

1、字符流

1. FileReader

1.逐字符读取

2.字符数组读取(对read()的升级操作)

3.利用 Scanner 进行字符读取

2.FileWriter

3.FileReader 和 FileWriter 举例

2、字节流

1.InputStream

1.逐字节读取

2.字节数组读取

3.利用 Scanner 进行字符读取

2.OutputStream

1.字节数组输出

2. PrintWriter 类输出

3. 举例

1、字符流

1. FileReader

1. 逐字符读取

```
1 public static void main(String[] args) throws IOException {  
2     // 这个创建实例的过程,就是在打开文件  
3     // 要先打开文件.然后才能进行练习  
4     File file = new File("hello.txt");  
5     try(FileReader fr = new FileReader(file);) {
```

```

6 // 逐个字节的方式把文件内容读取出来
7 while(true){
8 // 每次调用 read 就可以读取一些数据出来
9 // read 提供了好几个版本,其中无参数版本就是一次读取一个字节
10 // 对于无参数版本的 read 来说,返回值就是这次读到的字节
11 // 这个结果的范围是 0 ~ 255
12 // 如果读到文件夹末尾(EOF,end of file),此时继续进行 read .就会返回
    -1;
13 int b = fr.read();
14 if(b == -1){
15     break;
16 }
17 System.out.printf("%c", b);
18 }
19 } catch (IOException e) {
20     e.printStackTrace();
21 }
22 }

```

说明点:

1. read()的理解: 返回读入的一个字符。如果达到文件末尾, 返回-1
2. 异常的处理: 为了保证流资源一定可以执行关闭操作。需要使用try-catch-finally处理
3. 读入的文件一定要存在, 否则就会报FileNotFoundException。
2. 字符数组读取(对read()的升级操作)

```

1 public static void main(String[] args) throws IOException {
2     File file = new File("hello.txt");
3     try (FileReader fr = new FileReader(file)) {
4         //read(char[] cbuf):返回每次读入cbuf数组中的字符的个数。如果达到文件
        末尾, 返回-1
5         byte[] buffer = new byte[1024];
6         while(true){
7             int len = inputStream.read(buffer);
8             if(len == -1){
9                 break;
10            }

```

```

11  for(int i = 0;i < len;i ++){
12  System.out.printf("%c",buffer[i]);
13  }
14  // String str = new String(buffer,0,len);
15  // System.out.print(str)
16  }
17  }catch(IOException e){
18  e.printStackTrace();
19  }
20 }

```

注:

```

1  //错误的写法
2  for(int i = 0;i < cbuf.length;i++){
3  System.out.print(cbuf[i]);
4  }
5  //正确的写法
6  for(int i = 0;i < len;i++){
7  System.out.print(cbuf[i]);
8  }
9  //方式二:
10 //错误的写法,对应着方式一的错误的写法
11 String str = new String(cbuf);
12 System.out.print(str);
13 正确的写法
14 String str = new String(cbuf,0,len);
15 System.out.print(str);

```

3. 利用 Scanner 进行字符读取

```

1  public static void main(String[] args) {
2  // 尝试从文件中读取中文,借助标准库中的处理字符的方式
3  // Scanner 不光能从控制台读取标准输入,也可以从文件中读取数据
4  File file = new File("print.txt");
5  try (InputStream inputStream = new FileInputStream(file)) {
6  try (Scanner scan = new Scanner(inputStream, "UTF-8")) {
7  while(scan.hasNext()){

```

```

8 String s = scan.nextLine();
9 System.out.println(s);
10 }
11 }
12 }catch(Exception e){
13 e.printStackTrace();
14 }
15 }

```

2. FileWriter

从内存中写出数据到硬盘的文件里。

```

1 public static void main(String[] args) {
2 // 1.提供 File 类的对象
3 File file = new File("test1.txt");
4 // 2.提供 FileWriter 的对象
5 try (FileWriter fw = new FileWriter(file)) {
6 // 3.写出操作
7 fw.write("I have a dream!");
8 fw.write("you need to have a dream!");
9 fw.flush();
10 }catch(Exception e){
11 e.printStackTrace();
12 }
13 }

```

说明:

1. 输出操作，对应的File可以不存在的。并不会报异常
2. File对应的硬盘中的文件如果不存在，在输出的过程中，会自动创建此文件。

File对应的硬盘中的文件如果存在：

如果流使用的构造器是：FileWriter(file,false) / FileWriter(file):对原有文件的覆盖

如果流使用的构造器是：FileWriter(file,true):不会对原有文件覆盖，而是在原有文件基础上追加内容

3. FileReader 和 FileWriter 举例

```

1 public static void main(String[] args) {
2     // 1.创建File类的对象，指明读入和写出的文件
3     File srcFile = new File("test1.txt");
4     File destFile = new File("test2.txt");
5     // 2.创建输入流和输出流的对象
6     try (FileReader fr = new FileReader(srcFile);
7         FileWriter fw = new FileWriter(destFile)) {
8         // 3.数据的读入和写出操作
9         int len = 0; // 记录每次读入到buff数组中的字符的个数
10        char[] buff = new char[1024];
11        while((len = fr.read(buff)) != -1){
12            // 每次写出len个字符
13            fw.write(buff,0,len);
14        }
15    }catch(Exception e) {
16        e.printStackTrace();
17    }
18 }

