流与文件（二）

武永亮

讲授思路

- 字符流
- 对象流
- 其他常用流

讲授思路-字符流

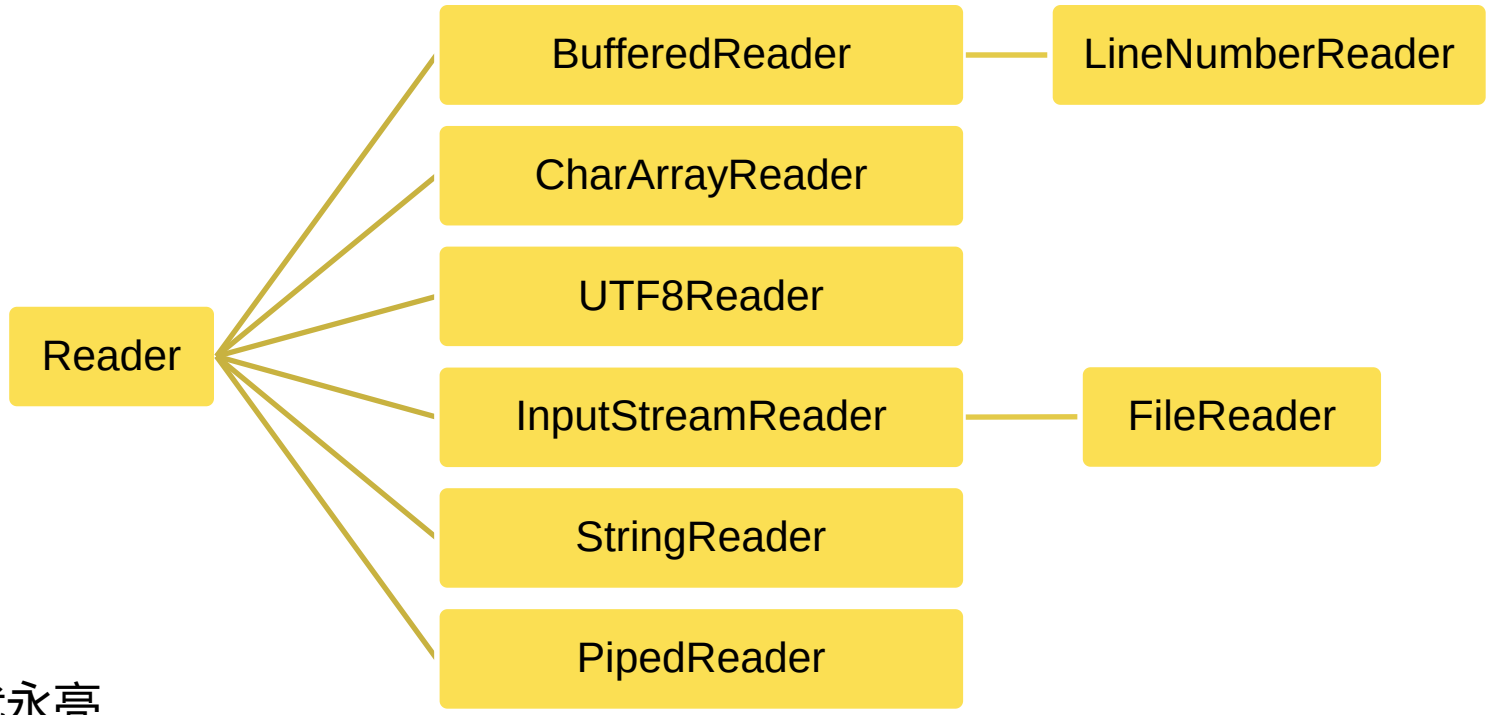
- 字符流的基本概念
- Java中的字符输入流Reader
- Java中的字符输出流Writer

Java中字符流的处理

- 字符流的输入输出主要以字符为单位。主要用于外部设备中字符序列的读取和外部设备中字符序列的显示。Java采用16位的Unicode来表示字符串和字符。
- 对于字符流的资源比较单一，就是字符序列。

