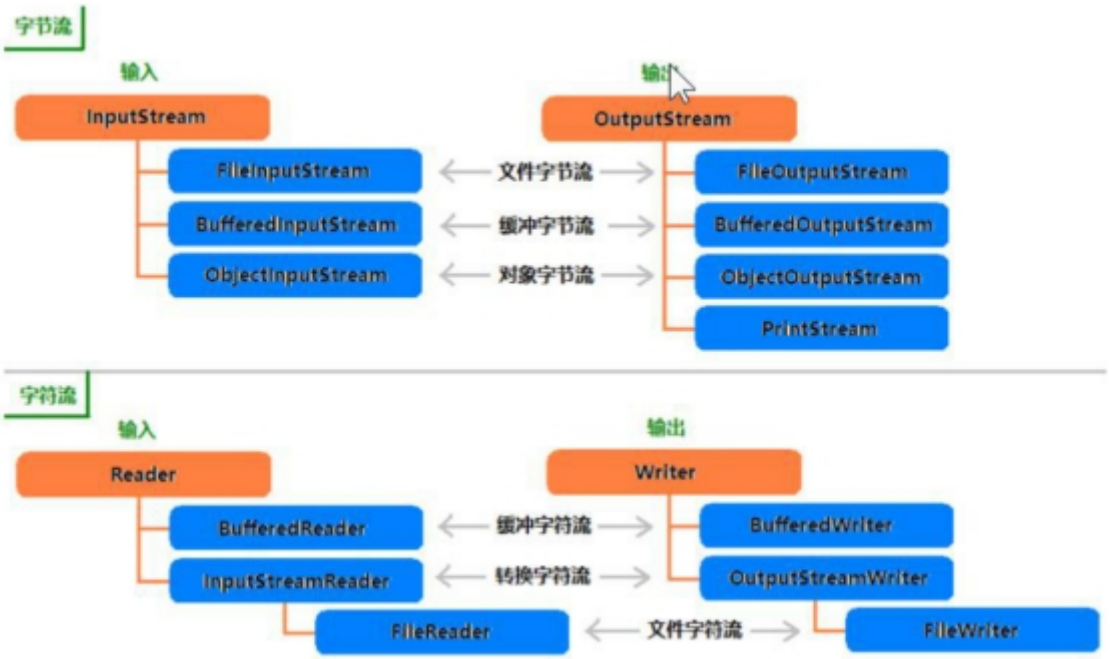

IO的概念和分类

IO流: 输入(Input)输出(Output)流

流: 数据流

输入: 从硬盘(文件)读取到内存(Java程序)

输出: 从内存(Java程序)写入到硬盘(文件)

(入和出都是相对于内存来说的)

```
          +---------+
 输入     |         |  输出
--------->|  内存   | -------->
          |         |
          +---------+
```

IO流的分类:

字节流 (读写字节: byte, 可以读写所有类型的文件, 包括视频, 图片, 音频, 文本等)

字节输入流: java.io.InputStream 抽象类

字节输出流: java.io.OutputStream 抽象类

字符流 (读写字符: char, String, 只能读写文本文件)

字符输入流: java.io.Reader 抽象类

字符输出流: java.io.Writer 抽象类

字节输出流: OutputStream和FileOutputStream

java.io.OutputStream抽象类: 字节输出流 (顶层类)

// 成员方法

void close() ： 释放资源

void flush() ： 刷新缓冲区(对于字节流来说没有作用)

// 写字节的成员方法

abstract void write(int b): 一次写一个字节 (参数int便于传递byte的整数不用强转)

void write(byte[] b): 一次写一个字节数组

void write(byte[] b, int offset, int len): 一次写一个字节数组的一部分

java.io.FileOutputStream类: 文件字节输出流 (向文件写数据)

// 构造方法

FileOutputStream(String name): 通过文件路径创建文件字节输出流

FileOutputStream(File file): 通过File对象创建文件字节输出流

构造方法的作用:

1. 创建一个FileOutputStream对象
2. 根据构造方法传递的路径, 在磁盘上创建一个空文件 ("如果文件存在则会清空数据")
3. 将创建的FileOutputStream对象指向这个磁盘上的文件

