'e':
d
Please enter a number to determine the figure size:
9
# # # # # # # # #
  #           #
    #       #
      #   #
        #
      #   #
    #       #
  #           #
# # # # # # # # #
```

5.输出 10 行“e”：

```
Please enter 'a' or 'b' or 'c' or 'd' or 'e':
e
Please enter a number to determine the figure size:
10
# # # # # # # # # #
# #             # #
#   #         #   #
#     #     #     #
#       # #       #
#       # #       #
#     #     #     #
#   #         #   #
# #             # #
# # # # # # # # # #
```

总结与收获：

1. 编写代码中尝试过使用“==”来判定输入的字母是否与“a”，“b”，“c”，“d”，“e”中的一个相等，结果都是 false，通过查阅资料后弄清楚了“==”的使用条件，且掌握了 equals（）方法的使用，并在后续代码中引入了 equals（）函数，使用 if 语句将输入的字母准确的对应到该字母的预期图形的编写程序上，以便正确调用函数输出正确的图形；
2. 编写代码时原本将第一行和最后一行图形都分别写在了每个图形函数里面，造成了大量的重复代码，后续直接将第一行和最后一行代码提出至 main（）函数中输出，减少重复的代码。

题目 4：摩斯密码

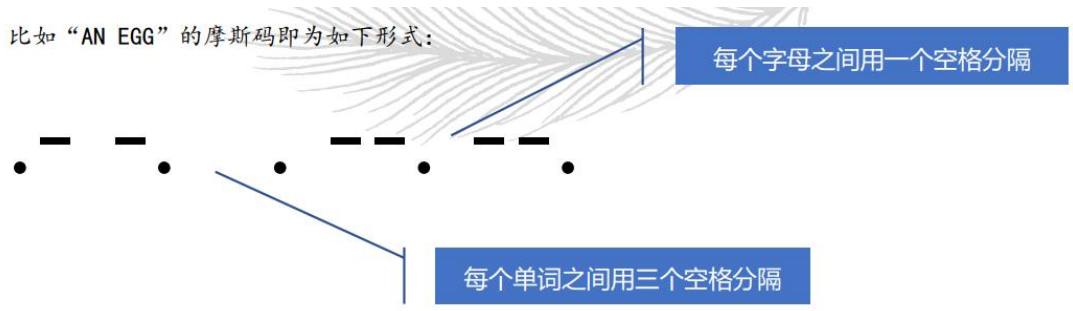
莫尔斯电码（又译为摩斯密码，Morse code）是一种时通时断的信号代码，通过不同的排列 顺序来表达不同的英文字母（不区分大小写）、数字和标点符号。它发明于 1837 年，发明者有争议， 是美国人塞缪尔.莫尔斯或者艾尔菲德.维尔。摩斯密码是一种早期的数字化通信形式，但是它不 同于现代只使用零和一两种状态的二进制码，它的代码包括点、划、点和划之间的停顿、每个词 之间的中等的停顿以及句子之间长的停顿。题目中的停顿用空格字符表示，摩斯码之间的停顿为 一个空格，摩斯码构成的单词和单词之间的停顿用三个空格表示。具体的摩斯编码如下：

Letter	Morse
A	.-
B	-...
C	-.-.
D	-..
E	.
F	..-.
G	--.
H	....
I	..
J	.---
K	-.-
L	.-..
M	--

Letter	Morse
N	-.
O	---
P	.--.
Q	--.-
R	.-.
S	...
T	-
U	..-
V	...-
W	.--
X	-..-
Y	-.--
Z	--..

Digit	Morse
0	-----
1	.----
2	..---
3	...--
4	....-
5	.....
6	-....
7	--...
8	---..
9	----.

Punctuation Mark	Morse
Error (also <HH>)	....
& Ampersand	.-...
' Apostrophe	.----.
@ At sign	.--.-.
) Bracket, close (parenthesis)	-.--.-
(Bracket, open (parenthesis)	-.--.
: Colon	---...
, Comma	--..--