```

2、字节流

1. InputStream

1. 逐字节读取

```

1 public static void main(String[] args) {
2     File file = new File("test1.txt");
3     try (InputStream is = new FileInputStream(file)) {
4         int data = 0;
5         while((data = is.read()) != -1){
6             System.out.print((char)data);
7         }
8     }catch(Exception e){
9         e.printStackTrace();
10    }
11 }

```

2. 字节数组读取

```

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

```

```

2  File file = new File("test1.txt");
3  try (InputStream is = new FileInputStream(file)) {
4      int len = 0;
5      byte[] cubf = new byte[1024];
6      while((len = is.read(cubf)) != -1){
7          String str = new String(cubf,0,len);
8          System.out.println(str);
9      }
10     }catch(Exception e){
11         e.printStackTrace();
12     }
13 }

```

3.利用 Scanner 进行字符读取

```

1  public static void main(String[] args) {
2      File file = new File("test1.txt");
3      try (InputStream is = new FileInputStream(file)) {
4          try (Scanner scan = new Scanner(is, "UTF-8")) {
5              while(scan.hasNext()){
6                  String str = scan.nextLine();
7                  System.out.println(str);
8              }
9          }catch (Exception e){
10             e.printStackTrace();
11         }
12     }catch(Exception e){
13         e.printStackTrace();
14     }
15 }

```

2. OutputStream

1. 字节数组输出

使用字节流FileInputStream处理文本文件，可能出现乱码.所以需要
使用指定字符编码集.

```

1  public static void main(String[] args) {
2      File file = new File("test4.txt");

```

```

3  try (OutputStream os = new FileOutputStream(file)) {
4  String str = new String("你好 世界");
5  byte[] buffer = str.getBytes("UTF-8");
6  os.write(buffer);
7  os.flush();
8  }catch(Exception e){
9  e.printStackTrace();
10 }
11 }

```

2. PrintWriter 类输出

```

1  public static void main(String[] args) {
2  File file = new File("test5.txt");
3  try (OutputStream os = new FileOutputStream(file)) {
4  // 使用的字符编码集是 UTF-8
5  try (OutputStreamWriter osWriter = new OutputStreamWriter(os,
    "UTF-8")) {
6  try (PrintWriter writer = new PrintWriter(osWriter)) {
7  writer.print("计算机技术!");
8  }catch(Exception e){
9  e.printStackTrace();
10 }
11 }catch(Exception e){
12 e.printStackTrace();
13 }
14 }catch(Exception e){
15 e.printStackTrace();
16 }
17 }

```

3. 举例

1.实现对图片的复制操作

```

1  public static void main(String[] args) {
2  File srcFile = new File("1.jpg");
3  File destFile = new File("2.jpg");
4
5  try (InputStream is = new FileInputStream(srcFile);

```

```

6  OutputStream os = new FileOutputStream(destFile)) {
7  int len = 0;
8  byte[] cubf = new byte[1024];
9  while((len = is.read(cubf)) != -1){
10   os.write(cubf,0,len);
11  }
12  os.flush();
13  }catch(Exception e){
14   e.printStackTrace();
15  }
16 }

```

2.指定路径下文件的复制

```

1  public static void main(String[] args) {
2  long start = System.currentTimeMillis();
3  String srcPath = "G:\\尚硅谷java\\尚硅谷Java学科全套教程（总207.7GB）\\1.尚硅谷全套JAVA教程--基础必备（67.32GB）" +
4  "\\尚硅谷宋红康Java核心基础_好评如潮（30天入门）\\Java基础全套视频教程\\1.zip";
5  String destPath = "G:\\尚硅谷java\\尚硅谷Java学科全套教程（总207.7GB）\\1.尚硅谷全套JAVA教程--基础必备（67.32GB）" +
6  "\\尚硅谷宋红康Java核心基础_好评如潮（30天入门）\\Java基础全套视频教程\\2.zip";
7
8  copyFile(srcPath,destPath);
9
10 long end = System.currentTimeMillis();
11 System.out.println("复制花费的时间:" + (end - start));
12 }
13
14 // 复制指定路径下的文件
15 private static void copyFile(String srcPath, String destPath) {
16 File srcFile = new File(srcPath);
17 File destFile = new File(destPath);
18
19 try (InputStream is = new FileInputStream(srcFile);
20 OutputStream os = new FileOutputStream(destFile)) {

```



```
21  int len = 0;
22  byte[] cubf = new byte[4096];
23  while((len = is.read(cubf)) != -1){
24      os.write(cubf,0,len);
25  }
26  }catch (Exception e){
27      e.printStackTrace();
28  }
29 }
```

结论:

1. 对于文本文件(.txt,.java,.c,.cpp), 使用字符流处理
2. 对于非文本文件(.jpg,.mp3,.mp4,.avi,.doc,.ppt,...), 使用字节流处理