

11 IO流_字符流

JAVAAEE高级

一：字符流描述

- 1、字符流是可以直接读写字符的IO流
- 2、符流读取字符、就要先读取到字节数据、然后转为字符.
- 3、如果要写出字符、 需要把字符转为字节再写出.
- 4、读:字符流FileReader

```
1.      @Test
2.      public void getVoid() throws IOException {
3.          FileReader fr = new FileReader("D:\\IO\\修改后.txt");
4.          int c;
5.          while ((c = fr.read()) != -1) { // 通过项目默认的码表一次读取一个字符
6.              System.out.print((char) c);
7.          }
8.          fr.close();
9.      }
```

5、写:字符流FileWriter

```
1.      @Test
2.      public void getVoid() throws IOException {
3.          FileWriter fw = new FileWriter("D:\\IO\\修改后.txt",true);
4.          fw.write("大家好大家好大家好大家好大家好大家好");
5.          fw.write(97);
6.          fw.close();
7.      }
```

二：字符流的拷贝

1、拷贝文件

```
1.  @Test
2.      public void getVoid() throws IOException {
3.          FileReader fr = new FileReader("D:\\IO\\修改后.txt");
4.          FileWriter fw = new FileWriter("D:\\IO\\修改后2.txt");
5.          int c;
6.          while ((c = fr.read()) != -1) {
7.              fw.write(c);
8.          }
9.          fr.close();
10.         fw.close();
11.     }
```

2、拷贝纯文本<是不拷贝纯文本文件的>

```
1.  @Test
2.      public void getVoid() throws IOException {
3.          FileReader fr = new FileReader("D:\\IO\\微信图片.png");
4.          FileWriter fw = new FileWriter("D:\\IO\\微信图片4.png");
5.          int c;
6.          while ((c = fr.read()) != -1) {
7.              fw.write(c);
8.          }
9.          fr.close();
10.         fw.close();
11.     }
```

3、使用数组的方式进行拷贝

```
1.  @Test
2.      public void getVoid() throws IOException {
3.          FileReader fr = new FileReader("D:\\IO\\修改后.txt");
4.          FileWriter fw = new FileWriter("D:\\IO\\修改后111.txt");
5.          char[] arr = new char[1024];
6.          int len;
```

```

7.         while ((len = fr.read(arr)) != -1) { // 将文件上的数据读取到字符数组
中
8.             fw.write(arr, 0, len); // 将字符数组中的数据写到文件上
9.         }
10.        fr.close();
11.        fw.close();
12.    }

```

三：带缓冲的字符流

- 1、BufferedReader的read()方法读取字符时会一次读取若干字符到缓冲区, 然后逐个返回给程序, 降低读取文件的次数, 提高效率
- 2、BufferedWriter的write()方法写出字符时会先写到缓冲区, 缓冲区写满时才会写到文件, 降低写文件的次数, 提高效率

```

1.    @Test
2.        public void getVoid() throws IOException {
3.            BufferedReader br = new BufferedReader(new FileReader("D:\\IO\\
修改前.txt"));
4.            BufferedWriter bw = new BufferedWriter(new FileWriter("D:\\IO\\
修改后111.txt"));
5.            int c;
6.            while ((c = br.read()) != -1) {
7.                bw.write(c);
8.            }
9.            br.close();
10.           bw.close();
11.        }

```

①、readLine()方法

- 1、可以读取一行字符(不包含换行符号)

```

1.    @Test
2.        public void getVoid() throws IOException {
3.            BufferedReader br = new BufferedReader(new FileReader("D:\\IO\\
修改后.txt"));

```

```

4.         BufferedWriter bw = new BufferedWriter(new FileWriter("D:\\IO\\
修改后1212.txt"));
5.         String line;
6.         while((line = br.readLine()) != null) {
7.             bw.write(line);
8.             bw.newLine();
9.             System.out.println(line);
10.            //写出回车换行符
11.            bw.write("\r\n");
12.        }
13.        br.close();
14.        bw.close();
15.    }

```

四：LineNumberReader

- 1、LineNumberReader是BufferedReader的子类, 具有相同的功能, 并且可以统计行号
- 2、getLineNumber()方法可以获取当前行号
- 3、setLineNumber()方法可以设置当前行号

```

1.    @Test
2.        public void getVoid() throws IOException {
3.            // 创建输入输出流
4.            LineNumberReader lnr = new LineNumberReader(new FileReader("D:\\
\\IO\\修改后.txt"));
5.            String line;
6.            lnr.setLineNumber(10);
7.            while ((line = lnr.readLine()) != null) {
8.                System.out.println(lnr.getLineNumber() + ":" + line);
9.            }
10.           lnr.close();
11.        }

```

五：案例

1、<文件反转>

1、将一个文本文档上的文本反转,第一行和倒数第一行交换,第二行和倒数第二行交换

2、分析

创建输入输出流

创建集合

将读到的数据存储在集合

倒着遍历集合将数据写到文件上

关流

```
1.  @Test
2.      public void getVoid() throws IOException {
3.          // 创建输入输出流
4.          BufferedReader br = new BufferedReader(new FileReader("D:\\IO\\
修改后.txt"));
5.          // 创建集合
6.          ArrayList<String> list = new ArrayList<>();
7.          //将读到的数据存储在集合中
8.          String line;
9.          while ((line = br.readLine()) != null) {
10.              list.add(line);
11.          }
12.          br.close(); // 关流
13.
14.          // 倒着遍历集合将数据写到文件上
15.          BufferedWriter bw = new BufferedWriter(new FileWriter("D:\\IO\\
修改后2.txt"));
16.          for (int i = list.size() - 1; i >= 0; i--) {
17.              bw.write(list.get(i));
18.              bw.newLine();
19.          }
20.          // 关流
21.          bw.close();
22.      }
```

2、<获取文件出现的字符的次数>

1、获取一个文本上每个字符出现的次数,并将结果写在.txt文件上

```
1.  @Test
```

```
2.     public void getVoid() throws IOException {
3.         //1创建带缓冲区的输入流对象
4.         BufferedReader br = new BufferedReader(new FileReader("D:\\IO\\
修改后.txt"));
5.         //2创建双列集合对象,目的是把字符当作键,把字符出现的次数当作值
6.         HashMap<Character, Integer> hm = new HashMap<>();
7.         //3通过读取不断向集合中存储,存储的时候要判断,如果不包含这个键就将键和值为1
        存储,如果包含就将键和值加1存储
8.         int c;
9.         while((c = br.read()) != -1) {
10.             char ch = (char)c;
11.             hm.put(ch, !hm.containsKey(ch)? 1 : hm.get(ch) + 1);
12.         }
13.         //4关闭输入流
14.         br.close();
15.         //5创建输出流对象
16.         BufferedWriter bw = new BufferedWriter(new FileWriter("D:\\IO\\
修改后1212.txt"));
17.         //6将结果写出
18.
19.         for (Character key : hm.keySet()) {
20.             bw.write(key + "=" + hm.get(key));
21.         }
22.         bw.close();
23.     }
```