**java提供了许多实现文件输入/输出的类。这些类可以分为文本I/O类和二进制I/O类。**

**File对象封装了文件或路径属性，但是不包含从/向文件读/写数据的方法。为了进行I/O操作，需要使用正确的javaI/O类创建对象。这些对象包含从/向文件中读/写数据的方法。例如，为了将文本写入一个名为temp.txt的文件中，可以使用PrintWriter类按如下方式创建一个对象：**

**PrintWriter output=new PrintWriter("temp.txt");**

**现在，可以调用该对象的print方法向文件写入一个字符串。例如**

**output.print("java101");**

**output.close();**

**java有许多用于各种目的的I/O类。通常，可以将它们分为输入类和输出类。输入类包含读数据的方法，而处处类包含写数据的方法。PrintWriter是一个输出类的例子，而Scanner是一个输入类的例子。例如**

**Scanner input=new Scanner(new File("temp.txt"));**

**System.out.println(input.nextLine());**

**输入对象从文件中读取数据流，输出对象将数据流写入文件。输入对象也称为输入流，输出对象称为输出流。**

**文本I/O与二进制I/O**

**二进制I/O不涉及编码和解码，因此比 文本I/O更加高效。**

**计算机并不区分二进制文件和文本文件。所有的文件都是以二进制形式来存储的，因此，从本质上说，所有的文件都是二进制文件。文件I/O建立在二进制I/O的基础上，它能提供一层抽象，用于字符层次的编码和解码。对于文本I/O而言，编码和解码都是自动进行的。**

**在写入一个字符时，java虚拟机会将统一码转化为文件指定的编码，而在读字符时，将文件指定的编码转化为统一码。**

**二进制I/O不需要转化。如果使用二进制I/O向文件写入一个数值，就是将内存中的那个值复制到文件中。**

**一般而言，对于文本编辑器或文本输出程序创建的文件，应该使用文本输入来读取，对于java二进制输出程序创建的文件，应该使用二进制输入来读取。**

**由于二进制I/O不需要编码和解码，所以，它比文本I/O效率高。二进制文件与主机的编码方案无关，因此，它是可移植的。在任何机器上的java程序可以读取java程序所创建的二进制文件。这就是为什么java的类文件存储为二进制文件的原因。java类文件可以在任何具有java虚拟机的机器上运行。**

**.txt命名文本文件，.dat命名二进制文件。**

**二进制I/O类**

**抽象类InputStream是读取二进制数据的根类，抽象类OutputStream是写入二进制数据的根类。**

**java.io.InputStream**

**read():int**

**read(b:byte[]):int**

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

**available():int**

**close():void**

**skip(n:long):long**

**markSupported():boolean**

**mark(readlimit:int):void**

**reset():void**

**java.io.OutputStream**

**write(int:b):void**

**write(b:byte[]):void**

**write(b:bte[],off:int,len:int):void**

**close():void**

**flush():void**

**二进制I/O类中的所有方法都声明为抛出java.io.IOException或java.io.IOException的子类。**

**FileInputStream和FileOutputStream**

**FileInputStream类和FileOutputStream类用于从/向文件读取/写入字节。它们的所有方法都是从InputStream类和OutputStream类继承的。FileInputStream类和FileOutputStream类没有引入新的方法。**

**java.io.FileInputStream**

**FileInputStream(file:File)**

**FileInputStream(filename:String)**

**如果试图为一个不存在的文件创建FileInputStream对象，将会发生java.io.FileNotFoundException异常。**

**java.io.OutputStream**

**FileOutputStream(file:File)**

**FileOutputStream(filename:String)**

**FileOutputStream(file:File,append:boolean)**

**FileOutputStream(filename:String,append:boolean)**

**如果append为true，数据将追加到已经存在的文件中**

**几乎所有的I/O类中的方法都会抛出异常java.io.IOException。因此，必须在方法中声明会抛出java.io.IOException异常，或者将代码放在try-catch块中。**

**FileInputStream类的实例可以作为参数去构造一个Scanner对象，而FileOutputStream类的实例可以作为参数构造一个PrinterWriter对象。可以创建一个PrinterWriter对象向文件中追加文本。如果temp.txt不存在，就会创建这个文件。如果temp.txt文件已经存在，就将新数据追加到这个文件中。**

**new PrintWriter(new FileOutputStream("temp.txt"),true));**

**FilterInputStream和FilterOutputStream**

**过滤器数据流（filter stream）是为了某种目的过滤字节的数据流。基本字节输入流提供的读取方法read只能用来读取字节。如果要读取整数值、双精度值或字符串，那就需要一个过滤器类来包装字节输入流。使用过滤器类就可以读取整数值、双精度值和字符串，而不是字节或字符。FilterInputStream类和FilterOutputStream类是过滤数据的基类。需要处理基本数值类型时，就是用DataInputStream和DataOutputStream类来过滤字节。**

