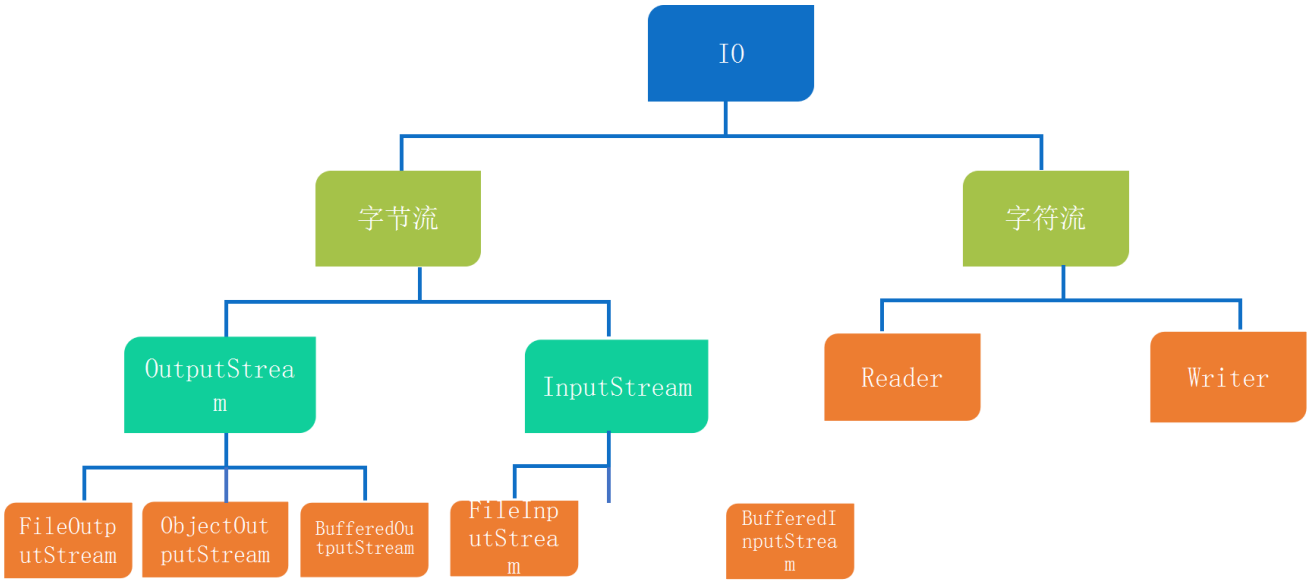
Java I/O

1、IO分类



2、File类

2-1File类创建

文件是组织一堆数据的方式。类型有很多：`.txt、.doc、.mp3、.jpg、.rmvb`等应用程序经常需要对文件进行操作，Java提供了`java.io.File`类用于对文件的封装File类中定义了一些与平台无关的方法供我们来操作文件，例如创建、删除、重命名文件等。在Java中，目录(文件夹)也是用File类来进行封装的。

```
1  public class Test01_创建file对象并创建文件
2  {
```

```
    public static void main(String[] args) throws IOException
    {
        File file1 = new File("src\\upload\\a.txt");

        File file2 = new File("src"+File.separator+"upload"+File.separator+"b.txt");

        File file3 = new File("C:\\863\\IO.txt");

        System.out.println(file1);
        System.out.println(file2);

        file1.createNewFile();
        file2.createNewFile();
        file3.createNewFile();
    }
}
```

