文件与数据流

主讲老师：申雪萍

主要内容

- **控制台读入数据、标准输入输出、Scanner类（基础的输入输出）**
- Java文件管理（File类）
- 输入/输出流类及其派生类的使用（字节流的顺序读写）
- 随机访问文件
- 读写器及其派生类的使用（字符流的顺序读写）
- 对象序列化（对象流的读写）

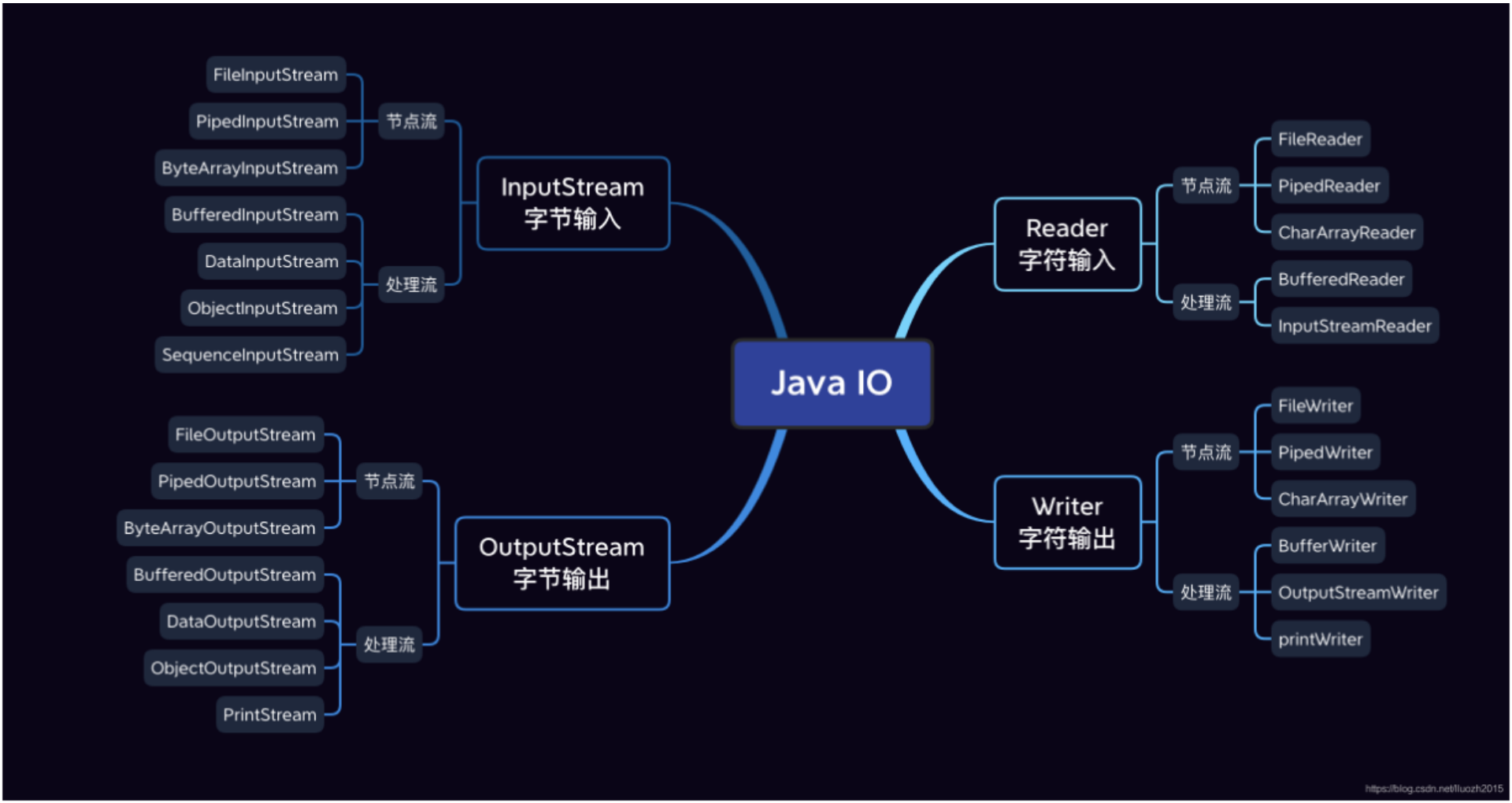
问答题

功能需求

- Java支持字符读写吗？支持字节读写吗？
- 如何接受命令行输入？
- 如何进行标准输入和标准输出？
- 如何读写文件？如何了解文件的属性?随机读还是顺序读？
- 如何读格式化的文件？
- 如何读内存？
- Java支持分布式编程，如何在节点之间读写对象？
- Java支持线程间通信吗？

