[IO 2](#_Toc529173556)

[File类 2](#_Toc529173557)

[字节流 2](#_Toc529173558)

[字节输入流 2](#_Toc529173559)

[字节输出流 2](#_Toc529173560)

[字符流 2](#_Toc529173561)

[字符输入流 2](#_Toc529173562)

[字符输出流 2](#_Toc529173563)

[其它流 2](#_Toc529173564)

[转换流 2](#_Toc529173565)

[缓冲流 3](#_Toc529173566)

[打印流 3](#_Toc529173567)

[线程 3](#_Toc529173568)

[Lock 3](#_Toc529173569)

[BlockQueue 3](#_Toc529173570)

[课堂练习 3](#_Toc529173571)

[家庭作业 3](#_Toc529173572)

# IO

## File类

File位于java.io包下.在该类的类头有这样一段注释



这句可以理解为.在Java中一个File对象可以代表一个文件或者是代表一个文件夹.

既然是代表文件或者文件夹,那我们通过File对象.可以直接对文件或文件夹本体执行任何操作(排除读和写).这些操作指的是删除.新建.修改等.

### File的基本API

1. /\*\*
2. \*
3. \*/
4. **package** com.hwua.demo01;
6. **import** java.io.File;
8. /\*\*
9. \* @author Administrator
10. \*
11. \*/
12. **public** **class** Demo01 {
14. **public** **static** **void** main(String[] args) {
15. //-- 用File指向文件
16. File file1 = **new** File("E:\\TC35\\hotlist.txt");
17. File file2 = **new** File("E:\\TC35","hotlist.txt");
19. //-- 用File指向文件夹
20. File directory1 = **new** File("E:\\TC35\\A");
21. //-- B 在TC35目录下是没有的.
22. File directory2 = **new** File("E:\\TC35\\B");
24. //-- 验证文件或文件夹是否存在.在本地磁盘上确实有这个文件或文件夹存在.
25. System.out.println(file1.exists());
26. System.out.println(file2.exists());
27. System.out.println(directory1.exists());
28. //-- 如果没有返回值为false
29. System.out.println(directory2.exists());
31. //-- 获取文件或文件夹的名称
32. System.out.println(file1.getName());
34. //-- 获取文件或文件夹的路径
35. System.out.println(file1.getPath());
36. //-- 获取文件或文件夹的绝对路径
37. System.out.println(file1.getAbsolutePath());
38. //-- 获取文件或文件夹的上级目录
39. File parentFile = file1.getParentFile();
40. System.out.println(parentFile.getPath());
42. //-- 获取文件的大小 单位是字节
43. System.out.println(file1.length());
45. //-- 获取空间大小.获取的是文件所在的盘符的总空间大小和剩余空间大小
46. System.out.println(file1.getTotalSpace());
47. System.out.println(file1.getFreeSpace());
48. }
49. }

### 创建文件

创建文件是属于File的基本操作.通过API方法来创建指定指定后缀的文件.

利润上可以创建所有类型文件(包括已知和未知)

1. //-- 1.创建文件
3. //-- 1.1 构建指向要创建的文件的File对象.
4. File parentFile = **new** File("E:\\TC35\\A");
5. File childFile = **new** File(parentFile,"A.txt");
7. //-- 1.2 利用API方法来新建文件
8. **try** {
9. childFile.createNewFile();
10. } **catch** (IOException e) {
11. // TODO Auto-generated catch block
12. e.printStackTrace();
13. }

### 创建文件夹

1. /\*
2. \* mkdirs和mkdir都是用来建立文件夹的区别在于
3. \* mkdir  只能建立一层文件夹.如果层级关系中有不存在的会创建失败!
4. \* mkdirs 无论你有多少层.无论是存在还是不存在.都可以创建
5. \*/
6. //-- 2.4 当路径不存在时,把路径创建完成
7. file1.mkdirs();

