第12章 Java输入输出流知识点

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- Java流的类层次结构
- 字节流
  - 抽象类InputStream和OutputStream
  - 文件流FileInputStream和FileOutputStream
  - 缓冲流BufferedInputStream和BufferedOutputStream
  - 数据过滤流DataInputStream和DataOutputStream
  - 打印流PrintStream
  - 序列化接口Serializable
  - 对象流ObjectInputStream和ObjectOutputStream
  - 字节数组流ByteArrayInputStream和ByteArrayOutputStream

本章知识点

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 字符流
  - 抽象类Reader和Writer
  - 转换流InputStreamReader和OutputStreamWriter
  - FileReader和FileWriter
  - BufferedReader类
  - PrintWriter类
- RandomAccessFile类

- 操作文件的类和接口

输入输出流

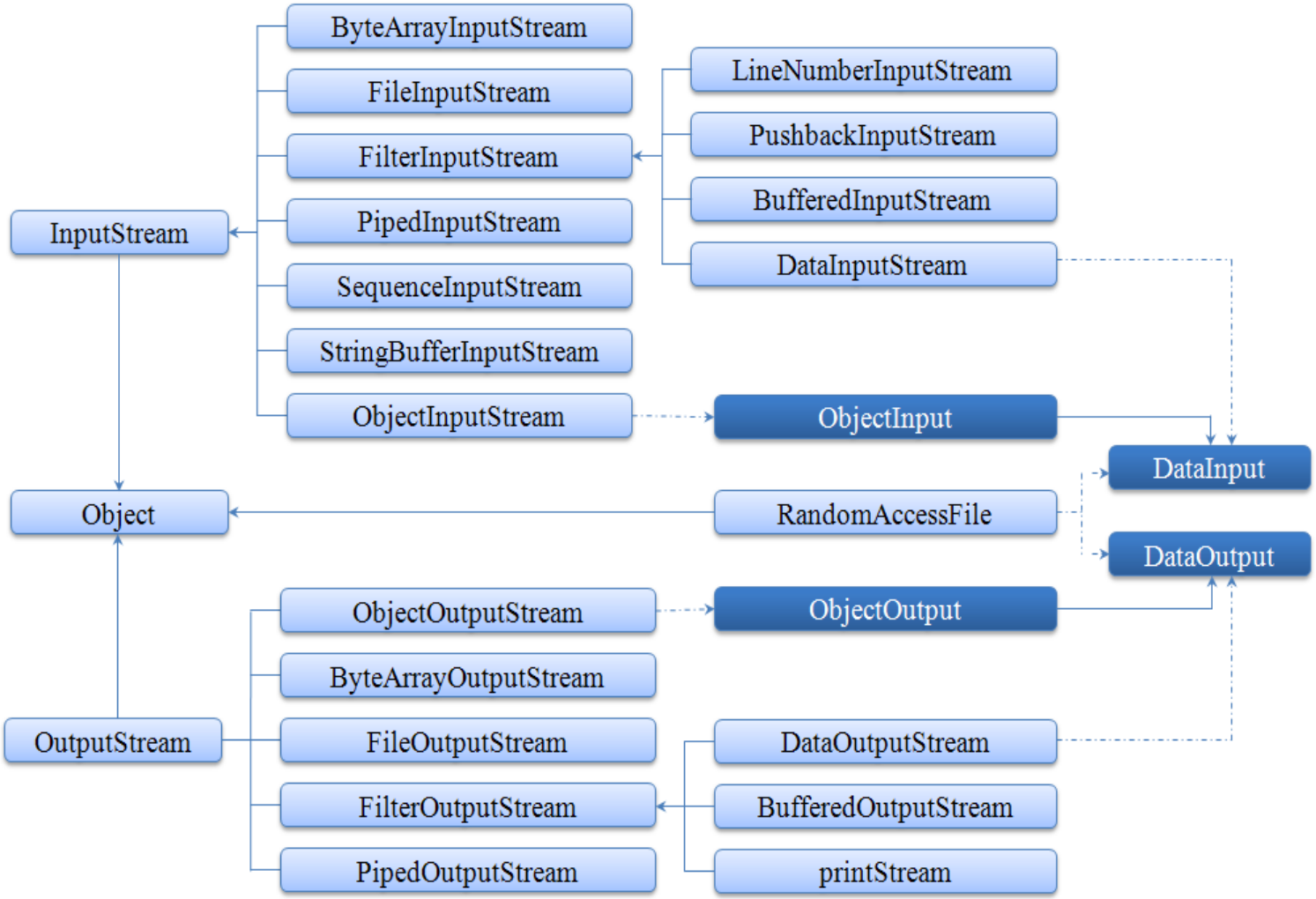
- Java中的输入/输出（Input/Output，简称I/O）：程序与外部设备或其他计算机进行数据交换的过程。

- Java用“流”的方式处理I/O，对应不同类型的I/O问题，会有相应的流对象提供解决方案。
  - JDK 1.0版本中设计了面向字节的I/O流。
  - JDK 1.1 对基本的I/O流类库进行了重大的修改，增加了面向字符的I/O流，这些流对应的类和接口位于java.io包下。
  - 在JDK1.4中添加了nio类用于改进性能和功能，这些类位于java.nio包下。

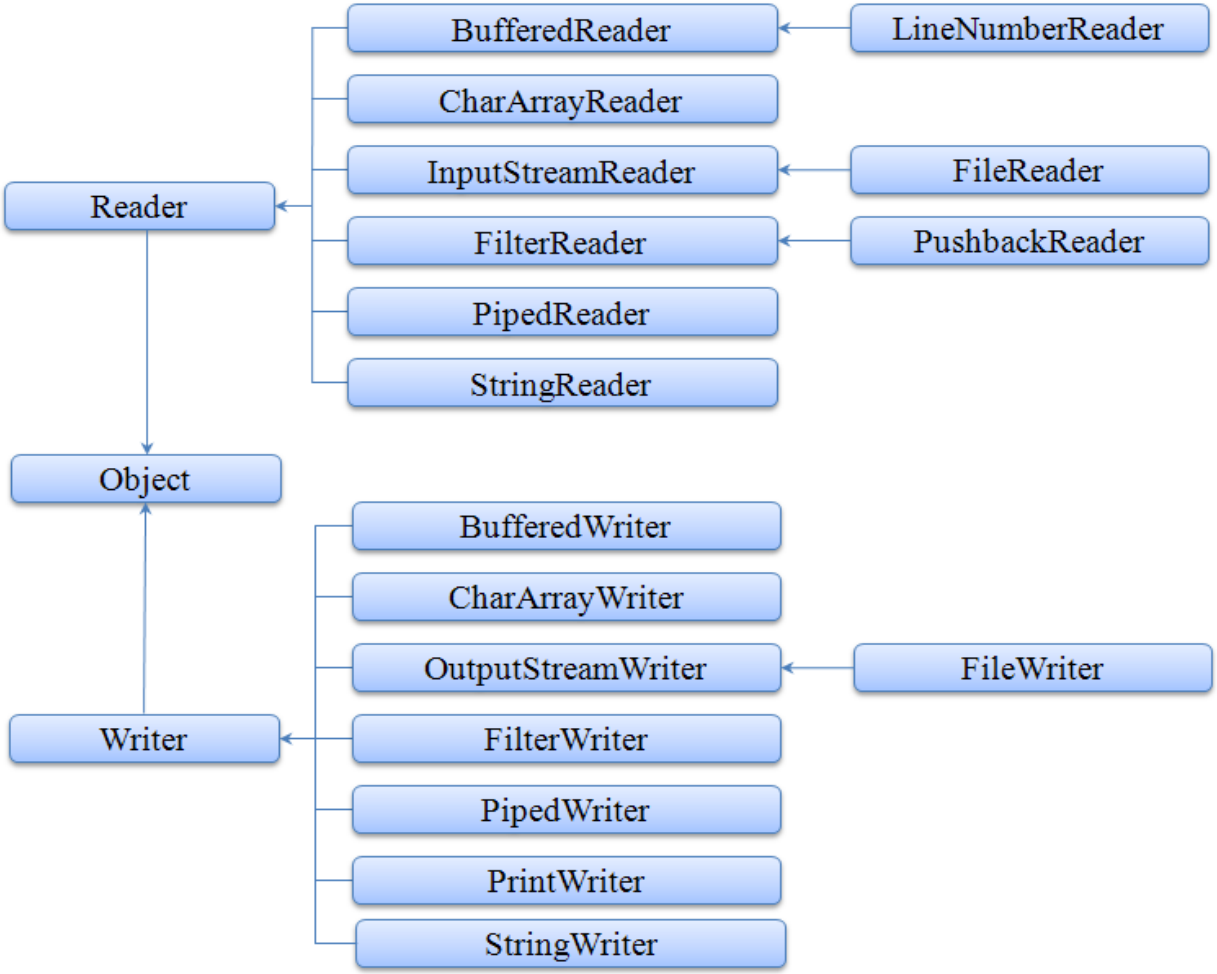
输入输出流

- 按流的方向分为**输入流**和**输出流**。
- 按数据传输单位分为**字节流**和**字符流**。
- 按功能分为底层的**节点流**和上层的**处理流**（或称为包装流）
  - 节点流用于和底层不同的物理存储节点（如文件、内存、控制台、网络连接等）关联
  - 处理流用于对节点流进行包装，丰富其功能、提供统一的操作方式，实现使用统一的代码读取不同的物理存储节点。

Java流的类层次结构--字节流层次结构



字符流层次结构


BufferedReader
LineNumberReader
CharArrayReader
InputStreamReader
FileReader
Reader
FilterReader
PushbackReader
PipedReader
StringReader
Object
BufferedWriter
CharArrayWriter
OutputStreamWriter
FileWriter
Writer
FilterWriter
PipedWriter
PrintWriter
StringWriter

12.2字节流

- 字节流以8位的字节为读写单位。
- 字节流体系：以抽象类InputStream和OutputStream作为顶层，向下包含了众多子类成员，可以按照输入流/输出流的方式完成各种基本数据类型数据、数组、字符串、对象、文件等的读入和写出。

