**文件**

File类

1.构造函数

File（String path）

File（String path，String name）

File（File dir，String name）

2.是否存在

public boolean exists（）

3.判断是文件还是目录

public boolean isFile（）

public boolean isDirectory（）

4.获取文件名或者路径

public String getName（）

public String getPath（）

5.文件长度

public long length（）

6.文件读写属性

public boolean canRead（）

public boolean canWrite（）

7.获取目录中的文件

public String[]list（）

public File[]listFiles（）

过滤，筛选符合条件的文件

list（new FilenameFilter（）{

public boolean accept（File dir，String name）{

}

}）

8.文件/目录操作

重命名 public boolean renameTo（File new）

删除 public boolean delete（）

创建目录 public boolean mkdir（）

mkdirs（）

**输入输出**

输入：

1.使用Scanner类进行控制台输入

Scanner scanner = new Scanner(System.in);

String userInput = scanner.nextLine();

System.out.println("You entered: " + userInput);

scanner.close();

2.使用BufferedReader类进行控制台输入

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String userInput = reader.readLine();

System.out.println("You entered: " + userInput);

reader.close();

3.使用Scanner读取文本文件

Scanner scanner = new Scanner(new File("example.txt"));

while (scanner.hasNextLine()) {

String line = scanner.nextLine();

System.out.println(line);

}

scanner.close();

4.使用BufferedReader读取文本文件

BufferedReader reader = new BufferedReader(new FileReader("example.txt"))

String line;

while ((line = reader.readLine()) != null) {

System.out.println(line);

}

输出：

文件输出

1.使用PrintWriter写入文本文件

PrintWriter writer = new PrintWriter(new FileWriter("output.txt"))

writer.println("Hello, World!");

writer.close();

2.使用BufferedWriter写入文本文件

BufferedWriter writer = new BufferedWriter(new FileWriter("output.txt"))

writer.write("Hello, World!");

writer.newLine();

**多线程**

1.实现线程

1）继承Thread类

2）实现Runnable接口

关键都是重载run函数

注意：实现接口的话需要Thread包装。Thread selThread = **new** Thread(**new** sell());

要实现信息传递，可以定义新变量

2.锁

1）synchronized（lock）{}修饰代码段，lock是object类型

2）ReentrantLock类型的lock，函数有lock（）、unlock（）、

trylock（）：成功的话不用再lock（），已经加锁了

trylock（2，TimeUnit.SECONDS）

3.Thread.sleep（）

4.wait（）、notify（）需要在synchronized方法或者代码段中

**随机数**

在Java中，你可以使用`java.util.Random`类来生成随机数。以下是一个简单的例子：

import java.util.Random;

public class RandomExample {

public static void main(String[] args) {

// 创建一个Random对象

Random random = new Random();

// 生成一个随机整数

int randomNumber = random.nextInt();

System.out.println("随机整数: " + randomNumber);

// 生成一个指定范围的随机整数（例如，生成1到100之间的随机数）

int randomInRange = random.nextInt(100) + 1;

System.out.println("1到100之间的随机整数: " + randomInRange);

// 生成一个随机浮点数（0.0到1.0之间）

double randomDouble = random.nextDouble();

System.out.println("随机浮点数: " + randomDouble);

}

}

```

在上面的例子中，`nextInt()`方法用于生成随机整数，而`nextDouble()`方法用于生成随机浮点数。你可以根据需要调整生成随机数的范围。要注意，`nextInt(n)`生成的随机整数范围是0（包括）到n（不包括）。

**字符串处理**

1）String

2）StringBuilder

3）String[]

4）char[]

1.把数字数组转换为字符串

private static String concatenateIntegers(int[] integers) {

StringBuilder builder = new StringBuilder();

for (int i : integers) {

builder.append(i).append(" ");

}

return builder.toString().trim(); // 移除末尾的空格

}

2.统计字符串中大小写字母数目

String inputString = scanner.nextLine();

for (char c : inputString.toCharArray()) {

if (Character.isUpperCase(c)) {

uppercaseCount++;

} else if (Character.isLowerCase(c)) {

lowercaseCount++;

}

}