Java中字符输入流处理Reader

- Java中所有的字符输入流都用Reader表示，读取单位为1字符，2字节（16位）
- Reader类是一个抽象类，我们一般通过它的子类来使用



Java中字符输入流处理Reader

- Java中Reader的常用方法：
  - int read();读取单个字符。
  - int read(char c[ ]);将字符读取数组 。
  - int read(char c[ ], int off, int len);将字符读入数组的某一部分。
  - void mark(int readlimit);标记流中的当前位置。
  - void reset();重置该流。
  - void close();关闭此输入流并释放与该流关联的所有系统资源。
  - long skip(long n);跳过字符
- 参考JDK_API 1.6.0 文档

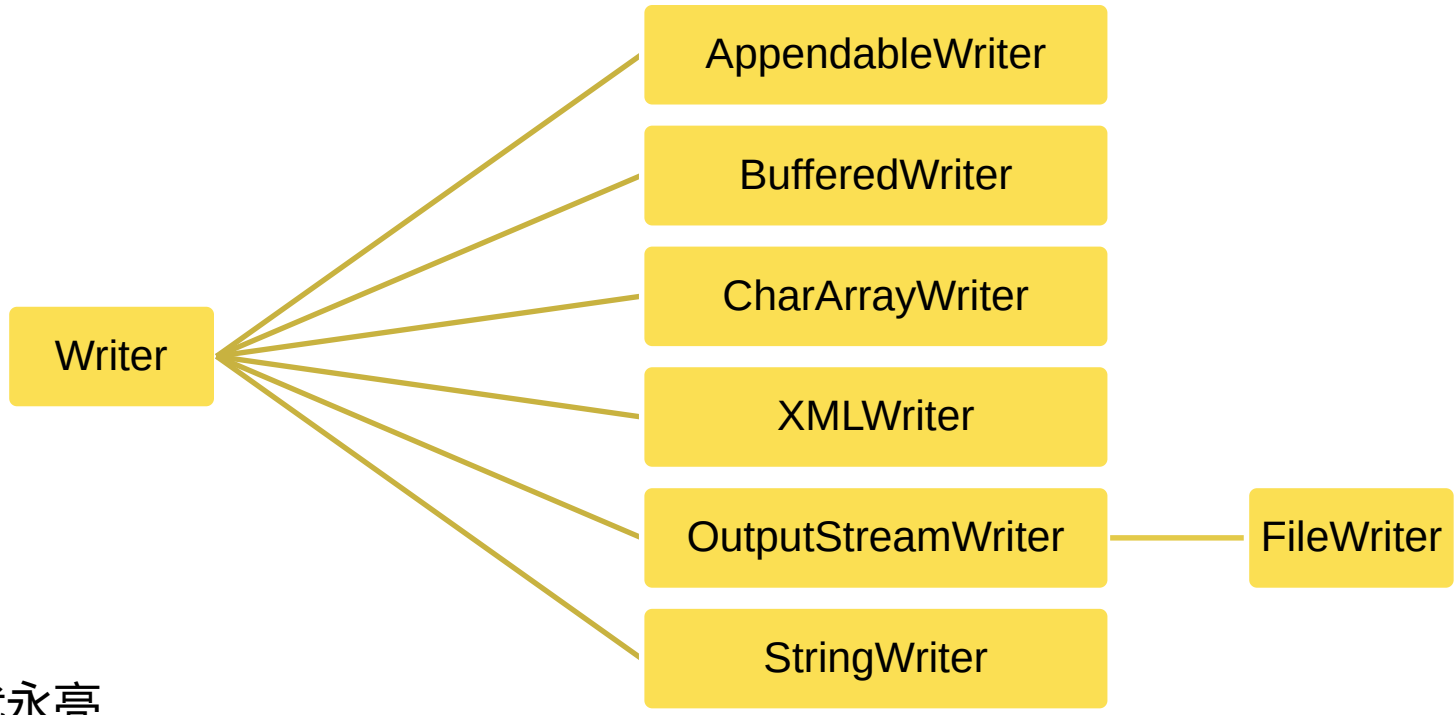
课堂练习：

- 读取本地路径的一个文件C:/a.txt文件

```
//通过字符流读取文件内容
Reader r = new FileReader(new File("C://test.txt"));
char[] cbuf = new char[256];
int len = r.read(cbuf);
System.out.println(new String(cbuf, 0, len));
r.close();
```

