**1,从键盘接收一个文件夹路径,统计该文件夹大小**

1. package demo;
3. import java.io.File;
4. import java.util.Scanner;
6. public class Test1 {
8. public static void main(String[] args) {
9. File file = getDirectory();
10. System.out.println(fileSize(file));
11. }
13. // 判断是否是文件夹
14. public static File getDirectory() {
15. Scanner sc = new Scanner(System.in);
16. System.out.print("请输入文件路径:");
18. while (true) {
19. String path = sc.next();
20. File file = new File(path);
21. if (!file.exists()) {
22. System.out.print("文件不存在，请重新输入:");
23. } else if (file.isFile()) {
24. System.out.print("您输入的是文件路径,请重新输入:");
25. } else {
26. return file;
27. }
28. }
30. }
32. // 使用递归判断文件夹大小
33. public static int fileSize(File file) {
35. int size = 0;
37. File[] files = file.listFiles();
39. for (File file1 : files) {
40. if (file1.isFile()) {
41. size += file1.length();
42. } else {
43. size += fileSize(file1);
44. }
45. }
47. return size;
48. }
50. }

**2,从键盘接收一个文件夹路径,删除该文件夹**

1. package demo;
3. import java.io.File;
4. import java.util.Scanner;
6. public class Test2 {
8. public static void main(String[] args) {
9. File file = getDirectory();
10. deleteDirectory(file);
11. if (file.exists()) {
12. System.out.println("删除失败");
13. } else {
14. System.out.println("删除成功");
15. }
16. }
18. // 判断是否是文件夹
19. public static File getDirectory() {
20. Scanner sc = new Scanner(System.in);
21. System.out.print("请输入文件路径:");
23. while (true) {
24. String path = sc.next();
25. File file = new File(path);
26. if (!file.exists()) {
27. System.out.print("文件不存在，请重新输入:");
28. } else if (file.isFile()) {
29. System.out.print("您输入的是文件路径,请重新输入:");
30. } else {
31. return file;
32. }
33. }
35. }
37. // 使用递归删除文件夹
38. public static void deleteDirectory(File file) {
40. File[] files = file.listFiles();
42. for (File file1 : files) {
43. if (file1.isFile()) {
44. file1.delete();
45. } else {
46. deleteDirectory(file1);
47. }
48. }
50. // 当文件夹里没有文件时，删除文件
51. file.delete();
52. }
54. }

**3,从键盘接收两个文件夹路径,把其中一个文件夹中(包含内容)拷贝到另一个文件夹中**

1. package demo;
3. import java.io.BufferedInputStream;
4. import java.io.BufferedOutputStream;
5. import java.io.File;
6. import java.io.FileInputStream;
7. import java.io.FileNotFoundException;
8. import java.io.FileOutputStream;
9. import java.io.IOException;
10. import java.util.Scanner;
12. public class Test3 {
14. public static void main(String[] args) throws IOException {
15. File src = getDirectory();
16. File dest = getDirectory();
17. if (src.equals(dest)) {
18. System.out.println("目标文件夹是原文件夹的子文件夹");
19. } else {
20. copyFile(src, dest);
21. }
23. }
25. // 判断是否是文件夹
26. public static File getDirectory() {
27. Scanner sc = new Scanner(System.in);
28. System.out.print("请输入文件路径:");
30. while (true) {
31. String path = sc.next();
32. File file = new File(path);
33. if (!file.exists()) {
34. System.out.print("文件不存在，请重新输入:");
35. } else if (file.isFile()) {
36. System.out.print("您输入的是文件路径,请重新输入:");
37. } else {
38. return file;
39. }
40. }
42. }
44. // 使用递归拷贝文件夹
45. public static void copyFile(File src, File dest) throws IOException {
47. // 先要在目的文件夹里创建原文件夹
48. File newFile = new File(dest, src.getName());
49. newFile.mkdir();
51. // 遍历原文件里的文件和文件夹
52. File[] files = src.listFiles();
54. for (File file1 : files) {
55. if (file1.isFile()) {
56. BufferedInputStream bis = new BufferedInputStream(new FileInputStream(file1));
57. BufferedOutputStream bos = new BufferedOutputStream(
58. new FileOutputStream(new File(newFile, file1.getName())));
59. int b;
60. while ((b = bis.read()) != -1) {
61. bos.write(b);
62. }
63. bis.close();
64. bos.close();
65. } else {
66. copyFile(file1, newFile);
67. }
68. }
69. }
71. }