### 把建立文件夹和文件整合在一起

1. //-- 2.创建文件.如果文件的上级目录中有不存在的文件夹怎么办
2. //-- 2.1 构建File对象
3. File file = **new** File("E:\\TC35\\A\\B\\C\\C.txt");
4. //-- 2.2 怀疑上级目录中不存在.所以要验证下
5. File file1 = file.getParentFile();
6. //-- 2.3 验证是否存在
7. **if**(!file1.exists()) {
8. /\*
9. \* mkdirs和mkdir都是用来建立文件夹的区别在于
10. \* mkdir  只能建立一层文件夹.如果层级关系中有不存在的会创建失败!
11. \* mkdirs 无论你有多少层.无论是存在还是不存在.都可以创建
12. \*/
13. //-- 2.4 当路径不存在时,把路径创建完成
14. file1.mkdirs();
15. }
17. //-- 2.5.构建文件
18. **try** {
19. file.createNewFile();
20. } **catch** (IOException e) {
21. // TODO Auto-generated catch block
22. e.printStackTrace();
23. }

### 重命名

在Java中利用API对文件进行重命名的本质就是新建一个文件.

1. /\*\*
2. \*
3. \*/
4. **package** com.hwua.demo01;
6. **import** java.io.File;
8. /\*\*
9. \* @author Administrator
10. \*
11. \*/
12. **public** **class** Demo03 {
13. **public** **static** **void** main(String[] args) {
14. System.out.println("\n");
15. System.out.println("\t");
16. //-- 在所有的语言中 \ 代表转移字符.
17. File file = **new** File("E:\\TC35\\A\\B\\C\\C.txt");
18. //-- 对上面的文件进行重命名.这个target不需要实际存在.只要路径准确就可以了
19. File target = **new** File("E:\\TC35\\A\\B\\C\\D.txt");
20. System.out.println(file.renameTo(target));
21. }
23. }

### 文件移动

利用重命名效果来实现文件的移动.因为renameTo方法,.如果修改了文件的路径.会发现原始路径下的文件没有了.只有新指定的路径下有文件存在!

### 文件删除

1. /\*\*
2. \*
3. \*/
4. **package** com.hwua.demo01;
6. **import** java.io.File;
8. /\*\*
9. \* @author Administrator
10. \*
11. \*/
12. **public** **class** Demo04 {
13. **public** **static** **void** main(String[] args) {
14. System.out.println("\n");
15. System.out.println("\t");
16. //-- 在所有的语言中 \ 代表转移字符.
17. File file = **new** File("E:\\TC35\\A\\B\\C\\D.txt");
18. System.out.println(file.delete());
19. //-- 验证文件删除成功还是失败!
20. System.out.println(file.exists());
22. }
24. }

## 字节流

对于任何一门语言流都是非常重要的一部分.起名为流是因为它很像水一样从A点流到B点.

所有的语言中都基本已内存作为参照物.流向内存是输入流.从内存流出的是输出流.

同时又根据所使用的单位是字节还是字符又分为字符流和字节流.

对字符和字节进行包装就是包装流.无论哪一种流它们的根本都是字节流.

### 字节输入流

使用字节流程内存的.就是字节输入流.比如从文件中读取内容就是输入流.

它的写法流程是固定的.可以遵循以下几个步骤:

1. 构建一个File对象指向.要被读取的文件
2. 利用文件来构建一个字节输入流对象
3. 利用流对象的API方法进行读取.
   1. 读取到哪里
   2. 读取几次
   3. 每次又读多少
4. 关闭流.
5. /\*\*
6. \*
7. \*/
8. **package** com.hwua.demo02;
10. **import** java.io.File;
11. **import** java.io.FileInputStream;
12. **import** java.io.FileNotFoundException;
13. **import** java.io.IOException;
14. **import** java.io.InputStream;
16. /\*\*
17. \* @author Administrator
18. \*
19. \*/
20. **public** **class** Demo04 {
22. **public** **static** **void** main(String[] args) {
23. // -- 1.构建File
24. File file = **new** File("E:\\TC35\\A\\B\\B.txt");
25. InputStream is = **null**;
26. **try** {
27. **try** {
28. // -- 2.利用File构建流对象 凡是带有Stream结尾的都是字节流
29. is = **new** FileInputStream(file);
30. // -- 3.利用API方法来读取
31. // -- 3.1 提供一个字节数组
32. **byte**[] buffer = **new** **byte**[1024];
33. // -- read方法有返回值.如果是-1代表读取到文件末尾.利用该值构建循环条件
34. // -- 不确定循环次数时,最好的选择就是while
35. // -- 3.2 为了解决数组默认值和读取内容长度不足数组长度问题
36. // -- 保存每次实际读取的长度
37. **int** length = 0;
38. // -- 3.3 提供一个可变长度的容器用来保存每次读取的内容
39. StringBuilder sb = **new** StringBuilder();
40. // -- 3.4 把内容读取到数组中并获取当前读取的字节数来作为循环条件
41. **while** (-1 != (length = is.read(buffer))) {
42. // -- 3.5 把读取的内容转变成字符串
43. String str = **new** String(buffer, 0, length, "GBK");
44. // -- 3.6 如果读取了多次.第二次是否把第一次的该覆盖了
45. // -- 把当前读取的给保存下来
46. sb.append(str);
47. }
48. // -- 3.7 把读取到的内容转换成字符串并打印.为了保证不会乱码.要指定编码格式
49. System.out.println(sb.toString());
50. // -- 3.8 关闭流
51. } **finally** {
52. is.close();
53. }
54. } **catch** (FileNotFoundException e) {
55. // TODO Auto-generated catch block
56. e.printStackTrace();
57. } **catch** (IOException e) {
58. // TODO Auto-generated catch block
59. e.printStackTrace();
60. }
61. }
62. }

### 字节输出流

使用输出流的流程和使用输入流的流程一样.都可以分为4个步骤.只是所调用的API方法需要变动下

1. /\*\*
2. \*
3. \*/
4. **package** com.hwua.demo02;
6. **import** java.io.FileNotFoundException;
7. **import** java.io.FileOutputStream;
8. **import** java.io.IOException;
9. **import** java.io.OutputStream;
11. /\*\*
12. \* @author Administrator
13. \*
14. \*/
15. **public** **class** Demo06 {
17. **public** **static** **void** main(String[] args) {
18. //-- 思考下: 是否要保障文件的父级目录是必须存在
19. //-- 1.声明流
20. OutputStream os = **null**;
21. **try** {
22. **try** {
23. //-- 2.实例化流对象 默认写入是覆盖的.想追加
24. os = **new** FileOutputStream("E:\\TC35\\A\\B\\C\\C.txt",**true**);
26. //-- 3.利用API方法写文件
27. os.write("\r\n".getBytes());
28. os.write("这是新插入的".getBytes());
29. os.write("\r\n".getBytes());
31. //-- 4.刷新流.必须的
32. os.flush();
33. }**finally** {
34. os.close();
35. }
36. }**catch** (FileNotFoundException e) {
37. // TODO: handle exception
38. }**catch** (IOException e) {
39. // TODO: handle exception
40. }
41. }
42. }

## 字符流

字符流是对字节流进行包装得到的.所采用的包装方式就是装饰设计模式.

它的写法和字节流基本一样.只是使用的类不一样.API方法类似.

既然有了字节流为什么还需要字符流呢?因为中文.因为中文占两个字节.所以就引用了字符流.因为字符char占两个字节可以完美的契合中文.

