

| * 学习目标

* 能够掌握IO流的分类和继承结构

- * InputStream OutputStream
 - * FileInputStream FileOutputStream
 - * ObjectInputStream ObjectOutputStream
 - * SequenceInputStream PrintStream
 - * BufferedInputStream BufferdOutputStream
 - * ByteArrayInputStream ByteArrayOutputStream
- * Reader Writer
 - * BufferedReader BufferedWriter
 - * InputStreamReader OutputStreamWriter
 - * FileReader FileWriter
 - * StringReader StringWriter
 - PrintWriter

* 能够使用字节输出流 (OutputStream) 写出数据到文件

- * write(int),write(byte[]),write(byte[],0,len) ,close,flush
- * FileOutputStream
 - * \r\n

* 能够使用字节输入流(InputStream)读取数据到程序

- * read, (-1)
 - int len=-1;
 - byte[] b=new byte[1024];
 - while((len=is.read(b))!=-1){
 - // 处理len和b
 - }

* 能够使用字符输入流Reader读取数据到程序

- * FileReader

- * read

- * 能够使用字符输出流Writer的写出数据到文件

- * FileWriter

- * flush

- * write(String)

- * 能够处理IO的异常

- * jdk1.7之前try{}catch(){} finally{close}

- * jdk1.7 AutoClosable：自动关闭流：try(FileWriter fw=new FileWriter("")){ }catch{}

- * 能够掌握Properties的使用

- * 回顾

- * IO概述

- * 输入（读数据），输出流（写数据）

- * File

- * 构造器：File (pathName) ,File(parent,pathName),File(File,pathName)

- * 常用的方法

- * getName,length,getPath , getAbsolutePath, getCononicalPath

- * exists,isFile,isDirectory,createNewFile,delete,mkdir,mkdirs,listRoots , lastModifiy

- * list(),listFiles,FileFilter

- * 文件夹遍历

- * 能够掌握IO流的分类和继承结构

- * IO流概述

- * 应用程序与设备之间的数据传输

- * 键盘：输入数据

- * 显示器：显示数据

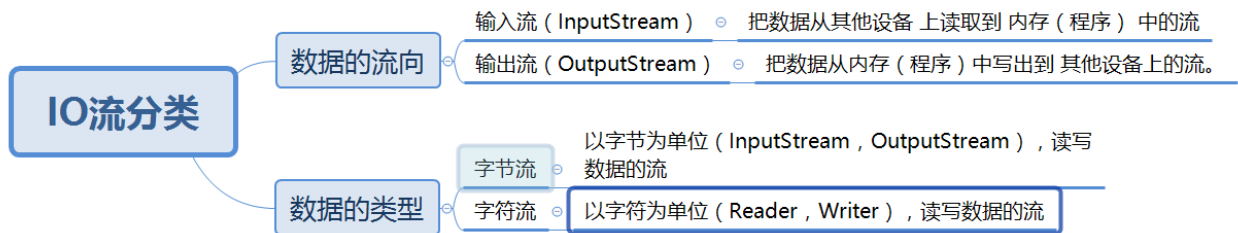
- * 在Java中，通过不同输入输出设备之间的数据传输抽象的表述为“流”，程序允许通过流的方式与输入输出设备进行数据传输。

