|  |  |
| --- | --- |
| **面向对象程序设计 作业报告** | |
| 第 一 次 | |
| 图片包含 标牌  已生成极高可信度的说明 | |
|  | |
|  | |
|  | |
| **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

实验1

1. 题目

信用卡号码的校验

1. 数据设计
2. String类型变量input：以字符串形式存储读入的卡号
3. 布尔类型变量isValid： 用于存储判断卡号是否正确的结果
4. Long类型变量cardNumber：定义在函数isValidCardNumber中，将卡号从字符串类型转换为长整型
5. String类型变量cardNumberStr：定义在判断函数中，用于将可成功转换的卡号转回字符串形式
6. Int类型变量sum：用于检验卡号是否正确
7. I、digit、doubleDigit等：由检验步骤需要所设置的用于循环、加和、判断等的变量
8. 算法设计

主函数：读入卡号存储在input中🡪交由isValidCardNumber函数判断，将结果存储在isValid中🡪根据isValid输出结果

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

1. 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.length() > 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(cardNumberStr.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. }
30. 运行结果展示





1. 总结和收获
2. 学习运用Scanner类，使用其读入String类型变量
3. 学习利用Long.parseLong完成String类型到long的转换
4. 学习运用try{}-catch语法，利用其判断转换是否正确进行
5. 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

1. 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。

1. 主干代码说明
2. public static void printCalendar(int year, int month) {
3. Calendar calendar = Calendar.getInstance();
4. calendar.set(year, month - 1, 1);
5. SimpleDateFormat sdf = new SimpleDateFormat("MMMM yyyy");
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\_WEEK) - 1;
9. int maxDays = calendar.getActualMaximum(Calendar.DAY\_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。

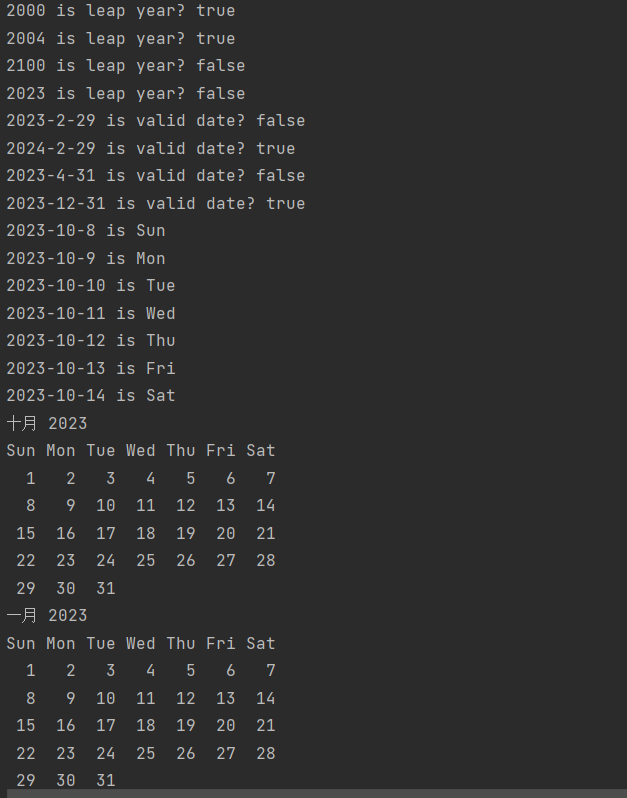
（4）使用set方法将Calendar对象的日期设置为指定月份的第一天。这里将日期设置为1，表示从该月的第一天开始打印。