1、InputStream类

- 用来表示那些从不同数据源产生输入的类。
- InputStream的主要子类
  - ByteArrayInputStream：把内存中的一个缓冲区作为InputStream，读取该缓冲区内容。
  - StringBufferInputStream：把一个字符串对象转换成InputStream。
  - FileInputStream：把文件作为InputStream，用于从文件中读取信息。
  - PipedInputStream：实现了管道（pipe）的概念，产生用于写入PipedOutputStream的数据，在线程通信中使用。
  - SequenceInputStream：把多个InputStream合并为一个InputStream。

12.2.1 抽象类InputStream和OutputStream

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- InputStream中的方法
  - int read()
  - int read(byte[] b)
  - int read(byte[] b, int off, int len)
  - int available()：返回流中可用字节数。
  - void close()：关闭输入流，并释放与该流关联的所有系统资源。

2．OutputStream

- 用于表示输出要去往的目标
- OutputStream主要的子类
  - ByteArrayOutputStream：在内存中创建缓冲区，所有送往“流”的数据都先放置在此缓冲区。
  - FileOutputStream：用于将信息写入文件。
  - PipedOutputStream：配合PipedInputStream类在线程间通信。

- OutputStream类中的方法
  - write(int b)
  - write(byte[] b)
  - write(byte[] b, int off, int len)
  - void close()：关闭输出流。
  - void flush()：强制刷新输出缓冲区内容，将输出缓冲区内容写入流。

12.2.2 文件流FileInputStream和FileOutputStream

- FileInputStream和FileOutputStream：基于字节、广泛用于操作文件。
  - FileInputStream类用于读取一个文件，FileOutputStream类用于将数据写入一个文件。
  - FileInputStream常用构造方法
    - FileInputStream(String name)
    - FileInputStream(File file)

- 它们都会打开一个到实际文件的连接，并创建一个文件输入流，如果该文件不存在，或者它是一个目录，或者因为其他原因而无法打开，会抛出FileNotFoundException异常。前者文件通过路径名name指定，后者通过File对象指定。

【例12-1】编写一个对指定文件进行加密复制的程序。

- 分析：复制文件就是将一个文件读出再写入另一个文件的过程，无论文件是何类型，都可以按照字节一一读出，一一写入，所以使用FileInputStream和FileOutputStream可以实现任意文件的复制。
- read()方法每次从文件中读取一个字节，范围从0到255间，如果已到达文件的末尾，则返回-1，所以循环读取的条件为：检测读到的数据是否为-1。
- 所谓加密复制就是在写入数据前对其进行加密处理，可以采取异或运算。

【例12-2】利用try-with-resource处理机制重写文件加密复制程序。

- 分析：传统的输入输出流的关闭过程代码繁琐，从Java SE 7引入了try-with-resource处理机制。
- 如果一个类实现了AutoClosable接口，那么创建这个类的对象就可以写在try-catch的try后面的括号中，并且能在try代码块正常结束前或者因发生异常要抛出异常前自动执行close()方法，保证资源正常关闭，不再需要书写finally代码块。

12.2.2 文件流FileInputStream和FileOutputStream

【例12-2】利用try-with-resource处理机制重写文件加密复制程序。

- Java SE 7后java.io中的类都实现了AutoClosable接口，可以直接使用try-with-resource。

compact1, compact2, compact3

java.io

Class FileInputStream

java.lang.Object
  java.io.InputStream
    java.io.FileInputStream

All Implemented Interfaces:

Closeable, AutoCloseable

12.2.3 缓冲流BufferedInputStream和BufferedOutputStream

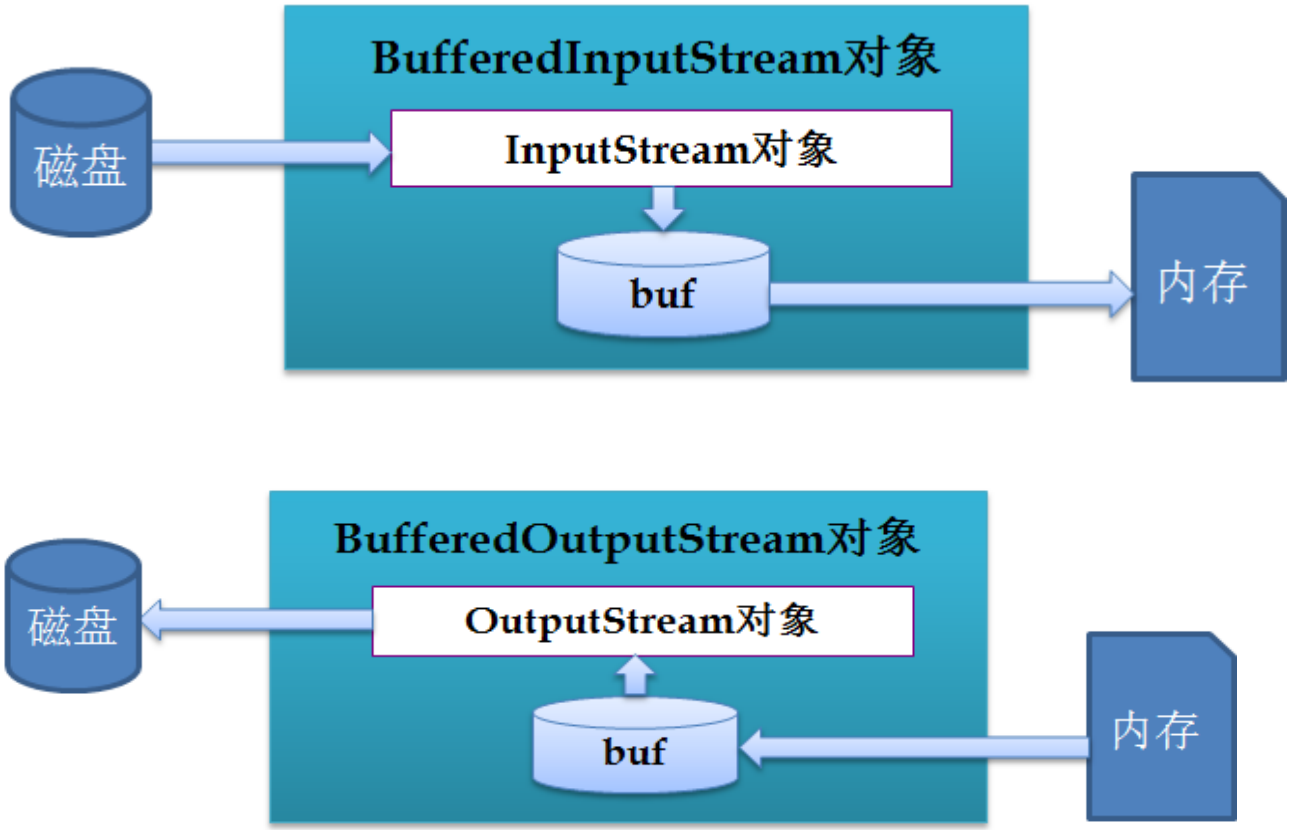
- BufferedInputStream和BufferedOutputStream是FilterInputStream和FilterOutputStream的子类，为普通的字节流增加了缓冲区功能，将InputStream和OutputStream对象包装为一个带缓冲的字节流。

- 原理：因为内存的读写速度快，而磁盘的读写速度慢，二者间的数据传输堵塞严重。所以，为了减少对磁盘的存取，通常在内存和磁盘间建立一个缓冲区，从磁盘中读数据时一次读入一个缓冲区大小的数量，数据写入磁盘时也是先将缓冲区装满后，再将缓冲区数据一次性写入到磁盘，由此提高了文件存取的效率。

12.2.3 缓冲流BufferedInputStream和BufferedOutputStream

- BufferedInputStream的构造方法：
  - BufferedInputStream(InputStream in)：包装InputStream对象、创建BufferedInputStream对象，并创建一个默认大小（8192字节）的字节数组做缓冲区。
  - BufferedInputStream(InputStream in, int size)：包装InputStream对象、创建指定缓冲区大小的BufferedInputStream对象。

12.2.3 缓冲流BufferedInputStream和BufferedOutputStream



12.2.3 缓冲流BufferedInputStream和BufferedOutputStream

- 【例12-3】对比带缓冲区的文件复制与不带缓冲区文件复制的性能。
  - 通过完成相同的复制工作所花费的时间对比得到带缓冲区与不带缓冲区间的差别。

12.2.3 缓冲流BufferedInputStream和BufferedOutputStream

- 【关于flush()刷新】对于BufferedOutputStream，只有缓冲区满时，才会将数据真正送到输出流。那么有些情况下，就需要人为地调用flush()方法将尚未填满的缓冲区中的数据送出。例如，数据已经读取完毕，但缓冲区尚未装满，这时必须由程序调用flush()方法强制刷新缓冲区。
- 一般情况下，如果调用close()方法，会隐含flush()操作。但是有些特殊情况下通信双方需要保持通信，建立的流不能关闭。如两台计算机使用QQ软件聊天，流对象需长期保持连接，而聊天数据都是在本地计算机的输出缓冲区，不一定被装满，此时不能关闭流，每次必须调用flush()操作将数据发送给对方。

12.2.4 数据过滤流DataInputStream和DataOutputStream

- 数据过滤流类DataInputStream和DataOutputStream是FilterInputStream和FilterOutputStream的子类。

- 它们将字节数据按照指定形式结合成有意义的基本数据类型数据、以及String类型数据。

12.2.4 数据过滤流DataInputStream和DataOutputStream

DataInputStream除父类InputStream中的3个read()方法外，还有8种基本数据类型数据的读方法，如果读取过程中输入流已到达文件末尾，则抛出EOFException异常。

- short readShort()
- int readInt()
- long readLong()
- byte readByte()
- boolean readBoolean()
- char readChar()
- float readFloat()
- double readDouble()

