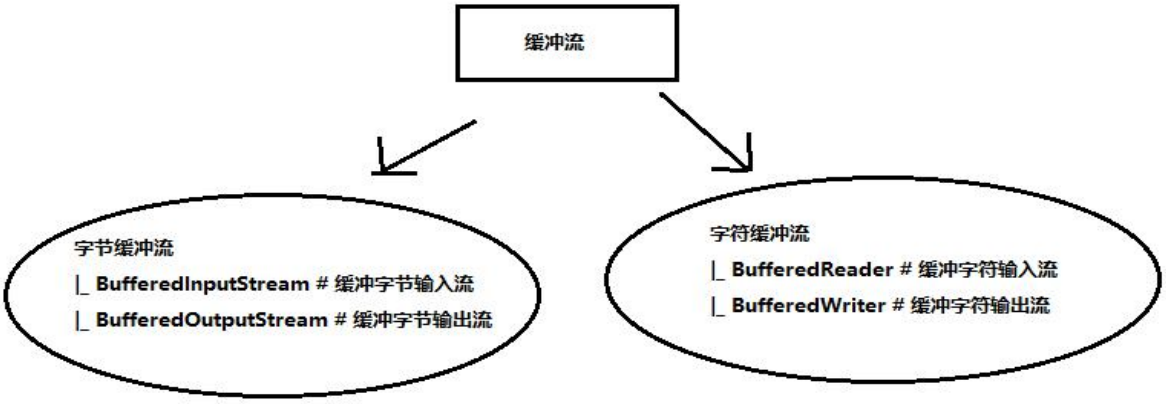
day10

缓冲流

缓冲流的原理

底层也是使用基本流来读写
但缓冲流内部定义了一个缓冲数组，在读的时候类似于我们一次读一个数组的方式，减少了磁盘操作次数，提高了程序效率



BufferedOutputStream 缓冲字节输出流

java.io.BufferedOutputStream extends OutputStream

```
//构造方法
    BufferedOutputStream(OutputStream out)  创建一个新的缓冲输出流，以将数据写入指定的底层输出流。
    BufferedOutputStream(OutputStream out, int size)  创建一个新的缓冲输出流，以将具有指定缓冲区大小的数据写入指定的底层输出流。
```

继承自父类的成员方法：

```
// 成员方法
    void close() : 释放资源
```

```
    void flush() ：刷新缓冲区(对于字节流来说没有作用)
// 写字节的成员方法
    abstract void write(int b)：一次写一个字节（参数是 int 是为了代码中写表示 byte 的整数方便不用强转)
    void write(byte[] b)：一次写一个字节数组
    void write(byte[] b,int off,int len)：一次写一个字节数组的一部分
```

BufferedInputStream 类：缓冲字节输入流

java.io.BufferedInputStream 类：缓冲字节输入流

```
// 构造方法
    BufferedInputStream(InputStream in)：使用基本流创建一个缓冲字节输入流
    BufferedInputStream(InputStream in, int size)：使用基本流创建一个缓冲字节输入流，设置缓冲区大小
```

BufferedWriter 类：

java.io.BufferedWriter 类 extends Writer

```
// 构造方法
    BufferedWriter(Writer out)：使用基本流创建一个缓冲字符输出流
    BufferedWriter(Writer out, int size)：使用基本流创建一个缓冲字符输出流，设置缓冲区大小
// 特有方法
    void newLine()：写入一个换行符，换行符自动根据当前系统确定
```

BufferedReader 类

java.io.BufferedReader 类：缓冲字符输入流

```
// 构造方法
    BufferedReader(Reader in)：使用基本流创建一个缓冲字符输入流
    BufferedReader(Reader in, int size)：使用基本流创建一个缓冲字
```

```
符输入流，设置缓冲区大小
// 特有方法
    String readLine(): 一次读一行字符串，"不包含换行符". 读到文件末尾返回 null
```