**DataInputStream和DataOutputStream**

**DataInputStream从数据流读取字节，并且将它们转换为合适的基本类型值或字符串。DataOutputStream将基本类型的值或字符串转换为字节，并且将字节输出到数据流。**

**DataInputStream类扩展FilterInputStream类，并实现DataInput接口。**

**DataOutputStream类扩展FilterOutputStream类，并实现DataOutput接口。**

**DataInputStream实现了定义在DataInput接口中的方法来读取基本数据类型值和字符串。DataOutputStream实现了定义在DataOutput接口中的方法来写入基本数据类型值和字符串。基本类型的值不需要做任何转化就可以从内存中复制到输出流数据。**

**BufferedInputStream和BufferedOutputStram**

**BufferedInputStream类和BufferedOutputStream类可以通过减少磁盘读写次数来提高输入和输出的速度。使用BufferedInputStream时，磁盘上的整块数据一次性地读入到内存中的缓冲区中。然后从缓冲区中将个别的数据传递到程序中。使用BufferedOutputStream，个别的数据首先写入到内存中缓冲区中。当缓冲区以满时，缓冲区中的所有数据一次性写入到磁盘中。**

**可以使用其构造方法包装在任何一个InputStream类和OutputStream类上的BufferedOutputStream类和BufferedInputStream类。**

**DataOutputStream output=new DataOutputStream(**

**new BufferedOutputStream(new FileOutputStream("temp.dat")));**

**应该总是使用缓冲区I/O来加速输入和输出。对于小文件，我们可能注意不到性能的提升。但是，对于超过100MB的大文件，我们将会看到使用缓冲的I/O带来的实质性的提升。**

**示例学习：复制文件**

**import java.io.\*;**

**public class Copy {**

**/\*\* Main method**

**@param args[0] for sourcefile**

**@param args[1] for target file**

**\*/**

**public static void main(String[] args) throws IOException {**

**// Check command-line parameter usage**

**if (args.length != 2) {**

**System.out.println(**

**"Usage: java Copy sourceFile targetfile");**

**System.exit(1);**

**}**

**// Check if source file exists**

**File sourceFile = new File(args[0]);**

**if (!sourceFile.exists()) {**

**System.out.println("Source file " + args[0]**

**+ " does not exist");**

**System.exit(2);**

**}**

**// Check if target file exists**

**File targetFile = new File(args[1]);**

**if (targetFile.exists()) {**

**System.out.println("Target file " + args[1]**

**+ " already exists");**

**System.exit(3);**

**}**

**try (**

**// Create an input stream**

**BufferedInputStream input =**

**new BufferedInputStream(new FileInputStream(sourceFile));**

**// Create an output stream**

**BufferedOutputStream output =**

**new BufferedOutputStream(new FileOutputStream(targetFile));**

**) {**

**// Continuously read a byte from input and write it to output**

**int r, numberOfBytesCopied = 0;**

**while ((r = input.read()) != -1) {**

**output.write((byte)r);**

**numberOfBytesCopied++;**

**}**

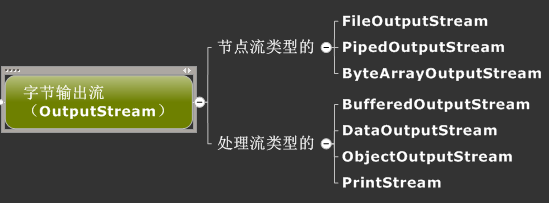
**// Display the file size**

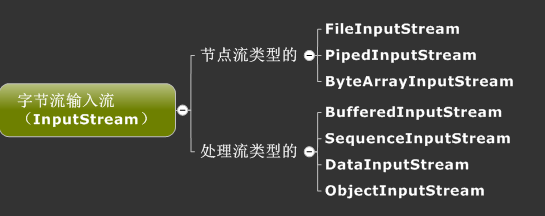
**System.out.println(numberOfBytesCopied + " bytes copied");**

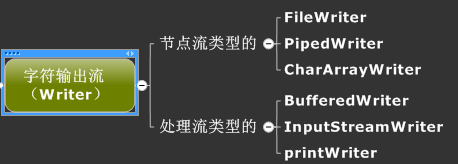
**}**

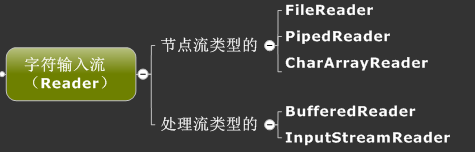
**}**

**}**









**InputStream类和OutputStream类是所有二进制I/O类的根类。FileInputStream类和FileOutputStream类关联一个文件用于输入/输出。BufferedInputStream类BufferedOutputStream类可以包装任何一个二进制输入/输出流以提高其性能。DataInputStream类和DataOutputStream类可以用来读写基本类型数据和字符串。**

**ObjectInputStream类和ObjectOutputStream类除了可以读写基本类型数据值和字符串，还可以读写对象。为实现对象的可序列化，对象的定义类必须实现java.io.Serializable标记接口。**

**RandomAccessFile类允许对文件读写数据。可以打开一个模式为“r”的文件，这个模式表示文件是只读的，也可以打开一个模式为"rw"的文件，这个模式表示文件是可更新的。由于RandomAccessFile类实现了DataInput和DataOutput接口，所以，RandomAccessFile中的许多方法都与DataInputStream和DataOutputStream中的方法一样。**