2-2、常用文件方法

方法	描述
public File(String pathname)	根据传入的完整路径创建File类对象
public boolean exists()	判断文件是否存在
public boolean createNewFile()	创建新文件，路径必须存在
public long lastModified()	文件最后一次被修改的时间
public long length()	返回文件内容长度，单位字节
public String getName()	获得文件的名字
public String getPath()	返回文件的路径信息
public boolean isFile()	判断给定的路径是否是一个文件
public boolean delete()	删除文件
public static final String separator	与系统平台有关的默认路径分隔符

```
public class Test02_创建文件的常用方法
{
    public static void main(String[] args) throws IOException
    {
        String url_true = "src\\upload\\a.txt";
        String url_false = "src\\uploads\\c.txt";
        File file = new File(url_true);
        File file2 = new File(url_false);

        System.out.println(file.canRead());

        System.out.println(file2.canRead());
```

```
12
13          System.out.println(file.exists());
14          System.out.println(file2.exists());
15
16          System.out.println(file.isFile());
17          System.out.println(file2.isFile());
18  //          file.createNewFile();
19  //          file2.createNewFile();
20
21          System.out.println(file.getAbsoluteFile());
22          System.out.println(file.getAbsolutePath());
23
24          System.out.println(Arrays.toString(file.list()));
25      }
26  }
```

2-3、常用文件夹方法

方法	描述
public File(String pathname)	根据传入的完整路径创建File类对象
public boolean exists()	判断目录是否存在
public boolean mkdir()	根据路径创建单级目录
public boolean mkdirs()	根据路径创建目录且允许创建多级目录
public boolean isDirectory()	判断给定的路径是否为目录
public String getName()	获得目录的名字
public File[] listFiles()	列出指定目录的全部文件
public String[] list()	列出指定目录的全部文件的名称
public boolean delete()	删除目录(空)

```
1   public class Test04_file练习题
2   {
3       public static void main(String[] args) throws Exception
4       {
5           Scanner sc = new Scanner(System.in);
6           System.out.println("请输入查询目录：");
7           String url = sc.next();
8
9           File file = new File(url);
10          if(!file.isDirectory())
11          {
12              System.out.println("不是一个目录");
13              return;
14          }
15
```

```
        search(file);


    }

    public static String trim = "";
    public static void search(File file)
    {
        File[] fs = file.listFiles();


        for(File f : fs)
        {
            if(f.isFile())
            {
                System.out.println(trim+"\t"+"文件："+f.getName()+"\t\t"+f.length());
            }
        }
        for(File f : fs)
        {
            if(f.isDirectory())
            {
                System.out.println(trim+"\t"+"文件夹："+f.getName());
                trim += "\t";
                search(f);
            }
        }
        trim = "";

    }
}
```

2-4、File过滤器

使用`FileFilter`或者`FilenameFilter`可以实现文件过滤功能

```
public class Test05_过滤器
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入查询目录：");
        String url = sc.next();

        File file = new File(url);
        if(!file.isDirectory())
        {
            System.out.println("不是一个目录");
            return;
        }

        FileFilter fnf = new NameFilter();
```

```
16
17          File[] fs = file.listFiles(fnf);
18          for(File f : fs)
19          {
20              System.out.println(f.getName());
21          }
22
23
24      }
25  }
26  class NameFilter implements FileFilter
27  {
28      @Override
29      public boolean accept(File pathname)
30      {
31          return pathname.getName().endsWith(".java");
32      }
33  }
```

2-5、递归

递归做为一种算法在程序设计语言中广泛应用.是指函数/过程/子程序在运行过程中直接或间接调用自身而产生的重入现象。(自己调用自己，有结束条件)注意：递归时一定要明确结束条件。（递归次数过多也会内存溢出）

```
1   public class Test06_递归
2   {
3       public static void main(String[] args) throws Exception
4       {
5           Scanner sc = new Scanner(System.in);
6           System.out.println("请输入查询目录：");
7           String url = sc.next();
8
9           File file = new File(url);
10          if(!file.isDirectory())
11          {
12              System.out.println("不是一个目录");
13              return;
14          }
15
16
17          search(file);
18
19
20      }
21
22      public static String trim = "";
23      public static void search(File file)
24      {
25          File[] fs = file.listFiles();
26
27
28          for(File f : fs)
```

```
        {
            if(f.isFile())
            {
                System.out.println(trim+"\t"+"文件："+f.getName()+"\t\t"+f.length());
            }
        }
        for(File f : fs)
        {
            if(f.isDirectory())
            {
                System.out.println(trim+"\t"+"文件夹："+f.getName());
                trim += "\t";
                search(f);
            }
        }
        trim = "";

    }
}
```

