第二十一章 IO数据访问

1. I/O流概述

1. I/O（Input/Output）：读写
2. 流：从源节点到目标节点数据的流动
3. 源节点：称为输入流，用来读取数据
4. 目标节点：称为输出流，用来写入数据
5. 举例：百度网盘下载资料到本地
    a. 源节点：百度网盘
    b. 目标节点：本地电脑

2. 流的分类

1. 从传输的角度来说，流分为字节流和字符流
2. InputStream和OutputStream是字节流的父类
3. Reader和Writer字符流的父类

分类	字节流	字符流
输出流	InputStream	Reader
输入流	OutputStream	Writer
流中的数据	二进制	字符|字符串

3. 使用字符流读取文件

3.1 相对路径和绝对路径

- *FileNotFoundException*
- 相对路径和绝对路径

代码块

```java
package com.powernode.reader01;

import java.io.FileNotFoundException;
import java.io.FileReader;

public class Test01 {
    public static void main(String[] args) {
        /**
         * 1.FileNotFoundException:文件没找到
         *    1.文件不存在
         *    2.路径错误
         * 2.绝对路径和相对路径
         *    1.绝对路径：从盘符开始到文件，看到路径就知道具体位置
         *        D:\powernode\02-JavaSE\03-code\JavaProject\day21\src\fileTest.txt
         *    2.相对路径：从当前模块到文件，看到路径不知道具体位置
         *        .\day21\src\fileTest.txt
         *        也可以写成（省略.\）
         *        day21\src\fileTest.txt
         */
        try {
            FileReader fileReader = new FileReader("day21/src/fileTest.txt");
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }
}
```

3.2 字符流逐个读取文件中数据

代码块

```java
package com.powernode.reader01;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class Test02 {
    public static void main(String[] args) {
        try {
            //1.创建文件读取对象，并指定需要读取文件的路径
            FileReader fileReader = new FileReader("./day21/src/fileTest.txt");
            //2.读取文件，返回一个int类型的数据
            System.out.println(fileReader.read());//97 是 a 的ASCII
            System.out.println(fileReader.read());//98 是 b 的ASCII
```

```
            System.out.println(fileReader.read());//99 是 c 的ASCII
            System.out.println(fileReader.read());//-1 ,没有读取到数据
        } catch (FileNotFoundException e) {
            System.out.println("文件没有找到");
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

3.3 循环读取

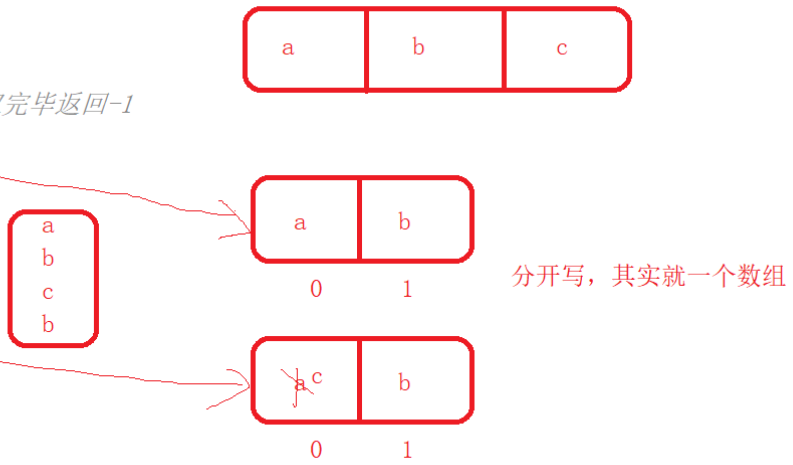
代码块

```
package com.powernode.reader01;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class Test03 {
    public static void main(String[] args) {
        FileReader fileReader = null;
        try {
            //1.创建一个文件读取对象，并指定需要读取的文件路径（建立了读取通道）
            fileReader = new FileReader("./day21/src1/fileTest.txt");
            //2.读取文件
            int read = fileReader.read();
            while(read != -1){
                System.out.println((char) read);
                read = fileReader.read();
            }
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }finally {
            //3.关闭流
            try {
                if (fileReader != null)
                    fileReader.close();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
    }
```