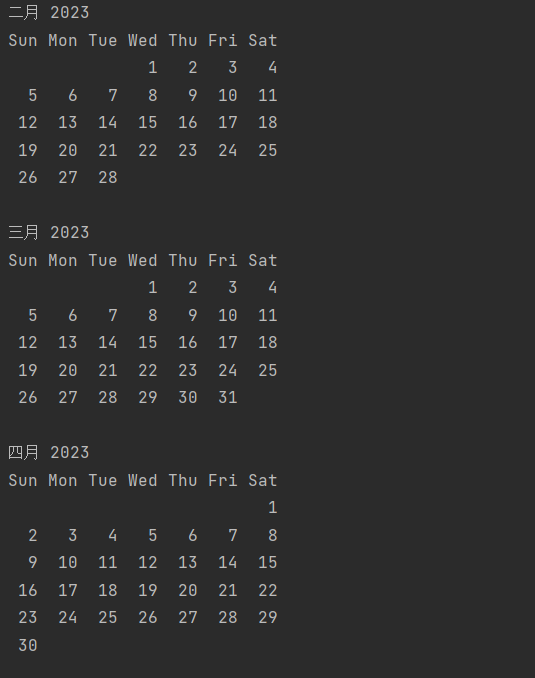
（5）使用get方法获取指定日期是星期几，并将其赋值给变量startDayOfWeek。注意，星期日的值是1，星期一的值是2，以此类推。

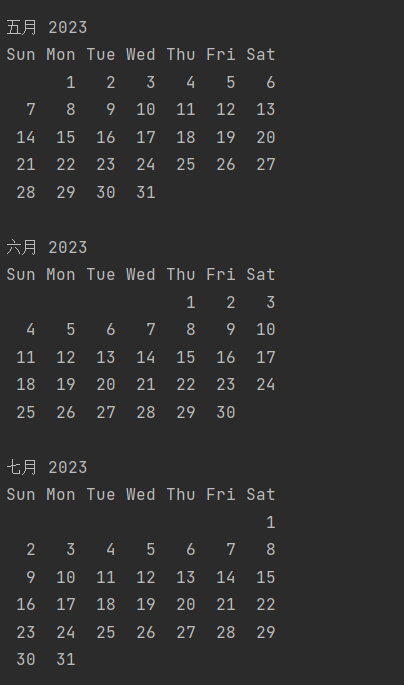
（6）打印日历的表头，即星期几的标识。使用一个循环，从星期日开始打印到星期六，使用printf方法打印，每个标识之间使用空格分隔。

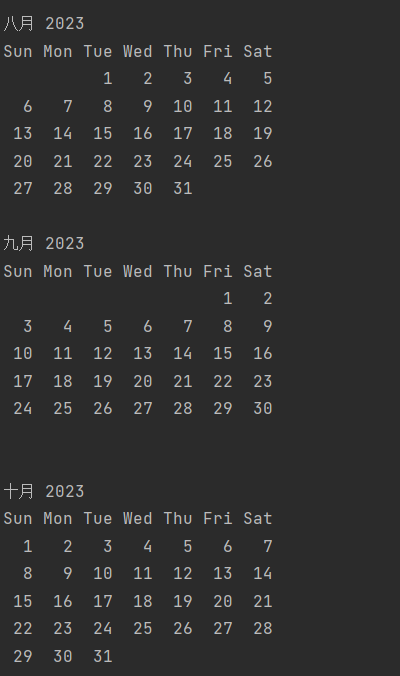
（7）根据startDayOfWeek的值，打印出空格，以对齐日期的位置。使用一个循环，从1打印到maxDays，表示该月的所有日期。在循环中，使用printf方法打印日期，并设置宽度为3，以对齐日期的位置。每打印一个日期，判断是否需要换行，即当前日期是星期六，如果是，则打印一个换行符。

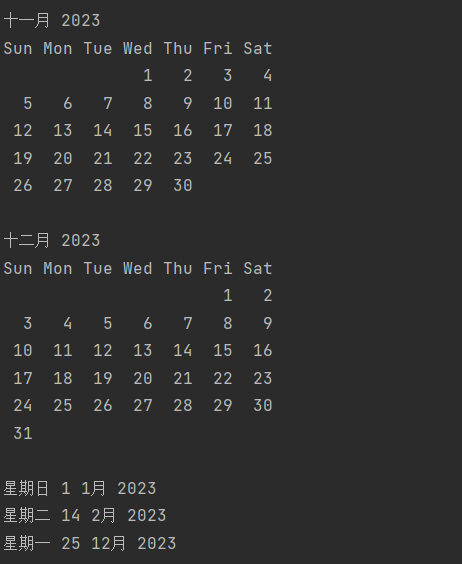
1. 运行结果展示











1. 总结和收获
2. 学习接触了Calendar类的相关知识并加以利用
3. 接触了将许多函数合为一类的思想
4. 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