- readUTF()：用于读取一个String。

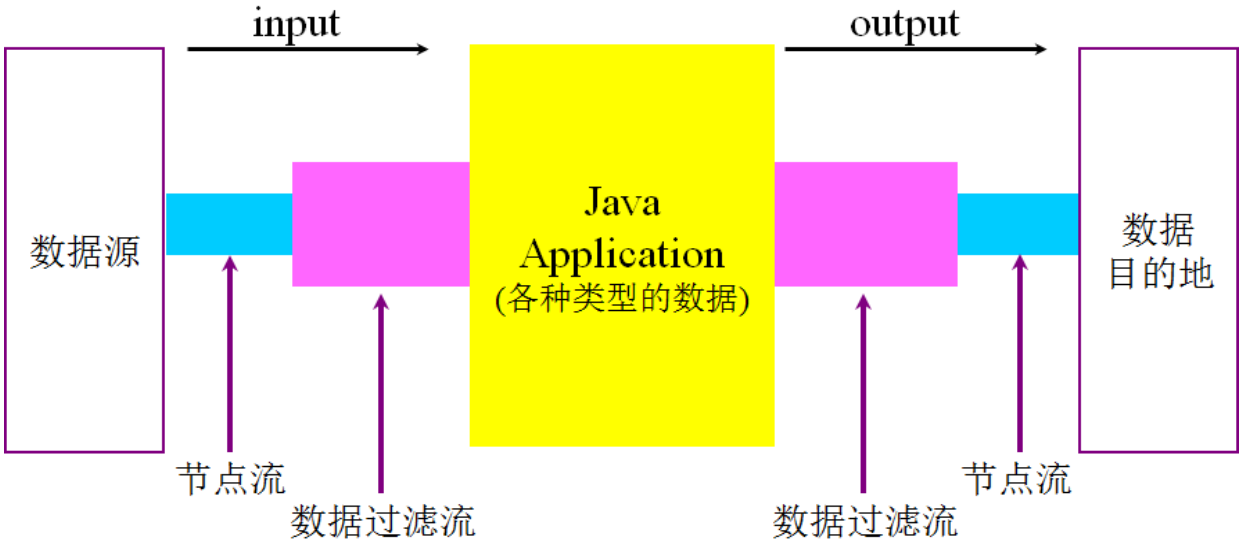
12.2.4 数据过滤流DataInputStream和DataOutputStream

- DataOutputStream除父类OutputStream中的和3个write()方法外，还有8种基本数据类型数据的写方法：
  - void writeShort(int)
  - void writeInt(int)
  - void writeLong(long)
  - void writeByte(int)
  - void writeChar(int)
  - void writeBoolean(boolean)
  - void writeFloat(float)
  - void writeDouble(double)

  - 用于写入一个String：
    - void writeChars(String)
    - void WriteUTF(String)

12.2.4 数据过滤流DataInputStream和DataOutputStream

- 它们的构造方法就是对已有的InputStream和OutputStream对象进行包装：DataInputStream(InputStream in)，DataOutputStream(OutputStream out)



12.2.4 数据过滤流DataInputStream和DataOutputStream

【例12-4】使用数据过滤流将1000-2000间的素数写到文件中持久化保存；将这些素数从文件读出、按每行10个的形式打印。

- 分析：使用自定义方法boolean isPrime(int)判断某数是否是素数。找到素数后，将其用writeInt()方法写入文件。读取文件时，每次使用readInt()方法读取一个int。

12.2.5 对象流与序列化接口Serializable

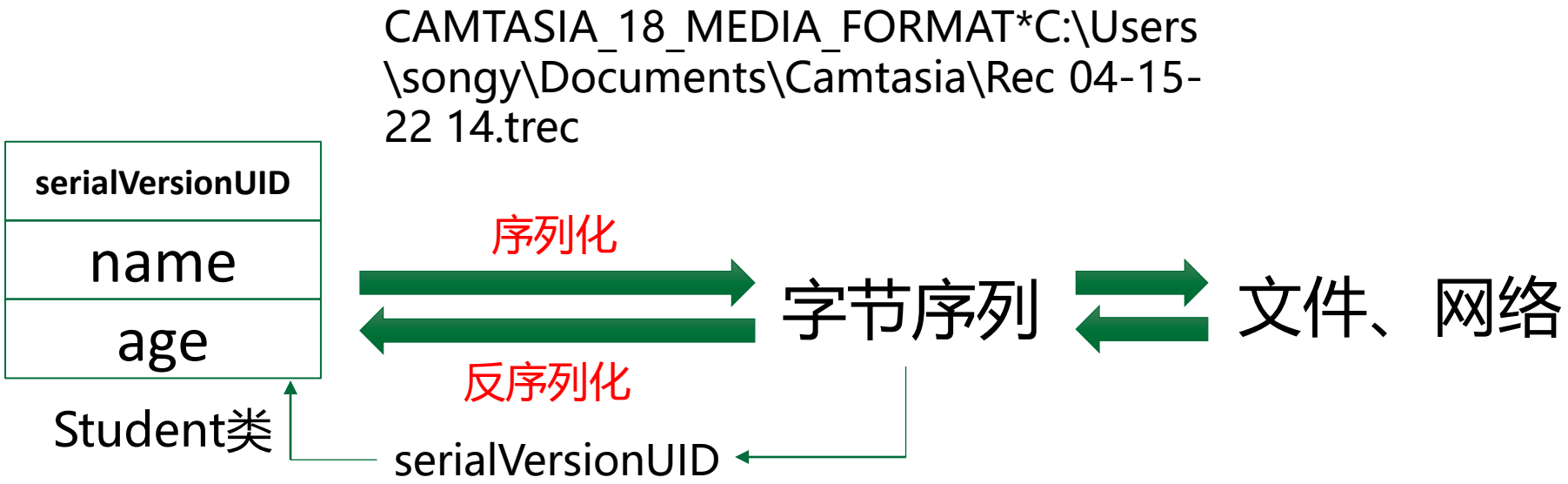
- 有时希望将以对象方式存在于内存中的数据存储至文件（持久化到文件），需要时再将其从文件中读出还原为对象，或者在网络上传送对象，这时可以使用Java提供的对象流ObjectInputStream和ObjectOutputStream。

12.2.5 对象流与序列化接口Serializable

- 如果对象需要被持久化到文件，或者在网络上传送对象，则定义该对象的类必须实现**Serializable**接口。
  - Serializable接口中并没有任何方法，这个接口只具有标识性的意义，代表该对象是可以序列化的。
  - 把Java对象转换为字节序列的过程称为对象的序列化，把字节序列恢复为Java对象的过程称为对象的反序列化。

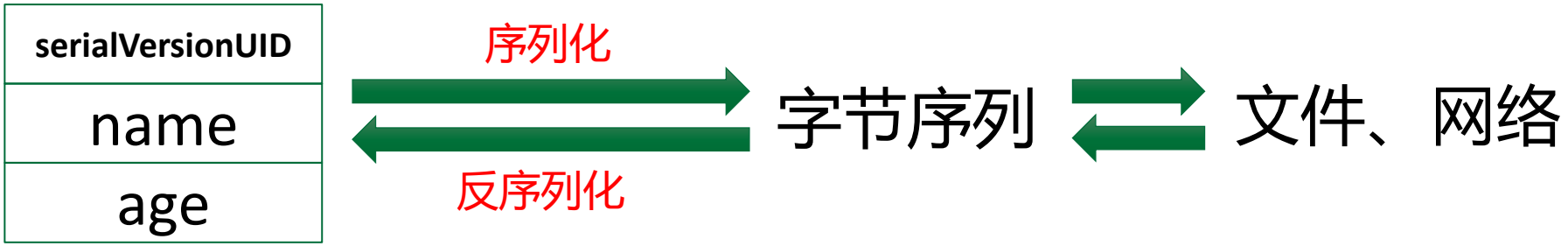
12.2.5 对象流与序列化接口Serializable

- JVM使用serialVersionUID来验证版本一致性。在进行反序列化时，JVM会把传来的字节流中的serialVersionUID与本地相应类的serialVersionUID进行比较，如果相同就认为是一致的，可以进行反序列化，否则就会出现序列化版本不一致的InvalidCastException异常。



12.2.5 对象流与序列化接口Serializable

- 支持序列化的类使用常量serialVersionUID标识可序列化对象的版本，从而维持版本信息的一致。
- 为了提高serialVersionUID的独立性和确定性，强烈建议显式定义serialVersionUID，为它赋予明确的取值。



Student类

```
public class Student implements Serializable{

    private static final long serialVersionUID = -7108027765951316257L;
}
```

12.2.5 对象流与序列化接口Serializable

【例12-5】将一个Student对象持久化到文件，并从文件读出、打印。

12.2.6 字节数组流ByteArrayInputStream和ByteArrayOutputStream

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 流的来源或目的地不一定是文件，也可以是内存中的一段空间。
  - 内存虚拟文件就是把内存中的数据缓存区虚拟成一个文件，原来应该写入到磁盘文件的内容，可以写入内存中；原来应该从磁盘文件中读取的内容，也可以从内存中读取。这样可以大大提高应用程序的性能和效率。

- 在Java中定义了ByteArrayInputStream和ByteArrayOutputStream，用于以I/O流的方式来完成对字节数组内容的读写，来支持类似内存虚拟文件的功能。
- ByteArrayInputStream和ByteArrayOutputStream作为与内存这个物理存储打交道的节点流，它们可以被处理流所包装，从而用更丰富的操作使用字节数组流中的字节数据。

12.2.6 字节数组流ByteArrayInputStream和ByteArrayOutputStream

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 构造方法：
  - ByteArrayInputStream(byte[] buf)：使用buf作为其缓冲区数组创建一个ByteArrayInputStream对象。
  - ByteArrayInputStream(byte[] buf, int offset, int length)：使用buf数组从offset位置开始的length长度空间作为缓冲区创建ByteArrayInputStream。
  - ByteArrayOutputStream()：创建一个新的字节数组输出流。
  - ByteArrayOutputStream(int size)：创建一个新的字节数组输出流，它具有指定大小的缓冲区容量。

