第十四天：

目标：

Stream I/O and Files, 共 44 个slide(411-455);

知识点：

一. 流的概念

java.io.InputStream和**java.io.OutputStream**分别表示**字节输入流**和**字节输出流**，**java.io.Reader**和**java.io.Writer**分别表示**字符输入流**和**字符输出流**。

二. 字节输入流和输出流概述

在java.io包中，**java.io.InputStream**表示字节输入流，**java.io.OutputStream**表示字节输出流，它们都是抽象类，不能被实例化。

InputStream类提供了一系列和读取数据有关的方法：文件中的数据都是string类型

1. **read**():

a. int read(): 从输入流读取一个8位的字节，把它转换为0-255之间的整数，并返回这一整数

1000000000001001(源码－－>补吗) 取反＋1 1111111101110111(只要后8位) 01110111

b. int read(byte[] b): 返回的整数表示读取的字节数。

使用byte[]数组可以降低物理读取次数

byte[]长度最大可以取到整个要读取的数据的长度

需要注意,数组设定的长度如果不能一次读取完内容.第二次读取时会先将剩下的内容放入到数组中(从第一个位置开始替换直到所有数据都进来,如果数组中还有剩余空间,后面没有替换的不受影响)

c. int read(byte[] b, int off, int len):

需要注意测试如果继续打印保存在数组中的内容,len后面的内容第一次没有读取到.会将他们覆盖到源来的数组中,后面没有覆盖的不影响.

2. void close(): 关闭输入流，InputStream类本身的close()方法不执行任何操作。它的一些子类覆盖了close()方法，

3. int available(): 返回可以从输入流中读取的字节数目；此方法是返回这个流中有多少个字节数，可以把数组长度定为这个

4. skip(long n): 从输入流中跳过参数n指定数目的字节.

5. boolean markSupported()，void mark(int)，void reset():

mark就像书签一样，在这个BufferedReader对应的buffer里作个标记，以后再调用reset时就可以再回到这个mark过的地方。mark方法有个参数，通过这个整型参数，你告诉系统，希

望在读出这么多个字符之前，这个mark保持有效。读过这么多字符之后，系统可以使mark不再有效，

需要注意还和buffer的缓冲区大小有关.

BufferedInputStream类调用mark(int readlimit)方法后读取多少字节标记才失效，是取**readlimit**和**BufferedInputStream**类的缓冲区大小两者中的最大值，而并非完全由readlimit确定。

1. write(): 向输出流写入数据：有三种重载形式：

a. void write(int b): 向输出流写入一个字节；read()

b. void write(byte[] b): 把参数b指定的字节数组中的所有字节写到输出流；read(byte[])

c. void write(byte[] b, int off, int len): 把参数b指定的字节数组中的所有字节写到输出流，参数off指定字节数组的起始下标，从这个位置开始输出由参数len指定数目的字节；

采用后面两个write方法可以减少进行物理读文件或键盘的次数，因此能提高I/O操作的效率。

2. void close(): 关闭输出流，OutputStream类本身的close()方法不执行任何操作。它的一些子类覆盖了close()方法，在close()方法中释放和流有关的系统资源。

3. void flush(): OutputStream类本身的flush()方法不执行任何操作，它的一些带有缓冲区的子类(比如BufferedOutputStream和PrintStream类)覆盖了flush()方法。通过带缓冲区的输出流写数据时，数据先保存在缓冲区中，积累到一定程度才会真正写到输出流中。缓冲区通常用字节数组实现，实际上是指一块内存空间。

flush:即使内容没有达到缓冲的标准大小,仍然输出到输出流中.

只要使用到buffer的流,就一定要进行flush.最后很可能有数据没有

完全达到要求而不能输出.

三. 输入流和输出流层级结构

1. ByteArrayInputStream: 把字节数组转换为输入流；

ByteArrayInputStream类有两个默认的构造函数：

ByteArrayInputStream(byte[] b): 使用一个字节数组当中所有的数据做为数据源，程序可以像输入流方式一样读取字节，可以看做一个虚拟的文件，用文件的方式去读取它里面的数据。

ByteArrayInputStream(byte[] b,int offset,int length): 从数组当中的第offset开始，一直取出length个这个字节做为数据源

ByteArrayOutputStream类也有两个默认的构造函数：

ByteArrayOutputStream(): 创建一个32个字节的缓冲区

ByteArrayOutputStream(int): 根据参数指定大小创建缓冲区

这两个构造函数创建的缓冲区大小在数据过多的时候都会自动增长，如果创建缓冲区以后，程序就可以把它像虚拟文件一样似的往它里面写入内容，当写完内容以后调用ByteArrayOutputStream()的方法就可以把其中的内容当作字节数组返回