1. 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
2. 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：

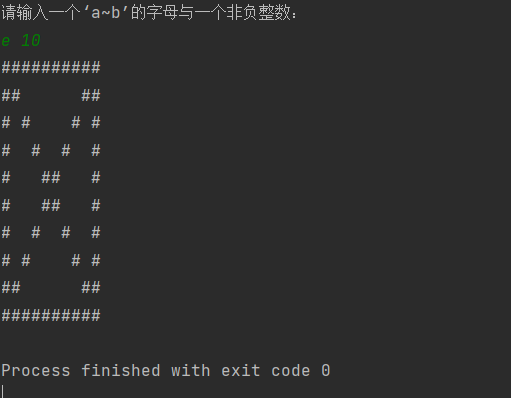
（使用多重循环语句，借用横纵坐标i、j）

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

1. 主干代码说明
2. switch(graphType){
3. *//判断打印图形的种类*
4. case 'a':
5. CreateA(graphSize);*//以图形大小为函数参数*
6. break;
7. case 'b':
8. CreateB(graphSize);
9. break;
10. case 'c':
11. CreateC(graphSize);
12. break;
13. case 'd':
14. CreateD(graphSize);
15. break;
16. case 'e':
17. CreateE(graphSize);*//用不同函数处理，打印不同图形*
18. }

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++){
3. *//保证行数，i为纵坐标*
4. if(i==0||i==SOG-1){
5. *//首尾行均全部输出*
6. for(int j=0;j<SOG;j++){
7. System.out.print('#');
8. }
9. *//输出完一行要换行*
10. System.out.print('\n');
11. }
12. else{
13. for(int j=0;j<SOG;j++){
14. *//内循环判断i、j关系*
15. if(i+j==SOG-1||i==j||j==0||j==SOG-1)
16. *//分别为左对角线、右对角线、最左列、最右列*
17. System.out.print('#');
18. else
19. System.out.print(' ');
20. }
21. *//换行*
22. System.out.print('\n');
23. }
24. }
25. }
26. 运行结果展示



1. 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
2. 获

实验4

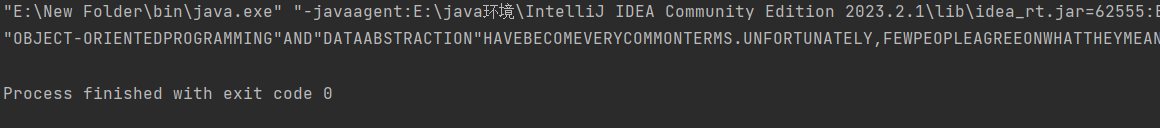
1. 题目

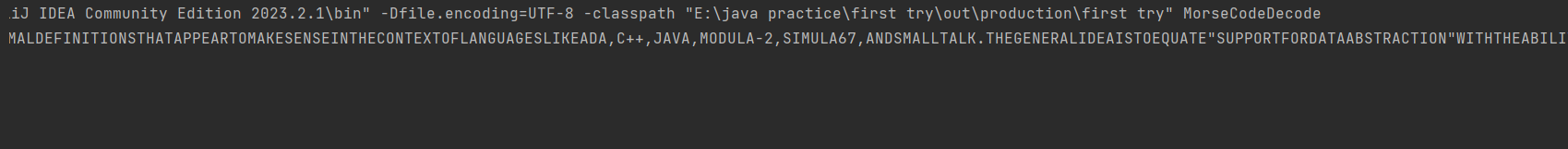
摩斯密码

1. 数据设计
2. 定义了file和scanner对象用于读取文件
3. 定义result对象以存储解密后的密码
4. String类型变量morseCode：存储截取出的摩斯密码
5. 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

