# 面向对象程序设计
# 作业报告

第 一 次



| | |
|---|---|
| 姓名 | 曹家豪 |
| 班级 | 软件 2204 班 |
| 学号 | 2226114017 |
| 电话 | 13572763245 |
| Email | caojiahao@stu.xjtu.edu.cn |
| 日期 | 2023-10-8 |

# 目录

isValidCardNumber 函数：将 input 储存的卡号转换为 long 型→判断是否全部转换成功：

（1）转换成功→转换回 String 类型→判断位数是否为 13-16 位 →否，返回“false”||是→定义 doubleDigit 用于判断位数奇偶→根据判断方法对偶数和奇数分别进行处理→将和存储在 sum 中→判断 sum 能否被 10 整除→能，返回“true” 反之返回“false“

（2）转换失败→返回 false

4、 主干代码说明

下图是 isValidCardNumber 函数主干：

```
1.          try{//将 String 转为 long 并判断是否转换成功
2.              long cardNumber=Long.parseLong(input);
3.          // 将卡号转换为字符串
4.          String cardNumberStr = Long.toString(cardNumber);
5.          // 检查卡号长度是否在 13-16 位之间
6.          if (cardNumberStr.length() < 13 || cardNumberStr.le
   ngth() > 16) {
7.              return false;
8.          }
9.          // 从右到左遍历卡号的每一位数字
10.         int sum = 0;
11.         boolean doubleDigit = false;
12.         for (int i = cardNumberStr.length() - 1; i >= 0; i-
   -) {
13.             int digit = Character.getNumericValue(cardNumbe
   rStr.charAt(i));
14.             // 如果是偶数位，则将数字翻倍
15.             if (doubleDigit) {
16.                 digit *= 2;
17.                 // 如果翻倍后的数字是两位数，则将两个数字相加
18.                 if (digit > 9) {
19.                     digit = digit % 10 + digit / 10;
20.                 }
21.             }
22.             sum += digit;
23.             doubleDigit = !doubleDigit;
24.         }
```

```
25.         // 检查校验和是否能被 10 整除
26.         return sum % 10 == 0;}
27.         catch(NumberFormatException e){
28.             return false;
29.         }
```

5、 运行结果展示

```
请输入信用卡号码：5566644588545646
输入的卡号是错误的。
```

```
请输入信用卡号码：&994556
输入的卡号是错误的。
```

6、 总结和收获

（1） 学习运用 Scanner 类，使用其读入 String 类型变量

（2） 学习利用 Long.parseLong 完成 String 类型到 long 的转换

（3） 学习运用 try{}-catch 语法，利用其判断转换是否正确进行

（4） 更熟练的掌握截取字符，字符、整型的转化等操作

实验 2

1、 题目

创建一个日期工具类 DateUtil

2、 算法设计

DateUtil 内包含多个不同功能的方法：

（1）isLeapYear(int year)函数是判断给定的年份是否为闰年。一个年份是闰年的条件是：能被 4 整除但不能被 100 整除，或者能被

400 整除。因此，可以使用条件判断来判断给定的年份是否满足这两个条件，如果满足则返回 true，否则返回 false。

(2)isValidDate(int year, int month, int day)函数判断给定的年份、月份和日期是否构成一个有效的日期。根据题目要求，可以使用条件判断来判断给定的年份、月份和日期是否满足条件，如果满足则返回 true，否则返回 false。

(3)getDayOfWeek(int year, int month, int day)函数计算给定日期是星期几。根据题目要求，假设给定的日期是有效的，因此可以使用 Calendar 类来进行日期计算和操作。首先，使用 Calendar 类设置给定的年份、月份和日期。然后，使用 get 方法获取星期几的值，根据星期几的定义，星期日的值是 1，星期一的值是 2，以此类推。因此，可以直接返回获取到的星期几的值减 1，即可得到对应的星期几。

(4)printCalendar(int year, int month)函数打印指定年份和月份的日历。首先，使用 Calendar 类设置给定的年份和月份。然后，使用循环和条件判断来打印日历的表头、日期部分和空行，以满足日历的格式要求。

(5)printCalendar(int year)函数与 printCalendar(int year, int month)函数类似，只是省略了月份的参数，直接打印指定年份的整个日历。

