

# 课程回顾

预习必要性

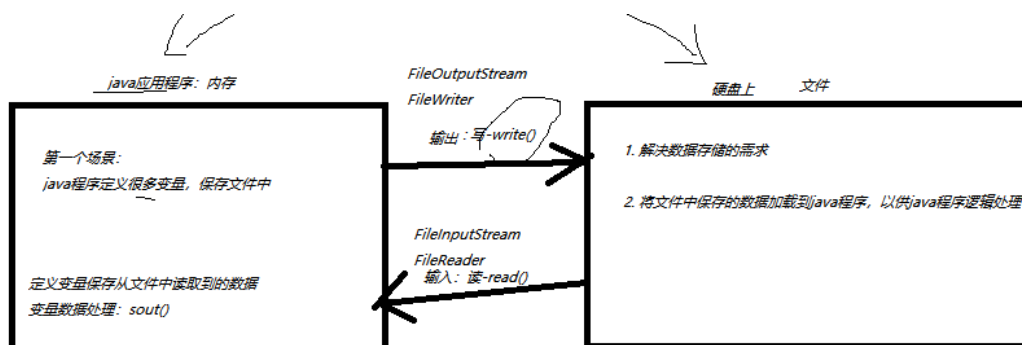
听课：记笔记（形式：纸质、文档）

教：保证听懂

学：实训课通过反复作业，达到代码熟练度

模仿，U2 U3 U4(业务分析：参考代码)

## 1 IO概念：输出流 输入流



- 1 IO作用: java应用程序和文件之间交互。
- 2 输出流:
- 3
- 4 输入流:
- 5

## 2 IO操作流程：套路

- 1 输出流
- 2 1.找对象 (new 字节流 字符流)
- 3 2.调用write(byte[]/char[]/String)
- 4 3.close()
- 5
- 6 输入流:
- 7 1.找对象 (new 字节流 字符流)
- 8 2.循环读取
- 9 while((len=read(字节数组 字符数组))!=-1){
- 10 //数据读取到java程序后, 程序想如何处理? ?
- 11 sout()
- 12
- 13 //输出其他文件
- 14 }
- 15 3.释放资源
- 16 close()

# 课程目标

1 缓冲流（提升读写效率） === 掌握

2 转换流（解决读写中文乱码问题） === 掌握

3 序列化流（解决对象读写问题） === 掌握

4 字节流和字符流区别

## 课程实施

### 1 字节流和字符流区别

#### 1-1 案例分析

##### 字节流读取文本文件

```
1 private static void readByStream() throws IOException {
2     //1.使用字节流读取文本文件
3     InputStream is=new FileInputStream("1.txt");
4     //2.读取文件
5     byte[] bs=new byte[2];
6     int len;//每次实际读取的字节数
7     while((len=is.read(bs))!=-1){
8         //从文件中读取的数据，想要如何处理？？
9         //数据统一保存bs中，sout(bs)
10        /**
11         * bs[] 保存全部-45 -98
12         * 解码: byte[]==>String
13         * 解码的代码实现: new String(bs,开始解析的位置, 解析字节个数)
14         */
15        String str=new String(bs,0,len);
16        //sout()输出的结果不是程序员，给客户看
17        System.out.print(str);
18    }
19    //3.标准流程 是否资源
20    is.close();
21 }
```

##### 字符读取文本文件

```
1 public static void readByReader(){
2     //1.使用字节流读取文本文件
3     Reader reader=new FileReader("1.txt");
4     //2.读取文件
5     char[] bs=new char[2];
6     int len;//每次实际读取的字节数
7     while((len=reader.read(bs))!=-1){
8         //从文件中读取的数据，想要如何处理？？
9         //数据统一保存bs中，sout(bs)
10        /**
11         * bs[] 保存全部-45 -98
```

```

12         * 解码: byte[]==>String
13         * 解码的代码实现: new String(bs,开始解析的位置, 解析字节个数)
14         */
15         String str=new String(bs,0,len);
16         //sout()输出的结果不是程序员, 给客户看
17         System.out.print(str);
18     }
19     //3.标准流程 是否资源
20     reader.close();
21 }

```

## 1-2 案例小结

```

1  字符流和字节流读取文本文件时的区别
2  举例: txt .java .html sql .css 使用记事本打开, 文件内容可以正常阅读
3  举例: .jpg .gif .doc .ppt .xls .png mp3 .vedio
4  字节流读取文本文件出现乱码的原因:
5  字节流处理读取的内容时, 一定有解码过程
6  byte[]获取时, 只拿到一个汉字的部分字节, 造成字节转换字符时问题(即乱码问题)
7  为了保证读到数据不出现乱码的风险, 优先建议按照字符一个一个读