1. 主干代码说明
2. try {
3. *//读取文件*
4. File file = new File("E:\\java practice\\first try\\src\\encode(1).txt");
5. Scanner scanner = new Scanner(file);
6. *// 解密摩斯密码*
7. StringBuilder result = new StringBuilder();
8. *//判断文件是否读完*
9. while (scanner.hasNext()) {
10. *//截取一个字母对应的摩斯密码*
11. String morseCode = scanner.next();
12. *//将对应字母加入result*
13. result.append(morseCodeMap.get(morseCode));
14. }
15. *// 输出解密结果*
16. System.out.println(result.toString());
17. } catch (FileNotFoundException e) {
18. *//未成功读取文件则输出not found*
19. System.out.println("File not found.");
20. }
21. 运行结果展示





1. 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

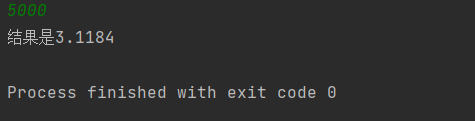
1. 题目

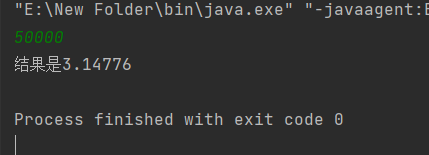
蒙特卡洛方法模拟

1. 数据设计
2. 定义reader对象用于读取数据
3. 定义random对象用于生成随机数
4. double类型变量x、y：代表随机点的横、纵坐标
5. 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
6. 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

1. 主干代码说明
2. 运行结果展示
3. *//定义n次循环*
4. for(int i=0;i<n;i++){
5. *//生成随机点*
6. double x= random.nextDouble();
7. double y= random.nextDouble();
8. *//判断随机点是否在阴影部分*
9. if(x\*x+y\*y<=1){
10. m++;*//增加m*
11. }
12. }
13. *//计算pi*
14. double probablePi=4.0\*m/n;





1. 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

1. 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

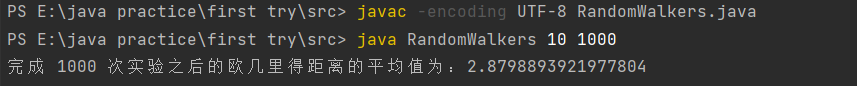
（3）double类型变量distance：用于存储距离

1. 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）

1. 主干代码说明
2. for (int i = 0; i < n; i++) {
3. *// 随机选择一个方向*
4. int direction = random.nextInt(4);
5. *// 根据方向更新坐标*
6. switch (direction) {
7. case 0: *// 北*
8. y++;
9. break;
10. case 1: *// 南*
11. y--;
12. break;
13. case 2: *// 西*
14. x--;
15. break;
16. case 3: *// 东*
17. x++;
18. break;
19. }
20. *// 计算欧几里得距离*
21. distance = Math.sqrt(x \* x + y \* y);
22. *// 输出当前步的坐标*
23. System.out.println("第 " + (i + 1) + " 步的坐标为：(" + x + ", " + y + ")");
24. }
25. 运行结果展示



1. 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.length() > 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(cardNumberStr.charAt(i));
28. *// 如果是偶数位，则将数字翻倍*
29. if (doubleDigit) {
30. digit \*= 2;
31. *// 如果翻倍后的数字是两位数，则将两个数字相加*
32. if (digit > 9) {
33. digit = digit % 10 + digit / 10;
34. }
35. }
36. sum += digit;
37. doubleDigit = !doubleDigit;
38. }
39. *// 检查校验和是否能被10整除*
40. return sum % 10 == 0;}
41. catch(NumberFormatException e){
42. return false;
43. }
44. }
45. }