### 字符输入流

1. /\*\*
2. \*
3. \*/
4. **package** com.hwua.demo03;
6. **import** java.io.File;
7. **import** java.io.FileNotFoundException;
8. **import** java.io.FileReader;
9. **import** java.io.IOError;
10. **import** java.io.IOException;
11. **import** java.io.Reader;
13. /\*\*
14. \* @author Administrator
15. \*
16. \*/
17. **public** **class** Demo01 {
18. **public** **static** **void** main(String[] args) {
19. // -- 1.构建File
20. File file = **new** File("C:\\C\\C.txt");
21. **try** {
22. Reader reader = **null**;
23. **try** {
24. // -- 2.构建流
25. reader = **new** FileReader(file);
26. // -- 3.利用循环和API方法重复读取
27. **int** length = 0;
28. StringBuilder sb = **new** StringBuilder();
29. **char**[] buffer = **new** **char**[1024];
30. **while** (-1 != (length = reader.read(buffer))) {
31. sb.append(**new** String(buffer, 0, length));
32. }
33. System.out.println(sb.toString());
34. } **finally** {
35. // -- 4.关闭流
36. reader.close();
37. }
38. } **catch** (FileNotFoundException e) {
39. // TODO Auto-generated catch block
40. e.printStackTrace();
41. } **catch** (IOException e) {
42. // TODO Auto-generated catch block
43. e.printStackTrace();
44. }
45. }
47. }

### 字符输出流

同输入流一样.字符输出流和字节输出流的写法也差不多

1. /\*\*
2. \*
3. \*/
4. **package** com.hwua.demo03;
6. **import** java.io.FileWriter;
7. **import** java.io.IOException;
8. **import** java.io.Writer;
10. /\*\*
11. \* @author Administrator
12. \*
13. \*/
14. **public** **class** Demo02 {
16. **public** **static** **void** main(String[] args) {
18. **try** {
19. Writer writer = **new** FileWriter("C:\\C\\C.txt", **true**);
20. writer.write("字符流支持直接写字符串");
21. // -- 字符流不刷新缓冲区是不会把内容写入文件
22. writer.flush();
23. writer.close();
24. } **catch** (IOException e) {
25. // TODO Auto-generated catch block
26. e.printStackTrace();
27. }
28. }
29. }

## 其它流

### 转换流

把字节转字符的流

### 缓冲流

1. /\*\*
2. \*
3. \*/
4. **package** com.hwua.demo04;
6. **import** java.io.BufferedReader;
7. **import** java.io.File;
8. **import** java.io.FileInputStream;
9. **import** java.io.FileNotFoundException;
10. **import** java.io.IOException;
11. **import** java.io.InputStreamReader;
12. **import** java.io.UnsupportedEncodingException;
14. /\*\*
15. \* @author Administrator
16. \*
17. \*/
18. **public** **class** Demo01 {
20. **public** **static** **void** main(String[] args) {
21. // -- 找到首富祖奶奶
22. File file = **new** File("data/我成了首富祖奶奶.txt");
23. System.out.println(file.exists());
24. // -- 读取文本一定是字符流
25. // -- 只有读取图片媒体.等二进制内容时才用字节
27. **try** {
28. // -- 对字符进行包装 得到包装流/缓冲流
29. BufferedReader br = **new** BufferedReader(
30. // InputStreamReader是转换流把字节转成字符
31. **new** InputStreamReader(**new** FileInputStream(file), "GBK"));
33. String line = "";
34. **while** (**null** != (line = br.readLine())) {
35. // System.out.println(line);
36. **if** (line.contains("第")) {
37. System.out.println(line);
38. }
39. }
41. } **catch** (UnsupportedEncodingException | FileNotFoundException e) {
42. // TODO Auto-generated catch block
43. e.printStackTrace();
44. } **catch** (IOException e) {
45. // TODO Auto-generated catch block
46. e.printStackTrace();
47. }
49. }
50. }

