**第8课 输入 / 输出与文件处理**

|  |  |  |
| --- | --- | --- |
| **课 题** | 输入 / 输出与文件处理 | |
| **课 时** | 6课时（270min）。 | |
| **教学目标** | **知识技能目标：**  1．掌握文件类的概念和使用方法。  2．掌握字节型输入 / 输出流和缓冲区输入 / 输出流的基本使用方法。  **思政育人目标：**  让学生通过学习输入 / 输出与文件处理，了解Java 中处理数据的输入和输出是使用流来实现的。 | |
| **教学重难点** | **教学重点：**使用输入 / 输出流  **教学难点：**创建文件和文件夹 | |
| **教学方法** | 讲授法、问答法、讨论法 | |
| **教学用具** | 电脑、投影仪、多媒体课件、教材 | |
| **教学设计** | 第1节课：考勤（2min）--知识讲解（40min）--作业布置（3min）  第2节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第3节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第4节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第5节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第6节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min） | |
| **教学过程** | **主 要 教 学 内 容 及 步 骤** | **设计意图** |
| **考勤**  **（2min）** | ■【教师】清点上课人数，记录好考勤  ■【学生】班干部报请假人员及原因 | 培养学生的组织纪律性,掌握学生的出勤情况 |
| **知识讲解**  （40min） | **【教师】**展示使用输入 / 输出流（一）  **任务描述**  编写程序，由字符串生成字节型输入 / 输出流，并使用适当方法将流内容显示在屏幕上。  **任务目标**  1. 能够掌握 Java 中流概念和流的分类；  2. 能够掌握 Java 中字节型输入 / 输出流和缓冲区输入 / 输出流的基本使用方法；  3. 能够掌握 Java 中字符型输入 / 输出流和缓冲区输入 / 输出流的基本使用方法。  **预备知识**  流是 Java 中处理输入 / 输出的形式。字节型输入 / 输出流由 InputStream 类和OutputStream 类派生而来，字符型输入 / 输出流由 Reader 类和 Writer 类派生而来。  **一、流**  所有的编程语言都支持读入输入和显示输出，Java 语言以流的形式处理所有的输入和输出。流是由源到目的地通过通信路径传递的一串字节或字符，流中的数据只能向一个方向流动，所以我们使用区分输入数据和输出数据的流。流在被接收时称为输入（读），在被发送时称为输出（写）。使用输入流可从文件或键盘读入数据，而使用输出流可将数据写入到文件或其他输出设备，如显示在屏幕上。  Java 中的流按处理的数据类型又可分为字节流和字符流，分别用于处理字节型数据和字符型数据。字节输入流的基类是 InputStream，字节输出流的基类是 OutputStream，字符输入流的基类是 Reader，字符输出流的基类是 Writer。这四个类都是抽象类，是所有其他流的父类。  接下来将介绍字节流的使用方法。  **二、字节型输入 / 输出流**  **（一）InputStream 类**  InputStream 类是一个抽象类，不能被实例化，仅用于派生各种字节型输入流，它的常用方法如表 8−1 所示，这些方法在 InputStream 的各种子类中也可以使用。    InputStream 类的常见子类有：  FileInputStream：文件输入流，用于读取本地文件系统的文件数据（信息）。  PipedInputStream：管道输入流，用于实现管道流的输入处理。  ByteArrayInputStream：字节数组输入流，允许将内存中的一个缓冲区作为输入流使用，实现内存的读取操作。  SequenceInputStream：连接输入流，将两个或更多的输入流对象连接起来。  StringBufferInputStream：用于读取 StringBuffer 类的可变字符串中的数据。  ObjectInputStream：对象输入流，用于输入流中读取对象类数据。  FilterInputStream：过滤输入流，重写了 InputStream 中的所有方法，使程序在读取数据的同时，还可以进行数据处理，它也是一个抽象类，有 BufferedInputStream、LineNumberInputStream、PushbackInputStream 和 DataInputStream 等子类。  **（二）OutputStream 类**  OutputStream 类也是一个抽象类，用于派生各种字节型输出流。  OutputStream 类的常见子类有：  ObjectOutputStream：对象输出流，用于直接进行对象的写入。  