二：

1. import java.text.SimpleDateFormat;
2. import java.util.Calendar;
3. import java.util.Date;
4. public class DateUtil {
5. public static boolean isLeapYear(int year) {
6. return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
7. }
8. public static boolean isValidDate(int year, int month, int day) {
9. if (year < 1 || year > 9999 || month < 1 || month > 12 || day < 1) {
10. return false;
11. }
12. int maxDays = 31;
13. if (month == 4 || month == 6 || month == 9 || month == 11) {
14. maxDays = 30;
15. } else if (month == 2) {
16. maxDays = isLeapYear(year) ? 29 : 28;
17. }
18. return day <= maxDays;
19. }
20. public static int getDayOfWeek(int year, int month, int day) {
21. Calendar calendar = Calendar.getInstance();
22. calendar.set(year, month - 1, day);
23. return calendar.get(Calendar.DAY\_OF\_WEEK) - 1;
24. }
25. public static void printCalendar(int year, int month) {
26. Calendar calendar = Calendar.getInstance();
27. calendar.set(year, month - 1, 1);
28. SimpleDateFormat sdf = new SimpleDateFormat("MMMM yyyy");
29. System.out.println(sdf.format(calendar.getTime()));
30. System.out.println("Sun Mon Tue Wed Thu Fri Sat");
31. int startDayOfWeek = calendar.get(Calendar.DAY\_OF\_WEEK) - 1;
32. int maxDays = calendar.getActualMaximum(Calendar.DAY\_OF\_MONTH);
33. for (int i = 0; i < startDayOfWeek; i++) {
34. System.out.print("    ");
35. }
36. for (int day = 1; day <= maxDays; day++) {
37. System.out.printf("%3d ", day);
38. if ((startDayOfWeek + day) % 7 == 0) {
39. System.out.println();
40. }
41. }
42. System.out.println();
43. }
44. public static void printCalendar(int year) {
45. for (int month = 1; month <= 12; month++) {
46. printCalendar(year, month);
47. System.out.println();
48. }
49. }
50. public static String formatDate(int year, int month, int day) {
51. Calendar calendar = Calendar.getInstance();
52. calendar.set(year, month - 1, day);
53. SimpleDateFormat sdf = new SimpleDateFormat("EEEE d MMM yyyy");
54. return sdf.format(calendar.getTime());
55. }
56. }
57. public class TestDateUtil {
58. public static void main(String[] args) {
59. testIsLeapYear();
60. testIsValidDate();
61. testGetDayOfWeek();
62. testPrintCalendar();
63. testFormatDate();
64. }
65. public static void testIsLeapYear() {
66. int[] years = {2000, 2004, 2100, 2023};
67. for (int year : years) {
68. boolean isLeap = DateUtil.isLeapYear(year);
69. System.out.println(year + " is leap year? " + isLeap);
70. }
71. }
72. public static void testIsValidDate() {
73. int[][] dates = {
74. {2023, 2, 29},
75. {2024, 2, 29},
76. {2023, 4, 31},
77. {2023, 12, 31}
78. };
79. for (int[] date : dates) {
80. boolean isValid = DateUtil.isValidDate(date[0], date[1], date[2]);
81. System.out.println(date[0] + "-" + date[1] + "-" + date[2] + " is valid date? " + isValid);
82. }
83. }
84. public static void testGetDayOfWeek() {
85. int[][] dates = {
86. {2023, 10, 8},
87. {2023, 10, 9},
88. {2023, 10, 10},
89. {2023, 10, 11},
90. {2023, 10, 12},
91. {2023, 10, 13},
92. {2023, 10, 14}
93. };
94. String[] daysOfWeek = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};
95. for (int[] date : dates) {
96. int dayOfWeek = DateUtil.getDayOfWeek(date[0], date[1], date[2]);
97. System.out.println(date[0] + "-" + date[1] + "-" + date[2] + " is " + daysOfWeek[dayOfWeek]);
98. }
99. }
100. public static void testPrintCalendar() {
101. DateUtil.printCalendar(2023, 10);
102. DateUtil.printCalendar(2023);
103. }
104. public static void testFormatDate() {
105. int[][] dates = {
106. {2023, 1, 1},
107. {2023, 2, 14},
108. {2023, 12, 25}
109. };
110. for (int[] date : dates) {
111. String formattedDate = DateUtil.formatDate(date[0], date[1], date[2]);
112. System.out.println(formattedDate);
113. }
114. }
115. }