3、字节流

IO流简介：（Input/Output）I/O类库中使用“流”这个抽象概念。Java对设备中数据的操作是通过流的方式。表示任何有能力产出数据的数据源对象，或者是有能力接受数据的接收端对象。“流”屏蔽了实际的I/O设备中处理数据的细节。IO流用来处理设备之间的数据传输。设备是指硬盘、内存、键盘录入、网络等。Java用于操作流的对象都在IO包中。IO流技术主要用来处理设备之间的数据传输。由于Java用于操作流的对象都在IO包中。所以使用IO流需要导包如：import java.io.*;IO流的分类 流按操作数据类型的不同分为两种：字节流与字符流。 流按流向分为：输入流，输出流（以程序为参照物，输入到程序，或是从程序输出）

什么是字节流 计算机中都是二进制数据,一个字节是8个2进制位.字节可以表示所有的数据,比如文本,音频,视频.图片,都是作为字节存在的.也就是说字节流处理的数据非常多。字节流的抽象基类： 输入流：`java.io.InputStream` 输出流：`java.io.OutputStream`

特点：

字节流的抽象基类派生出来的子类名称都是以其父类名作为子类名的后缀。

如：`FileInputStream`，`ByteArrayInputStream`等。

说明：

字节流处理的单元是一个字节，用于操作二进制文件（计算机中所有文件都是二进制文件）

3-1、`InputStream`

实现：显示指定文件内容。明确使用流，使用哪一类流？使用输入流，FileInputStream

第一步：1：打开流（即创建流）

第二步：2：通过流读取内容

第三步：3：用完后，关闭流资源
使用流就像使用水管一样，要打开就要关闭。所以打开流和关闭流的动作是比不可少的。

3-1-1、创建`InputStream`对象

```
public class Test01_创建输入流对象
{
    public static void main(String[] args) throws FileNotFoundException
    {
        String url = "src\\upload\\a.txt";

        InputStream in = new FileInputStream(url);

        File f = new File(url);
        InputStream in2 = new FileInputStream(f);

    }
}
```

3-1-2、读取一个字节

```
public class Test02_读取一个字节
{
    public static void main(String[] args) throws IOException
    {
        String url = "src\\upload\\a.txt";

        InputStream is = new FileInputStream(url);

        //一次读取一个字节
        int num = is.read();

        //类型转换
        System.out.print((char)num);

        //循环读取，但还是一次一个字节
        File f = new File(url);
        for(int i= 0 ; i < f.length();i++)
        {
            int n = is.read();
            System.out.print((char)n);
        }
```

```
22
23          is.close();
24
25      }
26  }
```