2. FileInputStream: 从文件中读取数据；

3. PipedInputStream: 连接一个PipedOutputStream;

4. SequenceInputStream: 把几个输入流转换为一个输入流；

5. ObjectInputStream: 对象输入流；

6. FilterInputStream:　　过滤器，扩展其它输入流功能；

讲解：ch14.ByteArrayTester.java

讲解：ch14.FileInputStreamTester.java

字符　　编码值
a　　->　　97
b　　->　　98
c　　->　　99
1　　->　　49
空格　　->　　32

讲解：ch14.UseBuffer.java

四. BufferedInputStream类

BufferedInputStream类覆盖了被过滤的输入流的读数据行为，利用缓冲区来提高读数据的效率。BufferedInputStream类先把一批数据读入到缓冲区，接下来 read()方法只需要从缓冲区内获取数据，就能**减少物理性读取数据的次数**。

. BufferedInputStream(InputStream in)——参数in指定需要被过滤的输入流。
. BufferedInputStream(InputStream in, int size)——参数in指定需要被过滤的输入流。参数size指定缓冲区的大小，以字节为单位。

五. DataInputStream 类

注意1：一般情况下在读入时尽量按照写入时的格式进行读取，
否则有可能会出现显示乱码或程序出现异常。
如首先写入文件用的是writeUTF()，在读取的时候如果不是用readUTF()就会出现乱码，
如果readUTF()读取的内容不是UTF-8格式的，程序就会抛出异常。
java.io.EOFException
注意2：如程序中注释所说，对于出现汉字字符的情况不能用writeBytes()，这会在**写入文件时丢弃汉字字符的第一个字节**从而在读取时出现错误。
注意3：所有的读取方法都是共享一个位置指示器的，即在前面的read方法执行后，后面再执行其他read方法都是从上一个read方法读取到的位置开始向后读取的。如开始执行了1次readByte()后面的readChar是从第2个字节开始读的。

DataInputStream 实现了DataInput接口，用于读取基本类型数据，如int, float, long, double和boolean等。
必须先使用DataOutputStream写入数据，然后使用DataInputStream读取数据方可
DataOutputStream本来就是用来写二进制数据的(效率高)，你直接打开文件,看到的是乱码
. readByte()——从输入流中读取1个字节，指它转换为byte类型的数据；
. readLong()——从输入流中读取8个字节，指它转换为long类型的数据；
. readFloat()——从输入流中读取4个字节，指它转换为float类型的数据；

. readUTF()—— 从输入流中读取1到3个字节，指它转换为UTF-8字符编码的字符串；

讲解：ch14.FormatDataIO.java

六. 管道输入类：PipedInputStream 类

管道输入流从一个管道输出流中读取数据。通常由一个线程向管道输出流写数据，由另一个线程从管理输入流中读取数据，

两个线程可以用管道来通信。

管道流，用于线程间的通信。一个线程的PipedInputStream对象从另外一个线程的PipedOutputStream对象读取输入。

要使管道流有用，必须同时构造管道输入流和管道输出流

讲解：ch14.PipedTest.java

七. Reader and Writer概述

InputStream和OutputStream类处理的是字节流，也就是说，数据流中的最小单元为一个字节，它包括8个二进制位。在许

多应用场合，Java应用程序需要读写文本文件。在文本文件中存放了采用特定字符编码的字符。为了便于读于各种字符

编码的字符，java.io包中提供了Reader/Writer类，它们分别表示字符输入流和字符输出流

window,mac,unix:utf-8

在处理字符流时，最主要的问题是进行字符编码的转换。Java语言采用Unicode字符编码。对于每一个字符，Java虚拟机会

为其分配两个字节的内存。而在文本文件中，字符有可能采用其他类型的编码，比如GBK和UTF-8字符编码等。

Reader类能够将输入流中采用其他编码类型的字符转换为Unicode字符，然后在内存中为这些Unicode字符分配内存。Writer

类能够把内存中的Unicode字符转换为其他编码类型的字符，再写到输出流中。

在默认情况下，Reader和Writer会在本地平台的字符编码和Unicode字符编码之间进行编码转换。

Writer的write()方法

使用Unicode字符编码的字 ----------> 使用本地平台的字符编码的字符串

符串(内存中) <---------- (数据源/数据目的地)