```

## 2 缓冲流

### 2-1 缓冲流概述

优势: 提升文件读写速度

**缓冲流: 装饰设计模式**

分两类:

缓冲字节流:

BufferedInputStream: 缓冲字节输入流

BufferedOutputStream: 缓冲字节输出流

缓冲字符流:

BufferedReader: 缓冲字符输入流

BufferedWriter: 缓冲字符输出流

### 2-2 缓冲流使用

结论: 想把一个流改造成读写速度快的对象, 解决方案就是: 将普通IO流对象通过缓冲流的构造方法进行装饰即可。

```

1  新对象=new BufferedReader(提供被装饰的对象)
2  读写: 被装饰的对象方式一样的
3  释放资源: 需要释放几个对象??
4  一个: 新对象释放即可

```

## 字节缓冲流

- 输出流使用案例

```
1 package cn.kgc.demo;
2
3 import java.io.BufferedOutputStream;
4 import java.io.FileOutputStream;
5 import java.io.IOException;
6
7 /**
8  * @Author: lc
9  * @Date: 2022/4/12
10  * @Description: 字节缓冲流的案例
11  * @Version: 1.0
12  */
13 public class BufferedDemo1 {
14     /**
15      * 输出: 数据从哪儿来
16      * 输入: 数据从哪儿来 输入为什么放在输出下面: 输入流都是读取输出流执行的结果
17      */
18     public static void main(String[] args) {
19         //1.缓冲流实现数据输出
20         //将java程序中的数据保存到文件中
21         //匿名对象: 适合用于对象作为参数传递的场景
22         //OutputStream os=new FileOutputStream("f:\\bos.txt", true);
23         BufferedOutputStream bos= null;
24         try {
25             //1-1 装饰成一个缓冲流
26             bos = new BufferedOutputStream(
27                 new FileOutputStream("f:\\bos.txt", true)
28             );
29
30             //输出操作: os还是bos? ? bos才是装饰后的对象, 拥有速度快
31             bos.write("Hello".getBytes());
32             bos.write("中国人".getBytes());
33         } catch (IOException e) {
34             e.printStackTrace();
35         } finally {
36             try {
37                 //释放资源
38                 if (bos!=null) {
39                     bos.close();
40                 }
41                 //os.close();//没毛病, 没必要
42             } catch (IOException e) {
43                 e.printStackTrace();
44             }
45         }
46     }
47 }
48 }
```

- 输入流课堂案例

```
1 package cn.kgc.demo;
2
```

```

3  import java.io.BufferedInputStream;
4  import java.io.FileInputStream;
5
6  /**
7   * @Author: lc
8   * @Date: 2022/4/12
9   * @Description: 字节缓冲流的输入案例
10  * @Version: 1.0
11  */
12  public class BufferedDemo2 {
13      public static void main(String[] args) throws Exception {
14          //1. 找对象
15          BufferedInputStream bis=new BufferedInputStream(
16              new FileInputStream("f:\\bos.txt")
17          );
18
19          //2.如何读取文件
20          byte[] bs=new byte[1024];
21          int len;//实际读取的字节个数
22          while((len=bis.read(bs))!=-1){
23              //2-2 处理读取到的数据  sout()
24              String str = new String(bs, 0, len);
25              System.out.println(str);
26          }
27
28          //3.释放资源
29          bis.close();
30      }
31  }

```

## 字符缓冲流

- 输出流使用案例
- `newLine()`实现换行的输出效果

```

1  package cn.kgc.demo;
2
3  import java.io.BufferedReader;
4  import java.io.BufferedWriter;
5  import java.io.FileReader;
6  import java.io.FileWriter;
7
8  /**
9   * @Author: lc
10  * @Date: 2022/4/12
11  * @Description: 字符缓冲流的读和写
12  * 资源释放: 先开后关
13  * @Version: 1.0
14  */
15  public class BufferedDemo3 {
16      public static void main(String[] args) throws Exception{
17          //输入
18          BufferedReader br=new BufferedReader(
19              new FileReader("f:\\bos2.txt")
20          );
21          //2-1 定义数组
22          String content;//实际读取的内容

```

```

23         while((content=br.readLine())!=null){
24             //使用读取的数据??
25             System.out.println(content);
26         }
27         /*char[] cs=new char[1024];
28         int len;
29         while((len=br.read(cs))!=-1){//readLine(): String 读取一行
30             //使用读取数据
31             String str = new String(cs, 0, len);
32             System.out.println(str);
33         }*/
34         //关闭
35         br.close();
36     }
37 }
38

