一、文件

1、简介

```
文件是保存数据的地方
```

2、文件流

```
文件在程序中是以流的形式操作的
1) 流：数据在数据源（文件）和程序（内存）之间经历的路径
2) 输入流：数据从数据源（文件）到程序（内存）的路径
3) 输出流：数据从程序（内存）到数据源（文件）的路径
```

二、常用文件操作

1、创建文件对象构造器

```
new File(String pathName)       // 根据路径创建一个File对象
new File(File parent, String child) // 根据父目录文件 + 子路径创建
new File(String parent, String child)   // 根据父目录 + 子路径创建
```

2、创建文件对象方法

```
createNewFile()     // 创建新文件
```

3、获取文件相关信息

```
getName          获取文件名字
getAbsolutePath  获取文件绝对路径
getParent        获取文件父级目录
length           文件大小（字节）
exists           文件是否存在
isFile           是不是一个文件
isDirectory      是不是一个目录
```

4、目录操作和文件删除

```
mkdir   创建一级目录
mkdirs  创建多级目录
delete  删除空目录或文件
```

三、IO流原理及流的分类

1、IO流原理

1）I/O是Input/Output的缩写，I/O技术是非常实用的技术，用于处理数据传输。如读/写文件，网络通讯等
2）Java程序中，对于数据的输入/输出操作以 "流（stream）"的方式进行
3）java.io包下提供了各种 "流" 类和接口，用以获取不同种类的数据，并通过方法输入或输出数据
4）输入input：读取外部数据（磁盘、光盘等存储设备的数据）到程序（内存）中
5）输出output：将程序（内存）数据输出到磁盘、光盘等存储设备中

2、流的分类

1）按操作数据单位不同分为：字节流（8 bit）二进制文件，字符流（按字符）文本文件
2）按数据流的流向不同分为：输入流，输出流
3）按流的角色的不同分为：节点流，处理流/包装流

(抽象基类)	字节流	字符流
输入流	InputStream	Reader
输出流	OutputStream	Writer

四、IO流体系图-常用类

1、InputStream-简介

ImputStream抽象类是所有类字节输入流的超类

2、InputStream-常用子类

```
FileInputStream          文件输入流
BufferedInputStream      缓冲字节输入流
ObjectInputStream        对象字节输入流
```

3、FileInputStream-构造器

构造器	说明
FileInputStream(File file)	通过打开一个到实际文件的连接来创建一个FileInputStream，该文件通过文件系统的File对象file指定
FileInputStream(FileDescriptor fdObj)	通过使用文件描述符 fdObj 创建一个 FileInputStream，该文件描述符表示到文件系统中某个实际文件的现有连接
FileInputStream(String name)	通过打开一个到实际文件的连接来创建一个FileInputStream，该文件通过文件系统中的路径名 name 指定

4、FileInputStream-方法

方法	说明
evailable()	返回下一次对此输入流调用的方法可以不受阻塞地从此输入流读取（或跳过）地估计剩余字节数
close()	关闭此文件输入流并释放与此流有关的所有系统资源
finalize()	确保在不再引用文件输入流时调用其close方法
getChannel()	返回与此文件输入流有关的唯一 FileChannel 对象
getFD()	返回表示到文件系统中实际文件的连接的 FileDescriptor 对象，该文件系统正被此 FileInputStream使用
read()	从此输入流中读取一个数据字节
read(byte[] b)	从此输入流中将最多 b.length 个字节的数据读入一个byte数组中
read(byte[] b, int off, int len)	从此输入流中将最多 len 个字节的数据读入一个 byte 数组中
skip(long n)	从输入流中跳过并丢弃n个字节的数据

5、FileInputStream-基本流程

```
1）创建FileInputStream对象
2）使用read()读取字符
3）关闭流，以释放资源
```

6、FileOutputStream-构造器

构造器	说明
FileOutputStream(File file)	创建一个向指定 File 对象表示的文件中写入数据的文件输出流
FileOutputStream(File file, boolean append)	创建一个向指定File对象表示的文件中写入数据的文件输出流
FileOutputStream(FileDescriptor fdObj)	创建一个向指定文件描述符处写入数据的输出文件流，该文件描述符表示一个到文件系统的某个位置的文件输出流
FileOutputStream(String name)	创建一个向具有指定名称的文件中写入数据的输出文件流
FileOutputStream(String name, boolean append)	创建一个向具有指定 name 的文件中写入数据的输出文件流