Reader的read()方法

如果要输入或输出采用特定类型编码的字符串，可以使用InputStreamReader类和OutputStreamWriter类。在它们的构造方法

中可以指定输入流或输出流的字符编码。

OutputStreamWriter的write()方法
使用Unicode字符编码的字 ----------> 使用本地平台的字符编码的字符串
符串(内存中) <---------- (数据源/数据目的地)
UTF-8 InputStreamReader的read()方法 EUC_CN

由于Reader和Writer采用了字符编码转换技术，Java I/O系统能够正确地访问采用各种字符编码的文本文件，另一方面，在为
字符分配内存时，虚拟机对字符统一采用Unicode字符编码，因此Java程序处理字符具有平台独立性。

CharArrayReader : 把字符数组转换为Reader，从字符数组中读取字符；
BufferedReader : 过滤器，为其他Reader提供读缓冲区，此外，它的readLine()方法能够读入一行字符串；
StringReader : 把字符串转换为Reader，从字符串中读取字符；
PipedReader : 连接一个PipedWriter;
PushBackReader : 能把读到的字符压回到缓冲区中，通常用做编译器的扫描器，在程序中一般很少使用它。
InputStreamReader : 过滤器，把InputStream转换为Reader，可以指定字符编码；
FileReader : 从文件中读取字符；

八. InputStreamReader类

InputStreamReader类把InputStream类型转换为Reader类型，构造方法：

. InputStreamReader(InputStream in): 按照本地平台的字符编码读取输入流中的字符；
. InputStreamReader(InputStream in, String charsetName): 按照指定的字符编码读取输入流中的字符；

九. FileReader类

InputStreamReader的一个子类，用于从文件中读取字符数据。该类只能按照本地平台的字符编码来读取数据，用户不能
指定其他字符编码类型。

. FileReader(File file): 参数file指定需要读取的文件；
. FileReader(String name): 参数name指定需要读取的文件的路径；

十. BufferedReader类

Reader类的read()方法每次都从数据源读入一个字符，BufferedReader带有缓冲区，它可以先把一批数据读到缓冲区内，

接下来的操作都从缓冲区内获取数据，避免每次都从数据源读取数据并进行字符编码转换，从而提高读操作的效率。

BufferedReader的readLine()方法可以一次读入一行字符，以字符形式返回。

. BufferedReader(Reader in): 指定被修饰的Reader类； 默认8192字节=8Mbit

. BufferedReader(Reader in, int sz): 参数in指定被装饰的Reader类，参数sz指定缓冲区的大小，以字符为单位。

动态扩展内存中的缓冲区,可以设定默认缓冲区的大小，但是使用默认和不默认区别是有的，在系统资源有限的紧迫的时候就很重要了，当然这个是在高级应用的时候才需要，不管有没有默认值只要超过长度就会自动增加，但是这个增加是需要额外的花费的。如果频繁增加更加影响效率。

大多数情况下，默认值就足够大了

十一. File类

File类提供管理文件或目录的方法。File实例表示真实文件系统中的一个文件或者目录。

1. 构造方法

. File(String pathname): 参数pathname表示文件路径或者目录路径；

. File(String parent, String child): 参数parent表示根路径，参数child表示子路径；

. File(File parent, String child): 参数parent表示根路径，参数child表示子路径；

只处理一个文件，使用第一个构造方法；如处理一个公共目录的若干子目录或文件，那么使用第二个或者第三个更方便。

2. 普通方法

路径问题:

getAbsolutePath():获取绝对路径(从根路径开始 /)

getParent():获取父目录的路径(如果构建file时使用的时绝对路径 此时获得父目录的绝对路径.如果传入时相对路径,此时获得父目录的相对路径)

getPath():获得构建file时创建时候路径(一模一样)

