输入流和输出流 – IS与OS

IO流：

1. InputStream：抽象类，所有字节输入流的父类
2. OutputStream：抽象类，所有字节输出流的父类
3. 输入和输出是以内存（虚拟内存）为主导，读取外界数据使用read()，写出程序中的数据到外界使用输出write()
4. 节点流（低级流）：可以从一个特定的地方读取/写入数据
5. 处理流（高级流）：对一个已经存在的流进行封装，它不会单独存在，是建立在低级流的基础之上，高级流是可以嵌套使用

第一组实现流： 低级流

1. FileInputStream：文件输入流
   1. Read() read(byte[] buf)
2. FileOutPutStream：文件输出流
   1. Write() write(byte[] buf)
      1. Wirte（byte[] buf, 0 ,length）
3. New FileOutputStream(String name, Boolean append)
   1. True – 追加
   2. False – 不追加
   3. 若是写false，等同于new FileOutputStream内容不追加，当前文本的内容都会被清除，那后在重新写入，这一点与RandomAccessFile不同

第二组实现流：缓冲字节流 高级流

1. BufferedInputStream：缓冲字节输入流—read()
2. BufferedOutputStream：缓冲字节输出流—write()
3. 这两个流内部各自维护着一个缓冲区（自带的缓冲区）
4. 自带缓冲区的输出流，只有在缓冲区存满的时候才会将缓冲区的数据自动写出，如果没有存满，则不写出
   1. 两种方式将未满的缓冲区数据写出：
      1. 关流
      2. 调用flush()方法，属于强制性操作（会降低读写效率，为了保证信息的及时性和有效性，需要调用该方法）
   2. 关流的时候只需要关闭高级流即可，低级流会随着高级流的关闭而关闭，
   3. 自带缓冲区和外加的缓冲区（字节数组）的区别：
   4. 在相同字节长度情况下，FileInputStream/FileOutputStream加换成字节数组，要比BufferedInputStream/BufferedOutputStream自带的缓冲区数据的读取和写入要快一些，因为少了一个“换车”的环节

第三组实现流： 对象流 高级流

1. 可以直接保存一个对象
   1. 将对象转成字节
   2. 长久保存在硬盘中
   3. 将一个特定的数据结构转成一组字节的过程—序列化
   4. 将数据写入到硬盘中长久保存的过程—持久化
   5. 存：先将对象转成字节—保存 序列化
   6. 取：将字节转成特定的数据结构—取出 反序列化
2. 序列化和反序列一般用于
   1. 传输
   2. 保存
3. ObjectInputStream:对象输入流  
   readObject():读取一个对象
4. ObjectOutputStream:对象输出流

witeObject()：写入一个对象

1. 如果一个类的对象想要实现序列化，必须先实现可序列化的接口，这时该类的对象，才有资格进行序列化
2. 序列化的接口：serializable
3. readObject():方法读取的是一个对象，返回值为Object类型，需要进行强制类型转换，才能进行对应类型中的属性和成员方法的调用
4. 该类在序列化之后，反序列化之前，对该类的属性/方法进行修改会出现报错，为了避免这种报错出现，需要在序列化之前增加一个serialVersionUID（版本号）可以避免这种出错
5. transient：修饰的成员变量在序列化时，可以忽略其属性值，在反序列化后，不保留该属性的值，但是会有该属性的存在，且有对应的默认值