12.2.6 字节数组流ByteArrayInputStream和ByteArrayOutputStream

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

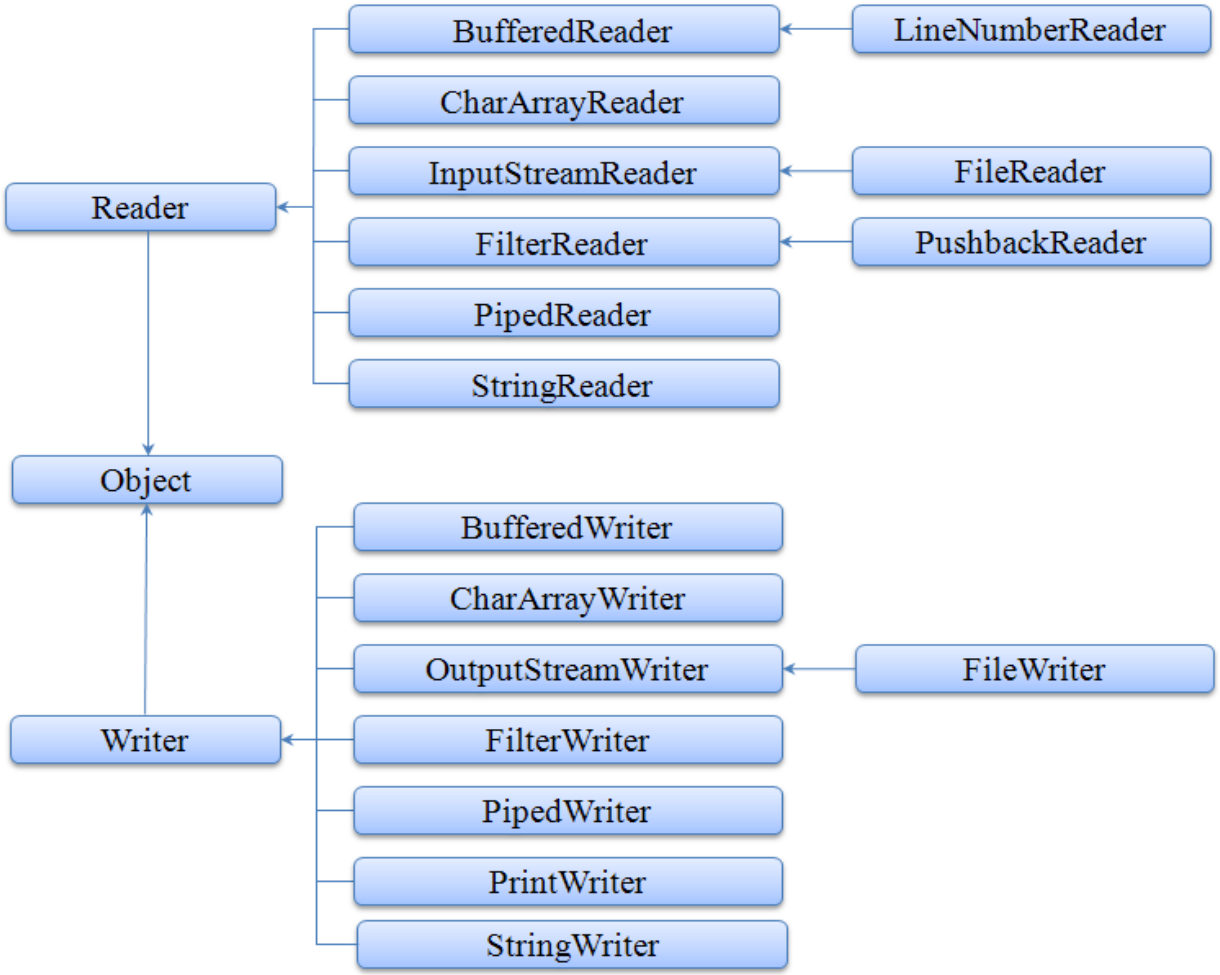
【例12-6】实现对象的深复制。

- 深复制的前提：对象以及对象内部所有引用到的对象都是可序列化的，即都实现了Serializable接口。
- 将对象持久化到文件中再读出可以获得原对象的一个拷贝，所以利用这个方法可以深复制对象。

- InputStream和OutputStream系列处理的是字节流，存取数据流时以字节为单位，但它们在读写文本字符、字符串时就不太方便。另外，字节流只支持ISO-8859-1编码，而Java本身使用Unicode编码，各个平台、系统中的字符还会使用其他编码方式。

- 为了便于读写字符型数据，让Java的I/O流都支持Unicode编码，并允许使用其他字符编码方案，Java 1.1的类库中增加了Reader和Writer继承层次结构，分别表示字符输入流和字符输出流。

12.3.1 抽象类Reader和Writer



12.3.1 抽象类Reader和Writer

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- Reader类中3个基本的读取数据的方法：
  - int read()：从输入流中读取单个字符。因为Java采用Unicode编码，每个字符分配2个字节的存储空间，所以read()方法将读取2个字节，返回所读取的字符数据的Unicode编码。
  - int read(char[] cbuf)：从输入流中最多读取cbuf.length个字符的数据，并将其存储在字符数组cbuf中，返回实际读取的字符数。
  - int read(char[] cbuf, int off, int len)：从输入流中最多读取len个字符的数据，并将其存储在字符数组cbuf中，从数组的off位置开始存储。
  - 所有read()方法在读取流数据时，如果已到达流的末尾，则返回-1。

12.3.1 抽象类Reader和Writer

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

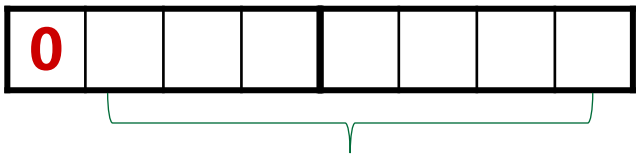
- Writer类中定义的向输出流写出数据的方法既包括写出字符、字符数组的，同时也包括写出字符串的，如下：
  - void write(int c)：将指定的字符输出到输出流。
  - void write(char[] cbuf)：将字符数组cbuf中的数据输出到指定输出流。
  - void write(char[] cbuf, int off, int len)：将字符数组cbuf从off位置开始的len个字符输出到指定输出流。
  - void write(String str)：将str字符串中的字符输出到指定输出流。
  - void write(String str, int off, int len)：将str字符串中从off位置开始的len个字符输出到指定输出流。

12.3.2 转换流InputStreamReader和OutputStreamWriter

- 把来自于“字节”层次结构中的类和“字符”层次结构中的类结合起来使用，为了实现这个目的，要用到InputStreamReader把InputStream转换为Reader，用OuputStreamWriter将OutputStream转换为Writer。

- ASCII码：用1个字节(8位二进制数)表示1个字符的编码

最高位取“0”

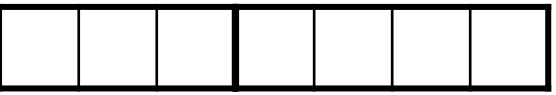


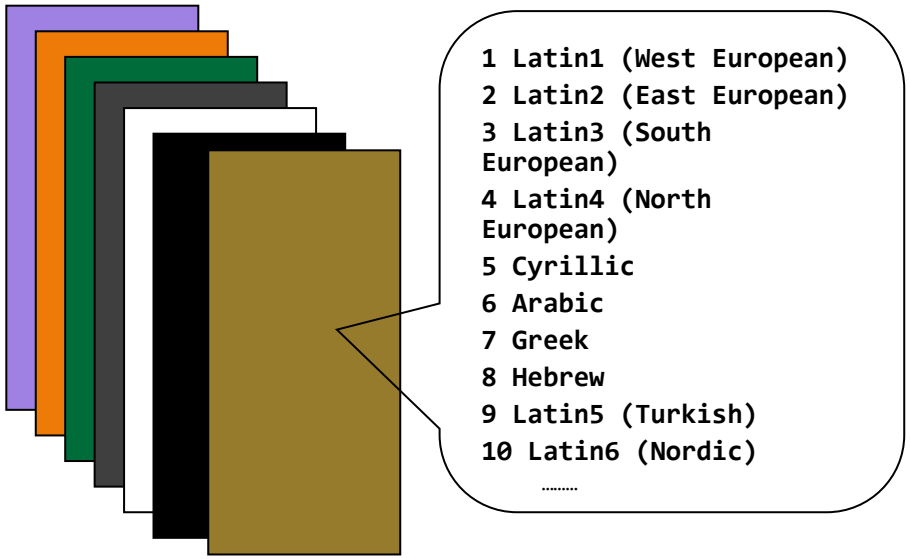
余下的7位可给出128个二进制编码
即0000 0000-0111 1111
可以表示128个字符

低4位 \ 高4位	0000	0001	0010	0011	0100	0101	0110	0111
0000	NUL	DLE	SPACE	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	ANK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

12.3.2 转换流InputStreamReader和OutputStreamWriter

- ISO8859：扩展ASCII码字符集





国际标准化组织（ISO）及国际电工委员会（IEC）联合制定的一系列8位元字符集的标准，现时定义了15个字符集。

- 高位为1的8位编码
- 均属于拉丁语系

12.3.2 转换流InputStreamReader和OutputStreamWriter

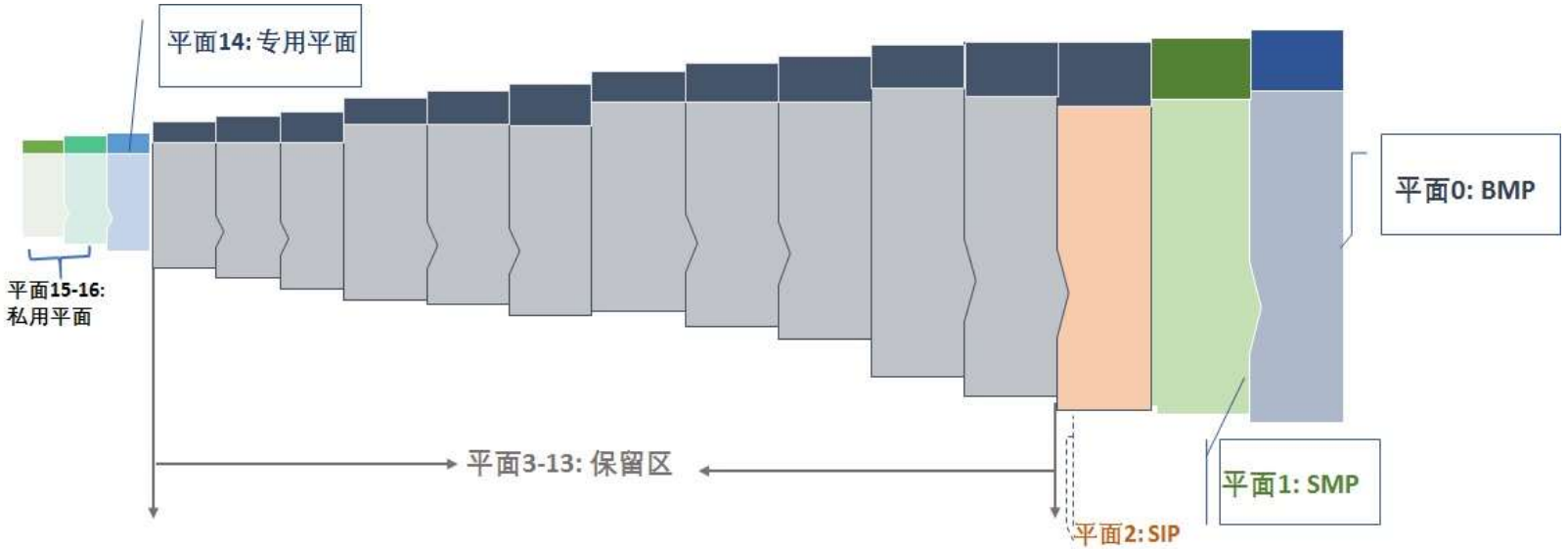
- 汉字编码：国标码GB2312、GBK

GBK是Windows中文系统的缺省字符集

12.3.2 转换流InputStreamReader和OutputStreamWriter

- unicode编码：国际化的编码方案
  - 规定了字符的码点
  - 常见编码方案：UTF-8、UTF-16

BMP(Basic Multilingual Plane)，多语言平面收集最常见的字符，编码U+0000到U+FFFF



总计 17 × 65,536 = 1,114,112 个码点

12.3.2 转换流InputStreamReader和OutputStreamWriter

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- UTF-8：可变长形式编码
  - 代码单元由 8 位组成
    - 单字节：ASCII字符，保持与ASCII编码的兼容性
    - 两字节：拉丁文等字符
    - 三字节：中（2万多汉字）日韩等字符
    - 四字节：扩充汉字