### 打印流

# 线程

## Lock

## BlockQueue

# 课堂练习

1. 在C盘建立文件夹C.里面建立文件C.txt写入Hello
2. 在D盘建立文件夹D.里面建立文件D.txt写入World
3. 在E盘建立文件夹E.里面建立文件E.txt写入从C.txt和D.txt中读取的内容(所涉及的所有文件和文件夹都是代码生成的.文件中的内容都是代码读取或写入的)
4. /\*\*
5. \*
6. \*/
7. **package** com.hwua.demo02;
9. **import** java.io.File;
10. **import** java.io.FileInputStream;
11. **import** java.io.FileOutputStream;
12. **import** java.io.IOException;
13. **import** java.io.InputStream;
14. **import** java.io.OutputStream;
16. /\*\*
17. \* @author Administrator
18. \*
19. \*/
20. **public** **class** Demo07 {
21. /\*\*
22. \*
23. \*/
24. **private** **static** **final** **int** BUFFER\_MAX\_SIZE = 1024;
26. **public** **static** **void** main(String[] args) {
27. Demo07 d = **new** Demo07();
28. File fileC = **null**;
29. File fileD = **null**;
30. File fileE = **null**;
31. **try** {
32. fileC = d.createFileOrDirectory("C:\\C\\C.txt");
33. fileD = d.createFileOrDirectory("D:\\D\\D.txt");
34. fileE = d.createFileOrDirectory("E:\\E\\E.txt");
36. d.writeIntoFile(fileC, "Hello");
37. d.writeIntoFile(fileD, "World");
39. String line = d.readFromFile(fileC, fileD);
40. d.writeIntoFile(fileE, line);
42. } **catch** (IllegalArgumentException | IOException e) {
43. // TODO Auto-generated catch block
44. e.printStackTrace();
45. }
47. }
49. /\*\*
50. \*
51. \* @param path
52. \* @throws IOException
53. \*/
54. **public** File createFileOrDirectory(String path) **throws** IllegalArgumentException, IOException {
55. **if** (path == **null** || path.length() == 0) {
56. **throw** **new** IllegalArgumentException("路径非法");
57. }
59. // -- 利用参数构建File对象
60. File file = **new** File(path);
61. // -- 获取父级目录验证父级目录是否存在
62. File parentFile = file.getParentFile();
63. // -- 验证
64. **if** (!parentFile.exists()) {
65. parentFile.mkdirs();
66. }
67. file.createNewFile();
68. **return** file;
69. }
71. /\*\*
72. \*
73. \* @param file    要写入的文件
74. \* @param content 要写入的内容
75. \*/
76. **public** **void** writeIntoFile(File file, String content) {
77. **if** (file == **null** || content == **null** || content.length() == 0) {
78. **throw** **new** IllegalArgumentException();
79. }
81. // -- 1.构建流对象
82. OutputStream os = **null**;
84. **try** {
85. **try** {
86. // -- 2.实例化流对象
87. os = **new** FileOutputStream(file, **false**);
88. // -- 3.写文件
89. os.write(content.getBytes());
90. // -- 4.刷新
91. os.flush();
92. } **finally** {
93. os.close();
94. os = **null**;
95. }
96. } **catch** (Exception e) {
97. // TODO: handle exception
98. }
99. }
101. /\*\*
102. \* 提供要被读取的文件
103. \*
104. \* @param file
105. \* @return 文件中内容的拼接
106. \*/
107. **public** String readFromFile(File... files) {
108. String line = "";
109. **if** (files == **null** || files.length == 0) {
110. **throw** **new** IllegalArgumentException();
111. }
113. InputStream is = **null**;
114. **int** length = 0;
115. **byte**[] buffer = **new** **byte**[BUFFER\_MAX\_SIZE];
116. StringBuilder sb = **new** StringBuilder();
117. **try** {
118. **try** {
119. **for** (**int** i = 0; i < files.length; i++) {
120. // -- 利用文件构建输入流对象
121. is = **new** FileInputStream(files[i]);
122. // -- 利用流读取文件
123. **while** (-1 != (length = is.read(buffer))) {
124. sb.append(**new** String(buffer, 0, length, "GBK"));
125. }
126. // -- 把读取的内容拼接到line上
127. line = line + sb.toString();
128. // -- 清空StringBuilder中 避免第二次的内容包含第一次的内容
129. sb.delete(0, sb.length());
130. sb.setLength(0);
131. }
132. } **finally** {
133. is.close();
134. }
135. } **catch** (Exception e) {
136. }
137. **return** line;
138. }
140. }

# 家庭作业

1. 读取一本TxT小说.把小说的章节目录读取出来.保存到集合中.选择指定的章节直接显示对应章节的内容
2. 提供上一章和下一章功能(选)
3. 提供百分比进度拖动功能(选)