(6)formatDate(int year, int month, int day)函数将给定的日期按照指定的格式进行格式化。根据题目要求，格式为"xxxday d mmm yyyy"，即星期几的缩写、日期、月份的缩写和年份。可以使用 Calendar 类进行日期的格式化，然后将格式化后的日期和其他信息拼接成字符串，

最后返回结果。

3、 主干代码说明

```
1. public static void printCalendar(int year, int month) {
2.         Calendar calendar = Calendar.getInstance();
3.         calendar.set(year, month - 1, 1);
4.
5.         SimpleDateFormat sdf = new SimpleDateFormat("MMMM y
   yyy");
6.         System.out.println(sdf.format(calendar.getTime()));
7.         System.out.println("Sun Mon Tue Wed Thu Fri Sat");
8.         int startDayOfWeek = calendar.get(Calendar.DAY_OF_W
   EEK) - 1;
9.         int maxDays = calendar.getActualMaximum(Calendar.DA
   Y_OF_MONTH);
10.        for (int i = 0; i < startDayOfWeek; i++) {
11.            System.out.print("    ");
12.        }
13.        for (int day = 1; day <= maxDays; day++) {
14.            System.out.printf("%3d ", day);
15.            if ((startDayOfWeek + day) % 7 == 0) {
16.                System.out.println();
17.            }
18.        }
19.        System.out.println();
20.    }
```

（1）定义了一个名为 printCalendar 的方法，接受两个参数：year 表示年份，month 表示月份。

（2）在方法内部，首先创建一个 Calendar 对象，用于进行日期计算和操作。然后，使用 set 方法设置 Calendar 对象的年份和月份，注意月份是从 0 开始计数的，所以需要将 month 减去 1。

（3）使用 getActualMaximum 方法获取指定月份的最大天数，并将其赋值给变量 maxDays。

（4）使用 set 方法将 Calendar 对象的日期设置为指定月份的第一天。这里将日期设置为 1，表示从该月的第一天开始打印。

（5）使用 get 方法获取指定日期是星期几，并将其赋值给变量 startDayOfWeek。注意，星期日的值是 1，星期一的值是 2，以此类推。

（6）打印日历的表头，即星期几的标识。使用一个循环，从星期日开始打印到星期六，使用 printf 方法打印，每个标识之间使用空格分隔。

（7）根据 startDayOfWeek 的值，打印出空格，以对齐日期的位置。使用一个循环，从 1 打印到 maxDays，表示该月的所有日期。在循环中，使用 printf 方法打印日期，并设置宽度为 3，以对齐日期的位置。每打印一个日期，判断是否需要换行，即当前日期是星期六，如果是，则打印一个换行符。

4、 运行结果展示

```
2000 is leap year? true
2004 is leap year? true
2100 is leap year? false
2023 is leap year? false
2023-2-29 is valid date? false
2024-2-29 is valid date? true
2023-4-31 is valid date? false
2023-12-31 is valid date? true
2023-10-8 is Sun
2023-10-9 is Mon
2023-10-10 is Tue
2023-10-11 is Wed
2023-10-12 is Thu
2023-10-13 is Fri
2023-10-14 is Sat
十月 2023
Sun Mon Tue Wed Thu Fri Sat
  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
一月 2023
Sun Mon Tue Wed Thu Fri Sat
  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
```

```
二月 2023
Sun Mon Tue Wed Thu Fri Sat
                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

三月 2023
Sun Mon Tue Wed Thu Fri Sat
                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

四月 2023
Sun Mon Tue Wed Thu Fri Sat
                        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
```

```
五月 2023
Sun Mon Tue Wed Thu Fri Sat
      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

六月 2023
Sun Mon Tue Wed Thu Fri Sat
                  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

七月 2023
Sun Mon Tue Wed Thu Fri Sat
                          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
```

```
八月 2023
Sun Mon Tue Wed Thu Fri Sat
                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

九月 2023
Sun Mon Tue Wed Thu Fri Sat
                        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


十月 2023
Sun Mon Tue Wed Thu Fri Sat
  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
```

```
十一月 2023
Sun Mon Tue Wed Thu Fri Sat
                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

十二月 2023
Sun Mon Tue Wed Thu Fri Sat
                        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

星期日 1 1月 2023
星期二 14 2月 2023
星期一 25 12月 2023
```