```

- 输入流课堂案例
- readLine():String 一次性读取一行数据

```

1  package cn.kgc.demo;
2
3  import java.io.BufferedReader;
4  import java.io.BufferedWriter;
5  import java.io.FileReader;
6  import java.io.FileWriter;
7
8  /**
9   * @Author: lc
10  * @Date: 2022/4/12
11  * @Description: 字符缓冲流的读和写
12  * 资源释放: 先开后关
13  * @Version: 1.0
14  */
15  public class BufferedDemo3 {
16      public static void main(String[] args) throws Exception{
17          //输入
18          BufferedReader br=new BufferedReader(
19              new FileReader("f:\\bos2.txt")
20          );
21          //2-1 定义数组
22          String content;//实际读取的内容
23          while((content=br.readLine())!=null){
24              //使用读取的数据??
25              System.out.println(content);
26          }
27          /*char[] cs=new char[1024];
28          int len;
29          while((len=br.read(cs))!=-1){//readLine(): String 读取一行
30              //使用读取数据
31              String str = new String(cs, 0, len);
32              System.out.println(str);
33          }*/
34          //关闭
35          br.close();
36      }
37  }

```

## 2-3 缓冲流实现文件复制

### 基础流复制代码

```

1 private static void copyBy() throws IOException {
2     FileInputStream fis=new
FileInputStream("g:\\FlashFXP_v5.4.0.3970.exe");
3     FileOutputStream fos=new FileOutputStream("g:\\copy.exe");
4
5     byte[] bs=new byte[1024];
6     int len;
7     long start = System.currentTimeMillis();
8     while((len=fis.read(bs))!=-1){
9         //使用
10        fos.write(bs,0,len);
11        fos.flush();
12    }
13    long end=new Date().getTime();
14    System.out.println("文件复制成功, 耗时: "+(end-start));
15    //释放 (先开后关)
16    fos.close();
17    fis.close();
18 }

```

### 缓冲流

```

1 public static void copyByBuffered {
2     /**
3      * 复制功能:
4      * 技术选型: 字节流
5      */
6     FileInputStream fis=new
FileInputStream("g:\\FlashFXP_v5.4.0.3970.exe");
7     FileOutputStream fos=new FileOutputStream("g:\\copy.exe");
8
9     //创建缓冲流
10    BufferedInputStream bis=new BufferedInputStream(fis);
11    BufferedOutputStream bos=new BufferedOutputStream(fos);
12
13    byte[] bs=new byte[1024];
14    int len;
15    long start = System.currentTimeMillis();
16    while((len=bis.read(bs))!=-1){
17        //使用
18        bos.write(bs,0,len);
19        bos.flush();
20    }
21    long end=new Date().getTime();
22    System.out.println("文件复制成功, 耗时: "+(end-start));
23    //释放 (先开后关)
24    fos.close();
25    fis.close();
26 }

```

## 3 转换流

### 3-1 使用场景

读写文件时，遇到中文出现乱码的情况

**乱码情况：中文出乱码**

- 1 ASCII：一个符号一个字节 7位，按照1个字节
- 2 gb2312:支持ASCII 一个汉字两个字节
- 3 gbk:支持ASCII 一个汉字两个字节
- 4 utf-8:支持ASCII 一个汉字三个字节
- 5 iso8859-1:支持ASCII,不支持中文码表 一个符号一个字节 8位 占用1个字节


### 3-2 乱码出现原因\*\*\*\*\*

- 1 java开发工具和记事本编码格式不一致造成
- 2
- 3 解决方案：
  1. 记事本格式：utf-8 远程电脑：只给读
  2. IDEA:GBK 开发工具：编码格式一旦修改，中文都会乱码
- 4 存在问题：可操作性不强
- 5
- 6
- 7
- 8 3.流操作，指定字符转换为字节、字节转换为字符使用码表名称

### 3-3 使用

- 1 隶于：字符流
- 2 输入流：InputStreamReader
- 3 输出流：OutputStreamWriter

### 课堂案例

 gbk.txt    编码格式是GBK格式    2022/4/12 16:57    文本文档    1 KB

### 转换输出流使用案例

```
1 package cn.kgc.demo;
2
3 import java.io.*;
4
5 /**
6  * @Author: lc
7  * @Date: 2022/4/12
8  * @Description: 转换输出流的案例
9  * @Version: 1.0
10 */
11 public class ConvertDemo2 {
12     public static void main(String[] args) throws Exception{
13         //基础字符流数据输出
14         //FileWriter fw=new FileWriter("f:\\gbk.txt",true);
15         FileOutputStream fw=new FileOutputStream("f:\\gbk.txt",true);
16         //基础流的编码格式明确的指定,编码格式设置与记事本编码格式一致
17         OutputStreamWriter isr=new OutputStreamWriter(fw,"gbk");
```



```

18         //解决中文乱码：记事本格式不让改、idea编码不让改
19         isr.write("真可爱");//计算机底层：转换为byte[] 默认形式：IDEA编码格式UTF-8
20         isr.close();
21     }
22 }