. boolean createNewFile()：创建一个新的文件，如果文件已经存在，则创建失败（返回false），否则创建成功（返回true）。

. boolean delete()：删除文件或者空目录

. boolean mkdir()/mkdirs()：创建一个或者多个目录（连续创建）->如果该目录的父目录不存在话，那么还会创建所有的父目录；

. boolean renameTo(File destination)：文件的改名

. boolean canRead()/canWrite()：判断指定的文件是否能读取或者写入数据

. boolean exists()：判断指定的文件或者目录是否存在

. String[] list()：返回指定目录下所有文件名或者子目录名所组成的字符串数组

. long lastModified()：返回指定文件最后一次被修改的时间（从1970年1月1日凌晨12点到这个文件的修改时间
之间所经历的毫秒数）

. String getPath()/getAbsolutePath()：返回指定文件或者目录的路径和绝对路径

. String getCanonicalPath(): 获取该File对象所代表的文件或者目录的正规路径
可以忽略掉'./'对应的内容

. String getParent()/getName()：返回指定文件或者目录的父目录（没有返回null）和名字

```
File f = new File(".\\test.txt"));
System.out.println(f.getCanonicalPath());        //c:\mypath\test.txt
System.out.println(f.getAbsolutePath());         //c:\mypath\ .\test.txt
System.out.println(f.getPath());                 //.\test.txt
if(!f.exists()) f.createNewFile());
        public static void test1(){
        File file1 = new File(".\\test1.txt");
        File file2 = new File("D:\\workspace\\test\\test1.txt");
        System.out.println("-----默认相对路径：取得路径不同------");
        System.out.println(file1.getPath());
        System.out.println(file1.getAbsolutePath());
        System.out.println("-----默认绝对路径:取得路径相同------");
        System.out.println(file2.getPath());
        System.out.println(file2.getAbsolutePath());
         }
        -----默认相对路径：取得路径不同------
        .\test1.txt
        D:\workspace\test\.\test1.txt
        -----默认绝对路径:取得路径相同------
        D:\workspace\test\test1.txt
        D:\workspace\test\test1.txt
```

讲解：ch04.FileEx.java

十一. FileInputStream and FileOutputStream

1. 当创建一个FileInputStream对象的时候，文件必须存在以及是可读的
FileInputStream(File file)
FileInputStream(String name)

2. 当创建一个FileOutputStream对象的时候，可以创建一个新的文件，也可以覆盖一个已经存在的同名文件。
FileOutputStream(File file)
FileOutputStream(File file, boolean append)
如果要创建的文件已经存在，可以选择向旧文件添加新的内容（append为true）或者新的内容覆盖旧文件的
内容（append为false）。

FileOutputStream(String name)
FileOutputStream(String name, boolean append)
如果要创建的文件已经存在，可以选择向旧文件添加新的内容（append为true）或者新的内容覆盖旧文件的
内容（append为false）。

注意：在这里写个com.briup.ch14.CopyTime.java

十二. FileReader and FileWriter

1. 读写字符文件方便
2. 构造器
FileReader(File file)
FileReader(String name)
FileWriter(File file)
FileWriter(String filename)

FileReader：new FileReader(“d:/back/string.txt”) =
new InputStreamReader(new FileInputStream(“d:/back/string.txt”));
FileWriter：new FileWriter(“d:/back/string.txt”) =
new OutputStreamWriter(new FileOutputStream(“d:/back/string.txt”));

注意：这里可以将com.briup.ch14.FileStringTest.java用FileReader修改一下，并打印系统默认的字符编码

十三. PrintWriter

可以输出基本数据类型、对象、字符（字符数组）和字符串，但是不能输出字节流。

PrintWriter类和BufferedWriter类
两种流可以相互封装
PrintStream类既可以输出字符串
Java代码 收藏代码
public void println(Object obj) {
write(String.valueOf(obj)+”\n”);
}
这是print方法的源码
可以看到其实print方法就是调用write方法实现的，只不过将object转换成String了，其它的应该没有区别

十四. Reading and Writing with RandomAccessFile

1. 实现数据的输入和输出

2. 用它能够提供读写文件的功能
3. 提供通过一个文件指针(开始读取位置)从文件的某一个断点开始读写数据的功能
4. 构造器
   RandomAccessFile(File file, String mode)
   RandomAccessFile(String filename, String mode)
   mode可以选择读、写和读写的方式
5. 方法
   read()/write() seek(long pointer) 定位到文件的断点

RandomAccessFile

RandomAccessFile是用来访问那些保存数据记录的文件的，你就可以用seek( )方法来访问记录，并进行读写了。这些记录的大小不必相同；但是其大小和位置必须是可知的。但是该类仅限于操作文件。

RandomAccessFile不属于InputStream和OutputStream类系的。实际上，除了实现DataInput和DataOutput接口之外(DataInputStream和DataOutputStream也实现了这两个接口)，它和这两个类系毫不相干，甚至不使用

InputStream和OutputStream类中已经存在的任何功能；它是一个完全独立的类，所有方法(绝大多数都只属于它自己)都是从零开始写的。这可能是因为RandomAccessFile能在文件里面前后移动，所以它的行为与其它的I/O类有些根