5、 总结和收获

(1) 学习接触了 Calendar 类的相关知识并加以利用

(2) 接触了将许多函数合为一类的思想

(3) 学习了利用另一个程序来测试结果

实验 3

1、 题目

模式化打印图形

2、 数据设计

（1） char 类型变量 graphType：用于存储打印图形的类型

（2） int 类型变量 graphSize：用于存储打印图形的大小

（3） int 型变量 i、j、k 等：用于循环的进行与终止

3、 算法设计

主函数：输入 graphType 与 graphSize→利用 switch 语句判断打印类型→调用不同函数打印图形

CreateA、CreateB、CreateC、CreateD、CreateE 等函数：（使用多重循环语句，借用横纵坐标 i、j）

首行与尾行均输出全部符号→根据不同图形分析 i 与 j 的关系→写出内循环

4、 主干代码说明

```
1、switch(graphType){
2、//判断打印图形的种类
3、            case 'a':
4、                CreateA(graphSize);//以图形大小为函数参数
5、                break;
6、            case 'b':
7、                CreateB(graphSize);
8、                break;
9、            case 'c':
10、                CreateC(graphSize);
11、                break;
12、            case 'd':
13、                CreateD(graphSize);
14、                break;
15、            case 'e':
16、                CreateE(graphSize);//用不同函数处理，打印不同图形
17、            }
```

下面以 CreateE 函数为例说明打印函数的构造

```
1.     public static void CreateE(int SOG){
2.         for(int i=0;i<SOG;i++){
```

```
            //保证行数，i 为纵坐标
            if(i==0||i==SOG-1){
                //首尾行均全部输出
                for(int j=0;j<SOG;j++){
                    System.out.print('#');
                }
                //输出完一行要换行
                System.out.print('\n');
            }
            else{
                for(int j=0;j<SOG;j++){
                    //内循环判断 i、j 关系
                    if(i+j==SOG-1||i==j||j==0||j==SOG-1)
                        //分别为左对角线、右对角线、最左列、最右列
                        System.out.print('#');
                    else
                        System.out.print(' ');
                }
                //换行
                System.out.print('\n');
            }
        }
    }
```

## 5、 运行结果展示

```
请输入一个'a~b'的字母与一个非负整数:
e 10
##########
##      ##
# #    # #
#  #  #  #
#   ##   #
#   ##   #
#  #  #  #
# #    # #
##      ##
##########

Process finished with exit code 0
```

6、 总结和收获

(1) 更加熟悉对 switch 语句的运用

(2) 更加熟练判断内外循环中变量的关系

1、 获

实验 4

1、 题目

摩斯密码

2、 数据设计

(1) 定义了 file 和 scanner 对象用于读取文件

(2) 定义 result 对象以存储解密后的密码

(3) String 类型变量 morseCode：存储截取出的摩斯密码

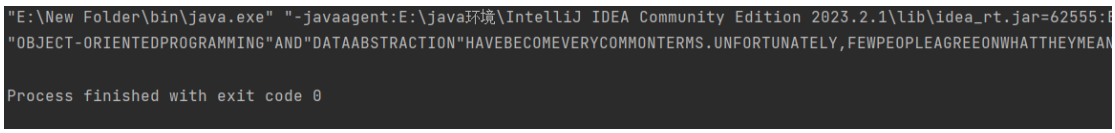
3、 算法设计

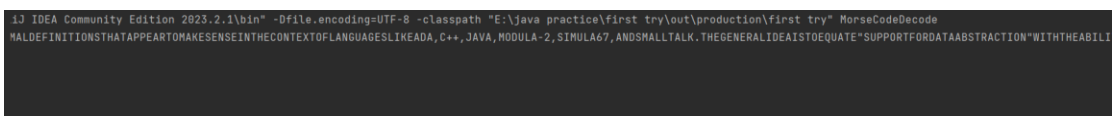
首先定义摩斯密码字典→读入并利用 try-catch 语句判断是否正确读入文件→定义空字符 result→判断文件是否读完→截取一个字母对应的摩斯密码存储在 morseCode 中→利用字典将 morseCode 对应字母加入 result→将 result 转为字符串类型并输出