```
33  }
```

3.4 使用字符数组做缓存进行优化

```
//3.创建字符数组做缓存
char[] cbuf = new char[2];
//4.读取数据到缓存，返回读取的字符个数，读取完毕返回-1
int read = fileReader.read(cbuf);
while(read != -1){
    for (int i = 0; i < cbuf.length; i++) {
        char c = cbuf[i];
        System.out.println(c);
    }
    read = fileReader.read(cbuf);
}
```



代码块

```
1   package com.powernode.reader01;
2
3   import java.io.FileNotFoundException;
4   import java.io.FileReader;
5   import java.io.IOException;
6
7   public class Test04 {
8       public static void main(String[] args) {
9           //1.确定文件读取路径
10          String path = "./day21/src/fileTest.txt";
11          FileReader fileReader = null;
12          try {
13              //2.创建文件读取对象
14              fileReader = new FileReader(path);
15              //3.创建字符数组做缓存
16              char[] cbuf = new char[2];
17              //4.读取数据到缓存，返回读取的字符个数，读取完毕返回-1
18              int read = fileReader.read(cbuf);
19              while(read != -1){
20                  for (int i = 0; i < read; i++) {//读取几个输出几个
21                      char c = cbuf[i];
22                      System.out.println(c);
23                  }
24                  read = fileReader.read(cbuf);
25              }
```

```
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }finally {
            if (fileReader != null) {
                try {
                    fileReader.close();
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
        }
    }
```

4. 使用字符流写入文件

4.1 把字符串写入文件

代码块

```
package com.powernode.writer02;

import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class Test01 {
    public static void main(String[] args) {
        //1.确定文件写入路径
        String path = "./day21/src/fileTest.txt";
        FileWriter fileWriter = null;
        try {
            //2.创建文件写入对象，并建立写入通道
            fileWriter = new FileWriter(path);
            /**
             * 3.写入文件:
             *    1.清空文件
             *    2.写入缓存（通道）
             */
            fileWriter.write("hello world");
        } catch (IOException e) {
            throw new RuntimeException(e);
        }finally {
```

```
            //4.关闭流
            //4.1清空缓存-物理写入
            //4.2关闭写入通道
            if (fileWriter != null) {
                try {
                    fileWriter.close();
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
        }
    }
}
```

4.2 使用flush实现物理写入

- 什么时候使用flush？什么时候使用close？

代码块

```
package com.powernode.writer02;

import java.io.FileWriter;
import java.io.IOException;

public class Test02 {
    public static void main(String[] args) {
        //1.确定文件写入路径
        String path = "./day21/src/fileTest.txt";
        FileWriter fileWriter = null;
        try {
            //2.创建文件写入对象，并建立写入通道
            fileWriter = new FileWriter(path);
            /**
             * 3.写入文件：
             *    1.清空文件
             *    2.写入缓存（通道）
             */
            fileWriter.write("下班去哪里玩？");
            //刷新缓存，写入硬盘
            fileWriter.flush();
            fileWriter.write("去后海吧！");
            fileWriter.flush();
            /**
             * 若需要【立即写入】数据且继续使用流，必须调用flush()。
             * 若操作结束要关闭流，close()会自动处理 flush，无需额外调用。
```

```
            * 对于带缓冲的流，仅靠close()虽然能保证最终数据写入，但无法满足实时性要求（如网络通信、实时日志）。
             */
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

4.3 文件续写

- *true表示追加内容到文件末尾处（续写）*

代码块

```
package com.powernode.writer02;

import java.io.FileWriter;
import java.io.IOException;

public class Test03 {
    public static void main(String[] args) {
        //1.确定文件写入路径
        String path = "./day21/src/fileTest.txt";
        FileWriter fileWriter = null;
        try {
            //2.创建文件写入对象
            fileWriter = new FileWriter(path,true);//true表示追加内容到文件末尾处（续写）
            //3.写入文件
            fileWriter.write("\n下班去哪里玩？");
        } catch (IOException e) {
            throw new RuntimeException(e);
        }finally {
            if (fileWriter!=null) {
                try {
                    fileWriter.close();
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
        }
    }
}
```