Java中字符输出流处理Writer

- Java中所有的字符输出流都用Writer表示，读取单位为1字符，2字节（16位）
- Writer类是一个抽象类，我们一般通过它的子类来使用



Java中字符输出流处理Writer

- Java中Writer的常用方法：
  - Writer append(char c);将指定字符添加到此 writer。
  - Writer append(CharSequence c);将指定字符序列添加到此 writer。
  - Writer append (CharSequence csq, int start, int end) ;将指定字符序列的子序列添加到此 writer.Appendable。
  - void flush();刷新该流的缓冲。
  - void write(int c);写入单个字符。
  - void write(String str);写入字符串。
  - void close();关闭此流，但要先刷新它。
- 参考JDK_API 1.6.0 文档

课堂练习：

- 读取本地路径的一个文件C:/a.txt文件
- 写入本地路径的一个文件D:/res.txt文件

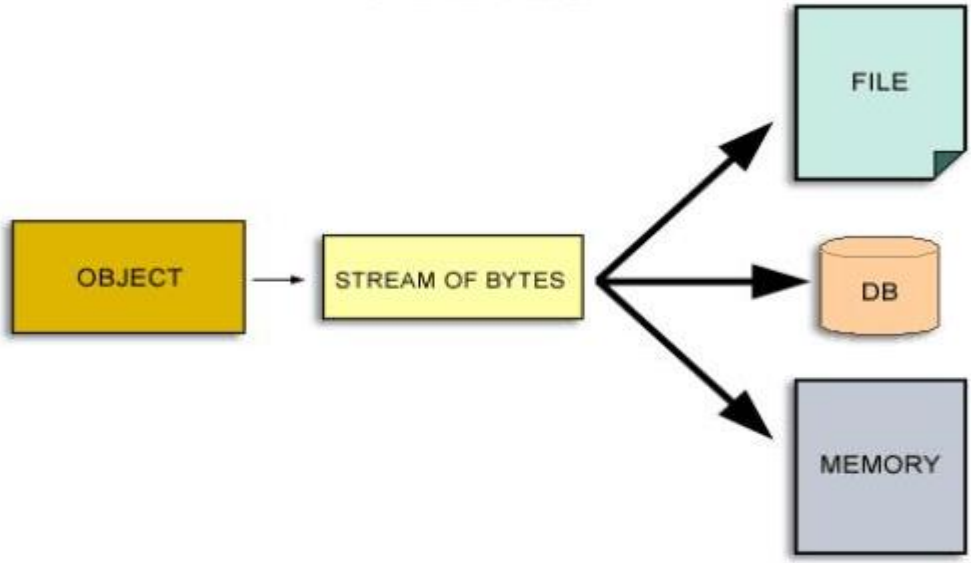
```
//通过字符流写入文件
Writer w = new FileWriter(new File("C://result.txt"));
w.write(cbuf);
w.close();
```

讲授思路-对象流

- 序列化的概念
- 序列化的步骤

序列化的概念

- Java允许我们在内存中通过定义对象来保存对象状态，但是当JVM停止时对象就无法保存。
- 如何持久化保存对象状态呢？
- Java对象序列化就能够帮助我们实现该功能。在保存对象时把其转换为一组字节，等待需要时将自己组装回之前的对象。



对象序列化注意事项

- 类必须实现Serializable或Externalizable接口才能被序列化
- 实现Serializable的类中若有不参与序列化的变量可用transient关键字修饰
- 用static修饰的静态成员变量，不参加序列化过程