FileOutputStream：文件输出流，用于将数据写入到本地文件系统的文件中。  PipedOutputStream：管道输出流，用于实现管道流的输出处理。  ByteArrayOutputStream：字节数组输出流，用于向内存中写入数据。  FilterOutputStream：过滤输出流，重写了 OutputStream 中的所有方法，使程序在写入数据的同时，还可以进行数据处理，它也是一个抽象类，有 DataOutputStream、BufferedOutputStream 和 PrintStream 等子类。  **三、字符型输入 / 输出流**  尽管字节流提供了处理任何类型输入 / 输出操作的足够的功能，它们不能直接操作Unicode 字符。Java 的一个主要目的是实现程序的“只写一次，到处运行”，那么包括直接的字符输入／输出是必要的。  **（一）Reader 类**  Reader 类是一个抽象类，不能被实例化，仅用于派生各种字符型输入流，该类的所有方法在出错情况下都将引发 IOException 异常。  Reader 类的常见子类有：  FileReader：文件输入流，用于读取本地文件系统的文件数据（信息）。  PipedReader：管道输入流，用于实现管道流的输入处理。  CharArrayReader：是字符数组输入流，用于读取字符数组，操作的数据是以字符为单位。  BufferedReader：是缓冲字符输入流，它的作用是为其他字符输入流添加一些缓冲功能。  InputStreamReader：将字节流转换为字符流。它是字节流通向字符流的桥梁，能将字节流输出为字符流，并且能为字节流指定字符集，可输出一个个的字符。如果不指定字符集编码，该解码过程将使用平台默认的字符编码，例如：GBK。  FilterReader：是实现自定义过滤输入字符流的基类，本身是一个抽象类，为所有装饰类提供一个标准，只是简单重写了父类 Reader 的所有方法，要求子类必须重写核心方法和提供具有自己特色的方法。这里没有像字节流那样有很多的子类来实现不同的功能，可能是因为字符流本来就是字节流的一种装饰，所以在这里没有必要再对其进行装饰，只是提供一个扩展的接口而已。  **（二）Writer 类**  OutputStream 类也是一个抽象类，用于派生各种字符型输出流，该类的所有方法都返回一个 void 值并在出错条件下引发 IOException 异常。  Writer 类的常见子类有：  FileWriter：文件输出流，用于将字符写入本地文件系统的文件。  PipedWriter：管道输出流，用于实现管道流的输出处理。  CharArrayWriter：字符数组输出流，实现了以数组为目标的输出流。  BufferedWriter：缓冲字符输出流，为其他字符输出流添加一些缓冲功能。  OutputStreamWriter：将输出的字符流转换为字节流，即将一个字符流的输出对象变为字节流的输出对象。它是字符流通向字节流的桥梁，可使用指定的 charset 将要写入流中的字符编码成字节。它使用的字符集可以由名称指定或显式给定，否则将接受平台默认的字符集。  PrintWriter：是字符类型的打印输出流。它用于向文本输出流打印对象的格式化表示形式。它实现在 PrintStream 中的所有 print 方法。它不包含用于写入原始字节的方法，对于这些字节，程序应该使用未编码的字节流进行写入。  **四、缓冲区输入 / 输出流**  缓冲区输入 / 输出流的特点是在内存中开辟一段空间作缓冲区，其基本操作就是往缓冲区中写入数据（输出流）或从缓冲区中读取数据（输入流）。缓冲区输入 / 输出流一共有以下三种。  **（一）ByteArrayInputStream 类**  字节数组输入流类，用于从一个字节数组取得输入数据。  **（二）ByteArrayOutputStream 类**  字节数组输出流类，用于把数据写进字节数组。  **（三）StringBufferInputStream 类**  它的构造函数以一个字符串为参数，原型为：  public StringBufferInputStream(String s)  其他常用方法均与 ByteArrayInputStream 中方法同名且基本功能相似。  **【学生】**思考、讨论。 | **展示使用输入 / 输出流（一），让学生更加仔细的阅读，从而激发学生的学习欲望。** |
| **作业布置**（3min） | **【教师】**布置课后作业  简述缓冲区输入 / 输出流的分类。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**展示使用输入 / 输出流（二）  **五、Java 的标准输入 / 输出**  Java 程序使用字符界面与系统标准输入 / 输出界面进行数据通信，即从键盘读入数据或向屏幕输出数据，这是十分常见的操作。为此而频繁创建输入 / 输出流类对象很不方便。因此，Java 系统事先定义了三个流对象，分别与系统的标准输入（stdin）、标准输出（stdout）和标准错误输出（stderror）相联系。  标准输入文件是键盘。标准输出文件是终端屏幕。标准错误输出文件也指向屏幕，如果有必要它也可以指向另一个文件以便和正常输出区分。  