7、FileOutputStream-方法

返回值	方法	说明
void	close()	关闭此文件输出流并释放与此流有关的所有系统资源
protected	finalize()	清理到文件的连接，并确保在不再引用此文件输出流时调用此方法的close方法
FileChannel	getChannel()	返回与此文件输出流有关的唯一 FileChannel对象
FielDescriptor	getFD()	返回与此流有关的文件描述符
void	write(byte[] b)	将b.length个字节从指定byte数组写入此文件输出流中
void	write(byte[] b, int off, int len)	将指定byte数组中从偏移量off开始的len个字节写入此文件输出流
void	write(int b)	将指定字节写入此文件输出流

8、FileOutputStream-基本流程

```
1) 创建FileOutputStream流对象
2) 写入
3) 关闭
```

9、FileReader FileWriter-简介

```
FileReader和FileWriter都是字符流，即按照字符来操作IO
```

10、FileReader-方法

```
1) new FileReader(File/String)
2) read：每次读取单个字符，返回该字符，如果到文件末尾返回-1
3) read(char[])：批量读取多个字符到数组，返回读取到的字符数，如果到文件末尾返回-1

相关API
1) new String(char[])：将char[]转换成String
2) new String(char[], off, len)：将char[]的指定部分转换成String
```

11、FileWriter-方法

1) new FileWriter(File/String)：覆盖模式，相当于流的指针在首端
2) new FileWriter(File/String, true)：追加模式，相当于流的指针在尾端
3) write(int)：写入单个字符
4) write(char[])：写入指定数组
5) write(char[], off, len)：写入指定数组的指定部分
6) write(String)：写入整个字符串
7) write(String, off, len)：写入字符串的指定部分

相关API
1) String类：toCharArray：将String转换成char[]

注意：
FileWrite使用后，必须要关闭(close)或刷新(flush)，否则写入不到指定的文件

12、FileReader-基本流程

```
# 逐个字符读取
    1) 创建filePath属性、FileReader属性、data属性
    2) new FileReader对象并将filePath作为参数
    3) 使用filereader的read方法，将读取到的数据赋给data，在 != -1的情况下，循环打印char类型的data
    4) 如果filereader不等于null，关闭filereader
```

```
# 字符数组读取
    1) 创建filePath属性，FileReader属性、readLen属性、数组buf属性
    2) new FileReader对象并将filePath作为参数
    3) 使用filereader的read方法，将读取到的数据赋给readLen，在 != -1的情况下，循环打印new String(buf, 0, readLen)
    4) 如果filereader不等于null，关闭filereader
```

13、FileReader-基本流程

1) 创建filePath属性、fileWriter属性
2) new FileWriter对象并将filePath作为参数
3) 使用filewriter的write方法，将指定的字符写入
4) 关闭流

五、节点流和处理流

1、简介

1) 节点流可以从一个特定的数据源读写数据，如FileReader、FileWriter
2) 处理流（也称包装流）是"连接"在已存在的流（节点流或处理流）之上，为程序提供更为强大的读写功能，也更加灵活，如BufferedReader、BufferedWriter

2、节点流和处理流的区别

1）节点流是底层流/低级流，直接跟数据源相接
2）处理流包装节点流，既可以消除不同节点流的实现差异，也可以提供更方便的方法来完成输入输出
3）处理流（也称包装流）对节点流进行包装，使用了修饰器设计模式，不会直接与数据源相连

3、处理流功能

1）提高性能：主要以增加缓冲的方式来提高输入输出的效率
2）方便操作：处理流可能提供了一系列便捷的方法来一次输入输出大批量的数据，使用更加灵活方便

4、BufferedReader-基本流程

1）创建缓冲流BufferedReader对象，传入节点流对象作为实参
2）读取文件
3）关闭外层流

5、BufferedWriter-基本流程

1）创建缓冲流BufferedWriter对象，传入节点流对象作为实参
2）写入数据
3）关闭外层流

6、Buffered字符拷贝-基本流程

1）创建srcFilePath属性、destFilePath属性、line属性
2）初始化BufferedReader对象、BufferderWriter对象
3）实例化缓冲流对象，传入节点流FileReader作为实参，并传入srcFilePath作为FileReader的实参，把这里的值赋给初始化的BufferedReader对象
4）实例化缓冲流对象，传入节点流FileWriter作为实参，并传入destFilePath作为FileWriter的实参，把这里的值赋给初始化的BufferedWriter对象
5）while循环使用BufferedWriter对象的readline方法，在不等于空的情况下，使用write方法写入line数据
6）当BufferedReader和BufferedWriter对象都不为空的情况下，关闭流