练习：文本排序

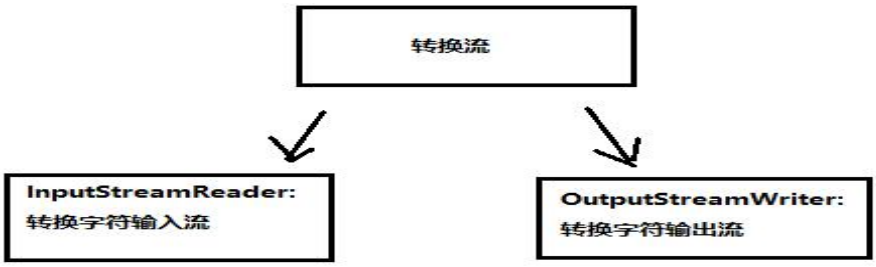
```
import java.io.*;
import java.util.Set;
import java.util.TreeMap;

//将以上文本按照序号排序，写入到当前模块下的 out.txt 中
public class Demo05 {
    public static void main(String[] args) throws IOException {
        //创建输入流对象
        BufferedReader br = new BufferedReader(new
FileReader("d:\\in.txt"));
        //创建 Map 集合对象
        TreeMap<String,String> map = new TreeMap<>();
        //按行读入
        String line;
        while ((line = br.readLine())!=null){
            String[] stringsArr = line.split("\\.");
            //添加键值对到 map 集合中，会按序号自动排好序
            map.put(stringsArr[0],stringsArr[1]);
        }
        //创建输出流对象，将数据输出到文件中
        BufferedWriter bw = new BufferedWriter(new
FileWriter("day10\\out.txt"));
        //遍历 map 集合
        Set<String> keySet = map.keySet();
        for (String key : keySet) {
            String value = map.get(key);
            //拼接，添加到输出流
            bw.write(key+value);
            bw.newLine();
        }
        //释放资源
        bw.close();
        br.close();
        }
    }
```

转换流

乱码问题: FileReader 读取 GBK 编码

知识点:
IDEA 是 UTF-8 编码，在 IDEA 中使用 FileReader 读取 GBK 编码的文件，能否读到正确的中文
乱码产生的原因是什么
乱码原因：读写编码不一致
GBK 文件中存储的是"你好"在 GBK 中对应的 byte
而 IDEA 中使用 FileReader 读取文件时，是将 byte 按照 UTF-8 编码表查找字符，结果找不到，就显示了问号



OutputStreamWriter 类

java.io.OutputStreamWriter 类: 输出转换流. 字符流通往字节流的桥梁

```
// 构造方法
    OutputStreamWriter(OutputStream out): 使用默认编码表创建转换流
    OutputStreamWriter(OutputStream out,String charsetName): 使用指定编码表创建转换流
```

使用步骤：

1、创建 OutputStreamWriter 对象，构造方法中传递字节输出流和指定的编码表

2、使用 OutputStreamWriter 对象中的方法 write

3、使用 OutputStreamWriter 对象的方法 flush

4、释放资源

InputStreamReader 类

java.io.InputStreamReader 类: 输入转换流. 字节流通往字符流的桥梁

```
// 构造方法
    InputStreamReader(InputStream in): 使用默认编码表创建转换流
    InputStreamReader(InputStream in,String charsetName): 使用指
定编码表创建转换流
```

序列化

ObjectOutputStream 类

java.io.ObjectOutputStream 类: 对象字节输出流

```
// 构造方法
    ObjectOutputStream(OutputStream out)
// 特有成员方法
    void writeObject(Object obj): 将对象写出
```

ObjectInputStream 类

java.io.ObjectInputStream 类: 对象字节输入流

```
// 构造方法
    ObjectInputStream(InputStream in)
// 特有成员方法
    Object readObject(): 读取对象
```

反序列化的前提:

1、类必须实现 Serializable 接口

2、类的文件必须存在

transient 关键字：瞬态关键字

静态的成员变量不能被序列化

被 transient 修饰的成员变量不能被序列化

InvalidClassException 异常

serialVersionUID 序列号的作用:

序列化操作时，会根据类生成一个默认的 serialVersionUID 属性，写入到文件中.