4、 主干代码说明

```
try {
//读取文件
            File file = new File("E:\\java practice\\first try\\src\\encode(1).txt");
            Scanner scanner = new Scanner(file);
            // 解密摩斯密码
            StringBuilder result = new StringBuilder();
//判断文件是否读完
            while (scanner.hasNext()) {
//截取一个字母对应的摩斯密码
                String morseCode = scanner.next();
//将对应字母加入 result
                result.append(morseCodeMap.get(morseCode));
                }
            // 输出解密结果
            System.out.println(result.toString());
        } catch (FileNotFoundException e) {
//未成功读取文件则输出 not found
            System.out.println("File not found.");
        }
```

5、 运行结果展示




6、 总结和收获

（1） 学习了使用 Hashmap 建立字典和映射关系

（2） 学习了使用 Scanner 类定义对象实现从文件中读取文本数据

（3） 对 Scanner 类的操作更加熟悉，学习了部分其中的方法

实验 5

1、 题目

蒙特卡洛方法模拟

2、 数据设计

（1） 定义 reader 对象用于读取数据

（2） 定义 random 对象用于生成随机数

（3） double 类型变量 x、y：代表随机点的横、纵坐标

（4） int 类型变量 n、m：代表随机点的总数和在阴影部分的点数量

3、 算法设计

读入 n→定义并初始化阴影面积内的点数量 m→开始 n 次循环→每次循环生成一个随机点，x、y 为其横、纵坐标→判断随机点是否在阴影部分→是则 m+1→循环结束后根据面积及模拟关系计算 pi

4、 主干代码说明

5、 运行结果展示

```
1. //定义 n 次循环
2. for(int i=0;i<n;i++){
3. //生成随机点
```

```
4.              double x= random.nextDouble();
5.              double y= random.nextDouble();
6. //判断随机点是否在阴影部分
7.              if(x*x+y*y<=1){
8.                  m++;//增加m
9.              }
10.         }
11.//计算pi
12.         double probablePi=4.0*m/n;
```

```
5000
结果是3.1184

Process finished with exit code 0
```

```
"E:\New Folder\bin\java.exe" "-javaagent:E
50000
结果是3.14776

Process finished with exit code 0
```

7、 总结和收获

随着 n 的增大，pi 会越来越接近其科学值

学习使用了 Random 类，了解了一定的生成随机数的方法

实验 6

1、 题目

一只兔子的行走

2、 数据设计

（1）int 类型变量 n：用于存储步数

（2）int 类型变量 direction，用以确定兔子走的方向

（3）double 类型变量 distance：用于存储距离

3、 算法设计

读入步数→初始化兔子起始坐标和原点距离→定义 n 次循环→生成整数随机数 direction（0~3）以确定兔子行走方向→模拟兔子移动，改变横纵坐标→计算距离原点距离→循环结束，输出最后一次行走离远点的距离

（加上 trials 后只需加和每次的最远距离并除以试验次数）

4、 主干代码说明

```
1.          for (int i = 0; i < n; i++) {
2.              // 随机选择一个方向
3.              int direction = random.nextInt(4);
4.              // 根据方向更新坐标
5.              switch (direction) {
6.                  case 0: // 北
7.                      y++;
8.                      break;
9.                  case 1: // 南
10.                     y--;
11.                     break;
12.                 case 2: // 西
13.                     x--;
14.                     break;
15.                 case 3: // 东
16.                     x++;
17.                     break;
18.             }
19.             // 计算欧几里得距离
20.             distance = Math.sqrt(x * x + y * y);
21.             // 输出当前步的坐标
22.             System.out.println("第 " + (i + 1) + " 步的坐标为：(" + x + ", " + y + ")");
23.         }
```

5、 运行结果展示

```
PS E:\java practice\first try\src> javac -encoding UTF-8 RandomWalkers.java
PS E:\java practice\first try\src> java RandomWalkers 10 1000
完成 1000 次实验之后的欧几里得距离的平均值为：2.8798893921977804
```