7、BufferedInputStream-简介

BufferedInputStream是字节流，在创建BufferedInputStream时，会创建一个内部缓冲区数组。

8、BufferedOutputStream-简介

BufferedOutputStream是字节流，实现缓冲的输出流，可以将多个字节写入底层输出流中，而不必对每次字节写入调用底层系统。

9、Buffered字节拷贝-基本流程

1）创建srcFilePath属性、destFilePath属性、buff数组、readLen属性
2）初始化BufferedInputStream对象、BufferedOutputStream对象
3）实例化缓冲流对象，传入节点流FileInputStream作为实参，并传入srcFilePath作为FileInputStream的实参，把这里的值赋给初始化的BufferedInputStream对象
4）实例化缓冲流对象，传入节点流FileOutputStream作为实参，并传入srcFilePath作为FileOutputStream的实参，把这里的值赋给初始化的BufferedOutputStream对象
5）while循环使用BufferedInput对象的read方法，在不等于-1的情况下，使用write方法写入数组，并指定开始位置和长度
6）在BufferedInputStream和BufferedOutputStream分别不等于空的情况下，关闭流

10、序列化和反序列化

1）序列化就是在保存数据时，保存数据的值和数据类型
2）反序列化就是在恢复数据时，恢复数据的值和数据类型
3）需要让某个对象支持序列化机制，则必须让其类时可序列化的，为了让某个类是可序列化的，该类必须实现如下两个接口之一：

```
Serializable    // 推荐
Externalizable
```

4）ObjectOutputStream提供序列化功能，ObjectInputStream提供反序列化功能

11、ObjectOutputStream序列化-基本流程

1）创建流对象，加入FileOutputStream实参和FileOutputStream的实参filePath
2）写入对象
3）关闭流

12、ObjectInputStream反序列化-基本流程

1）创建流对象，加入FileInputStream实参和FileInputStream的实参filePath
2）使用readObject方法反序列化数据，并且反序列化的顺序必须和序列化顺序一致
3）关闭流

13、序列化和反序列化-细节

1）读写顺序要一致
2）要求序列化或反序列化对象，需要实现 Serializable
3）序列化的类中建议添加SerialVersionUID，可以提高版本的兼容性
4）序列化对象时，默认将里面所有属性都进行序列化，但出了static或transient修饰的成员
5）序列化对象时，要求里面属性的类型也需要实现序列化接口
6）序列化具备可继承性，也就是如果某类已经实现了序列化，则它的所有子类也已经默认实现了序列化

14、标准输入输出流

```
System.in   标准输入
System.out  标准输出
```

15、转换流

1）InputStreamReader是Reader的子类，可以将InputStream(字节流)包装成Reader(字符流)
2）OutputStreamWriter是Writer的子类，可以将OutputStream(字节流)包装成Writer(字符流)
3）当处理纯文本数据时，如果使用字符流效率更高，并且可以有效解决中文问题，所以建议将字节流转换成字符流
4）可以在使用时指定编码格式(如 utf-8，gbk，gb2312，ISO8859-1等)

16、InputStreamReader-基本流程

1）创建filePath属性
2）实例化InputStreamReader，传入实参 new FileInputStream(filePath)，指定编码格式
3）实例化BufferedReader，传入实参InputStreamReader对象
4）使用BufferedReader的readLine方法，逐行读取内容
5）关闭BufferedReader流

17、OutputStreamWriter-基本流程

1）创建filePath属性、charSet属性
2）实例化OutputStreamWriter，传入实参 new FileOutputStream(filePath)和charSet
3）使用OutputStream的write方法，写入字符
4）关闭OutputStreamWriter流

六、打印流

1、简介

打印流分为PrintStream和PrintWriter，只有输出流没有输入流。

2、PrintStream-基本流程

1）将System.out赋值给PrintStream对象
2）使用PrintStream的print方法打印
3）关闭PrintStream流

3、PrintWriter-基本流程

1）实例化PrintWriter，传入实参 new FileWriter("e:\\xxx.txt")
2）使用PrintWriter的print方法，写入字符
3）关闭PrintWriter流

七、Properties类

1、简介

```
    Properties类是专门用于读写配置文件的集合类，配置文件格式：键=值
```

2、常用方法

```
    1) load              加载配置文件的键值对到Properties对象
    2) list              将数据显示到指定设备
    3) getProperty(key)  根据键获取值
    4) getProperty(key, value)  设置键值对到Properties对象
    5) store             将Properties中的键值对存储到配置文件，在idea中，保存信息到配置文
件，如果含有中文，会存储为unicode码
```