该版本号是根据类中成员计算出来的一个数值. 当类的成员发生改变时，序列版本号也会发生变化

当代码中的类和读取的对象序列版本号不一致时，就会抛出 InvalidClassException

为了避免这种问题，我们可以手动写死这个序列号，这样无论成员如何变化，序列版本号都是一样的

练习：序列化集合

当我们想在文件中保存多个对象时，可以把多个对象存储到一个集合中，对集合进行序列化和反序列化

```
import java.io.Serializable;

public class Person implements Serializable {
    private static final long serialVersionUID =
789661352862914726L;
    private String name;
    private int age;

    public Person() {
    }

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
```

```
    }

    @Override
    public String toString() {
        return "Person{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }

    public static long getSerialVersionUID() {
        return serialVersionUID;
    }

    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;
    }
}

/*需求:
        定义测试类:
        创建 ArrayList 集合, 存储 3 个 Person 对象:
        张三, 18
        李四, 19
        王五, 20
        使用序列化流将集合写入当前模块的 list.txt 文件中, 再反序列化
并遍历集合打印元素*/

import java.io.*;
import java.util.ArrayList;

public class Demo02 {
```

```
    public static void main(String[] args) throws IOException,
ClassNotFoundException {
        //创建 ArrayList 集合，存储 3 个 Person 对象
        ArrayList<Person> list = new ArrayList<>();
        list.add(new Person("张三",18));
        list.add(new Person("李四",19));
        list.add(new Person("王五",20));
        //创建序列化流对象
        ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("day10\\testpPerson.txt"));
        oos.writeObject(list);
        //创建反序列化流对象
        ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("day10\\testpPerson.txt"));
        Object o = ois.readObject();
        System.out.println(o);
        //释放资源
        ois.close();
        oos.close();
    }
}
```

打印流

PrintStream 类

java.io.PrintStream 类: 字节打印流

```
// 构造方法
    PrintStream(File file): 创建字节打印流，输出到一个文件
    PrintStream(OutputStream out): 创建字节打印流，输出到一个字节
输出流
    PrintStream(String fileName): 创建字节打印流，输出到一个文件路
径
```

特点：

1、只负责数据的输出，不负责读取

2、永远不会抛 IOException 异常

3、特有方法：print println

System 类:

java.lang.System 类:

```
// 静态方法
    static void setOut(PrintStream out): 设置 System.out 的输出目的地为参数的打印流
```

今日错题：

中文在 ASCII 码表中是不存在的，在 GBK 中占用两个字节，在 UTF-8 中占用三个字节

7、
已知idea工作环境默认编码方式是UTF-8。

观察下列代码，以下说法正确的是：( )

```
OutputStreamWriter osw = new OutputStreamWriter(new
FileOutputStream("E:\\out.txt"));
osw.write("你好");
osw.close();
```

- A： 会在E盘位置生成一个out.txt文件，文件中的内容是乱码
- B： 不会在E盘位置生成一个out.txt文件
- C： 会在E盘位置生成一个out.txt文件，文件中的内容是你好，并且文件大小是4个字节
- D： 会在E盘位置生成一个out.txt文件，文件中的内容是你好，并且文件大小是6个字节

我的答案： A

参考答案： D

在转换流的构造方法中，第二个参数不写则默认是 UTF-8 的编码格式，一个中文占用三个字节，所以“你好”一共占用 6 个字节

3、
已知在项目的模块itcast下存在copy.txt文件，文件中具有如下数据：

黑马程序员

观察下列代码，求最终结果：( )

```
BufferedReader br = new BufferedReader(new FileReader("itcast\\copy.txt"));
BufferedWriter bw = new BufferedWriter(new FileWriter("itcast\\copy.txt"));
String line=null;
while((line=br.readLine())!=null){bw.write(line);}
bw.close();
br.close();
```

- A： 黑马程序员
- B： 报异常
- C： 项目的模块itcast下的copy.txt文件没有任何内容
- D： 黑马程序员黑马程序员

当创建缓冲输出流时，如果构造方法中的文件已经存在则会全部覆盖，所以此时 copy 文件中已经是空的，读不出任何内容