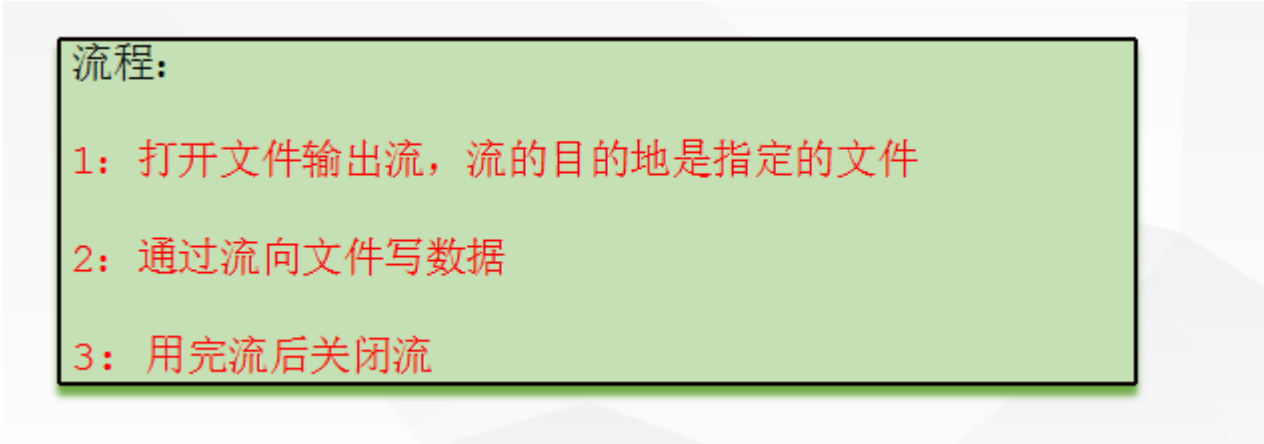
3-1-3、读取多个字节

```
1   public class Test03_读取多个字节
2   {
3       public static void main(String[] args) throws IOException
4       {
5           String url = "src\\upload\\a.txt";
6           File file = new File(url);
7           InputStream is = new FileInputStream(file);
8
9           //设置一次读取多个字节
10          byte[] bytes = new byte[1024];
11          int num;
12          String result = "";
13          while((num = is.read(bytes)) != -1)
14          {
15              String s = new String(bytes,0,num);
16              result += s;
17          }
18          System.out.println(result);
19      }
20  }
```

3-1-4、读取所有字节

```
1   public class Test04_读取全部字节
2   {
3       public static void main(String[] args) throws IOException
4       {
5           String url = "src\\upload\\a.txt";
6           File file = new File(url);
7           InputStream is = new FileInputStream(file);
8
9           //设置一次读取所有字节
10          byte[] bytes = new byte[(int)file.length()];
11          int num = is.read(bytes);
12          System.out.println(num);
13          String s = new String(bytes,0,num);
14          System.out.println(s);
15
16      }
17  }
```

3-2、`OutputStream`



3-2-1、创建`OutputStream`对象

```
public class Test01_创建对象
{
    public static void main(String[] args) throws FileNotFoundException
    {
        String url = "src\\upload\\a.txt";
        File file = new File(url);
        //覆盖掉之前的内容
        OutputStream os = new FileOutputStream(file);
        //在之前内容后追加
        OutputStream os2 = new FileOutputStream(file,true);
    }
}
```

3-2-2、从内存中读取内容向文件中写入内容

```
public class Test02_从内存中读取内容向文件中写入内容
{
    public static void main(String[] args) throws IOException
    {
        String url = "src\\upload\\a.txt";
        File file = new File(url);
        //覆盖掉之前的内容
        OutputStream os = new FileOutputStream(file);
        //在之前内容后追加
        OutputStream os2 = new FileOutputStream(file,true);


        String str = "abc";
        byte[] bs = str.getBytes();
        os.write(bs);
        os.close();

        os2.write("\tbbc".getBytes());
        os2.close();
    }

}
```

3-2-3、IO操作中文问题

```
public class Test03_IO操作中文问题
{
    public static void main(String[] args) throws Exception
    {
        String url = "src\\upload\\b.txt";
        File file = new File(url);
        //覆盖掉之前的内容
        OutputStream os = new FileOutputStream(file);
        //在之前内容后追加
        OutputStream os2 = new FileOutputStream(file,true);


        String str = "abc";
        byte[] bs = str.getBytes();
        os.write(bs);
        os.close();

        os2.write("\t中国".getBytes());
        os2.close();

        InputStream is = new FileInputStream(file);
        byte[] bytes = new byte[1024];
        int num;
        String result = "";
        while((num = is.read(bytes)) != -1)
        {
            result += new String(bytes,0,num);
        }
        System.out.println(result);


    }
}
```