```

## 转换输入流使用案例

```

1 package cn.kgc.demo;
2
3 import java.io.FileInputStream;
4 import java.io.InputStreamReader;
5
6 /**
7  * @Author: lc
8  * @Date: 2022/4/12
9  * @Description: 转换输入流的案例
10  * @Version: 1.0
11  */
12 public class ConvertDemo1 {
13     public static void main(String[] args) throws Exception{
14         //1.读取文件
15         FileInputStream fis=new FileInputStream("f:\\gbk.txt");
16         //2 转换流 码表单词不区分大小写,设置码表的名称与读取文件码表一致
17         InputStreamReader isr=new InputStreamReader(fis,"GBK");
18         char[] bs=new char[1024];//存入字节, gbk的编码字节
19         int len;
20         while((len=isr.read(bs))!=-1){
21             //解码: byte[]==>按照特定码表==>符号
22             System.out.println(new String(bs,0,len));//安装idea的编码解码
23         }
24         fis.close();
25     }
26 }

```

# 4 序列化流

## 4-1 概念

- 1 序列化：将程序中的对象存入文件的过程。其实就是对象输出流
- 2 反序列化：将文件中保存的对象加载到程序中使用过程。其实是对对象输入流。

## 4-2 使用场景

实现java对象的存取

## 4-3 序列化对象

- 1 ObjectOutputStream: 序列化流
- 2 ObjectInputStream: 反序列化流

## 4-4 序列化案例

```
Exception in thread "main" java.io NotSerializableException Create breakpoint : cn.kgc.demo.Person
    at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1184)
    at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java:348)
    at cn.kgc.demo.ObjectDemo1.main(ObjectDemo1.java:24) Person必须支持序列化
```

- Person类：必须实现Serializable接口

```
1 package cn.kgc.demo;
2
3 import java.io.Serializable;
4
5 /**
6  * @Author: lc
7  * @Date: 2022/4/12
8  * @Description: cn.kgc.demo
9  * @Version: 1.0
10 */
11 //Serializable被子类实现，不用实现任何方法，就是表示该类支持序列化
12 public class Person implements Serializable {
13     private String id;
14     private String name;
15
16     public String getId() {
17         return id;
18     }
19
20     public void setId(String id) {
21         this.id = id;
22     }
23
24     public String getName() {
25         return name;
26     }
27
28     public void setName(String name) {
29         this.name = name;
30     }
31
32     public Person(String id, String name) {
33         this.id = id;
34         this.name = name;
35     }
36
37     @Override
38     public String toString() {
39         final StringBuilder sb = new StringBuilder("Person{");
40         sb.append("id=").append(id).append('\n');
41         sb.append(", name=").append(name).append('\n');
42         sb.append('}');
43         return sb.toString();
44     }
45 }
```

- 序列化对象

```

1 private static void saveObject() throws IOException {
2     //1.序列化: 输出
3     ObjectOutputStream oos=new ObjectOutputStream(
4         new FileOutputStream("obj.txt")
5     );
6
7     //1-2 输出对象
8     oos.writeBoolean(true);
9     oos.writeInt(12);//12就是整型数值
10    oos.writeObject(new Person("001","李四"));
11
12    //释放资源
13    oos.close();
14    //2.字符流
15 }

```

## 4-5 反序列化案例

```

1 public class ObjectDemo1 {
2     public static void main(String[] args) throws Exception{
3         //反序列化 输入流
4         ObjectInputStream ois=new ObjectInputStream(
5             new FileInputStream("obj.txt")
6         );
7         //while()
8         //读取对象时, 按照保存顺序加载
9         boolean bool = ois.readBoolean();
10        System.out.println(bool);
11        int num = ois.readInt();
12        System.out.println(num);
13        Object obj = ois.readObject();
14        System.out.println(obj);
15
16        //释放资源
17        ois.close();
18    }
19 }

```

## 4-6 序列化小结

1 | 一定是先做序列化, 文件中有了对象, 才能使用反序列化加载文件中的对象使用。

# 课程总结

## 1 缓冲流: 提速

## 2 转换流: 解决中文乱码

## 3 序列化流: 读写对象

# 预习

线程!!!

HTML