6、 总结和收获

随着 n 增大，距离也越来越大但会到达一定限度

（1） 学习使用命令行输入参数

（2） 学会用随机数法模拟运动方向

附录：每个题的源代码

一：

```
1. import java.util.Scanner;
2. public class CreditCardValidator {
3.     public static void main(String[] args) {
4.         Scanner scanner = new Scanner(System.in);
5.         System.out.print("请输入信用卡号码：");
6.         String input=scanner.nextLine();
7.         boolean isValid = isValidCardNumber(input);
8.         if (isValid) {
9.             System.out.println("输入的卡号是正确的。");
10.        } else {
11.            System.out.println("输入的卡号是错误的。");
12.        }
13.    }
14.    public static boolean isValidCardNumber(String input) {
15.        try{//将 String 转为 Long 并判断是否转换成功
16.            long cardNumber=Long.parseLong(input);
17.        // 将卡号转换为字符串
18.        String cardNumberStr = Long.toString(cardNumber);
19.        // 检查卡号长度是否在 13-16 位之间
20.        if (cardNumberStr.length() < 13 || cardNumberStr.le
   ngth() > 16) {
21.            return false;
22.        }
23.        // 从右到左遍历卡号的每一位数字
24.        int sum = 0;
25.        boolean doubleDigit = false;
26.        for (int i = cardNumberStr.length() - 1; i >= 0; i-
   -) {
27.            int digit = Character.getNumericValue(cardNumbe
   rStr.charAt(i));
28.            // 如果是偶数位，则将数字翻倍
29.            if (doubleDigit) {
30.                digit *= 2;
```

```
                // 如果翻倍后的数字是两位数，则将两个数字相加
                if (digit > 9) {
                    digit = digit % 10 + digit / 10;
                }
            }
            sum += digit;
            doubleDigit = !doubleDigit;
        }
        // 检查校验和是否能被 10 整除
        return sum % 10 == 0;}
        catch(NumberFormatException e){
            return false;
        }
    }
}
```

二：

```
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
public class DateUtil {
    public static boolean isLeapYear(int year) {
    return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}
    public static boolean isValidDate(int year, int month, int day) {
        if (year < 1 || year > 9999 || month < 1 || month > 12 || day < 1) {
            return false;
        }
        int maxDays = 31;
        if (month == 4 || month == 6 || month == 9 || month == 11) {
            maxDays = 30;
        } else if (month == 2) {
            maxDays = isLeapYear(year) ? 29 : 28;
        }
        return day <= maxDays;
    }
    public static int getDayOfWeek(int year, int month, int day) {
        Calendar calendar = Calendar.getInstance();
        calendar.set(year, month - 1, day);
        return calendar.get(Calendar.DAY_OF_WEEK) - 1;
```

```
    }
    public static void printCalendar(int year, int month) {
        Calendar calendar = Calendar.getInstance();
        calendar.set(year, month - 1, 1);

        SimpleDateFormat sdf = new SimpleDateFormat("MMMM yyyy");
        System.out.println(sdf.format(calendar.getTime()));
        System.out.println("Sun Mon Tue Wed Thu Fri Sat");
        int startDayOfWeek = calendar.get(Calendar.DAY_OF_WEEK) - 1;
        int maxDays = calendar.getActualMaximum(Calendar.DAY_OF_MONTH);
        for (int i = 0; i < startDayOfWeek; i++) {
            System.out.print("    ");
        }
        for (int day = 1; day <= maxDays; day++) {
            System.out.printf("%3d ", day);
            if ((startDayOfWeek + day) % 7 == 0) {
                System.out.println();
            }
        }
        System.out.println();
    }
    public static void printCalendar(int year) {
        for (int month = 1; month <= 12; month++) {
            printCalendar(year, month);
            System.out.println();
        }
    }
    public static String formatDate(int year, int month, int day) {
        Calendar calendar = Calendar.getInstance();
        calendar.set(year, month - 1, day);

        SimpleDateFormat sdf = new SimpleDateFormat("EEEE d MMM yyyy");
        return sdf.format(calendar.getTime());
    }
}
```

```
public class TestDateUtil {
    public static void main(String[] args) {
    testIsLeapYear();
    testIsValidDate();
    testGetDayOfWeek();
```

```
6.      testPrintCalendar();
7.      testFormatDate();
8.  }
9.      public static void testIsLeapYear() {
10.         int[] years = {2000, 2004, 2100, 2023};
11.
12.         for (int year : years) {
13.             boolean isLeap = DateUtil.isLeapYear(year);
14.             System.out.println(year + " is leap year? " + isLeap)
    ;
15.         }
16.     }
17.     public static void testIsValidDate() {
18.         int[][] dates = {
19.                 {2023, 2, 29},
20.                 {2024, 2, 29},
21.                 {2023, 4, 31},
22.                 {2023, 12, 31}
23.         };
24.         for (int[] date : dates) {
25.             boolean isValid = DateUtil.isValidDate(date[0], date[
    1], date[2]);
26.             System.out.println(date[0] + "-" + date[1] + "-
    " + date[2] + " is valid date? " + isValid);
27.         }
28.     }
29.     public static void testGetDayOfWeek() {
30.         int[][] dates = {
31.                 {2023, 10, 8},
32.                 {2023, 10, 9},
33.                 {2023, 10, 10},
34.                 {2023, 10, 11},
35.                 {2023, 10, 12},
36.                 {2023, 10, 13},
37.                 {2023, 10, 14}
38.         };
39.
40.         String[] daysOfWeek = {"Sun", "Mon", "Tue", "Wed", "Thu",
     "Fri", "Sat"};
41.
42.         for (int[] date : dates) {
43.             int dayOfWeek = DateUtil.getDayOfWeek(date[0], date[1
    ], date[2]);
44.             System.out.println(date[0] + "-" + date[1] + "-
    " + date[2] + " is " + daysOfWeek[dayOfWeek]);
```

```
        }
    }
    public static void testPrintCalendar() {
        DateUtil.printCalendar(2023, 10);
        DateUtil.printCalendar(2023);
    }

    public static void testFormatDate() {
        int[][] dates = {
                {2023, 1, 1},
                {2023, 2, 14},
                {2023, 12, 25}
        };

        for (int[] date : dates) {
            String formattedDate = DateUtil.formatDate(date[0], date[1], date[2]);
            System.out.println(formattedDate);
        }
    }
}
```