System 类管理标准输入 / 输出流和错误流，System 类在 Java.lang 包中，System 类的所有属性和方法都是静态的。调用时需要以类名 System 为前缀。  System.in: 从标准输入获取输入（通常是键盘）。  System.out: 把输出送到默认的显示（通常是显示器）。  System.err: 把错误信息送到默认的显示。  每当 main( ) 方法被执行时，就自动生成上述 3 个对象。  **（一）标准输入**  Java 的标准输入 System.in 是 InputStream 类的对象，当程序需要从键盘读入数据的时候。只需调用 System.in 的 read( ) 方法即可。  如下面的语句将从键盘读入一个字节的数据：  char ch = System.in.read();  在使用 System.in.read( ) 方法读入数据时，需要注意以下几点。  （1）System.in.read( ) 语句必须包含在 try 语句块中，且 try 语句块后面应该有一个可接受 IOException 异常的 catch 块。  （2）执行 System.in.read( ) 方法将从键盘缓冲区读入一个字节的数据。然而返回的却是 16 位的整型量。其低位字节是真正输入的数据。高位字节全为零。作为 InputStream类的对象，System.in 只能从键盘读取二进制的数据，而不能把这些信息转换为整数、字符，浮点数或字符串等复杂数据类型的量。  （3）当键盘缓冲区中没有未被读取的数据时。执行 System.in.read( ) 将导致系统转入阻塞状态。在阻塞状态下。当前流程将停留在上述语句位置，整个程序被挂起。等用户输入一个键盘数据后才能继续运行下去。所以程序中有时利用 System.in.read( ) 语句来达到暂时保留屏幕的目的。  **（二）标准输出**  Java 标准输出 System.out 是 PrintStream 类的对象。PrintStream 是过滤输出流类FilterOutputStream 的一个子类，其中定义了向屏幕输出不同类型数据的方法 print( ) 和println( )。  println( ) 的作用是向屏幕输出其参数指定的变量或对象，然后再换行，使光标停留在屏幕下一行第一个字符的位置。print( ) 方法与 println( ) 不同的是输出对象后不换行，下一次输出时，将输出在同一行。  任务实施  /\* 源程序文件名：Ex0801.java\*/  import java.io.\*;  public class Ex0801  {  public static void main(String args[]) throws IOException  {  String s = "This is a test of I/O stream";  byte buffer[] = s.getBytes();  ByteArrayOutputStream baout = new ByteArrayOutputStream();  for(int i = 0; i<buffer.length; i++)  baout.write(buffer[i]);  ByteArrayInputStream bain = new ByteArrayInputStream  (baout.toByteArray());  boolean eof = false;  while(!eof)  {  int c = bain.read();  if(c==-1)  eof = true;  else  System.out.print((char)c);  }  System.out.println("\n'writeTo 方法可以达到同样的结果：");  baout.writeTo(System.out);  baout.close();  System.out.println("\n 缓冲输入流的长度 ( 未重置指针 )："+bain.available());  bain.reset();  System.out.println(" 缓冲输入流的长度 ( 重置指针后 )："+bain.available());  }  }  在这个例子中，我们将一个 String 对象转换成字节数组后，构造了一个字节数组输出流 ByteArrayOutputStream，这是 Java 输出流的一种。同时构造了一个字节数组输入流ByteArrayInputStream。然后使用不同的方法将输入流和输出流里的内容显示在屏幕上。最后还使用输入流的 available 方法和 reset 方法查看了不同情况下流内的可用字节数。  注意这些语句：  for(int i = 0; i<buffer.length; i++)  baout.write(buffer[i]);  在这个 for 循环中，通过每次向输出流 baout 写入一个字节内容，最终构造了一个内容和字符串 s 相同的字节输出流。  ByteArrayInputStream bain = new ByteArrayInputStream  (baout.toByteArray());  这条语句使用输出流转换后的字节数组构造了一个字节数组输入流。  