1、字符流

1. FileReader

1.逐字符读取

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

```
// 逐个字节的方式把文件内容读取出来
while(true){
// 每次调用 read 就可以读取一些数据出来
// 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
 // 对于无参数版本的 read 来说,返回值就是这次读到的字节
 // 这个结果的范围是 0 ~ 255
 // 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;
 int b = fr.read();
 if(b == -1){
 break;
 }
 System.out.printf("%c", b);
 }
 } catch (IOException e) {
 e.printStackTrace();
 }
}
```

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)

```
public static void main(String[] args) throws IOException {
 File file = new File("hello.txt");
 try (FileReader fr = new FileReader(file)) {
 //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
 byte[] buffer = new byte[1024];
 while(true){
 int len = inputStream.read(buffer);
 if(len == -1){
 break;
 }
```

```
 for(int i = 0;i < len;i ++){
 System.out.printf("%c",buffer[i]);
 }
 // String str = new String(buffer,0,len);
 // System.out.print(str)
 }
 }catch(IOException e){
 e.printStackTrace();
 }
}
```

注:

```
//错误的写法
 for(int i = 0;i < cbuf.length;i++){
 System.out.print(cbuf[i]);
 }
//正确的写法
 for(int i = 0;i < len;i++){
 System.out.print(cbuf[i]);
 }
//方式二:
 //错误的写法,对应着方式一的错误的写法
  String str = new String(cbuf);
  System.out.print(str);
 正确的写法
  String str = new String(cbuf,0,len);
  System.out.print(str);
```

3. 利用 Scanner 进行字符读取

```
public static void main(String[] args) {
 // 尝试从文件中读取中文,借助标准库中的处理字符的方式
 // 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
 File file = new File("print.txt");
 try (InputStream inputStream = new FileInputStream(file)) {
 try (Scanner scan = new Scanner(inputStream, "UTF-8")) {
 while(scan.hasNext()){
```

```
String s = scan.nextLine();
System.out.println(s);
 }
 }
 }catch(Exception e){
 e.printStackTrace();
 }
}
```

2. FileWriter

从内存中写出数据到硬盘的文件里。

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

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

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

2、字节流

1. InputStream

1. 逐字节读取

```java
public static void main(String[] args) {
 File file = new File("test1.txt");
 try (InputStream is = new FileInputStream(file)) {
 int data = 0;
 while((data = is.read()) != -1){
 System.out.print((char)data);
 }
 }catch(Exception e){
 e.printStackTrace();
  }
}
```

2. 字节数组读取

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

```
File file = new File("test1.txt");
try (InputStream is = new FileInputStream(file)) {
int len = 0;
byte[] cubf = new byte[1024];
while((len = is.read(cubf)) != -1){
String str = new String(cubf,0,len);
System.out.println(str);
}
 }catch(Exception e){
 e.printStackTrace();
 }
}
```

3.利用 Scanner 进行字符读取

```
public static void main(String[] args) {
 File file = new File("test1.txt");
 try (InputStream is = new FileInputStream(file)) {
 try (Scanner scan = new Scanner(is, "UTF-8")) {
 while(scan.hasNext()){
 String str = scan.nextLine();
 System.out.println(str);
 }
 }catch (Exception e){
  e.printStackTrace();
  }
  }catch(Exception e){
  e.printStackTrace();
  }
 }
```

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.

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

```
 try (OutputStream os = new FileOutputStream(file)) {
 String str = new String("你好 世界");
 byte[] buffer = str.getBytes("UTF-8");
 os.write(buffer);
 os.flush();
 }catch(Exception e){
 e.printStackTrace();
  }
}
```

2. PrintWriter 类输出

```
public static void main(String[] args) {
 File file = new File("test5.txt");
 try (OutputStream os = new FileOutputStream(file)) {
 // 使用的字符编码集是 UTF-8
 try (OutputStreamWriter osWriter = new OutputStreamWriter(os, "UTF-8")) {
 try (PrintWriter writer = new PrintWriter(osWriter)) {
 writer.print("计算机技术!");
 }catch(Exception e){
 e.printStackTrace();
  }
  }catch(Exception e){
  e.printStackTrace();
  }
  }catch(Exception e){
  e.printStackTrace();
  }
}
```

3. 举例

1.实现对图片的复制操作

```
public static void main(String[] args) {
 File srcFile = new File("1.jpg");
 File destFile = new File("2.jpg");

 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207.77
GB）\\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207.7
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