字节输出流: 一次写一个字节到文件

写数据的原理:

Java程序 -> JVM虚拟机 -> OS操作系统 -> OS调用写数据的方法 -> 将数据写入文件

使用字节输出流写数据到文件的步骤:

1. 创建对象: 创建FileOutputStream对象, 构造方法中传递文件路径
2. 写入数据: 使用FileOutputStream对象调用 write(int b) 方法, 将字节写入文件
3. 释放资源: 使用FileOutputStream对象调用 close() 方法, 关闭流对象释放资源

```
FileOutputStream fos = new FileOutputStream("模块名\\文件名.txt");
fos.write(97);
```

```
fos.close();
```

文件存储原理和记事本打开文件的原理

向文件中写入字节数据时, 十进制的数字会被转换为"二进制"的数字写入文件

文本编辑器打开文本文件时, 会先查询编码表, 将二进制数字转换为对应的字符进行显示

0-127: 查询ASCII码表　　-128~127

其他: 查询系统默认码表

Windows简体中文系统的程序打开是按 GBK 码表

IDEA中使用的是 UTF-8 码表

ASCII编码表: 1个byte就是一个字符　　97 a

GBK编码表: 2个byte数字组成一个汉字. "你": -60, -29

UTF-8编码表: 3个byte数字组成一个汉字. "你": -28, -67, -96

字节输出流: 续写, 换行

java.io.FileOutputStream类: 文件字节输出流

// 带有 续写 功能的构造方法, 不会清空文件

FileOutputStream(String name, boolean append): 通过文件路径创建流, true可以续写

FileOutputStream(File file, boolean append): 通过File对象创建流, true可以续写

换行符:

Windows系统: "\r\n"

Linux系统: "\n"

MacOS系统: "\r"

IO流操作中的异常处理

```
FileWriter fw = null;
    try {
        //IO流对象的创建, 操作等代码
        fw = new FileWriter("d:\\09_IOAndProperties\\g.txt", true);
        for (int i = 0; i <10 ; i++) {
            fw.write("HelloWorld"+i+"\r\n");
```

```
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        // 释放资源
        if(fw != null){
            try {
                fw.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

Properties集合

Properties存储数据和遍历

Properties双列集合:

键和值都是 String 类型

java.util.Properties类: 属性集, 属于Map的双列集合, 继承自Hashtable

// 构造方法

Properties(): 创建一个Properties集合

// 可以使用Map接口中的方法

// 特有成员方法

Object setProperty(String key, String value): 保存/替换键值对

String getProperty(String key): 通过键获取值. 键不存在返回null

Set<String> stringPropertyNames(): 返回键的集合

void store(OutputStream out, String comments): 将集合用字节流写入文件(不能中文), 并写入注释

void store(Writer writer, String comments): 将集合用字符流写入文件(可以中文),并写入注释

void load(InputStream inStream): 从配置文件中通过字节流加载数据到Properties集合(不能读中文)
void load(Reader reader): 从配置文件中通过字符流加载数据到Properties集合(可以读中文)

Properties: store()存储数据到文件

Properties将数据写入到文件的方法:
void store(OutputStream out, String comments): 将集合用字节流写入文件(不能中文), 并写入注释
void store(Writer writer, String comments): 将集合用字符流写入文件(可以中文), 并写入注释
使用步骤:
1.创建Properties集合对象,添加数据
2.创建字节输出流/字符输出流对象,构造方法中绑定要输出的目的地
3.使用Properties集合中的方法store,把集合中的临时数据,持久化写入到硬盘中存储
4.释放资源

Properties: load()从文件加载数据到集合

Properties将数据从文件加载到集合中的方法:
void load(InputStream inStream): 从配置文件中通过字节流加载数据到Properties集合(不能读中文)
void load(Reader reader): 从配置文件中通过字符流加载数据到Properties集合(可以读中文)

使用步骤:
1.创建Properties集合对象
2.使用Properties集合对象中的方法load读取保存键值对的文件
3.遍历Properties集合