第 5 章

CHAPTER 5

IO 字符输出流

IO 字符输出流通过数据流、文件系统提供字符输出功能。

5.1 Writer 抽象类

此抽象类定义了把字符数据转换为字节数据的规范方法。本节基于官方文档介绍常用方法，核心方法配合代码示例以方便读者理解。

1. append(char c)

将指定的字符添加到当前字符输出流中。接收 char 入参，作为指定的字符。

2. append(CharSequence csq)

将指定的字符序列添加到当前字符输出流中。接收 CharSequence 入参，作为指定的字符序列。

3. close()

关闭当前输出流并释放与此流相关联的任何资源。

4. flush()

清空当前输出流并强制写出任何缓冲的字符。

5. write(char[] cbuf)

将指定字符数组写出到当前输出流。接收 char[]入参，作为指定字符数组。

6. write(char[] cbuf, int off, int len)

将指定字符数组写出到当前输出流。接收 char[]入参，作为指定字符数组，接收 int 入参，作为数组的起始索引，接收 int 入参，作为写出的最大字符长度。

7. write(int c)

将指定字符数据写出到当前输出流。接收 int 入参，作为指定字符数据。

8. write(String str)

将指定字符串写出到当前输出流。接收 String 入参，作为指定字符串。

5.2 OutputStreamWriter 类

此类提供了字符流到字节流的编解码转换功能，可以使用指定的编解码字符集。

5.2.1 构造器

OutputStreamWriter 构造器见表 5-1。

表 5-1 OutputStreamWriter 构造器

构 造 器	描 述
OutputStreamWriter(OutputStream out)	构造新的对象，指定字节输出流
OutputStreamWriter(OutputStream out, String charsetName)	构造新的对象，指定字节输出流，指定字符集名称
OutputStreamWriter(OutputStream out, Charset cs)	构造新的对象，指定字节输出流，指定字符集
OutputStreamWriter(OutputStream out, CharsetEncoder enc)	构造新的对象，指定字节输出流，指定字符集编码器

5.2.2 常用方法

本节基于官方文档介绍常用方法，核心方法配合代码示例以方便读者理解。

1. close()

关闭当前输出流并释放与此流相关联的任何资源。

2. flush()

清空当前输出流并强制写出任何缓冲的字符。

3. getEncoding()

返回当前输出流所使用的字符集编码的名称。

4. append(char c)

将指定的字符添加到当前字符输出流中。接收 char 入参，作为指定的字符。

5. append(CharSequence csq)

将指定的字符序列添加到当前字符输出流中。接收 CharSequence 入参，作为指定的字符序列，代码如下：

```
//第 5 章/two/OutputStreamWriterTest.java
public class OutputStreamWriterTest {

    public static void main(String[] args) {
```

```
        try (OutputStreamWriter out = new OutputStreamWriter(
                new FileOutputStream("D:\\temp\\src\\test.txt"))) {
            out.write("hello world");
            out.append(System.lineSeparator()).append("end");
            out.flush();
        }catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

运行程序后查看文件内容，如图 5-1 所示。

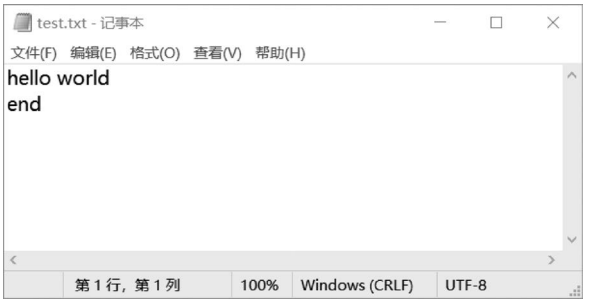

图 5-1　文件内容

6. write(char[] cbuf)

将指定字符数组写出到当前输出流。接收 char[]入参，作为指定字符数组。

7. write(char[] cbuf, int off, int len)

将指定字符数组写出到当前输出流。接收 char[]入参，作为指定字符数组，接收 int 入参，作为数组的起始索引，接收 int 入参，作为写出的最大字符长度。

8. write(int c)

将指定字符数据写出到当前输出流。接收 int 入参，作为指定字符数据，代码如下：

```
//第 5 章/two/OutputStreamWriterTest.java
public class OutputStreamWriterTest {

    public static void main(String[] args) {
        try (OutputStreamWriter out = new OutputStreamWriter(
                new FileOutputStream("D:\\temp\\src\\test.txt"))) {
            out.write(55401);
            out.write(57112);
//"\uD869\uDF18"
        }catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

运行程序后查看文件内容，如图 5-2 所示。

图 5-2 文件内容

注意：此文字属于是 Unicode 扩展区文字。

9. write(String str)

