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