LearnIO

一.File类

1.构造方法

```
#public File(String pathName):通过将给定的路径名字字符串转换为抽象路径名来创建新的File实例。
public File(String parent,String child):父路径和子路径创建新的File实例.
```

2.常用方法

1.文件名相关

```
file.getAbsolutePath():返回文件或目录的绝对路径
file.getPath():将此抽象路径名转换为路径名字符串
file.getName():返回文件名
file.renameTo(File file1):把文件剪切为一个新的路径
```

2.文件检测相关

```
boolean exists():判断file对应的文件或目录是否存在
boolean canWrite():判断file文件或目录是否可写。如果是目录，则是是否可创建文件。
boolean canRead():判断file对应的文件或目录是否可读
boolean isFile():判断file是否是文件，而不是目录
boolean isDirectory():判断file是否是目录，而不是文件
boolean isAbsolute():判断file对应的文件或目录是否为绝对路径。
```

3.获取文件的常规信息

```
long lastModified():返回文件的最后修改时间戳
long length():返回文件内容的长度
```

4.文件操作

```
boolean createNewFile():创建文件
boolean delete():删除文件
```

5.目录相关

```
boolean mkdir():创建file对应的一级目录，对象必须是一个目录，而不是一个文件。
boolean mkdirs():创建对应的目录(多级)，包括器父目录。
String[] list():列出file的所有子文件和路径，返回String数组.
File[] listFiles():列出file的所有子文件和路径，返回File数组.
```

二.字节流

1.什么是IO

> *IO: 以电脑内存为基准，输入是input,输出是output.*

2.IO分类

```
流向:
    输入流:输入内存，数据进入
    输出流:离开内存
类型:
        字节流：以细节为单位，读写数据的流
        字符流：以字符为单位，读写数据的流
```

3.一切都是字节

数据都是二进制传输，都是字节。

4.OutputStream

```
OutputStream是一个抽象类，表示字节输出流的所有类的父类，将主动的字节信息写到目的地。定义了基本功能方法:
    public void close():关闭此输出流冰释放相关资源
    public void flush():刷新此输出流冰强制输出缓冲区的数据
    public void write(byte[] b):将b.length字节从指定的字节数组写入输出流
    public void write(byte[] b,int off,int len):从指定的字节数组写入len字节，总偏移量off开始输出的输出流。
    piblic abstract void write(int b):将指定的字节输出流
```

1.FileOutputStream

```
FileOutputStream是文件输出流，数据写出到文件，是OutputStream的实现类，两个构造方法：
    public FileOutputStream(File file):创建文件输出流以写入指定的Fi了对象表示的文件
    public FileOutputStream(String name):创建文件出书刘以指定的名称入文件。
```

5.InputStream

```
InputStream是一个抽象类，表示字节输入流的所有类的父类，读取字节信息到内存中。定义了基本功能方法：
    public void close():关闭此输入流并释放相关资源
    public abstract int read():从输入流读取数据的下一个字节
    public int read(byte[] b):从输入流中读取一些字节数，并将他们存储到字节出租中，返回的是我们读取的长度。
```

1.FileInputStream

```
    FileInputStream(File file):通过打开一实际文件的链接来创建一个FileInputStream。
    FileInputStream(String name):通过打开一实际文件的链接来创建一个FileInputStream。
    当文件路径错误是，会抛出一个FileNotFoundException错误。
```

三.字符流

1.为啥用字符流

> *遇到中文字符或标点符号这些占用多个字节的情况，byte读取可能会出问题。*
>
> 所以，用字符流就正好

2.Writer

```
    Writer抽象类是英语写出字符流的父类(相当于OutputStream),将指定的字符信息写出到目的地。定义了输出流的基本方法:
    void write(int c):写入单个数组
    void write(char[] cbuf):写入字符数组
    void write(String str):写入字符串
    void write(String str,int len,int off):指定大小
    void flush():刷新该流的缓冲
    void close():关闭该流
```

1.FileWriter

```
构造方法:
    FileWriter(File file):
    FileWriter(String fileName):
```

3.Reader

```
    Reader抽象类是英语读取字符流的父类(相当于InputStream),读取内存信息到内存中。
    public void close():关闭此输入流并释放相关资源
    public int read():从输入流读取数据的下一个字符
```

1.FileReader

```
构造方法:
    FileReader(File file):
    FileReader(String fileName):
```

四.IO异常处理

1.java7中的try-with-resourse

```
    try-with-source
        try(){
            ....
        }catch{
            ....
        }finally{
            ....
        }
```

五.缓冲流

1.什么是缓冲流

> 在穿件对象时，会创建一个内置的默认大小的缓冲区数组，通过缓冲区的读写，减少系统IO的次数，从而提高读写效率。

2.字节缓冲流

```
    public BufferedInputStream(InputStream in):创建一个新的缓冲输入流
    public BufferedOutputStream(InputStream out):创建一个新的缓冲输出流
```

3.字符缓冲流

```
    public BufferedReader(Reader in):创建一个新的缓冲输入流
    public BufferedWriter(Wirter out):创建一个新的缓冲输出流

  特有方法:
    BufferedReader:public String readLine():读一行文字
    BufferedWriter:public void newLine():写一行分隔符，由系统定义符号。
```

六.转换流

> java.io.InputStreamReader：Reader的子类，读取字节，使用指定的字符集将其解码为字符。

```
    InputStreamReader(InputStream in):创建一个使用默认字符集的字符流
    InputStreamReader(InputStream in,String charsetName):创建一个指定字符集的字符流。
```

> java.io.OutputStreamReader：Reader的子类，写出字节，使用指定的字符集将其解码为字符。

```
    OutputStreamReader(InputStream in):创建一个使用默认字符集的字符流
    OutputStreamReader(InputStream in,String charsetName):创建一个指定字符集的字符流。
```

七.打印流

```
    printSteam():
 try (InputStream inputStream = new FileInputStream(
     "D:\\MySoftware\\IntelliJ IDEA 2020.3.1\\IDEAproject\\JavaProject\\LearnIO\\doc\\Printf.txt")){
            System.setIn(inputStream);
            Scanner scanner = new Scanner(System.in);
            while (scanner.hasNextLine()){
                String line = scanner.nextLine();
                System.out.println(line);
            }

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

八.RandomAccessFile

> RandomAccessFile，功能最丰富。你可以输入也可以输出。支持“随机访问”，可以直接跳转到任意位置来读写数据。

```
  构造器:
    RandomAccessFile(File file,String mode):随机访问
    RandomAccessFile(String name,String mode):具有指定名称的文件进行读取

   注：mode
        r:只读
        rw:写
```

九.对象序列化

1.使用

> 如果一个类可以被序列化，那么应该实现下面两个接口之一:
>
> Serializable
>
> Extemalizable
>
> 这两个接口的作用，就是标记一个对象是否可以被序列化，无需实现人和方法。

2.指定那些字段不需要序列化

> 用transient关键字修饰对应的字段就OK了
>
> eg:private transient int age; //表示序列化时，age字段保密

3.类升级问题

> 使用serialVersionUID解决问题