三：

1. import java.util.Scanner;
2. public class GraphPrinter {
3. public static void main(String[] args){
4. System.out.println("请输入一个‘a~b’的字母与一个非负整数：");
5. Scanner reader=new Scanner(System.in);
6. char graphType=reader.next().charAt(0);
7. int graphSize=reader.nextInt();
8. switch(graphType){
9. case 'a':
10. CreateA(graphSize);
11. break;
12. case 'b':
13. CreateB(graphSize);
14. break;
15. case 'c':
16. CreateC(graphSize);
17. break;
18. case 'd':
19. CreateD(graphSize);
20. break;
21. case 'e':
22. CreateE(graphSize);
23. }
24. }
25. public static void CreateA(int SOG){
26. for(int i=0;i<SOG;i++){
27. if(i==0||i==SOG-1){
28. for(int j=0;j<SOG;j++){
29. System.out.print('#');
30. }
31. System.out.print('\n');
32. }
33. else{
34. for(int j=0;j<SOG;j++){
35. if(j==0||j==SOG-1)
36. System.out.print('#');
37. else
38. System.out.print(' ');
39. }
40. System.out.print('\n');
41. }
42. }
43. }
44. public static void CreateB(int SOG){
45. for(int i=0;i<SOG;i++){
46. if(i==0||i==SOG-1){
47. for(int j=0;j<SOG;j++){
48. System.out.print('#');
49. }
50. System.out.print('\n');
51. }
52. else{
53. for(int j=0;j<SOG;j++){
54. if(j==i)
55. System.out.print('#');
56. else
57. System.out.print(' ');
58. }
59. System.out.print('\n');
60. }
61. }
62. }
63. public static void CreateC(int SOG){
64. for(int i=0;i<SOG;i++){
65. if(i==0||i==SOG-1){
66. for(int j=0;j<SOG;j++){
67. System.out.print('#');
68. }
69. System.out.print('\n');
70. }
71. else{
72. for(int j=0;j<SOG;j++){
73. if(i+j==SOG-1)
74. System.out.print('#');
75. else
76. System.out.print(' ');
77. }
78. System.out.print('\n');
79. }
80. }
81. }
82. public static void CreateD(int SOG){
83. for(int i=0;i<SOG;i++){
84. if(i==0||i==SOG-1){
85. for(int j=0;j<SOG;j++){
86. System.out.print('#');
87. }
88. System.out.print('\n');
89. }
90. else{
91. for(int j=0;j<SOG;j++){
92. if(i+j==SOG-1||i==j)
93. System.out.print('#');
94. else
95. System.out.print(' ');
96. }
97. System.out.print('\n');
98. }
99. }
100. }
101. public static void CreateE(int SOG){
102. for(int i=0;i<SOG;i++){
103. if(i==0||i==SOG-1){
104. for(int j=0;j<SOG;j++){
105. System.out.print('#');
106. }
107. System.out.print('\n');
108. }
109. else{
110. for(int j=0;j<SOG;j++){
111. if(i+j==SOG-1||i==j||j==0||j==SOG-1)
112. System.out.print('#');
113. else
114. System.out.print(' ');
115. }
116. System.out.print('\n');
117. }
118. }
119. }
120. }

四：

1. import java.io.File;
2. import java.io.FileNotFoundException;
3. import java.util.HashMap;
4. import java.util.Scanner;
5. public class MorseCodeDecode {
6. public static void main(String[] args) {
7. *// 定义摩斯密码字典*
8. HashMap<String, Character> morseCodeMap = new HashMap<>();
9. morseCodeMap.put(".-", 'A');
10. morseCodeMap.put("-...", 'B');
11. morseCodeMap.put("-.-.", 'C');
12. morseCodeMap.put("-..", 'D');
13. morseCodeMap.put(".", 'E');
14. morseCodeMap.put("..-.", 'F');
15. morseCodeMap.put("--.", 'G');
16. morseCodeMap.put("....", 'H');
17. morseCodeMap.put("..", 'I');
18. morseCodeMap.put(".---", 'J');
19. morseCodeMap.put("-.-", 'K');
20. morseCodeMap.put(".-..", 'L');
21. morseCodeMap.put("--", 'M');
22. morseCodeMap.put("-.", 'N');
23. morseCodeMap.put("---", 'O');
24. morseCodeMap.put(".--.", 'P');
25. morseCodeMap.put("--.-", 'Q');
26. morseCodeMap.put(".-.", 'R');
27. morseCodeMap.put("...", 'S');
28. morseCodeMap.put("-", 'T');
29. morseCodeMap.put("..-", 'U');
30. 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) {
75. System.out.println("File not found.");
76. }
77. }
78. }