while(!eof)  {  int c = bain.read();  if(c==-1)  eof = true;  else  System.out.print((char)c);  }  在 while 循环中，每次读取输入流的一个字节内容，如果值不为 −1，则将其转换为字符后输出，否则表示输入流已经结束。  baout.writeTo(System.out);  输出流的 writeTo 方法是将流内容写入到另一个输出流去，这里写入的是系统输出流 System.out，所以字节流的内容同样会显示在屏幕上。  available 方法用于查看缓冲区内的可用字节数。由于在前面的代码中，输入流内的内容已经全部读出，指针指向的是流的末尾，所以第一次的 available 方法得到的结果是零。在使用了 reset 方法后，指针已经重置到流的开头，所以第二次 available 方法得到了输入流的完整长度。  有关输入 / 输出流的概念和使用，请大家继续学习下面的任务。  **【学生】**思考、讨论。 | **通过教师讲解，了解使用输入 / 输出流（二）的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了使用输入 / 输出流（二），让学生知道字节型输入 / 输出流由 InputStream 类和OutputStream 类派生而来，字符型输入 / 输出流由 Reader 类和 Writer 类派生而来。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | **【教师】**布置课后作业  简述Java 中字符型输入 / 输出流和缓冲区输入 / 输出流的基本使用方法。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示创建文件和文件夹  **任务描述**  编写程序，建立几个测试用的文件和文件夹。  **任务目标**  能够掌握 Java 中 File 类的使用方法。  **预备知识**  对文件的操作首先是建立文件和文件夹，Java 中使用 File 类的相关方法来完成这一操作。Java 中的 File 类是顶层类 Object 类的子类，在 java.io 包中，它提供了类似Windows 资源管理器的一些功能，可以对文件和文件夹进行诸如创建和删除文件及文件夹、浏览文件等操作。  File 类的构造函数如下：  File（String pathname）：通过把给出的路径名转换为抽象路径名，创建一个新的 File 实例 ;  File（String parent, String child）：由父路径名和子路径名创建一个新的 File 实例 ;  File（File parent, String child）：由父 File 对象和子路径名创建一个新的 File 实例。  File 类的构造函数只是构造了一个 File 类对象，该对象既可以是文件夹，也可以是文件，需要调用相应的 mkdir( )、mkdirs( ) 和 createNewFile( ) 方法创建文件夹和文件。  其中的路径分隔符是双反斜杠“\\”，因为 Java 是跨平台的，在 Windows 操作系统下双反斜杠“\\”将转换为“\”，而在 linux 操作系统下将转换为“/”。  File 对象常用方法：  mkdir( ) 方法：用于创建路径中的当前文件夹，其父文件夹必须已经存在，否则创建不成功。  mkdirs( ) 方法：用于创建路径中的所有文件夹。  createNewFile( ) 方法：用于创建路径中的文件。它会抛出一个 IOException 异常，调用时必须捕捉这种异常。  String getName( ) 方法：返回路径或文件名。  String getPath( ) 方法：返回当前路径名。  String getAbsolutePath( ) 方法：返回当前绝对路径。  String getParent( ) 方法：返回当前路径的父目录；如果当前路径没有父目录，则返回null。  boolean exists( ) 方法：判断当前目录或文件是否存在。  boolean canWrite( ) 方法：判断当前文件是否可写。  boolean canRead( ) 方法：判断当前文件是否可读。  boolean isFile( ) 方法：判断当前是否是文件。  boolean isDirectory( ) 方法：判断当前是否是路径。  **任务实施**  /\* 源程序文件名：Ex0802.java\*/  import java.io.