**4,从键盘接收一个文件夹路径,把文件夹中的所有文件以及文件夹的名字按层级打印**

1. package demo;
3. import java.io.File;
4. import java.io.IOException;
5. import java.util.Scanner;
7. public class Test4 {
9. public static void main(String[] args) throws IOException {
10. File file = getDirectory();
11. tierPrint(file, 0);
12. }
14. // 判断是否是文件夹
15. public static File getDirectory() {
16. Scanner sc = new Scanner(System.in);
17. System.out.print("请输入文件路径:");
19. while (true) {
20. String path = sc.next();
21. File file = new File(path);
22. if (!file.exists()) {
23. System.out.print("文件不存在，请重新输入:");
24. } else if (file.isFile()) {
25. System.out.print("您输入的是文件路径,请重新输入:");
26. } else {
27. return file;
28. }
29. }
31. }
33. // 使用递归按层级打印
34. public static void tierPrint(File file, int lev) throws IOException {
36. File[] files = file.listFiles();
38. for (File file1 : files) {
39. for (int i = 0; i < lev; i++) {
40. System.out.print("\t");
41. }
42. System.out.println(file1.getPath());
43. if (file1.isDirectory()) {
44. tierPrint(file1, lev + 1);
45. }
46. }
47. }
49. }

**5,假设一对刚出生的小兔一个月后就能长成大兔，再过一个月就能生下一对小兔，并且此后每个月都生一对小兔，一年内没有发生死亡，  
问：一对刚出生的兔子，一年内繁殖成多少对兔子?  
规律:1 1 2 3 5 8 13 21**

1. package demo;
3. import java.util.Scanner;
5. public class Test5 {
7. public static void main(String[] args) {
9. // 规律: 1 1 2 3 5 8 13 21
10. Scanner sc = new Scanner(System.in);
11. System.out.print("请输入天数:");
12. int day = sc.nextInt();
13. System.out.println(recursion(day));
14. // System.out.println(array(day));
15. }
17. // 使用递归进行计算
18. public static int recursion(int day) {
20. if (day == 1 || day == 2) {
21. return 1;
22. } else {
23. return recursion(day - 2) + recursion(day - 1);
24. }
26. }
28. // 按数组方式计算
29. public static int array(int day) {
31. int[] is = new int[day];
33. is[0] = 1;
34. is[1] = 1;
36. for (int i = 2; i < is.length; i++) {
37. is[i] = is[i - 2] + is[i - 1];
38. }
39. return is[is.length - 1];
40. }
42. }

**6,约瑟夫环**

**N个人围成一圈，从第一个开始报数，每第三个人将被杀掉，最后剩下一个，其余人都将被杀掉。**

1. package demo;
3. import java.util.ArrayList;
5. public class Test6 {
7. public static void main(String[] args) {
8. System.out.println(getLucklyNum(8));// 7
9. }
11. public static int getLucklyNum(int num) {
13. ArrayList<Integer> list = new ArrayList<Integer>();
14. for (int i = 1; i <= num; i++) {
15. list.add(i);
16. }
17. // 计数
18. int count = 1;
19. for (int i = 0; list.size() != 1; i++) {
21. // 如果下标超过了最大下标，相当于又从一个人接着往下叫数
22. if (i == list.size()) {
23. i = 0;
24. }
26. // 每三个人就删除一个
27. if (count % 3 == 0) {
28. list.remove(i);
29. i--;
30. }
32. count++;
34. }
36. // 最后剩下的一个人返回
37. return list.get(0);
38. }
40. }