性能

- 如何提升读写性能


Java IO
InputStream
字节输入
节点流
FileInputStream
PipedInputStream
ByteArrayInputStream
处理流
BufferedInputStream
DataInputStream
ObjectInputStream
SequenceInputStream
OutputStream
字节输出
节点流
FileOutputStream
PipedOutputStream
ByteArrayOutputStream
处理流
BufferedOutputStream
DataOutputStream
ObjectOutputStream
PrintStream
Reader
字符输入
节点流
FileReader
PipedReader
CharArrayReader
处理流
BufferedReader
InputStreamReader
Writer
字符输出
节点流
FileWriter
PipedWriter
CharArrayWriter
处理流
BufferWriter
OutputStreamWriter
printWriter
https://blog.csdn.net/liuozh2015

Java输入输出（基础）

主讲老师：申雪萍

Java IO概述

- 输入：读取外部数据（磁盘、光盘等存储设备的数据）到程序（内存）中。
- 输出：将程序（内存）数据输出到磁盘、光盘等存储设备中
- Java 的 IO 流主要包括输入、输出两种 IO 流，每种输入、输出流又可分为字节流和字符流两大类：
  - 字节流以**字节**为单位来处理输入、输出操作
  - 字符流以**字符**为单位来处理输入、输出操作

如何接受命令行输入？Command-Line Arguments

- **These string arguments are placed on the command line to launch the Java interpreter, after the class name:**

  java TestArgs arg1 arg2 "another arg"

- **Each command-line argument is placed in the args array that is passed to the static main method:**

  public static void main(String[] args)

如何接受命令行输入？Command-Line Arguments

```
1   public class TestArgs {
2     public static void main(String[] args) {
3       for ( int i = 0; i < args.length; i++ ) {
4         System.out.println("args[" + i + "] is '" + args[i] + "'");
5       }
6     }
7   }
```

Example execution:

```
java TestArgs arg1 arg2 "another arg"
args[0] is 'arg1'
args[1] is 'arg2'
args[2] is 'another arg'
```

标准输入输出 Console I/O

- **The variable System.out enables you to write to *standard output*. It is an object of type PrintStream.**
- **The variable System.in enables you to read from *standard input*. It is an object of type InputStream.**
- **The variable System.err enables you to write to *standard error*. It is an object of type PrintStream.**

标准输出

- **The println methods print the argument and a newline character (\n).**
- **The print methods print the argument without a newline character.**
- **The print and println methods are overloaded for most primitive types (boolean, char, int, long, float, and double) and for char[], Object, and String.**
- **The print(Object) and println(Object) methods call the toString method on the argument.**

标准输入 Reading From Standard Input

```
1   import java.io.*;
2
3   public class KeyboardInput {
4     public static void main (String args[]) {
5       String s;
6       // Create a buffered reader to read
7       // each line from the keyboard.
8       InputStreamReader ir
9         = new InputStreamReader(System.in);
10      BufferedReader in = new BufferedReader(ir);
11
12      System.out.println("Unix: Type ctrl-d to exit." +
13                         "\nWindows: Type ctrl-z to exit");
```

标准输入 Reading From Standard Input

```
14      try {
15        // Read each input line and echo it to the screen.
16        s = in.readLine();
17        while ( s != null ) {
18          System.out.println("Read: " + s);
19          s = in.readLine();
20        }
21
22        // Close the buffered reader.
23        in.close();
24      } catch (IOException e) { // Catch any IO exceptions.
25        e.printStackTrace();
26      }
27    }
28  }
```

Simple Formatted Output

- You can use the formatting functionality as follows

  out.printf(“name count\n”);

  String s = String.format(“%s %5d%n”, user, total);