3-2-4、文件的拷贝

```
public class Test04_文件的拷贝
{
    public static void main(String[] args) throws IOException
    {
        File file_old = new File("src\\upload\\b.txt");
        File file_new = new File("src\\upload\\w.txt");

        file_new.createNewFile();

        InputStream is = new FileInputStream(file_old);
        OutputStream os = new FileOutputStream(file_new);

```

```
        byte[] bytes = new byte[1024];
        int num;
        while((num = is.read(bytes)) != -1)
        {
            os.write(bytes,0,num);
        }
        os.flush();
        os.close();
        is.close();


    }
}
```

4、字节缓冲流

我们为了提高流的使用效率,自定义了字节数组,作为缓冲区.Java其实提供了专门的字节流缓冲来提高效率.

BufferedInputStream和BufferedOutputStream
BufferedOutputStream和BufferedOutputStream类可以通过减少读写次数来提高输入和输出的速度。它们内部有一个缓冲区，用来提高处理效率。查看API文档，发现可以指定缓冲区的大小。其实内部也是封装了字节数组。没有指定缓冲区大小，默认的字节是8192。

显然缓冲区输入流和缓冲区输出流要配合使用。首先缓冲区输入流会将读取到的数据读入缓冲区，当缓冲区满时，或者调用flush方法，缓冲输出流会将数据写出。

注意：当然使用缓冲流来进行提高效率时，对于小文件可能看不到性能的提升。但是文件稍微大一些的话，就可以看到实质的性能提升了。

```
public class Test01_字节缓冲流
{
    public static void main(String[] args) throws Exception
    {
        //字节缓冲流的使用，需要借助于字节流
        String old_url = "src\\upload\\a.txt";
        String new_url = "src\\upload\\y.txt";

        File old_file = new File(old_url);
        File new_file = new File(new_url);

        new_file.createNewFile();

        //输入缓冲流
        BufferedInputStream bis = new BufferedInputStream(new FileInputStream(old_file));
        BufferedOutputStream bos = new BufferedOutputStream(new FileOutputStream(new_file));


        byte[] bs = new byte[1024*2];
        int num;
        while ((num = bis.read(bs))!= -1)
        {
```

```
            bos.write(bs);
        }

        bis.close();
        bos.close();
    }
}
```

5、字符流

什么是字符流

计算机并不区分二进制文件与文本文件。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。所以字符流是建立在字节流之上的，它能够提供字符层次的编码和解码。例如，在写入一个字符时，Java虚拟机会将字符转为文件指定的编码（默认是系统默认编码），在读取字符时，再将文件指定的编码转化为字符。

字符流就是：字节流 + 编码表，更便于操作文字数据。
字符流的抽象基类：
Reader ， Writer。

由这些类派生出来的子类名称都是以其父类名作为子类名的后缀，如FileReader、FileWriter。

5-1 Reader对象

```
public class Test01_Reader对象
{
    public static void main(String[] args) throws IOException
    {
        String url = "src\\upload\\a.txt";

        Reader reader = new FileReader(url);

        File f = new File(url);
        Reader reader1 = new FileReader(f);

        char[] c = new char[1024];
        int num;
        String result = "";
        while((num = reader.read(c)) != -1)
        {
            result += new String(c,0,num);
        }
        System.out.println(result);
        reader.close();

    }
}
```

5-2 Writer对象

```java
public class Test02_write对象
{
    public static void main(String[] args) throws IOException
    {
        String url = "src\\upload\\a.txt";

        Writer writer = new FileWriter(url);

        File f = new File(url);
        Writer writer1 = new FileWriter(f,true);

        String m = "祖国万岁";

        writer.write(m);

        writer.close();

        writer1.write(",祝世界和平");

        writer1.close();


    }
}
```

5-3文件拷贝

```java
public class Test03_文件拷贝
{
    public static void main(String[] args) throws IOException
    {
        File file_old = new File("src\\upload\\a.txt");
        File file_new = new File("src\\upload\\z.txt");
        file_new.createNewFile();

        Reader reader = new FileReader(file_old);
        Writer writer = new FileWriter(file_new);
        char[] c = new char[1024];
        int num;
        while((num = reader.read(c))!= -1)
        {
            writer.write(new String(c,0,num));
        }
        writer.close();
        reader.close();

    }
}
```