五：

1. import java.util.Random;
2. import java.util.Scanner;
3. public class PIByMonte{
4. public static void main(String args[]){
5. Scanner reader=new Scanner(System.in);
6. Random random=new Random();
7. int n=reader.nextInt();
8. int m=0;
9. for(int i=0;i<n;i++){
10. double x= random.nextDouble();
11. double y= random.nextDouble();
12. if(x\*x+y\*y<=1){
13. m++;
14. }
15. }
16. double probablePi=4.0\*m/n;
17. System.out.println("结果是"+probablePi);
18. }
19. }

六：

1. import java.util.Random;
2. public class RandomWalker {
3. public static void main(String[] args) {
4. *// 检查命令行参数数量是否正确*
5. if (args.length != 1) {
6. System.out.println("请提供一个整数参数:");
7. return;
8. }
9. *// 获取步数参数*
10. int n = Integer.parseInt(args[0]);
11. *// 初始化起始坐标和欧几里得距离*
12. int x = 0;
13. int y = 0;
14. double distance = 0.0;
15. *// 创建随机数生成器*
16. Random random = new Random();
17. *// 模拟 n 步随机游走*
18. for (int i = 0; i < n; i++) {
19. *// 随机选择一个方向*
20. int direction = random.nextInt(4);
21. *// 根据方向更新坐标*
22. switch (direction) {
23. case 0: *// 北*
24. y++;
25. break;
26. case 1: *// 南*
27. y--;
28. break;
29. case 2: *// 西*
30. x--;
31. break;
32. case 3: *// 东*
33. x++;
34. break;
35. }
36. *// 计算欧几里得距离*
37. distance = Math.sqrt(x \* x + y \* y);
38. *// 输出当前步的坐标*
39. System.out.println("第 " + (i + 1) + " 步的坐标为：(" + x + ", " + y + ")");
40. }
41. *// 输出最终位置和起始位置的欧几里得距离*
42. System.out.println("最终位置和起始位置的欧几里得距离为：" + distance);
43. }
44. }
45. import java.util.Random;
46. public class RandomWalkers {
47. public static void main(String[] args) {
48. *// 检查命令行参数数量是否正确*
49. if (args.length != 2) {
50. System.out.println("请提供两个整数参数。");
51. return;
52. }
53. *// 获取步数参数和实验次数参数*
54. int n = Integer.parseInt(args[0]);
55. int trials = Integer.parseInt(args[1]);
56. *// 初始化总距离*
57. double totalDistance = 0.0;
58. *// 创建随机数生成器*
59. Random random = new Random();
60. *// 进行 trials 次实验*
61. for (int i = 0; i < trials; i++) {
62. *// 初始化起始坐标和欧几里得距离*
63. int x = 0;
64. int y = 0;
65. double distance = 0.0;
66. *// 模拟 n 步随机游走*
67. for (int j = 0; j < n; j++) {
68. *// 随机选择一个方向*
69. int direction = random.nextInt(4);
70. *// 根据方向更新坐标*
71. switch (direction) {
72. case 0: *// 北*
73. y++;
74. break;
75. case 1: *// 南*
76. y--;
77. break;
78. case 2: *// 西*
79. x--;
80. break;
81. case 3: *// 东*
82. x++;
83. break;
84. }
85. *// 计算欧几里得距离*
86. distance = Math.sqrt(x \* x + y \* y);
87. }
88. *// 累加距离*
89. totalDistance += distance;
90. }
91. *// 计算平均距离*
92. double averageDistance = totalDistance / trials;
93. *// 输出平均距离*
94. System.out.println("完成 " + trials + " 次实验之后的欧几里得距离的平均值为：" + averageDistance);
95. }
96. }