字节数	表示码位的位数	Byte 1	Byte 2	Byte 3	Byte 4
1	7	0xxxxxxx			
2	11	110xxxxx	10xxxxxx		
3	16	1110xxxx	10xxxxxx	10xxxxxx	
4	21	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

12.3.2 转换流InputStreamReader和OutputStreamWriter

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

		I	空格	a	m	空格	中	国
ISO8859-1		49	20	61	6d	20	3f	3f
GB2312 GBK		49	20	61	6d	20	d6d0	b9fa
Unicode码点		49	20	61	6d	20	4e2d	56fd
UTF-16 大端（高前低后）	feff	0049	0020	0061	006d	0020	4e2d	56fd
UTF-8		49	20	61	6d	20	e4b8ad	e59bbd

结论：处理文本数据的时候，为防止乱码出现，需要指定字符编码或者进行编码转换。

- 2．利用转换流设置字符编码

  - Reader类能将输入流中采用其他编码方式的字节流转换为Unicode字符，然后在内存中为这些Unicode字符分配内存。Writer类能将内存中的Unicode字符转换为其他的编码方式的字节流，再写到输出流。
  - 在默认的情况下，Reader和Writer会在本地平台默认字符编码和Unicode编码间进行转换。中文Windows操作系统中默认的是GBK编码，中文Linux操作系统中默认的是UTF-8编码。
  - 如果需要输入、输出流采用特定编码方案，可以使用InputStreamReader和OutputStreamWriter类，它们在将字节流转换为字符流的同时，可以指定字符编码方式。

- 2．利用转换流设置字符编码
  - 如果需要输入、输出流采用特定编码方案，可以使用InputStreamReader和OutputStreamWriter类，它们在将字节流转换为字符流的同时，可以指定字符编码方式。

12.3.2 转换流InputStreamReader和OutputStreamWriter

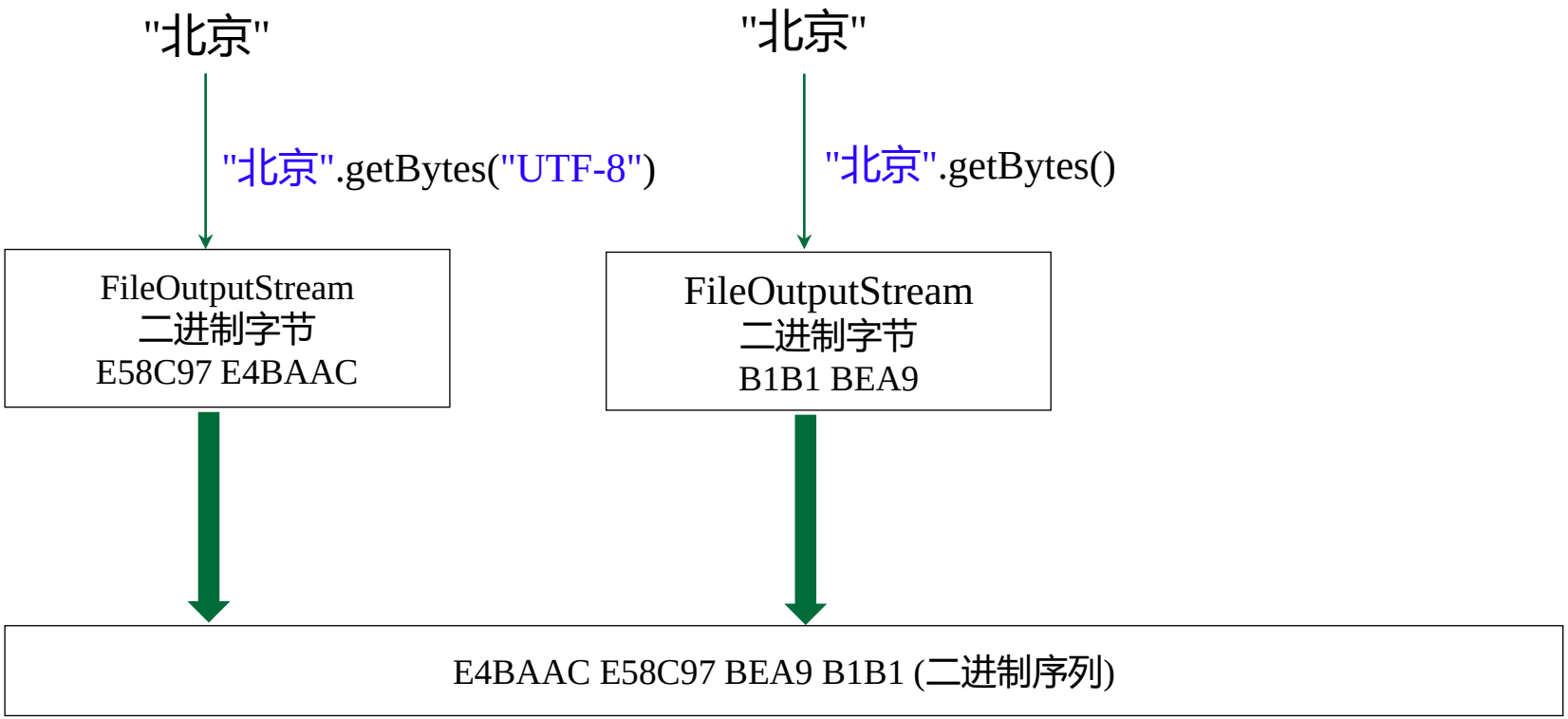
- InputStreamReader和OutputStreamWriter工作在字节流与字符流之间，被称作转换流
- InputStreamReader可以将一个字节流中的若干字节解码成字符，OutputStreamWriter可以将写入的字符编码成若干字节的二进制数据。
- 常用的构造方法如下：
  - InputStreamReader(InputStream in)：创建一个使用默认字符集的 InputStreamReader。
  - InputStreamReader(InputStream in,String charsetName)：创建使用指定字符集的 InputStreamReader。
  - OutputStreamWriter(OutputStream out)：创建使用默认字符编码的 OutputStreamWriter。
  - OutputStreamWriter(OutputStream out,String charsetName)：创建使用指定字符集的 OutputStreamWriter。

12.3.2 转换流InputStreamReader和OutputStreamWriter

【例12-7】向文件中写入一个中文字符串，再将其从文件读出。

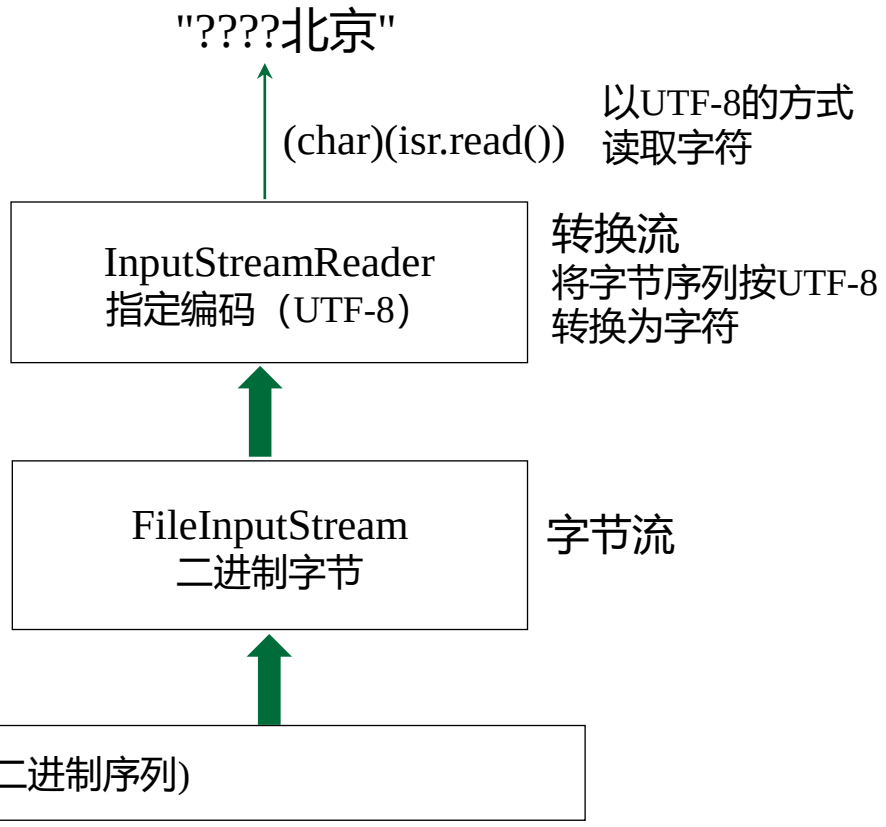
- 说明：为了演示汉字编码的使用，本例利用两种编码向输出流写入字符串“北京”，一次使用默认的本地编码方案GBK，一次指定编码方式UTF-8。从输入流读取数据时使用UTF-8编码。

```
try(
        FileOutputStream fos = new FileOutputStream(filename);
){
    fos.write("北京".getBytes());
    fos.write("北京".getBytes("UTF-8"));
}
```



```
try(
    FileInputStream fis = new FileInputStream(filename);
    InputStreamReader isr = new InputStreamReader(fis, "UTF-8");
) {
    while((ch=isr.read())!=-1){
        System.out.print((char)ch);
    }
}
```

