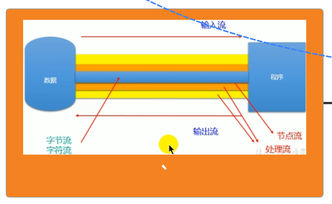
- File
  - 特点
    - java.io.File 类：文件和文件目录路径的抽象表示形式，与平台无关
    - File 类的一个对象，代表一个文件或一个文件目录
    - File 对象可以作为参数传递给流的构造器。
    - File 能新建、删除、重命名文件和目录，但 File 不能访问文件内容本身。如果需要访问文件内容本身，则需要使用输入 / 输出流。
  - 路径分隔符
    - 路径分隔符 windows:\\ unix:/
  - 创建 file 类的实例
    - File(String filePath)：以 filePath 为路径创建 File 对象，可以是绝对路径或者相对路径
    - **File(String parentPath(创建文件所在目录),String childPath(创建文件的名称))：以 parentPath 为父路径，childPath为子路径创建 File 对象。**
      File(File parentFile,String childPath)：根据一个父 File 对象和子文件路径创建 File 对象

      解释:
      举个例子，有一个文件my.txt，它的绝对路径为D:\Program\myfile\my.txt,这里有三个目录。
      D:\，D:\Program\，D:\Program\myfile\。前者是后者的父目录
  - 创建或删除文件或文件目录的一些方法
    - public boolean createNewFile()：创建文件。若文件存在，则不创建，返回 false；
      public boolean mkdir()：创建文件目录。如果此文件目录存在，就不创建了。如果此文件目录的上层目录不存在，也不创建。
      public boolean mkdirs()：创建文件目录。如果上层文件目录不存在，一并创建。

      删除磁盘中的文件或文件目录
      public boolean delete()：删除文件或者文件夹。
      删除注意事项：Java 中的删除不走回收站。
  - 查看文件属性一些方法
    - public String getAbsolutePath()：获取绝对路径
      public String getPath()：获取路径
      public String getName()：获取名称
      public String getParent()：获取上层文件目录路径。若无，返回 null
      public long length()：获取文件长度（即：字节数）。不能获取目录的长度。
      public long lastModified()：获取最后一次的修改时间，毫秒值
    - 如下的两个方法适用于文件目录：
      public String[] list()：获取指定目录下的所有文件或者文件目录的名称数组
      public File[] listFiles()：获取指定目录下的所有文件或者文件目录的 File 数组
    - File 类的重命名功能
      public boolean renameTo(File dest):把文件重命名为指定的文件路径 比如：file1.renameTo(file2) 为例：
      要想保证返回 true, 需要 file1 在硬盘中是存在的，且 file2 不能在硬盘中存在。
      File file1 = new File("hello.txt");
      File file2 = new File("D:\\book\\num.txt");
      boolean renameTo = file2.renameTo(file1);
      System.out.println(renameTo);
  - 判断文件某些属性是否存在的一些方法
    - **public boolean isDirectory()：判断是否是文件目录**
      public boolean isFile()：判断是否是文件
      public boolean exists()：判断是否存在
      public boolean canRead()：判断是否可读
      public boolean canWrite()：判断是否可写
      public boolean isHidden()：判断是否隐藏

关闭流的顺序和打开流的顺序相反。只要关闭最外层流即可，关闭最外层流也会相应关闭内层节点流。
flush() 方法的使用：手动将 buffer 中内容写入文件。
如果是带缓冲区的流对象的 close() 方法，不但会关闭流，还会在关闭流之前刷新缓冲区，关闭后不能再写出



- 字节流
  InputStream
  OutStream
  一般用于读写非文本

  读取字符容易产生乱码
  - 和字符流差不多,不过得用byte[] 数组
  - InputStream — FileInputStream
    - 构造方法
    - 方法
      - read()一次读取一个字节
        read(byte[])一次最多读取byte.length长度,返回每次读取的字节长度
      - 分支主题 2
  - OutputStream — FileOutputStream
    - 构造方法 — FileOutputStream(filePath) 会清空原文件内容后,填入内容
      FileOutputStream(filePath,true) 会在原文件的基础上追加内容
    - 方法 — write(数组,起始位置(start),结束位置(end)),将数组中start到end的内容,写入输入流
- 字符流
  Reader
  Writer
  - FileReader 中使用 read(char[] cbuf)读入数据

    ```
    // 1.File 类的实例化
      File file = new File("hello.txt");

      // 2.FileReader 流的实例化
      FileReader fr = new FileReader(file);
      //读取文件中的内容
      char[] cbuf = new char[1024];
      int len;
      while ((len = fr.read(cbuf)) != -1) {
       String str = new String(cbuf, 0, len);
       System.out.print(str);
    }
    fr.close()
    ```

    ● FileReader相关方法:
    1) new FileReader(File/String)
    2) read:每次读取单个字符，返回该字符，如果到文件末尾返回-1
    3) read(char[]):批量读取多个字符到数组，返回读取到的字符数，如果到文件末尾返回-1
    相关API:
    1) new String(char[]):将char[]转换成String
    2) new S：ing(char[],off,len):将char[]的指定部分转换成String
  - FileWriter 写出数据的操作写入文件

    ● FileWriter常用方法
    1) new FileWriter(File/String)：覆盖模式，相当于流的指针在首端
    2) new FileWriter(File/String,true)：追加模式，相当于流的指针在尾端
    *3) write(int):写入单个字符
    4) write(char[]):写入指定数组
    5) write(char[],off,len):写入指定数组的指定部分
    6) write（string）：写入整个字符串
    7) write(string,off,len):写入字符串的指定部分
    相关API：String类： toCharArray:将String转换成char[]
    ➢ 注意:
    FileWriter使用后，必须要关闭(close)或刷新(flush)，否则写入不到指定的文件!

