**Java学习day19**

**1.缓冲区的出现是为了抬高流的操作效率而出现的 对应01**

所以在创建缓冲区之前，必须要现有流对象。

该缓冲区提供了一个跨平台的换行符

把缓冲区形象的理解为一个瓶子， 如果没有瓶子，我们喝水的效率会很慢，缓冲区就相当于给我们提高效率使用的东西

思路：创建一个流对象， 然后再创建一个容器对象，将流对象装进容器对象之中。 然后使用容器的方法进行一些操作，写入，刷新和关闭

**2.字符读取流缓冲区 对应02**

该缓冲区提供了一次读取一行的方法readLine,方便与文本数据的获取。

当返回值为null时，说明已经读到了文本的末尾

1. **通过缓冲区复制一个java文件 对应03**
2. **IO流（readLine（））原理图例 对应04**

其实就是基于read方法一次读一行，但是这个readLine识别到关键字的时候就可以开始换行操作了。 readLIne()方法的返回类型为String,字符串类型

1. **IO流 MyBufferedReader 对应05**

**明白了BufferedReader类中的特有方法readLine()原理后，可以自定义一个类中包含的一个功能和readLine一致的方法。来模拟一下BufferedReader**

1. **装饰设计模式 对应06**

需求：定一个Person，只有吃饭的功能，但是这明显是不够的，所以我们新增一个SuperPerson增强其中的功能，同时接受Person类

装饰设计模式：当原有的类中的功能需要增强的时候，就需要用到装饰设计模式，

装饰设计模式是自己定义一个装饰类，同时将需要装饰的对象传入到装饰类之中，从而实现功能增强的目的。

被装饰的对象需要传入到 装饰类中的构造函数之中。

注意，BufferedRead也属于装饰功能，将流装入缓冲区中，并且使对象具有更强的功能，因为对象可以一次性读取一行（从原来的一个字符）。

复习：单例设计模式：就是确保一个类只有一个实例，有两种设计方法，分别是饿汉式与懒汉式。

1. **装饰和继承的区别： 对应07**

装饰模式要比集成要灵活，避免了集成体系的臃肿。

而且降低了累与泪之间的依赖关系。

装饰类因为增强已有对象，具备的功能和已有对象是相同的，只不过提供了更强的功能。所以装饰类预备装饰类通常在一个体系中。

1. **IO流（LineNumberReader） 对应08**

这是BUfferedReader中的子类，功能更强 查阅API然后做，就是行号的设置与获取。

// 留一个小练习： 模拟一个带行号的缓冲区对象。

/\*

int getLineNumber()

获取当前的行数。

void setLineNumber(int lineNumber)

设置当前行数。

int getLineNumber()

获取当前的行数。

**9.现在开始来模拟,MyLineNumberReader这个类的实现 对应09 10**

将几个主要的函数复现出来即可

/\*

int getLineNumber()

获取当前的行数。

void setLineNumber(int lineNumber)

设置当前行数。

String MyreadLine()

Myclose()

有的一些功能可以从MyBufferedReader 中给继承下来。

1. **需求：使用字节流写入文件 用到FileOutputStream 这一个类。 对应11**

想要操作图片数据的时候应该用到字节流

然后使用字节流读取文件，用到FileInputStream 这一个类。(这里字节流的读取共有3种方法，普通，创数组，特有(available这一个方法))