Java中的String.format使用

转 换 符	说　明	示　例
%s	字符串类型	"mingrisoft"
%c	字符类型	'm'
%b	布尔类型	true
%d	整数类型（十进制）	99
%x	整数类型（十六进制）	FF
%o	整数类型（八进制）	77
%f	浮点类型	99.99
%a	十六进制浮点类型	FF.35AE
%e	指数类型	9.38e+5
%g	通用浮点类型（f和e类型中较短的）	
%h	散列码	
%%	百分比类型	%
%n	换行符	
%tx	日期与时间类型（x代表不同的日期与时间转换符	

Java中的String.format使用

标　志	说　明	示　例	结　果
+	为正数或者负数添加符号	("%+d",15)	+15
-	左对齐	("%-5d",15)	|15　|
0	数字前面补0	("%04d", 99)	0099
空格	在整数之前添加指定数量的空格	("% 4d", 99)	| 99|
,	以“,”对数字分组	("%,f", 9999.99)	9,999.990000
(	使用括号包含负数	("%(f", -99.99)	(99.990000)
#	如果是浮点数则包含小数点，如果是16进制或8进制则添加0x或0	("%#x", 99) ("%#o", 99)	0x63 0143
<	格式化前一个转换符所描述的参数	("%f和%<3.2f", 99.45)	99.450000和99.45
$	被格式化的参数索引	("%1$d,%2$s", 99,"abc")	99,abc

Simple Formatted Output

- Common formatting codes are listed in this table.

Code	Description
`%s`	Formats the argument as a string, usually by calling the `toString` method on the object.
`%d %o %x`	Formats an integer, as a decimal, octal, or hexadecimal value.
`%f %g`	Formats a floating point number. The `%g` code uses scientific notation.
`%n`	Inserts a newline character to the string or stream.
`%%`	Inserts the `%` character to the string or stream.

Scanner类的使用

- Scanner的作用：通过分隔符模式将输入分解为标记，默认情况下该分隔符模式与空白匹配。
- 然后可以使用不同的 next 方法将得到的标记转换为不同类型的值。
- 通过 Scanner 类的 next() 与 nextLine() 方法获取输入的字符串，在读取前我们一般需要使用 hasNext 与 hasNextLine 判断是否还有输入的数据
- **A Scanner can be used with console input streams as well as file or network streams.**

使用 next 方法：

```
$ javac ScannerDemo.java
$ java ScannerDemo
next方式接收：
runoob com
输入的数据为：runoob
```

```java
import java.util.Scanner;

public class ScannerDemo {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        // 从键盘接收数据

        // next方式接收字符串
        System.out.println("next方式接收：");
        // 判断是否还有输入
        if (scan.hasNext()) {
            String str1 = scan.next();
            System.out.println("输入的数据为：" + str1);
        }
        scan.close();
    }
}
```

使用 nextLine 方法

```
$ javac ScannerDemo.java
$ java ScannerDemo
nextLine方式接收：
runoob com
输入的数据为：runoob com
```

```java
import java.util.Scanner;

public class ScannerDemo {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        // 从键盘接收数据

        // nextLine方式接收字符串
        System.out.println("nextLine方式接收：");
        // 判断是否还有输入
        if (scan.hasNextLine()) {
            String str2 = scan.nextLine();
            System.out.println("输入的数据为：" + str2);
        }
        scan.close();
    }
}
```

```java
import java.util.Scanner;

public class ScannerDemo {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        // 从键盘接收数据
        int i = 0;
        float f = 0.0f;
        System.out.print("输入整数：");
        if (scan.hasNextInt()) {
            // 判断输入的是否是整数
            i = scan.nextInt();
            // 接收整数
            System.out.println("整数数据：" + i);
        } else {
            // 输入错误的信息
            System.out.println("输入的不是整数！");
        }
        System.out.print("输入小数：");
        if (scan.hasNextFloat()) {
            // 判断输入的是否是小数
            f = scan.nextFloat();
            // 接收小数
            System.out.println("小数数据：" + f);
        } else {
            // 输入错误的信息
            System.out.println("输入的不是小数！");
        }
        scan.close();
    }
}
```

```
$ javac ScannerDemo.java
$ java ScannerDemo
输入整数：12
整数数据：12
输入小数：1.2
小数数据：1.2
```