三：

```
import java.util.Scanner;

public class GraphPrinter {
    public static void main(String[] args){
        System.out.println("请输入一个‘a~b’的字母与一个非负整数: ");
        Scanner reader=new Scanner(System.in);
        char graphType=reader.next().charAt(0);
        int graphSize=reader.nextInt();
        switch(graphType){
            case 'a':
                CreateA(graphSize);
                break;
            case 'b':
                CreateB(graphSize);
                break;
            case 'c':
                CreateC(graphSize);
                break;
            case 'd':
                CreateD(graphSize);
                break;
```

```
            case 'e':
                CreateE(graphSize);
        }
    }
    public static void CreateA(int SOG){
        for(int i=0;i<SOG;i++){
            if(i==0||i==SOG-1){
                for(int j=0;j<SOG;j++){
                    System.out.print('#');
                }
                System.out.print('\n');
            }
            else{
                for(int j=0;j<SOG;j++){
                    if(j==0||j==SOG-1)
                        System.out.print('#');
                    else
                        System.out.print(' ');
                }

                System.out.print('\n');
            }
        }
    }
    public static void CreateB(int SOG){
        for(int i=0;i<SOG;i++){
            if(i==0||i==SOG-1){
                for(int j=0;j<SOG;j++){
                    System.out.print('#');
                }
                System.out.print('\n');
            }
            else{
               for(int j=0;j<SOG;j++){
                   if(j==i)
                       System.out.print('#');
                   else
                       System.out.print(' ');
               }
               System.out.print('\n');
            }
        }
    }
    public static void CreateC(int SOG){
        for(int i=0;i<SOG;i++){
```

```
67.             if(i==0||i==SOG-1){
68.                 for(int j=0;j<SOG;j++){
69.                     System.out.print('#');
70.                 }
71.                 System.out.print('\n');
72.             }
73.             else{
74.                 for(int j=0;j<SOG;j++){
75.                     if(i+j==SOG-1)
76.                         System.out.print('#');
77.                     else
78.                         System.out.print(' ');
79.                 }
80.                 System.out.print('\n');
81.             }
82.         }
83.     }
84.     public static void CreateD(int SOG){
85.         for(int i=0;i<SOG;i++){
86.             if(i==0||i==SOG-1){
87.                 for(int j=0;j<SOG;j++){
88.                     System.out.print('#');
89.                 }
90.                 System.out.print('\n');
91.             }
92.             else{
93.                 for(int j=0;j<SOG;j++){
94.                     if(i+j==SOG-1||i==j)
95.                         System.out.print('#');
96.                     else
97.                         System.out.print(' ');
98.                 }
99.                 System.out.print('\n');
100.            }
101.        }
102.    }
103.    public static void CreateE(int SOG){
104.        for(int i=0;i<SOG;i++){
105.            if(i==0||i==SOG-1){
106.                for(int j=0;j<SOG;j++){
107.                    System.out.print('#');
108.                }
109.                System.out.print('\n');
110.            }
111.            else{
```

```
                    for(int j=0;j<SOG;j++){
                        if(i+j==SOG-1||i==j||j==0||j==SOG-1)
                            System.out.print('#');
                        else
                            System.out.print(' ');
                    }
                    System.out.print('\n');
                }
            }
        }

    }
```

四:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.HashMap;
import java.util.Scanner;
public class MorseCodeDecode {
    public static void main(String[] args) {
        // 定义摩斯密码字典
        HashMap<String, Character> morseCodeMap = new HashMap<>();
        morseCodeMap.put(".-", 'A');
        morseCodeMap.put("-...", 'B');
        morseCodeMap.put("-.-.", 'C');
        morseCodeMap.put("-..", 'D');
        morseCodeMap.put(".", 'E');
        morseCodeMap.put("..-.", 'F');
        morseCodeMap.put("--.", 'G');
        morseCodeMap.put("....", 'H');
        morseCodeMap.put("..", 'I');
        morseCodeMap.put(".---", 'J');
        morseCodeMap.put("-.-", 'K');
        morseCodeMap.put(".-..", 'L');
        morseCodeMap.put("--", 'M');
        morseCodeMap.put("-.", 'N');
        morseCodeMap.put("---", 'O');
        morseCodeMap.put(".--.", 'P');
        morseCodeMap.put("--.-", 'Q');
        morseCodeMap.put(".-.", 'R');
        morseCodeMap.put("...", 'S');
        morseCodeMap.put("-", 'T');
        morseCodeMap.put("..-", 'U');
        morseCodeMap.put("...-", 'V');
```

```
31.         morseCodeMap.put(".--", 'W');
32.         morseCodeMap.put("-..-", 'X');
33.         morseCodeMap.put("-.--", 'Y');
34.         morseCodeMap.put("--..", 'Z');
35.         morseCodeMap.put("-----", '0');
36.         morseCodeMap.put(".----", '1');
37.         morseCodeMap.put("..---", '2');
38.         morseCodeMap.put("...--", '3');
39.         morseCodeMap.put("....-", '4');
40.         morseCodeMap.put(".....", '5');
41.         morseCodeMap.put("-....", '6');
42.         morseCodeMap.put("--...", '7');
43.         morseCodeMap.put("---..", '8');
44.         morseCodeMap.put("----.", '9');
45.         morseCodeMap.put(".-.-.-", '.');
46.         morseCodeMap.put("--..--", ',');
47.         morseCodeMap.put("..--..", '?');
48.         morseCodeMap.put(".----.", '\'');
49.         morseCodeMap.put("-.-.--", '!');
50.         morseCodeMap.put("-..-.", '/');
51.         morseCodeMap.put("-.--.", '(');
52.         morseCodeMap.put("-.--.-", ')');
53.         morseCodeMap.put(".-...", '&');
54.         morseCodeMap.put("---...", ':');
55.         morseCodeMap.put("-.-.-.", ';');
56.         morseCodeMap.put("-...-", '=');
57.         morseCodeMap.put(".-.-.", '+');
58.         morseCodeMap.put("-....-", '-');
59.         morseCodeMap.put("..--.-", '_');
60.         morseCodeMap.put(".-..-.", '"');
61.         morseCodeMap.put(".--.-.", '@');
62.         // 读取文件
63.         try {
64.             File file = new File("E:\\java practice\\first
    try\\src\\encode(1).txt");
65.             Scanner scanner = new Scanner(file);
66.             // 解密摩斯密码
67.             StringBuilder result = new StringBuilder();
68.             while (scanner.hasNext()) {
69.                 String morseCode = scanner.next();
70.                 result.append(morseCodeMap.get(morseCode));
71.                 }
72.             // 输出解密结果
73.             System.out.println(result.toString());
74.         } catch (FileNotFoundException e) {
```