* 设备：键盘，内存，显示器，网络等

* 输入流和输出流是相对于内存设备而言的，将外设中的数据读取到内存中即输入，将内存的数据写入到外设中即输出。

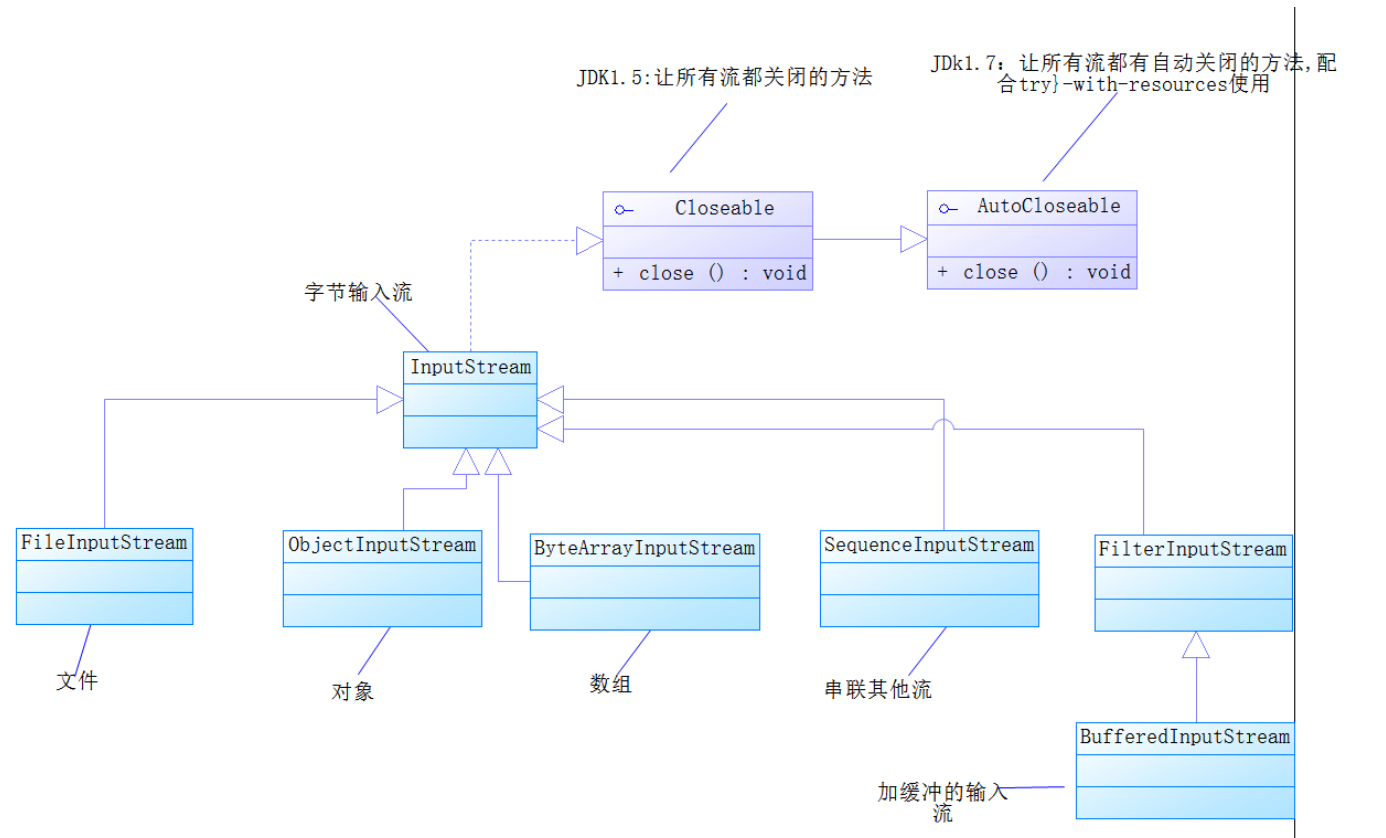
* Java中的“流”主要位于java.io包中，称之为IO（输入输出）流。输入也叫做读取数据，输出也叫做作写出数据。