字节流可以不用flush每次写入的时候刷。

1. **图片的操作是字节流的操作 对应12**

**BufferedInputStream 相应的缓冲区技术的引入。 对应13.**

**11.自定义字节流的缓冲区，read和write的特点， 明白一下这个创建的过程，需要数组、计数器与指针三个元素缺一不可。对应14**

因为byte是8位的，而int是32位的，直接将byte转为int强转的化，可能会使前面的位数补1，从而导致整个32位的int类型的值为-1，

所以应该&上一个int类型的255来完成最后8位的保留。

总结一下read和write的特点：

节流的读一个字节的read方法为什么返回值类型不是byte，而是int。

因为有可能会读到连续8个二进制1的情况，8个二进制1对应的十进制是-1；那么就会数据还没有读完，就结束的情况。因为我们判断读取结束是通过结尾标记-1来确定的。

所以，为了避免这种情况将读到的字节进行int类型的提升。并在保留原字节数据的情况前面了补了24个0，变成了int类型的数值。

而在write写入数据时，只写该int类型数据的最低8位。

**12.读取键盘录入 对应 15**

System.out 标准的输出，控制台

System.in 标准的输入，键盘

需求：通过键盘录入数据。

当录入一行数据的时候，就将该数据打印。

如果录入的数据是over，则停止打印退出。

自己先来测试一下 System.in的功能

**13.IO 流（读取转换流） IO流(写入转换流) 16，17,18**

键盘录入的read方法都是字节流InputStream的方法，

能否将字节流转换成字符流，然后使用字符流的readLine方法呢，

通过查阅API 我们看到了一个类InputStreamReader 通过它，能够有效地将字节转换成字符

需求：通过InputStreamReader将字节流转换成字符流，然后输出为大写。

键盘录入最常见的写法，就是使用OutputStreamWriter 这一个类，通过它，可以不用System.out.println这样的输出语句

将字符流转换成字节流，

需求1：通过InputStreamReader将字节流转换成字符流，然后输出为大写。

源：键盘录入。 目的：控制台。

需求2：将键盘录入的数据存储到一个文件中

源：键盘录入。 目的：文件

需求3：将一个文件的数据输出在控制台上

源：文件 目的：控制台

流的最基本规律：

最痛苦的就是流对象有很多，不知道该用哪个

通过三个明确来完成。

1.明确源和目的。（共2个选择）

源：输入流。 InputStream Reader

目的：输出流。 OutputStream Writer

2.操作的数据是否为纯文本。

是：字符流。

不是：字节流。

3.当体系明确后，在明确要使用哪个具体的对象。

通过设备来进行区分：

源设备：内存，硬盘，键盘。

目的设备：内存，硬盘，控制台。

1.将一个本文文件中的数据存储到另一个文件中，复制文件。

源：因为是源，所以使用读取流。InputStream Reader 这两个都可以选择

是不是操作文本文件。

是！这时就可以选择Reader

这样体系就明确了。

接下来要明确要使用体系中的那个对象。 明确设备：硬盘上的一个文件。

Reader体系中可以操作文件的对象是 FileReader

是否需要提高效率：是！ 加入Reader体系中缓冲区 BufferedReader。

FileReader fr = new FileReader("a.txt");

BufferedRead sb = new BufferedRead(fr);

目的: OutputStream Writer 这两个都可以选择。

是否为纯文本。

是！Writer。

设备：硬盘，一个文件。

Writer体系中可以操作文件的对象FileWriter

是否需要提高效率：是！加入Writer体系中的缓冲区 BufferedWriter

FIleWriter fw = new FileWriter("a\_copy.txt");

BufferedWriter sb1 = new BufferedWriter(fw);

**留一个练习**：将一个图片文件中数据存储到另一个文件，复制文件。

源：因为是源所以有两个选择：InputStream 和Read

因为是图片 所以要用字节流，所以是InputStream

因为是文件，所以明确了体系，是FileInputStream

体系明确之后，知道了是在硬盘中的文件

目的：因为是目的所以有两个选择，OutputStream 和Write

因为存储到另一个的应该是图片文件

在明确了体系中后知道了是硬盘中的文件。

14.IO流（流操作规律2）对应19

搞一个改良，就是希望存储的文件格式有不同的类型GBK 和 utf-8，将数据存到文件中。

OutputStreamWriter(OutputStream out, CharsetEncoder enc)

创建一个outputstreamwriter使用给定的字符集编码。

目的：OutputStream Writer

是否是文本？ 是！Writer

设备但是存储的时候，需要加入指定编码表utf-8。而指定的编码表只有转换流可以指定。所以要使用的对象是OutputStreamWriter，

而该转换对象要接受一个字节输出流，而且还可以操作的文件的字节输出流。FileOutputStream

所以，记住。转换流什么时候使用。字符和自己额之间的桥梁，通常涉及字符编码转换时，需要用到转换流。

--------------------------------------------

**练习**：将一个文本数据打印在控制台上，要按照以上三个格式完成三个明确 System.out

**15.I/O流（改变标准输入输出设备） 对应20**

用到了System.setIn 和System.setOut 这两个方法，记得自己查对应的API。

**16.IO流（异常的日志信息）**

并且添加一个报错的时间点,并且用一个SimpleDateFormat来做一个时间的格式化

**17.IO流（系统信息）**  并且添加时间信息

list(PrintWriter out)

将此属性列表打印到指定的输出流中。