将指定字符串写出到当前输出流。接收 String 入参，作为指定字符串。

5.3 CharArrayWriter 类

CharArrayWriter 类包装了一个字符数组缓冲区。适用于需要重复使用数据的场景，多线程并发安全。

5.3.1 构造器

CharArrayWriter 构造器见表 5-2。

表 5-2 CharArrayWriter 构造器

构 造 器	描 述
CharArrayWriter()	构造新的对象，默认为无参构造器
CharArrayWriter(int initialSize)	构造新的对象，指定缓冲区的容量大小

5.3.2 常用方法

本节基于官方文档介绍常用方法，核心方法配合代码示例以方便读者理解。

1. close()

此方法空实现。

2. flush()

此方法空实现。

3. append(char c)

将指定的字符添加到当前字符输出流中。接收 char 入参,作为指定的字符。

4. append(CharSequence csq)

将指定的字符序列添加到当前字符输出流中。接收 CharSequence 入参,作为指定的字符序列。

5. write(char[] cbuf)

将指定字符数组写出到当前输出流。接收 char[]入参,作为指定字符数组。

6. write(char[] cbuf, int off, int len)

将指定字符数组写出到当前输出流。接收 char[]入参,作为指定字符数组,接收 int 入参,作为数组的起始索引,接收 int 入参,作为写出的最大字符长度。

7. write(int c)

将指定字符数据写出到当前输出流。接收 int 入参,作为指定字符数据。

8. write(String str)

将指定字符串写出到当前输出流。接收 String 入参,作为指定字符串。

9. writeTo(Writer out)

将当前缓冲区内的有效字符数据写到指定的输出流。接收 Writer 入参,作为指定的输出流,代码如下:

```
//第 5 章/three/CharArrayWriterTest.java
public class CharArrayWriterTest {

   public static void main(String[] args) throws IOException {
      CharArrayWriter charArrayWriter = new CharArrayWriter();
      charArrayWriter.append("hello").append("world");
      charArrayWriter.write(System.lineSeparator());
      charArrayWriter.write("end");
      System.out.println(charArrayWriter);

      try (OutputStreamWriter out = new OutputStreamWriter(
            new FileOutputStream("D:\\temp\\src\\test.txt"))) {
         charArrayWriter.writeTo(out);
         charArrayWriter.writeTo(out);
         charArrayWriter.writeTo(out);
      }catch (Exception e){
         e.printStackTrace();
      }
   }
}
```

执行结果如下:

```
helloworld
end
```

运行程序后查看文件内容，如图 5-3 所示。

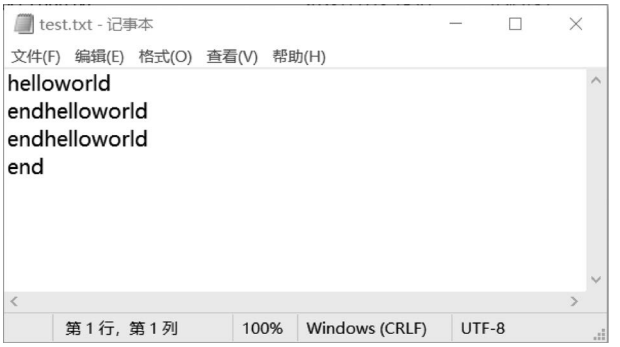

图 5-3 文件内容

10. size()

返回当前缓冲区有效字符的数据长度。

11. reset()

重置当前缓冲区的数据指针，效果类似清空缓冲区的数据。

小结

字符输出流用于输出文字型数据，并封装了字符数据转换为字节数据的功能。

习题

1. 判断题

(1) 单个 char 型数据可能无法表示一个完整的文字。(　　)

(2) 字符编码和解码必须使用相同的字符集。(　　)

(3) 使用 UTF-8 编码的数据可以用 GBK 解码。(　　)

2. 选择题

(1) Writer 抽象类的抽象方法有(　　)。(多选)

A. close()　　B. flush()

C. write(char[] cbuf, int off, int len)　　D. write(String str)

(2) 可以达到字符缓存效果的类是(　　)。(单选)

A. OutputStreamWriter　　B. CharArrayWriter

C. FileWriter　　D. Object

(3) 常用的字符集编码有(　　)。(多选)

A. UTF-8　　B. GBK

C. UTF-16　　D. Unicode

3. 填空题

查看执行结果并补充代码,代码如下:

```
//第 5 章/answer/OneAnswer.java
public class OneAnswer {
   public static void main(String[] args) throws IOException {
      CharArrayWriter charArrayWriter = new CharArrayWriter();
      charArrayWriter.append("hello").append("world");
      charArrayWriter.write(System._____________);
      charArrayWriter.write("end");
      System.out.println(charArrayWriter);
      charArrayWriter._____________;
      System.out.println(charArrayWriter);
   }
}
```

执行结果如下:

```
helloworld
end
```