本性的不同。总而言之，它是一个直接继承Object的，独立的类。

基本上，RandomAccessFile的工作方式是，把DataInputStream和DataOutputStream结合起来，再加上它自己的一些方法，比如定位用的getFilePointer( )，在文件里移动用的seek( )，以及判断文件大小的length( )、

skipBytes()跳过多少字节数。此外，它的构造函数还要一个表示以只读方式("r")，还是以读写方式("rw")打开文件的参数 (和C的fopen( )一模一样)。它不支持只写文件。

只有RandomAccessFile才有seek搜寻方法，而这个方法也只适用于文件。BufferedInputStream有一个mark( )方法，你可以用它来设定标记(把结果保存在一个内部变量里)，然后再调用reset( )返回这个位置，但是它的功能太

弱了，而且也不怎么实用

"r" 以只读方式打开。调用结果对象的任何 write 方法都将导致抛出 IOException。

"rw"打开以便读取和写入。如果该文件尚不存在，则尝试创建该文件。

"rws"打开以便读取和写入，对于 "rw"，还要求对文件的内容或元数据的每个更新都同步写入到底层存储设备。

"rwd"打开以便读取和写入，对于 "rw"，还要求对文件内容的每个更新都同步写入到底层存储设备。 JDK 1.6上面写的

每次write数据时，"rw"模式，数据不会立即写到硬盘中；而“rwd”，数据会被立即写入硬盘。如果写数据过程发生异常，“rwd”模式中已被write的数据被保存到硬盘，而“rw"则全部丢失

十五. 对象的序列化和反序列化

对象的序列化： 把对象写到一个输出流；

对象的反序列化：从一个输入流中读取一个对象；

1. 对象的持久化
2. 仅仅是一个对象的数据被序列化（将对象的数据序列化成字节流）
3. 标识为transient的数据不能被序列化 例如：transient 类名 表示该类不能被序列化 或者transient 字段
4. 要序列化的对象必须实现java.io.Serializable接口

对象的序列化主要用于：
1. 网络中传输的是字节流的数据，网络中对象的传输，是将对象的数据经过序列化后转换成字节流。
2. 将对象数据序列化到文件中，将对象数据转换成字节流存储到文件中。
从文件中读取字节流数据并转换成对象叫做对象的反序列化。

ObjectInputStream 和ObjectOutputStream(对象输入和输出流，可以读写基本数据类型和对象)
1. ObjectInputStream 和ObjectOutputStream为应用程序提供对象的持久化存储
2. 一个ObjectInputStream 可以反序列化通过ObjectOutputStream写入的基本数据类型和对象
3. 构造器
ObjectInputStream(InputStream in)
ObjectOutputStream(OutputStream out)
4. 方法
readObject()/writeObject() 将对象写入到输出流中或者从输入流中读取对象

注意：这里写com.briup.ch14.ObjectSerial.java 将一个Person类的对象序列化到文件中，再将对象从文件中反序
列化出来。
(Person和Address先不序列化测试一下，再序列化测试一下)
com.briup.ch14.CollectionSerialTest.java 将一个集合序列化到文件中，再将集合从文件中反序列化出来。
com.briup.ch14.BufferSerialTest.java 将一个Person类的对象序列化到缓存中，再将对象从缓存中反序列化出来。

Review Questions :

1.(Level 1)What are the main classes for input stream?
答：InputStream, DataInputStream, FileInputStream, ByteInputStream, BufferedInputStream
2.(Level 1)What are reader/writer classes?
答：Reader类能够将输入流中采用其他编码类型的字符转换为Unicode字符，然后在内存中为这些Unicode字符分配内存。 Writer类能够把内存中的Unicode字符转换为其他编码类型的字符，再写到输出流中。
3.(Level 1)What are the bridges between InputStream and Reader?
答：InputStreamReader
4.(Level 1)What is object serialization?
答：把对象写到一个输出流；
5.(Level 2)The File class contains a method that changes the current working directory
A.True B.False
答：B

6.(Level 2)It is possible to use the File class to list the contents of the current working directory

A.True B.False

答：A

7.(Level 2)Readers have methods that can read and return floats and doubles

A.True B.False

答：B

8.(Level 3)How many bytes does the following code write to file dest ?

```
1.try{
2.FileOutputStream fos = new FileOutputStream(“dest”);
3.DataOutputStream dos = new DataOutputStream(fos);
4.dos.writeInt(3);
5.dos.writeDouble(0.001);
6.dos.close();
7.fos.close();
8.}catch(IOException e){}
```

A.2
B.8
C.12
D.16
E.It depends on the underlving system

答：C