对象的序列化实现

ObjectInputStream

- 对象输入流，它的readObject( ) 方法从一个源输入流中读取字节序列，再反序列化为一个对象

ObjectOutputStream

- 对象输出流，它的writeObject( )方法可对参数指定的对象序列化并把得到的字节序列写到目标输出流中

对象的序列化、反序列化的步骤

- 对象序列化步骤
  - 创建一个ObjectOutputStream对象
  - 通过ObjectOutputStream对象的writeObject(object)方法输出对象
- 对象反序列化步骤
  - 创建一个ObjectInputStream
  - 通过ObjectInputStream的readObject()方法读取对象

课堂练习

- 自定义类Person，具有name，life属性。希望能够在运行中保存对象数据。

```
//将对象序列化后存入文件person.out
ObjectOutputStream oout = new ObjectOutputStream(new FileOutputStream(file));
oout.writeObject(p);
oout.close();

//读取文件，反序列化后得到对象
ObjectInputStream oin = new ObjectInputStream(new FileInputStream(file));
Object newPerson = oin.readObject();
oin.close();
System.out.println(newPerson);
```

讲授思路-其他常用流

- BufferedReader和BufferedWriter
- DataOutputStream和DataInputStream
- InputStreamReader和OutputStreamWriter

BufferedReader和BufferedWriter

- BufferedReader和BufferedWriter：可以直接读一行字符或写一行字符。
- BufferedReader：从字符输入流中读取文本，缓冲各个字符，从而实现字符、数组和行的高效读取。
  - 建议用BufferedReader如 FileReader 和 InputStreamReader等，因为他们的read()操作可能开销很高。
- BufferedWriter：将文本写入字符输出流，缓冲各个字符，从而提供单个字符、数组和字符串的高效写入。
  - 建议用BufferedWriter包装FileWriter、OutputStreamWriter等，因为他们的write()操作开销很高。

BufferedReader和BufferedWriter

- 使用BufferedReader和BufferedWriter对Unicode字符文件进行读取和写入操作？

```
File file = new File("d://帐户信息.txt");
FileReader fr = new FileReader(file);
BufferedReader br = new BufferedReader(fr);
//使用BufferedReader包装FileReader
File file_copy = new File("D://帐户信息-copy.txt");
FileWriter fw = new FileWriter(file_copy);
BufferedWriter bw = new BufferedWriter(fw);
//使用BufferedWriter包装FileWriter
String str = br.readLine();
while (str != null) {
                bw.write(str);
                str = br.readLine();
}
bw.flush();
bw.close();
fw.close();
br.close();
fr.close();
```

DataOutputStream和DataInputStream

- DataInputStream和DataOutputStream:数据输入/输出流。
- 允许应用程序以与机器无关方式从底层输入流中读取基本 Java 数据类型，两个类中提供了读取和写入基本数据类型的特定方法。
- 比如：double readDouble()、void writeDouble()、int readInt()、void writeInt() 等等。

DataOutputStream和DataInputStream

```
public class Main {
        public static void main(String[] args) throws FileNotFoundException,
                                        IOException, ClassNotFoundException {
                DataOutputStream dos =
 new DataOutputStream(new FileOutputStream("C:/a.txt"));
                dos.writeUTF("身高：");
                dos.writeDouble(15.2);dos.writeUTF("年龄：");
                dos.writeInt(15);
                dos.flush();
                dos.close();
                DataInputStream dis = new DataInputStream(new
FileInputStream("C:/a.txt"));
                String title1 =  dis.readUTF();
                double d = dis.readDouble();
                System.out.println(title1+d);
                String title2 = dis.readUTF();
                int i = dis.readInt();
                System.out.println(title2+i);
                dis.close();
        }
}
```

InputStreamReader和OutputStreamWriter

- 用来在字节流和字符流之间做中介，是字节流通向字符流的桥梁。
- 例如:
  - 在网络编程中，通常需要使用InputStreamReader和OutputStreamWriter在字节流和字符流之间进行转换。

```
InputStream
in=socket.getInputStream();
InputStreamReader inreader=new
InputStreamReader(in);
```

讲授思路

- 字符流
- 对象流
- 其他常用流

Thank You