5. 字符流实现文件备份

代码块

```
 package com.powernode.backup03;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class Test01 {
    public static void main(String[] args) {
        /**
         * - 文件备份，copy一个副本
         *    1.读取文件
         *    2.创建目标文件
         *    3.写入目标文件
         */
        //1.确定读写文件路径
        String sourcePath = "./day21/src/com/powernode/backup03/Test01.java";
        String targetPath = sourcePath + ".bak";
        FileReader fileReader = null;
        FileWriter fileWriter = null;
        try {
            //2.创建文件读写对象
            fileReader = new FileReader(sourcePath);
            fileWriter = new FileWriter(targetPath);
            //3.创建一个字符串数组缓存
            char[] cbuf = new char[500];
            int read = fileReader.read(cbuf);
            while(read!=-1){
                //5.写入文件
                fileWriter.write(cbuf,0,read);
                read = fileReader.read(cbuf);
            }
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }finally {
            //6.关闭流（先创建后关闭）
            if (fileWriter!= null){
                try {
                    fileWriter.close();
                } catch (IOException e) {
```

```
                    throw new RuntimeException(e);
                }
            }
            if (fileReader!= null) {
                try {
                    fileReader.close();
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }

        }
    }
}
```

6. 使用try()对代码进行优化

代码块

```
package com.powernode.backup03;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
class Student implements AutoCloseable{
    @Override
    public void close() throws Exception {
        System.out.println("Student.close");
    }
}
public class Test02 {
    public static void main(String[] args) {
        /**
         * - 文件备份，copy一个副本
         *    1.读取文件
         *    2.创建目标文件
         *    3.写入目标文件
         */
        //1.确定读写文件路径
        String sourcePath = "./day21/src/com/powernode/backup03/Test01.java";
        String targetPath = sourcePath + ".bak";
        try (
                //1.try()创建文件读写对象，try执行完毕最后会自动调用close方法关闭资源
                FileReader fileReader = new FileReader(sourcePath);
                FileWriter fileWriter = new FileWriter(targetPath);
```

```
                    /**
                     * try() 中创建的对象，该类必须实现AutoCloseable
                     * 重新AutoCloseable的close() 方法会自动执行
                     */
                    Student student = new Student();
            ){
                    //创建字符数组作为缓存
                char[] cbuf = new char[2];
                //读取文件
                int read = fileReader.read(cbuf);
                while(read != -1){
                    //写入文件
                    fileWriter.write(cbuf,0,read);
                    read = fileReader.read(cbuf);
                }
            } catch (FileNotFoundException e) {
                throw new RuntimeException(e);
            } catch (Exception e) {
                throw new RuntimeException(e);
            }


        }
    }
```

7. 使用字节流copy视频

代码块

```
package com.powernode.backup03;

import java.io.*;

public class Test03 {
    public static void main(String[] args) {
        //1.确定文件读写路径
        String sourcePath = "D:\\powernode\\02-JavaSE\\04-video\\day01\\01-认识Java语言.mp4";
        String targetPath = "D:\\01-认识Java语言.mp4";


        try(
                //2.创建文件读写对象
                FileInputStream fileInputStream = new FileInputStream(sourcePath);
```

```
                FileOutputStream fileOutputStream = new
    FileOutputStream(targetPath);
            ) {
                    //3.提供缓存
                byte[] bbuf = new byte[1024];
                //4.读取文件
                int read = fileInputStream.read(bbuf);
                while(read!=-1){
                    //5.写入文件
                    fileOutputStream.write(bbuf,0,read);
                    read = fileInputStream.read(bbuf);
                }
            } catch (FileNotFoundException e) {
                throw new RuntimeException(e);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
    }
```

8.