```
            System.out.println("File not found.");
        }
    }
}
```

五：

```
import java.util.Random;
import java.util.Scanner;
public class PIByMonte{
    public static void main(String args[]){
        Scanner reader=new Scanner(System.in);
        Random random=new Random();
        int n=reader.nextInt();
        int m=0;
        for(int i=0;i<n;i++){
            double x= random.nextDouble();
            double y= random.nextDouble();
            if(x*x+y*y<=1){
                m++;
            }
        }
        double probablePi=4.0*m/n;
        System.out.println("结果是"+probablePi);
    }
}
```

六：

```
import java.util.Random;
public class RandomWalker {
    public static void main(String[] args) {
        // 检查命令行参数数量是否正确
        if (args.length != 1) {
            System.out.println("请提供一个整数参数:");
            return;
        }
        // 获取步数参数
        int n = Integer.parseInt(args[0]);
        // 初始化起始坐标和欧几里得距离
        int x = 0;
        int y = 0;
        double distance = 0.0;
        // 创建随机数生成器
        Random random = new Random();
        // 模拟 n 步随机游走
        for (int i = 0; i < n; i++) {
```

```
            // 随机选择一个方向
            int direction = random.nextInt(4);
            // 根据方向更新坐标
            switch (direction) {
                case 0: // 北
                    y++;
                    break;
                case 1: // 南
                    y--;
                    break;
                case 2: // 西
                    x--;
                    break;
                case 3: // 东
                    x++;
                    break;
            }
            // 计算欧几里得距离
            distance = Math.sqrt(x * x + y * y);
            // 输出当前步的坐标
            System.out.println("第 " + (i + 1) + " 步的坐标为: (" + x + ", " + y + ")");
        }
        // 输出最终位置和起始位置的欧几里得距离
        System.out.println("最终位置和起始位置的欧几里得距离为: " + distance);
    }
}
```

```
import java.util.Random;
public class RandomWalkers {
    public static void main(String[] args) {
        // 检查命令行参数数量是否正确
        if (args.length != 2) {
            System.out.println("请提供两个整数参数。");
            return;
        }
        // 获取步数参数和实验次数参数
        int n = Integer.parseInt(args[0]);
        int trials = Integer.parseInt(args[1]);
        // 初始化总距离
        double totalDistance = 0.0;
        // 创建随机数生成器
        Random random = new Random();
        // 进行 trials 次实验
        for (int i = 0; i < trials; i++) {
```

```
18.             // 初始化起始坐标和欧几里得距离
19.             int x = 0;
20.             int y = 0;
21.             double distance = 0.0;
22.             // 模拟 n 步随机游走
23.             for (int j = 0; j < n; j++) {
24.                 // 随机选择一个方向
25.                 int direction = random.nextInt(4);
26.
27.                 // 根据方向更新坐标
28.                 switch (direction) {
29.                     case 0: // 北
30.                         y++;
31.                         break;
32.                     case 1: // 南
33.                         y--;
34.                         break;
35.                     case 2: // 西
36.                         x--;
37.                         break;
38.                     case 3: // 东
39.                         x++;
40.                         break;
41.                 }
42.
43.                 // 计算欧几里得距离
44.                 distance = Math.sqrt(x * x + y * y);
45.             }
46.             // 累加距离
47.             totalDistance += distance;
48.         }
49.         // 计算平均距离
50.         double averageDistance = totalDistance / trials;
51.         // 输出平均距离
52.         System.out.println("完成 " + trials + " 次实验之后的
    欧几里得距离的平均值为: " + averageDistance);
53.     }
54.}
```