字节数	表示码位的位数	Byte 1	Byte 2	Byte 3	Byte 4
1	7	0xxxxxxx			
2	11	110xxxxx	10xxxxxx		
3	16	1110xxxx	10xxxxxx	10xxxxxx	
4	21	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx



12.3.3 FileReader和FileWriter

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 如果存取的是一个文本文件，可以直接使用FileReader和FileWriter类，它们分别继承自InputStreamReader和OutputStreamWriter。
  - FileReader类用于文本文件的读，每次读取一个字符或一个字符数组。FileWriter类用于文本文件的写，每次写入一个字符、一个数组或一个字符串。通常可以将FileReader对象看作一个以字符为单位的无格式的字符输入流，将FileWriter对象看作是以字符为单位的无格式的字符输出流。
  - FileReader和FileWriter类只能按照平台默认的字符编码进行字符的读写，若要指定编码，则还是使用InputStreamReader和OutputStreamWriter。

【例12-11】将九九乘法表保存在文本文件中。

12.3.4 BufferedReader类

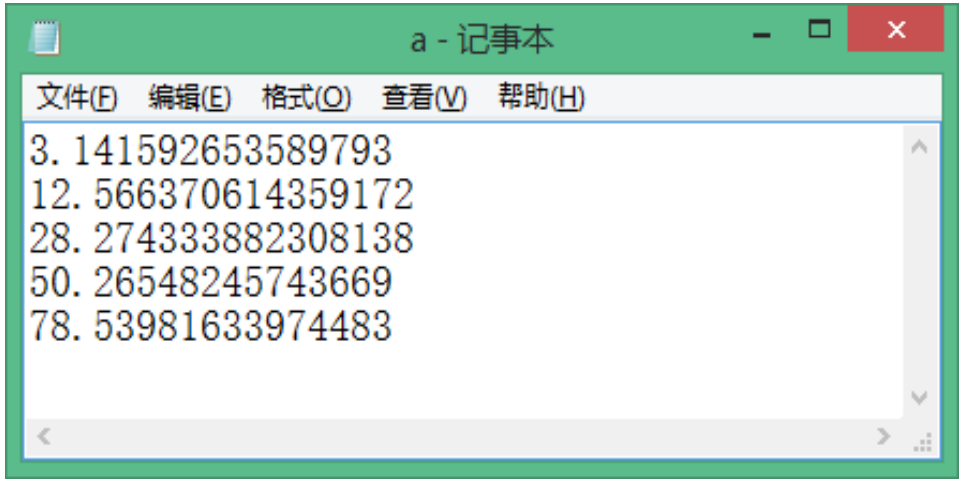
- BufferedReader和BufferedWriter类都带有8192个字符的缓冲区，缓冲区的作用与BufferedInputStream和BufferedOutputStream类相同。
  - BufferedReader类还可以“文本行”为基本单位读取数据，文本行是以回车换行结束的字符序列。
  - String readLine()：从输入流中读取一行字符，如果读遇到流结束，则返回null。

12.3.5 PrintWriter类

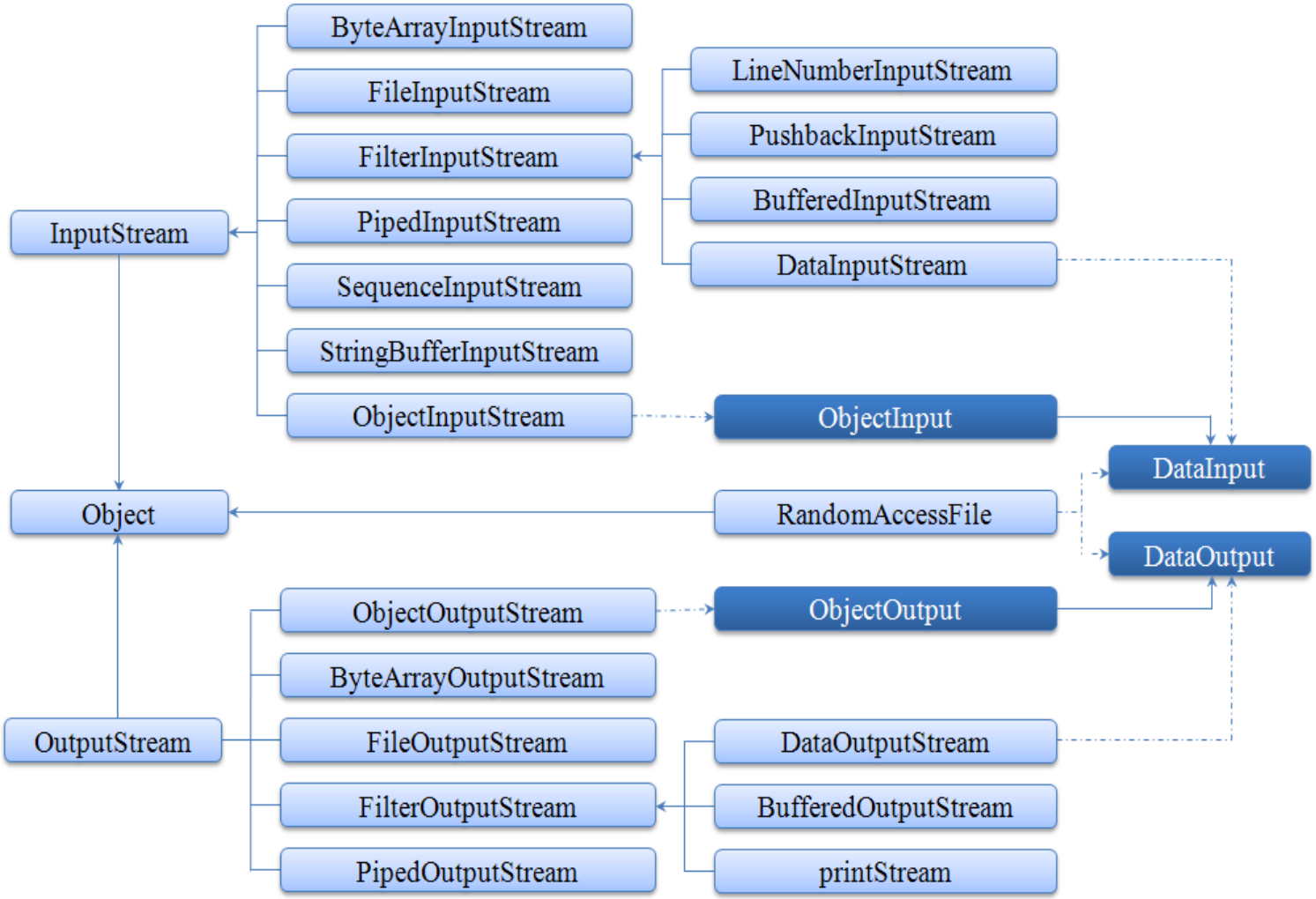
北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- PrintStream的问题是它不支持国际化，不能用与平台无关的方式处理换行。所以在JDK 1.1中引入了PrintWriter类，它是字符流，依旧使用与PrintStream相同的格式化接口print()和println()方法，但提供了国际化支持。在输出方面，PrintWriter比PrintStream更为合适。
  - PrintWriter提供了既能接收Writer对象又能接收OutputStream对象的构造方法，简化了输出流对象的创建过程。
  - PrintWriter也可以设置缓冲区的自动刷新。PrintWriter仅在调用println()方法时自动刷新；而PrintStream只要输出遇到换行符（调用println()方法、输出换行符等），缓冲区的内容就会被强制输出。

【例12-12】将计算得到的几个圆的面积写入一个文件；再将文件中的信息读取出来在控制台打印。



12.4 RandomAccessFile类



1．RandomAccessFile的特点

- RandomAccessFile类以字节为单位进行文件内容的存取。

- （1）RandomAccessFile提供了文件的“随机访问”方式，在RandomAccessFile类中定义了文件记录指针，标识当前正在读写的位置，它可以指向文件中的任意位置。
- （2）RandomAccessFile既可以读取文件的内容，也可以向文件输出数据，集读、写功能于一身。

2．RandomAccessFile中的读写方法

- 查看API文档

- 3、RandomAccessFile中的其他方法
  - long getFilePointer()：返回文件记录指针的当前位置。
  - void seek(long pos)：将文件记录指针定位在pos位置。
  - 通过它们可以获取和操作文件记录指针，实现RandomAccessFile与众不同的定位随机访问。

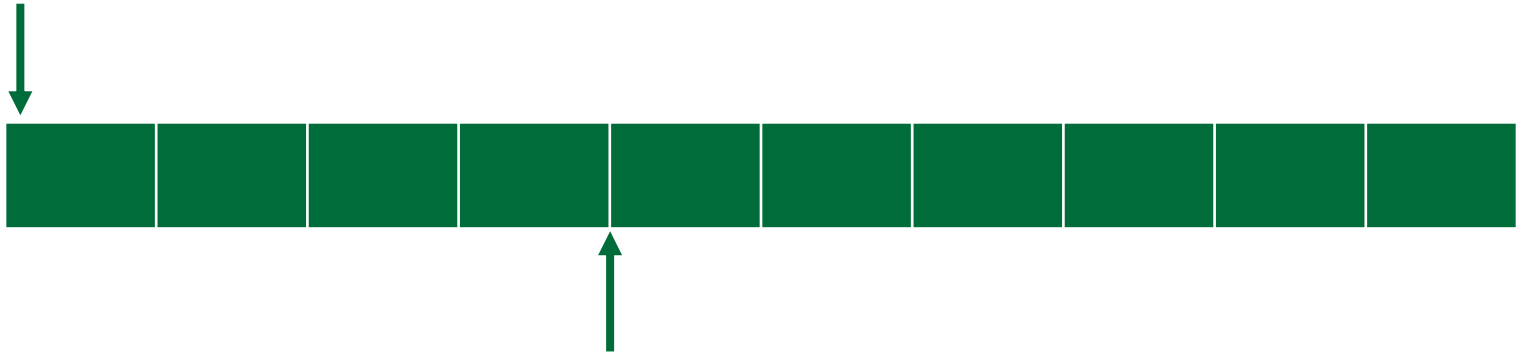
北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