    - 创建流对象，建立数据存放文件。
      FileWriterfw= new FileWriter(new File(“Test.txt”));

      调用流对象的写入方法，将数据写入流。
      fw.write(“atguigu-songhongkang”);

      关闭流资源，并将流中的数据清空到文件中。
      fw.close();
    - File 对应的硬盘中的文件如果不存在，在输出的过程中，会自动创建此文件

      如果流使用的构造器是：FileWriter(file,false) / FileWriter(file):对原有文件的覆盖。
      如果流使用的构造器是：FileWriter(file,true):不会对原有文件覆盖，而是在原有文件基础上追加内容。
- 节点流 — 从一个特定的数据源读写数据
- 缓冲流
  - BufferedInputStream 和 BufferedOutputStream
    - 当使用 BufferedInputStream 读取字节文件时，BufferedInputStream 会一次性从文件中读取 8192 个 (8Kb)，存在缓冲区中，直到缓冲区装满了，才重新从文件中读取下一个 8192 个字节数组
    - 向流中写入字节时，不会直接写到文件，先写到缓冲区中直到缓冲区写满，BufferedOutputStream 才会把缓冲区中的数据一次性写到文件里。使用方法 **flush()** 可以强制将缓冲区的内容全部写入输出流。
  - BufferedReader 和 BufferedWriter
- 对象处理流
  - 序列化: 保存值和数据类型
  - 反序列化: 恢复数据时,恢复数据值和数据的类型

    一个类要实现序列化和反序列化必须实现 Serializable(推荐)/Externalizable(有方法需要实现) 接口
  - ➢ 注意事项和细节说明

    1) 读写顺序要一致
    2) 要求序列化或反序列化对象 ，需要 实现 Serializable
    3) 序列化的类中建议添加SerialVersionUID,为了提高版本的兼容性
    4) 序列化对象时，默认将里面所有属性都进行序列化，但除了static或transient修饰的成员
    5) 序列化对象时，要求里面属性的类型也需要实现序列化接口
    6) 序列化具有可继承性，也就是如果某类已经实现了序列化，则它的所有子类也已经默认实现了序列化

    Java对应序列化/反序列化对应类
    - ObjectOutputStream

      ```
      //1.创建流 ...
      ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("src\\data.dat"));
      // 2.写入对象 (序列化)
      // 注意 100 也保存的，以integer 进行装箱 Integer 实现了 Serializable
      oos.writeInt(100);// 模型型值100写入流中
      // oos.write(100);//
      // Boolean 实现了 Serializable
      oos.writeBoolean(true);//
      // Charater 实现了 Serializable
      oos.writeChar('a');//
      // Double 实现了 Serializable
      oos.writeDouble(9.5);//
      // String 实现了 Serializable
      oos.writeUTF("韩三丰");//写入整个字符串
      // Dog 实现了 Serializable
      oos.writeObject(new Dog("小黄", 10));// 序列化对象
      oos.writeObject(new Dog("小花", 2));// 序列化对象
      oos.writeObject(new Dog("小白", 2));// 序列化对象
      // 3.关闭
      oos.close();
      System.out.println("以序列的方式保存ok~");
      ```

      分支主题 1
    - ObjectInputStream

      ```
      System.out.println(ois.readInt());
      System.out.println(ois.readBoolean());
      System.out.println(ois.readChar());
      System.out.println(ois.readDouble());
      System.out.println(ois.readUTF());
      Object dog = ois.readObject();
      System.out.println("运行类型=" + dog.getClass());
      System.out.println("dog信息=" + dog);//底层 Object -> Dog
      ```

      注意:返回的对象是Object类型,需要强转为原类型

      注意:读取(即:反序列化)的顺序要与保存时的顺序相同,否则会报错
  - 保存值
- 标准输入输出流

  ```
  //System 类 的 public final static InputStream in = null;
  // System.in 编译类型   InputStream
  // System.in 运行类型   BufferedInputStream
  // 表示的是标准输入 键盘
  System.out.println(System.in.getClass());

  //装饰解读
  //1. System.out public final static PrintStream out = null;
  //2. 编译类型 PrintStream
  //3. 运行类型 PrintStream
  //4. 表示标准输出 显示器
  System.out.println(System.out.getClass());
  ```

  分支主题 1
- 转换流
  字节流 -> 字符流
  - InputStreamReader — InputStreamReader abc = new InputStreamReader(new FileInputStream(文件路径), "utf-8");
    BufferedReader bufferedReader = new BufferedReader(abc);
  - OutStreamWriter — OutputStreamWriter outputStreamWriter = new OutputStreamWriter(new FileOutputStream(文件路径),"utf-8");