\*;  public class Ex0802  {  public static void main(String args[])  {  File dirobj1 = new File("c:\\FileTest");  File dirobj2 = new File("c:\\FileTest\\File1");  File dirobj3 = new File(dirobj1,"File2");  File dirobj4 = new File("c:\\FileTest","File3");  dirobj1.mkdir();  dirobj2.mkdirs();  dirobj3.mkdirs();  dirobj4.mkdirs();  File fileobj1 = new File ("c:\\FileTest\\Text1.dat");  File fileobj2 = new File("c:\\FileTest","Text2.dat");  File fileobj3 = new File(dirobj1,"Text3.dat");  try  {  fileobj1.createNewFile();  fileobj2.createNewFile();  fileobj3.createNewFile();  }  catch(IOException e)  {  System.out.println(" 创建文件时异常！ ");  e.printStackTrace();  }  }  }  **【学生】**思考、讨论。 | **教师通过创建文件和文件夹展示，让学生了解议论文阅读与写作的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了创建文件和文件夹，能够掌握 Java 中 File 类的使用方法。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **思考一下，File 对象常用方法。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示文件输入 / 输出操作  **任务描述**  编写程序，从一个文件中读出数据，把数据显示到屏幕，并写入到另一个文件。  **任务目标**  1. 能够掌握 Java 中文件输入 / 输出的基本操作方法。  2. 能够掌握 Java 中字节流文件的读取。  3. 能够掌握 Java 中字符流文件的读取。  **预备知识**  在创建了文件夹和文件后，可以使用字节流类、字符流类、随机存取文件类访问文件，读出或写入数据。  **一、字节流文件的读取**  **（一）FileInputStream**  FileInputStream 类用于文件的输入操作，主要用于文件的读取，其构造函数如下：  FileInputStream(String filename)：使用文件名创建一个新的 FileInputStream 对  象 , 该对象和参数所指定的文件相关联 , 该文件必须是在系统中已经存在的 ;  FileInputStream(File file)：使用文件对象创建一个新的 FileInputStream 对象。构造函数会抛出 FileNotFoundException 异 常，read 方法和 close 方法会抛出IOException 异常，使用该类时必须捕获该异常。  **（二）FileOutputStream**  FileOutputStream 类用于文件的输出操作，主要用于文件的写入，FileOutputStream 类的构造函数如下：  FileOutputStream(String filename): 使用文件名创建一个新的 FileOutputStream对象 , 该对象和参数所指定的文件相关联 , 该文件必须是在系统中已经存在的 ;  FileOutputStream(File file): 使用文件对象创建一个新的 FileOutputStream 对象。  IOException 异常，使用该类时必须捕获该异常。  **二、字符流文件的读取**  在 JDK1.1 之前，Java 中只有普通的字节流，这种流对于处理以 16 位的 Unicode 码表示的字符流很不方便。从 JDK1.1 开始，Java 中加入了专门用于字符流处理的类，它们是从 Reader 和 Writer 类派生出来的。  **（一）FileReader**  FileReader 类用于文件的输入操作，主要用于文件的读取，其构造函数如下：  FileReader(String filename): 使用文件名创建一个新的 FileReader 对象 , 该对象和参数所指定的文件相关联 , 该文件必须是在系统中已经存在的 ;  FileReader(File file): 使用文件对象创建一个新的 FileReader 对象。  构造函数会抛出 FileNotFoundException 异常，read 方法和 close 方法会抛出 IOException异常，使用该类时必须捕获该异常。  **（二）FileWriter**  FileWriter 类用于文件的输出操作，主要用于文件的写入，FileWriter 类的构造函数如下：  FileWrite(String filename): 使用文件名创建一个新的 FileWriter 对象 , 该对象和参数所指定的文件相关联 , 该文件必须是在系统中已经存在的 ;  FileWriter(File file): 使用文件对象创建一个新的 FileWriter 对象。  构造函数会抛出 FileNotFoundException 异 常，write 方法和 close 方法会抛出IOException 异常，使用该类时必须捕获该异常。  **三、缓冲流的应用**  内存的读取速度在所有 I/O 操作对象中是最快的，将输入 / 输出流缓冲到内存中是提高程序性能的一个方法。