6、字符缓冲流

查看Reader 发现Reader,操作的是字符,我们就不需要进行编码解码操作,由字符流读到二进制,自动进行解码得到字符,写入字符自动编码成二进制.
Reader有一个子类BufferedReader。子类继承父类显然子类可以重写父类的方法,也可以增加自己的新方法。例如一次读一行就是常用的操作.那么BufferedReader 类就提供了这个方法,可以查看readLine()方法具备 一次读取一个文本行的功能。很显然,该子类可以对功能进行增强。

```
public class Test02_字符缓冲流
{
    public static void main(String[] args) throws IOException
    {
        String old_url = "src\\upload\\a.txt";
        String new_url = "src\\upload\\m.txt";

        File old_file = new File(old_url);
        File new_file = new File(new_url);

        new_file.createNewFile();

        Reader reader = new FileReader(old_url);
        Writer writer = new FileWriter(new_file,true);
        //缓冲对象
        BufferedReader bufferedReader = new BufferedReader(reader);
        BufferedWriter bufferedWriter = new BufferedWriter(writer);

        String s;
       while( (s = bufferedReader.readLine()) != null )
       {
           System.out.println(s);
           bufferedWriter.write("-------------------------");
           bufferedWriter.newLine();
           bufferedWriter.write(s);
           bufferedWriter.newLine();
       }
       //必须要关闭，不关不给用
       bufferedReader.close();
       bufferedWriter.close();
 /*
        字符流拷贝
        char[] cs = new char[3];
        int num;
        while((num = reader.read(cs)) != -1)
        {
            writer.write(cs);
        }

        reader.close();
        writer.close();*/



    }
}
```

7、Properties文件处理

`Properties 类位于 java.util.Properties`，是Java 语言的配置文件所使用的类， `Xxx.properties` 为Java 语言常见的配置文件，如数据库的配置 `jdbc.properties`，系统参数配置 `system.properties`。 这里，讲解一下Properties 类的具体使用。 以key=value 的 键值对的形式进行存储值。 key值不能重复。

```
public class Test04_properties
{
    public static void main(String[] args) throws Exception
    {
        Properties prop = new Properties();

//        InputStream is = Test04_properties.class.getResourceAsStream("../upload/test.properties");

        File f = new File("src\\upload\\test.properties");

        InputStream is = new FileInputStream(f);
        //数据就从硬盘（文件）输入到了内存（程序）中的prop集合中了
        prop.load(is);

        is.close();

        String name = prop.getProperty("name");
        String age = prop.getProperty("age");
        String sex = prop.getProperty("sex");

        System.out.println(name+"\t"+age+"\t"+(sex.equals("w") ? "男" : "女"));

        //将文件中的数据存入到实体类对象中
        Student s = new Student();
        s.setName(name);
        s.setAge(  Integer.valueOf(age) );
        s.setSex(sex);

        //尝试转码
        System.out.println(new String(sex.getBytes(),"utf-8"));
        //可以使用对象
        System.out.println(s.getName()+"\t"+s.getAge()+"\t"+s.getSex());
        System.out.println("------------OutPutStream------------------");

        File newFile = new File("src\\upload\\test2.properties");
        newFile.createNewFile();

        Properties prop2 = new Properties();

        prop2.setProperty("nickname","sniper");
        prop2.setProperty("address","zhengzhoushi");
        prop2.setProperty("age",String.valueOf(18));


```

```
        OutputStream os = new FileOutputStream(newFile);
        prop2.store(os,"info");

        os.flush();
        os.close();
    }
}
```