4．RandomAccessFile的常用访问方式

- 创建RandomAccessFile对象时，需要指定文件的访问方式，常用的包括下面两种：
- “r”：以只读方式打开指定文件。
- “rw”：以读、写方式打开指定文件，如果文件不存在则尝试创建该文件。

- RandomAccessFile没有只写的访问方式。

【例12-10】向data.dat文件中写入10个int型随机整数；从键盘上输入1~10这些序号，在控制台打印出对应数字。



12.5 Java输入流输出流汇总

- 节点流：直接与不同的物理存储打交道。
- 处理流：对节点流进行包装，或提供缓存、或提供数据的格式化读写、打印，总之使输入输出更简单便捷，执行效率更高。

12.5 Java输入流输出流汇总

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 节点流
  - 对内存进行读写的类（包括字符数组、字节数组、字符串）：CharArrayReader、CharArrayWriter、ByteArrayInputStream、ByteArrayOutputStream、StringReader、StringWriter。
  - 对文件进行读写的类：FileReader、FileWriter、FileInputStream、FileOutputStream。
  - 来自和写入网络Socket端口的数据以字节的形式读写，也是一种节点流。

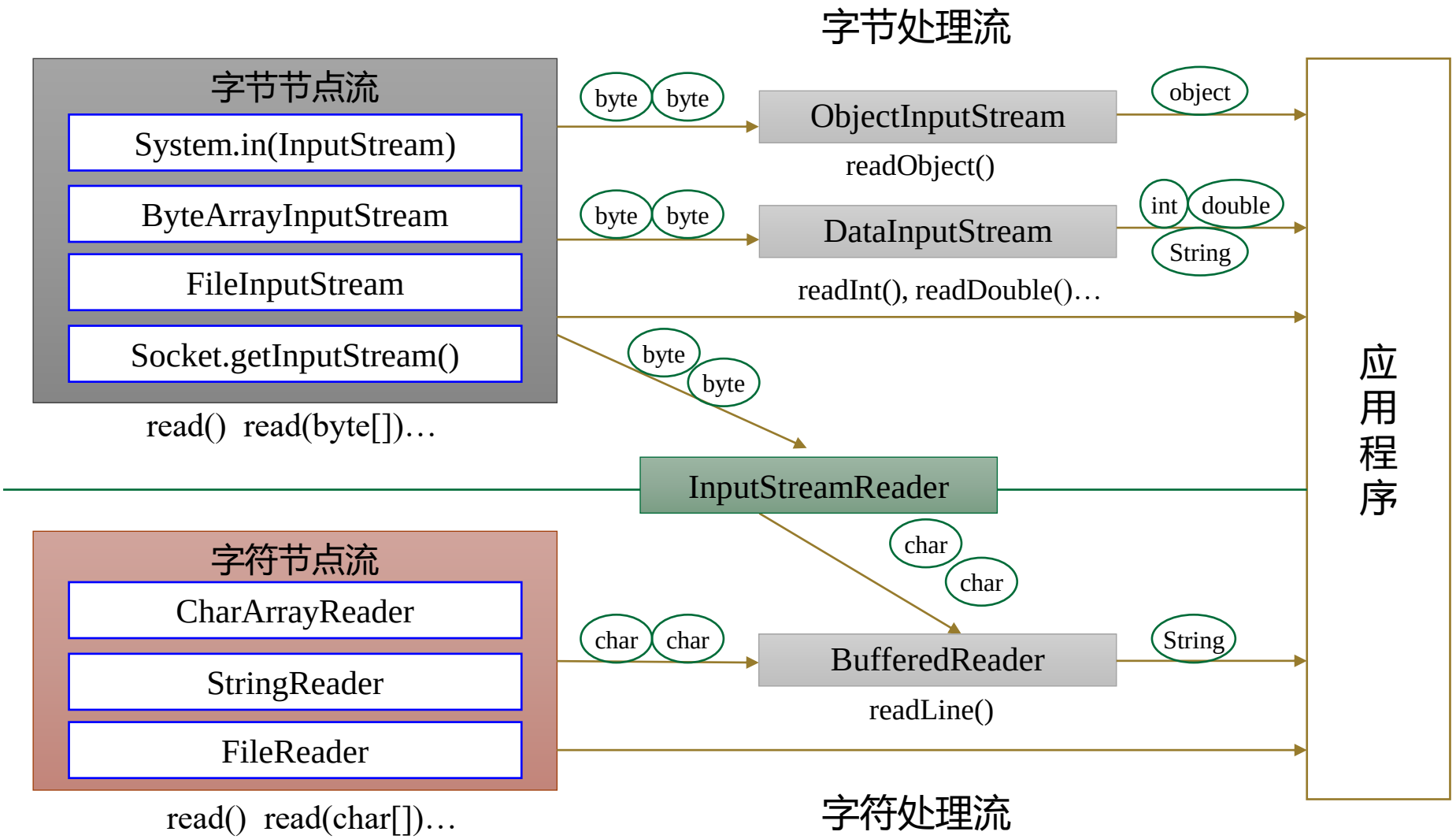
12.5 Java输入流输出流汇总

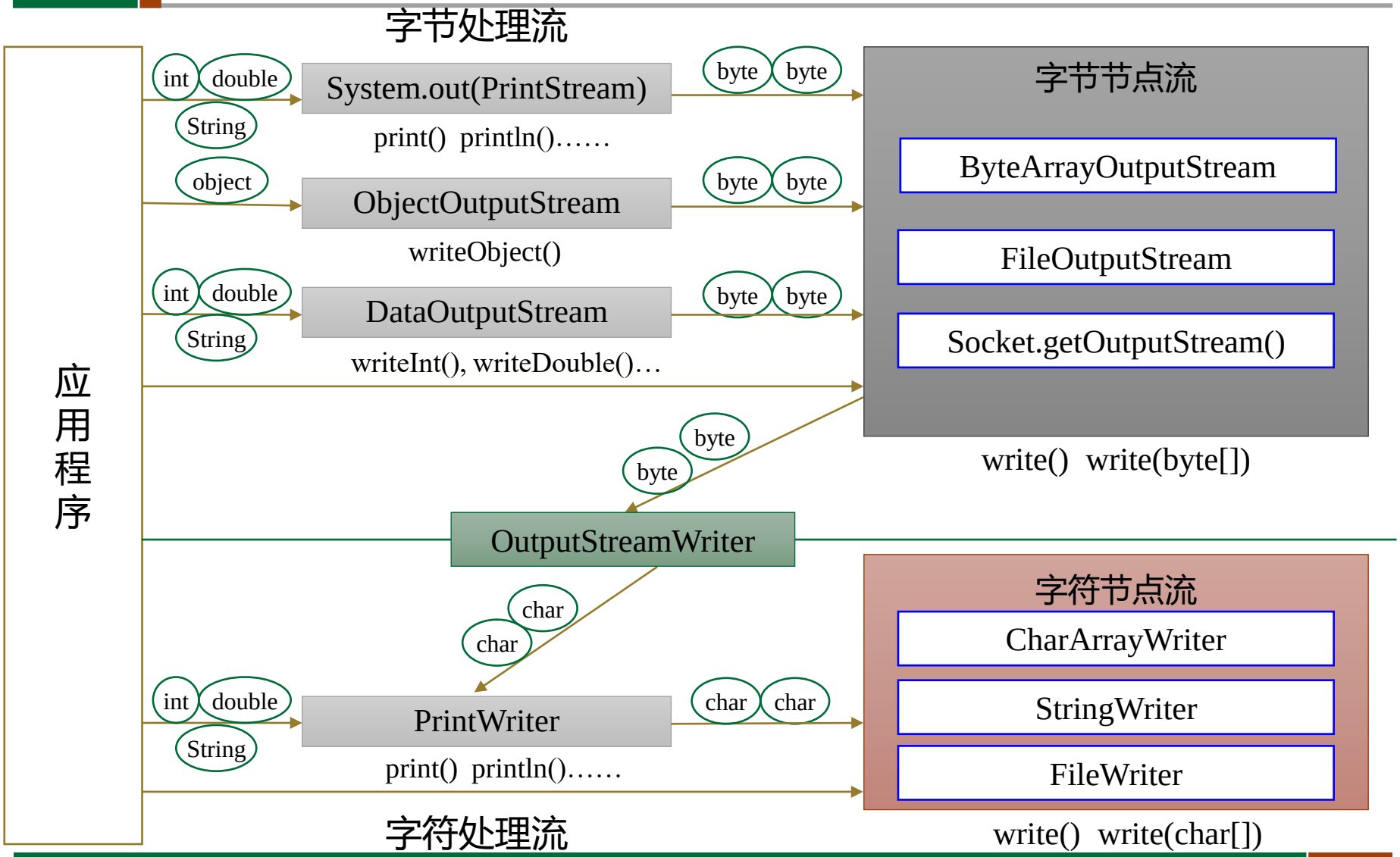
北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 处理流
  - 对对象进行读写的类：ObjectInputStream、ObjectOutputStream。
  - 按Java基本数据类型读写的类：DataInputStream、DataOutputStream。
  - 格式化打印输出的类： PrintWriter、PrintStream。
  - 读写时对数据进行缓存的类：BufferedReader、BufferedWriter、BufferedInputStream、BufferedOutputStream。
  - 按照一定的编码标准将字节流转换为字符流的类：InputStreamReader、OutputStreamWriter。

12.5 Java输入流输出流汇总

- 转换流：工作在字节流和字符流间，提供加入编码方式的字节数据与字符数据间的转换。


字节处理流
字节节点流
System.in(InputStream)
ByteArrayInputStream
FileInputStream
Socket.getInputStream()
read() read(byte[])…
byte
byte
ObjectInputStream
readObject()
object
byte
byte
DataInputStream
readInt(), readDouble()…
int
double
String
byte
byte
InputStreamReader
char
char
字符节点流
CharArrayReader
StringReader
FileReader
read() read(char[])…
char
char
BufferedReader
readLine()
String
字符处理流
应用程序


字节处理流
应用程序
int
double
String
System.out(PrintStream)
print() println()……
object
ObjectOutputStream
writeObject()
int
double
String
DataOutputStream
writeInt(), writeDouble()…
byte
byte
字节节点流
ByteArrayOutputStream
FileOutputStream
Socket.getOutputStream()
write() write(byte[])
byte
byte
OutputStreamWriter
char
char
字符节点流
CharArrayWriter
StringWriter
FileWriter
int
double
String
PrintWriter
print() println()……
char
char
write() write(char[])
字符处理流