将缓冲数据保存在内存中可以提供给以后的程序使用。当Java 程序需要读取输入流时，首先会查询缓冲，如果存在则直接读取，这就大大提高了读取数据的速度。  **（一）基于字节的缓冲流**  Java 的 BufferedInputStream 类可以对任何的 InputStream 进行带缓冲区的封装以达到性能的改善。要将输入流保存到缓冲区中，可以使用 BufferedInputStream 和FileInputStream 的对象相接创建一个缓冲流，当从输入流读取数据时就被保存到缓冲区了。  BufferedInputStream 的构造函数如下：  3BufferedInputStream(InputStream in): 使用字节输入流对象创建一个新的缓冲区大小默认的 BufferedInputStream 对象 ;  BufferedInputStream(InputStream in,int size): 使用字节输入流对象创建一个新的缓冲区大小为 size 的 BufferedInputStream 对象。  输出缓冲使用 BufferedOutputStream 类，和基于字节的输入缓冲流用法相同。BufferedOutputStream 类在方法上和 OutputStream 类没有太多的变化，只是增加了一个flush( ) 方法，这个方法用来将缓冲区内的数据强制写入目标设备中。  **（二）基于字符的缓冲流**  Java 的 BufferedReader 类可以对任何的 Reader 进行带缓冲区的封装以达到性能的改善。要将输入流保存到缓冲区中，可以使用 BufferedReader 和 FileReader 的对象相接创建一个缓冲流，当从输入流读取数据时就被保存到缓冲区了。  BufferedReader 的构造函数如下：  BufferedReader(Reader in): 使用字符输入流对象创建一个新的缓冲区大小默认的BufferedReader 对象 ;  BufferedReader(Reader in, int size): 使用字符输入流对象创建一个新的缓冲区大小为 size 的 BufferedReader 对象。  输出缓冲使用 BufferedWriter 类，和基于字符的输入缓冲流用法相同。BufferedReader类在方法上和 Writer 类变化不大。  BufferedReader 类的方法 readLine( ) 能够读出一行字符，而 newLine( ) 方法则是写入一行字符。flush( ) 方法用来将缓冲区内的数据强制写入目标设备中。  请看下面利用字符缓冲流 BufferedReader 类读取文件内容并显示的例子。  import java.io.\*;  public class BufferedReaderDemo  {  public static void main(String[] args)throws IOException  {  String str;  FileReader fr = new FileReader("d:\\javacode\\myProject\\source.  txt");  BufferedReader bfr = new BufferedReader(fr);  while((str = bfr.readLine())!=null)  System.out.println(str);  fr.close();  }  }  请看下面利用字符缓冲流 BufferedWriter 类把 5 个随机数写入文件的例子。  import java.io.\*;  public class BufferedWriterDemo  {  public static void main(String[] args)throws IOException  {  FileWriter fw = new FileWriter("d:\\javacode\\myProject  \\target.txt");  BufferedWriter bfw = new BufferedWriter(fw);  for(int i = 1;i<=5;i++)  {  bfw.write(Double.toString(Math.random()));  bfw.newLine();  }  bfw.flush();  fw.close();  }  }  **四、随机存取文件类 RandomAccessFile**  随 机 访 问（RandomAccess）是 指： 数 据 可 读 或 写 到 文 件 中 的 随 机 单 元。 在FileInputStream 和 FileOutputStream 中，数据将作为连续信息流分别进行读写。Java 提供的 RandomAccessFile 类来执行文件内特定单元的输入和输出操作，即使用该类的对象既可以从文件读取数据，也可以将数据写到文件。RandomAccessFile 类也支持读和写的许可，允许文件以只读或只写的方式被访问。