8、CSV文件处理

逗号分隔值（Comma-Separated Values，**CSV**，有时也称为**字符分隔值**，因为分隔字符也可以不是逗号），其文件以纯文本形式存储表格数据（数字和文本）。纯文本意味着该文件是一个字符序列，不含必须像二进制数字那样被解读的数据。CSV文件由任意数目的记录组成，记录间以某种换行符分隔；每条记录由字段组成，字段间的分隔符是其它字符或字符串，最常见的是逗号或制表符。通常，所有记录都有完全相同的字段序列。通常都是纯文本文件。

```
public class Test05_csv文件
{
    public static void main(String[] args) throws Exception
    {
        File f = new File("src\\upload\\teacher.csv");
        Reader reader = new FileReader(f);
        BufferedReader bufferedReader = new BufferedReader(reader);
        String msg;
       List<Student> list = new ArrayList<Student>();
       while( ( msg = bufferedReader.readLine()  ) != null)
       {
           //清洗数据
           String[] ss = msg.split(",");
           Student s = new Student();

           s.setName(ss[0]);
           s.setAge(Integer.valueOf(ss[1]));
           s.setSex(ss[2]);

           list.add(s);
       }
       bufferedReader.close();

       for(Student s :  list)
       {
           System.out.println(s.getName()+"\t"+s.getAge()+"\t"+s.getSex());
       }
        System.out.println("----------------------------------");
       File newFile = new File("src\\upload\\student2.csv");
       newFile.createNewFile();
        Writer writer = new FileWriter(newFile,true);
        BufferedWriter bufferedWriter = new BufferedWriter(writer);

        for(Student s  : list)

        {
```

```
            bufferedWriter.write(s.getName()+","+s.getAge()+","+s.getSex());
            bufferedWriter.newLine();
        }


        bufferedWriter.close();


    }
}
```

9、序列化与反序列化

当创建对象时，程序运行时它就会存在，但是程序停止时，对象也就消失了。但是如果希望对象在程序不运行的情况下仍能存在并保存其信息，将会非常有用，对象将被重建并且拥有与程序上次运行时拥有的信息相同。可以使用对象的序列化。

对象的序列化：　将内存中的对象直接写入到文件设备中
对象的反序列化：　将文件设备中持久化的数据转换为内存对象

基本的序列化由两个方法产生：一个方法用于序列化对象并将它们写入一个流，另一个方法用于读取流并反序列化对象。ObjectOutputStream和ObjectInputStream 对象分别需要字节输出流和字节输入流对象来构建对象。也就是这两个流对象需要操作已有对象将对象进行本地持久化存储。

```
public class Test03_序列化与反序列化
{
    public static void main(String[] args) throws Exception
    {
        Student s = new Student();
        s.setName("王杨");
        s.setAge(18);

        Student s1= new Student();
        s1.setName("王杨");
        s1.setAge(18);

        File f = new File("src\\upload\\student.txt");
        f.createNewFile();
        //字节流
        FileOutputStream fos = new FileOutputStream(f);
        //序列化对象
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        //写入对象
        oos.writeObject(s);
        //关闭
        oos.close();

        //读取文件中的内容到内存中
        InputStream is = new FileInputStream(f);
        ObjectInputStream ois = new ObjectInputStream(is);


        Object obj =  ois.readObject();
```

```
        if(obj != null)
        {
            if(obj instanceof Student)
            {
                Student s3 = (Student)obj;
                System.out.println(s3.getName()+"\t"+s3.getAge());
            }
        }

        ois.close();
    }
}
//实现这个接口，系统会自动为我们的对象提供ID，便于区分
class Student implements Serializable
{

    String name;
    int age;
    String sex;

    public String getName()
    {
        return name;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public int getAge()
    {
        return age;
    }

    public void setAge(int age)
    {
        this.age = age;
    }

    public String getSex()
    {
        return sex.equals("m") || sex.equals("男") ? "男" : "女";
    }

    public void setSex(String sex)
    {
        this.sex = sex;
    }
}
```