= Equals sign	-...-
! Exclamation mark	-.-.--
. Full-stop (period)	.-.-.-
- Hyphen	-....-
× Multiplication sign (also x)	-..-
% Percentage (literally 0/0)	----- -..-. -----
+ Plus sign	.-.-.
" Quotation marks	.-..-.
? Question mark (query)	..--..
/ Slash	-..-.



题目中提供一个包含摩斯密码的文本文件，名为：encode.txt。

请编写一个 MorseCodeDecode.java 程序，该程序的输入数据从文件 encode.txt 中读取，最终 给出对该文件所包含的摩斯密码的解密结果。

备注：为了简化问题，encode.txt 中并不涉及 Error 和%这两个字符对应的摩斯码。

设计:

1. 先创建类类型 MorseCodeTurnToLetter{}，将所有材料中提到的字母、数字、字符及其对应的摩斯密码输入;

2. 创建类类型 MorseCodeDecode{}，将所需解密的摩斯码以字符串的形式储存;

3. 运用 StringBuffer 的方法复制该摩斯码，以便后续与 MorseCodeTurnToLetter 类的对照表进行对照;

4. 编写 Decryption（）函数，目的是解密输入的摩斯码;

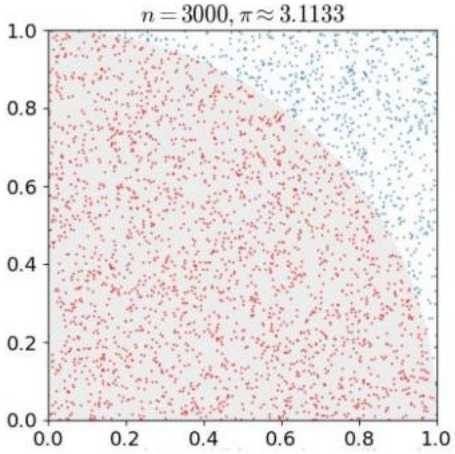
5. 在主程序中建立新的 MorseCodeDecode{}类，并对其运行 Decryption（）函数，使其输出摩斯密码对应的英语字母。

运行代码展示:

代码未成功运行...

题目 5：蒙特·卡洛方法模拟

蒙特·卡罗方法（Monte Carlo method），也称统计模拟方法，是二十世纪四十年代中期 由于科学技术的发展和电子计算机的发明，而被提出的一种以概率统计理论为指导的一类非常 重要的数值计算方法，是指使用随机数（或更常见的伪随机数）来解决很多计算问题的方法， 与它对应的是确定性算法。蒙特·卡罗方法在金融工程学，宏观经济学，计算物理学（如粒子 输运计算、量子热力学计算、空气动力学计算）等领域应用广泛。题目要求使用蒙特卡洛方法求解圆周率Π。具体的过程可以按照下面的方式来模拟：下图为边长为 1 的正方形，随机的向 该正方形中投入 n 个点，统计落入 1/4 正切圆中的点的个数为 m，那么可以通过 m/n 的比例间 接求解圆周率。如图所示，当投入的点数为 3000 时，统计出来的Π近似为 3.1133。



编写一个 PIByMonte.java 程序，该程序接收 1 个命令行参数 n（意义如题中描述），运行 结束后，输出在给定的 n 下，求解的近似圆周率值。另外，尝试将 n 的值不断增长，看看程序 输出结果的变化趋势。

设计：

1. 建立 MonteCarlo（）函数；

2. 设立整数参数 num，用于计算符合要求的点数为多少；

3. 运用 for 循环，循环次数为使用者输入的数字的大小，设立 double x 和 double y 作为平面直角坐标系的两垂直坐标向量，分别使用 Math.random（）函数给 x 和 y 赋值，若 x 的平方加上 y 的平方小于 1，则判断其在四分之一圆内，此时 num 的值加 1，否则 num 的值不变；

4. 设立 double pi，其为计算所得的π的值，其大小为 4 倍 num 的大小除以使用者输入的 n 的值;

5. 在主程序中调用 MonteCarlo（）函数，输出计算所得的π的值。

运行结果展示:

1.n=3000 时:

```
Please enter the size of n:
3000
3.148
```

2.n=8000 时:

```
Please enter the size of n:
8000
3.094
```

3.n=9999999 时:

```
Please enter the size of n:
9999999
3.1421067142106716
```

总结与收获:

1. 通过翻看教材及查阅资料，融会贯通了 Math 类的部分函数的使用方法，并将其运用到了代码之中（对 x 和 y 赋予 0 到 1 的随机值）；

2. 了解并掌握了蒙特卡洛方法的原理，熟知了该方法计算圆周率π的步骤。

题目 6：一只兔子地行走

一只兔子在我校地东操场漫无目的地行走，假设这只兔子每一步都走 1 米（有点夸张，但 并不影响模拟地结论），且它地行走方向只能是东、西、南、北四个方向,选择每个方向的概 率都是一样的,那么当这只兔子行走 n 步之后，离它出发的点有多远呢？这个过程就是著名的 二维随机游走问题。

1. 编写一个 RandomWalker.java 程序,该程序接收一个整数类型的命令行参数 n，功能是 模拟题干中描述的二维随机游走 n 步的过程。程序要求输出：每一步的坐标（包括起始步的坐 标，假设每次起始步的坐标都是（0，0）），除此以外，还需要给出 n 步之后最终位置和起始位 置的欧几里得距离。

2. 编写一个 RandomWalkers.java 程序，该程序接收两个整数类型的命令行参数 n 和 trials。n 的意义如同 1 中描述的含义，trials 表示的是完成实验的次数（一次实验即一只兔 子完成 n 步行走）。程序要求输出：完成 trials 次实验之后的欧几里得距离的平均值。 备注：执行多次该程序，调整 n 参数和 trials 参数，观察是否随着 n 的增加，兔子行走 的距离会离原点越来越远？如果不是的话，那么距离和 n 的值有关系吗？

设计：

一．RandomWalker{}类的设计

1. 运用 Scanner 函数，让用户能输入一串数字，这串数字即为该次试验中兔子走的总步数；

2. 建立平面直角坐标系，分别设 int x=0 和 int y=0，并使用 System 函数输出当前坐标“initial coordinate is （0,0）”；

3. 使用 for 循环，循环次数为用户输入的数字的大小，设立 double number=Math.random（），所得的 number 的值为 0 至 1 中的随机数；

4. 假设 0 至 0.25 为东，则 number 处于此区间时 x 的值加一，假设 0.25 至 0.50 为西，则 number 处于此区间时 x 的值减一，假设 0.50 至 0.75 为北，则 number 处于此区间时 y 的值加一，假设 0.75 至 1 为南，则 number 处于此区间时 y 的值减一；

5. 运用 System 函数输出每次循环后兔子的实时坐标；

6. 在循环外输入 Math.sqrt（）函数计算 x 的平方加 y 的平方后开根的结果，并使用 System 函数将其输出，此结果即为兔子的最终坐标与初始坐标（0,0）的直线距离。

二. RandomWalkers{}类的设计

1. 运用两次 Scanner 函数，第一次让用户输入进行实验的次数，第二次让用户输入每次实验兔子走的总步数；

2. 设立 double sum 作为计数变量，叠加每次实验所计算的距离的值；

3. 使用 for 循环，循环次数为用户输入进行实验的次数，每次实验均使用 RandomWalker{}类的方法计算距离，并通过 sum 进行累加；

4. 使用 System 函数输出最终的结果，即每一实验中兔子与初始坐标距离的平均值，其值为 sum 除以进行实验的次数。

运行结果展示：

1. 展示 RandomWalker{}类的运行结果）：

（1）兔子走了 5 步：

```
Please enter the number of steps the rabbit walks:
5
initial coordinates is: (0,0)
after 1 steps the coordinates is:(1,0)
after 2 steps the coordinates is:(1,-1)
after 3 steps the coordinates is:(1,0)
after 4 steps the coordinates is:(0,0)
the final coordinates is:(0,1)
the distance between the final coordinate and the initial coordinate is:1.0
```

（2）兔子走了 50 步：

```
the final coordinates is: (-2,0)
the distance between the final coordinate and the initial coordinate is:2.0
```

（3）兔子走了 500 步：

```
the final coordinates is: (26,8)
the distance between the final coordinate and the initial coordinate is:27.202941017470888
```

（4）兔子走了 5000 步：

```
the final coordinates is: (30,-18)
the distance between the final coordinate and the initial coordinate is:34.9857113690718
```

（5）兔子走了 50000 步：

```
the final coordinates is: (176,-236)
the distance between the final coordinate and the initial coordinate is:294.40108695451516
```

2. 展示 RandomWalkers{}类的运行结果：

（1）10 次实验，每次走 50 步；

```
Please enter the number of trails:
10
Please enter the number of steps the rabbit walks:
50
the average of distance is:5.602724214818981
```

（2）100 次实验，每次走 50 步；

```
Please enter the number of trails:
100
Please enter the number of steps the rabbit walks:
50
the average of distance is:6.609948414622803
```

（3）10 次实验，每次走 500 步；

```
Please enter the number of trails:
10
Please enter the number of steps the rabbit walks:
500
the average of distance is:19.764755365358962
```

（4）100 次实验，每次走 500 步；

```
Please enter the number of trails:
100
Please enter the number of steps the rabbit walks:
500
the average of distance is:19.373930204317055
```

总结：

根据实验结果显示，多次执行该程序，调整兔子走的步数 n 及实验次数 trails 的值，发现随着 n 的增加，兔子行走的距离会离原点越来越远。