有两种方式来创建随机访问文件，使用作为路径名或 File 类对象，这两种构造函数的原型分别为：  public RandomAccessFile(String filename, String mode) throws FileNot  FoundException  public RandomAccessFile(File file, String mode) throws FileNotFound  Exception  其中的字符串参数 mode 用于指定文件的访问方式为只读、只写或可读写，分别用“r”“w”和“rw”表示。  **任务实施**  /\* 源程序文件名：Ex0803.java\*/  import java.io.\*;  public class Ex0803  {  public static void main(String args[])  {  try  {  FileInputStream inobj = new FileInputStream("c:\\FileTest\\Text1.  dat");  FileOutputStream outobj = new FileOutputStream("c:\\FileTest\\Text2.  dat");  byte byarray[] = new byte[2];  while(inobj.read(byarray)!=-1)  {  String strobj = new String(byarray);  System.out.print(strobj);  outobj.write(byarray);  }  inobj.close();  outobj.close();  }  catch(FileNotFoundException e)  {  System.out.println(" 未找到指定文件！ ");  e.printStackTrace();  }  catch(IOException e)  {  System.out.println(" 文件读写错误！ ");  e.printStackTrace();  }  }  }  **【学生】**思考、讨论。 | **教师通过文件输入 / 输出操作展示，让学生了解文件输入 / 输出操作的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了文件输入 / 输出操作，能够掌握 Java 中文件输入 / 输出的基本操作方法。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **思考一下，Java 中字节流文件如何读取？** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示思考与练习  **一、简答题**  （1）什么是流？ Java 中的流有哪几种？  （2）File 类的作用是什么？有哪些常用方法？  （3）向文件中写入数据和从文件读出数据需要使用到哪些类和方法？  **二、选择题**  （1）对应标准输入设备键盘的是下列哪一项？（　　）  A. System.in B. System.out  C. System.err D. System.gc( )  （2）对应标准输出设备屏幕的是下列哪一项？（　　）  A. System.in B. System.out  C. System.err D. System.gc( )  （3）在 java.io 类库中，InputStream 和（　　）是处理字节数据的基本输入 / 输出类。  A. Stream B. File  C. OutputStream D. Reader  （4）字符流与字节流的区别在于（　　）。  A. 前者带有缓冲，后者没有  B. 前者是块读写，后者是字节读写  C. 二者没有区别，可以互换使用  D. 每次读写的字节数不同  （5）FileInputStream 构造方法的有效参数是（　　）。  A. 无参数 B. InputStream 对象  C. File 对象 D. 以上所有  **【学生】**思考、讨论。 | **教师通过思考与练习展示，让学生了解思考与练习的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了思考与练习，让学生能够巩固你所学知识。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **RandomAccessFile 类的作用是什么？** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示实训任务  （1）编写程序：从键盘接收用户输入的字符串。  （2）编写程序：接收用户输入的学生姓名和成绩，并将其存入一个文本文件中。  （3）编写程序：从上题创建的文件中读取指定学生成绩。  （4）尝试使用字符流完成以上各题。  **【学生】**思考、讨论。 | **教师通过实训任务展示，让学生了解实训任务的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了实训任务，增强学生的实际操作能力。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **用字符流实现文件的复制。** | 通过课后练习，使学生巩固所学新知识 |
| **教学反思** | 课堂上精讲多练，着重点放在后三分之二同学身上，让他们能听懂学会。 | |