- 除了存取文件的内容之外，文件管理还包括创建文件/目录、获取文件信息、复制移动删除文件等。
- Java SE 7在java.nio.file包中新增了Path接口、Paths类和Files类。

12.6.1 Paths类和Path接口

- 路径在文件系统用于定位文件，由一系列目录名组成，其后还可以跟着一个文件名，这些元素由一个分隔符分隔，Windows是“\”，Unix/Linux是“/”。

12.6.1 Paths类和Path接口

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- Paths类只包含了静态的get()方法，用于构建路径对象。
  - static Path get(String first, String... more)
  - 将一系列路径字符串转换为路径。如果more没有指定任何元素，那么第一个参数的值就是要转换的路径字符串；如果more指定了一个或多个元素，那么每个非空字符串（包括first）被视为一个名称元素序列，连接起来形成一个路径字符串，路径之间的分隔符视操作系统而定。
  - get()方法的返回值类型为Path。

12.6.1 Paths类和Path接口

- Path接口 用于表示依赖于操作系统的文件路径。
- 访问路径组件或其名称元素的子序列
  - getFileName()
  - getParent()
  - getRoot()
  - subpath(int beginIndex, int endIndex)

- 组合路径的方法
  - Path resolve(String other)：对路径进行解析，将给定的路径字符串转换为路径。以p.resolve(other)为例，它会按照如下规则返回路径：如果other是绝对路径则结果就是other；否则，将p路径后面组合上other作为结果。
  - 例如，在absolute路径（\home\source）的基础上调用resolve()方法
    - absolute.resolve("java")的结果为“\home\source\java”
    - absolute.resolve("/java")的结果为“\java”

- 组合路径的方法
  - Path resolveSibling(Path other)：通过解析other的父路径从而产生其兄弟路径。这在需要用另一个文件名替换文件名时非常有用。
  - 例如，假设other路径为“dir1\dir2\foo”
    - 使用路径“bar”调用此方法将返回路径“dir1\dir2\bar”
    - 如果other没有父路径，或者other是绝对路径，则此方法返回other。

- 组合路径的方法
  - Path relativize(Path other)：在调用方法的路径和other路径之间构造相对路径，即，返回从调用方法路径到达other路径的相对路径表示。
  - 例如：
    - Path path1 = Paths.get("/home/demo.html");
    - Path path2 = Paths.get("/home/images/logo.png");
    - Path relativePath = path1.relativize(path2);
    - 从path1如何到达path2呢?
    - 返回值为“..\images\logo.png”
    - 即从demo.html出发，用“..”达到它的父目录“home”，向下与“images\logo.png”组合。

12.6.2 File类

- Files类包含了对文件、目录进行操作的静态方法。
  - 创建文件和目录
  - 获取文件信息
  - 读写文件

（1）static Path createDirectory(Path dir, FileAttribute<?>... attrs)

- 在磁盘上创建一个新目录，其中参数dir中除最后一部分外，其他部分都必须是已存在的。
- 如果尚未存在则会抛出NoSuchFileException异常
- 如果创建已存在的目录，抛出FileAlreadyExistsException异常。
- 如果要创建路径中所有不存在的父目录时，应该使用createDirectories()方法。
- attrs参数是可选的文件属性，可在创建文件时自动设置。

（2）static Path createDirectories(Path dir, FileAttribute<?>... attrs)

- 通过先创建所有不存在的父目录来创建目录。与createDirectory()方法不同，如果由于目录已经存在而无法创建该目录不会引发异常。
- 例如，有如下路径定义：
- Path sourceDir = Paths.get("chap12", "ex13");
- 如果当前目录下已经存在“chap12”文件夹，则Files.createDirectory(sourceDir)即可实现创建“chap12\ex13”文件夹；
- 如果当前目录下“chap12”文件夹尚未存在，则需要使用Files.createDirectories(sourceDir)才能完成创建。

（3）static Path createFile(Path path, FileAttribute<?>... attrs)

- 与前两个方法不同，createFile()方法在文件系统中创建一个新的空文件，如果该文件已存在，则抛出FileAlreadyExistsException异常。
- 例如，有如下路径定义：
- Path fileName = Paths.get("config.properties");
- 那么，Files.createFile(fileName)即可实现在当前目录下创建一个空的新文件config.properties。

1. 创建文件和目录

创建一个Path对象和创建路径对应的目录或文件在Java中是两个不同的概念。前者是在JVM中创建了一个表示路径的对象，随着JVM的关闭这个对象也随之消失。而Files类中的createXXX()方法真正的在操作系统中完成了目录和文件的物理创建。

- Files类中设计了一些方法用于获取文件的基本信息。
  - static FileTime getLastModifiedTime(Path path, LinkOption... options)：获取文件最后被修改的时间，返回值类型为FileTime。
  - static long size(Path path)：获取文件的大小（字节）。
  - static boolean exists(Path path, LinkOption... options)：文件是否存在。
  - static boolean isDirectory(Path path, LinkOption... options)：是否是目录。
  - isHidden(Path)、isExecutable(Path)、isReadable(Path)、isWritable(Path)

2. 获取文件信息

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

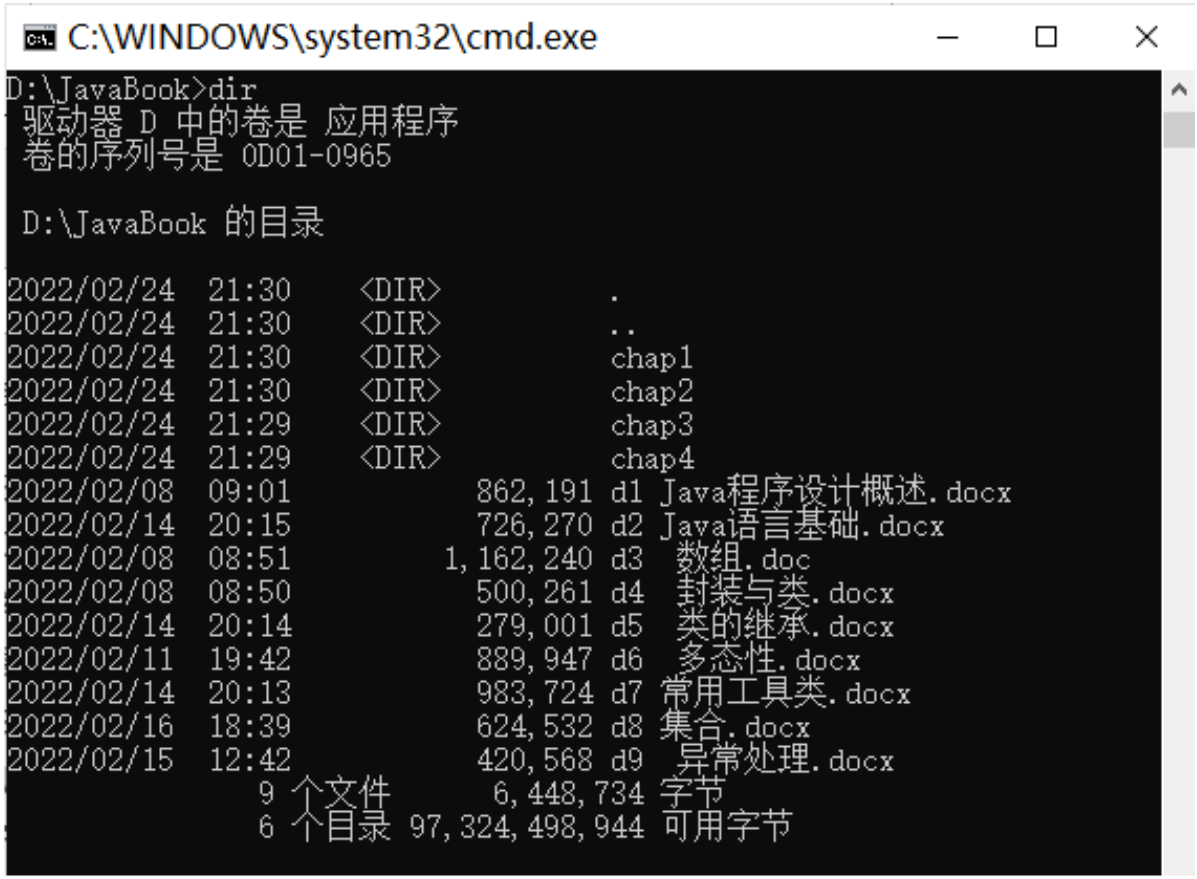
- static Stream<Path> list(Path dir)：返回一个可以读取目录中各个项的Stream<Path>对象。
- Stream是Java SE 8中新增的位于java.util.stream包中的接口，Stream对 Java 集合运算和表达进行了更高阶的抽象。Stream使用lambda表达式改进了集合的遍历代码；Stream提供了链式操作使以集合作为主语的方法调用可以依次连续书写。
- Stream API提高了Java程序员的生产力，使程序员能够写出高效率、干净、简洁的代码。

2. 获取文件信息

- Stream的void forEach(Consumer<? super T> action)方法以lambda表达式作为参数描述对集合成员的遍历操作。
- 例如，将List集合words转换为Stream对象后，使用forEach()方法利用lambda表达式即可组织wordStream的遍历。

```
List<String> words = Arrays.asList("Spring","Spring MVC","MyBatis");
Stream<String> wordStream = words.stream();
//word：集合中的每个元素
wordStream.forEach( word-> System.out.println(word) );
```

【例12-11】实现Windows环境下的dir命令。



- Files类不仅可以进行文件管理，也设计了对文本文件内容进行读写的方法。
  - static byte[] readAllBytes(Path path)：读取文件中的所有字节。
  - static List<String> readAllLines(Path path)：使用UTF-8编码读取文件中的所有行。
  - static Path write(Path path, byte[] bytes, OpenOption... options)：将字节数组写入文件。
  - static Path write(Path path, Iterable<? extends CharSequence> lines, OpenOption... options)：使用UTF-8编码将文本行写入文件。

- Fils中的这些方法可以一次性的读取文件中的所有内容，或者将数据一次性的写入目标文件。但是这种方式只适用于处理中等长度的文本文件，如果文件规模比较大，或者对二进制文件，则仍使用I/O流进行处理。

本章思维导图