* IO流的分类

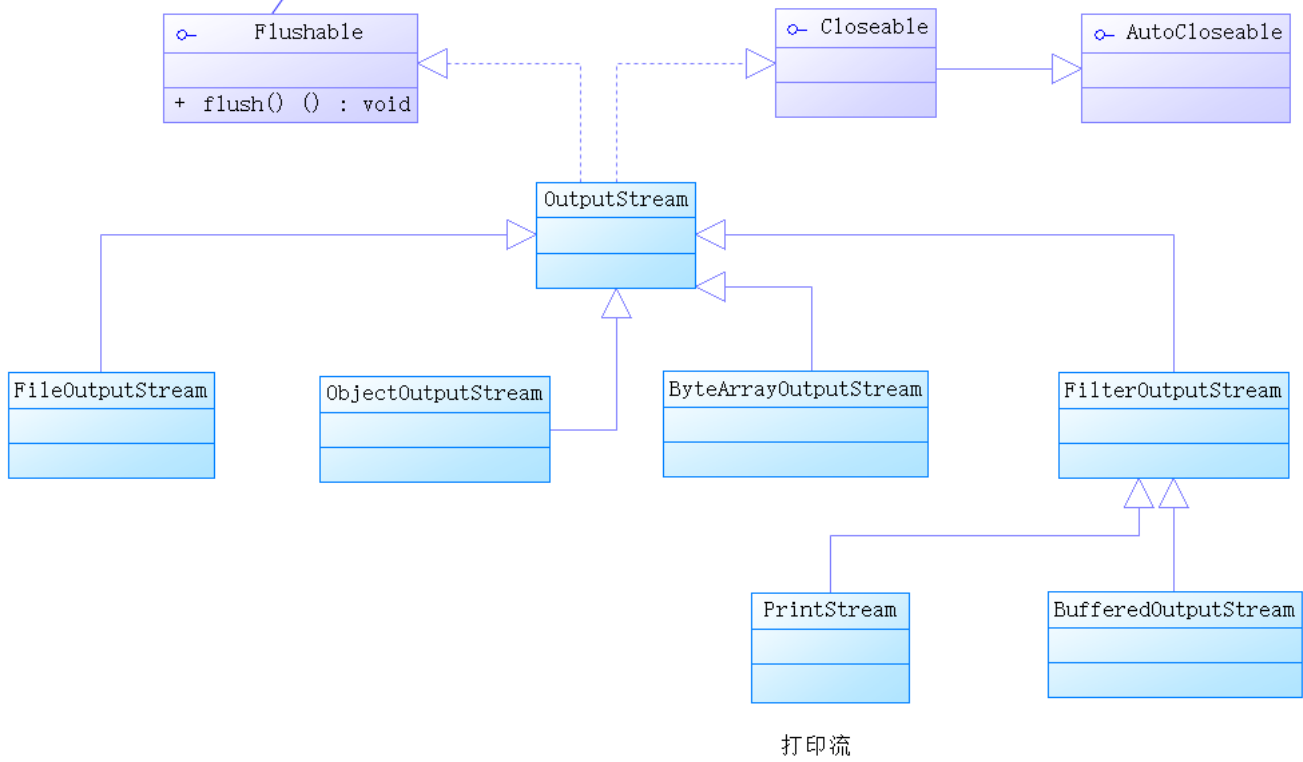


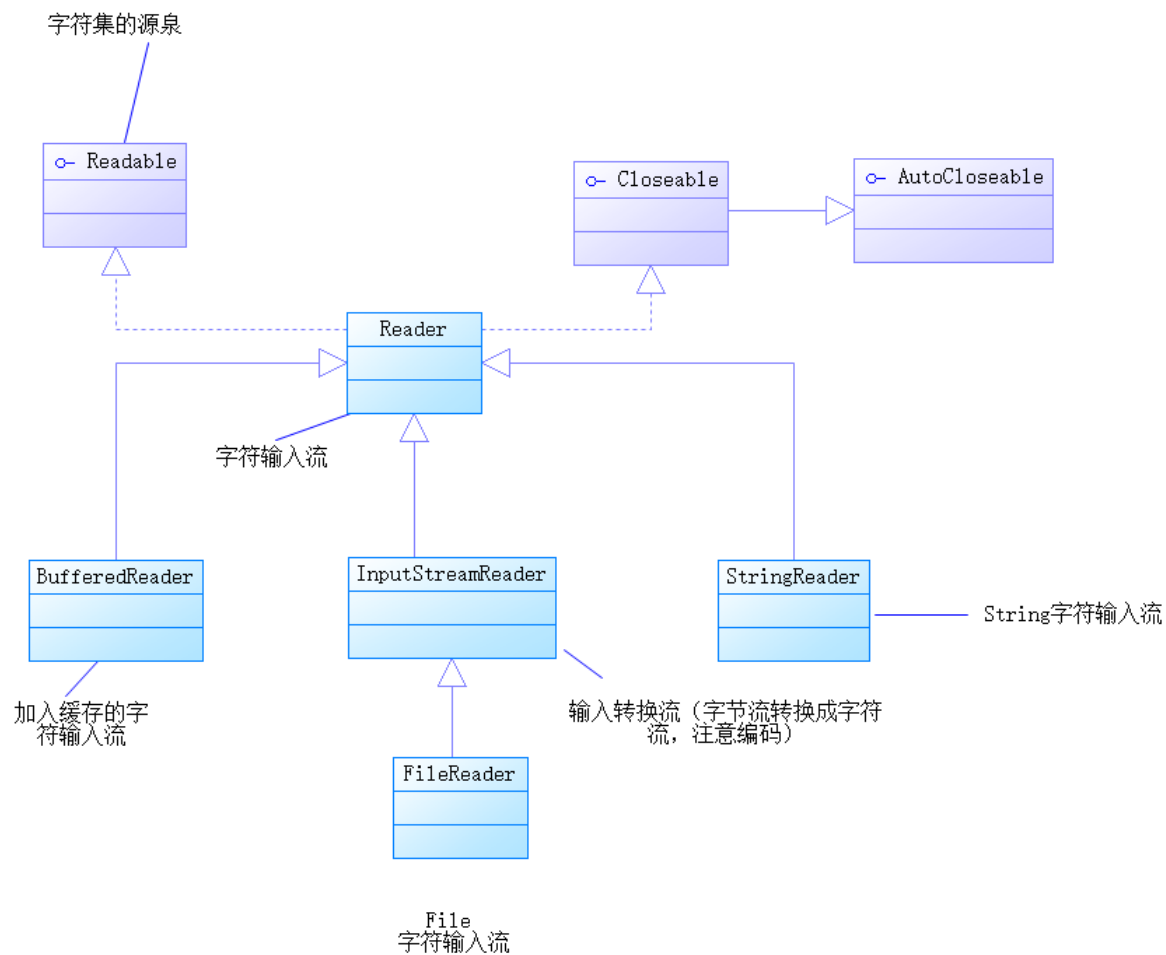
* 温馨提醒：字节流可以传输任意文件数据（图片，视频，文本...），底层传输的始终为二进制数据。

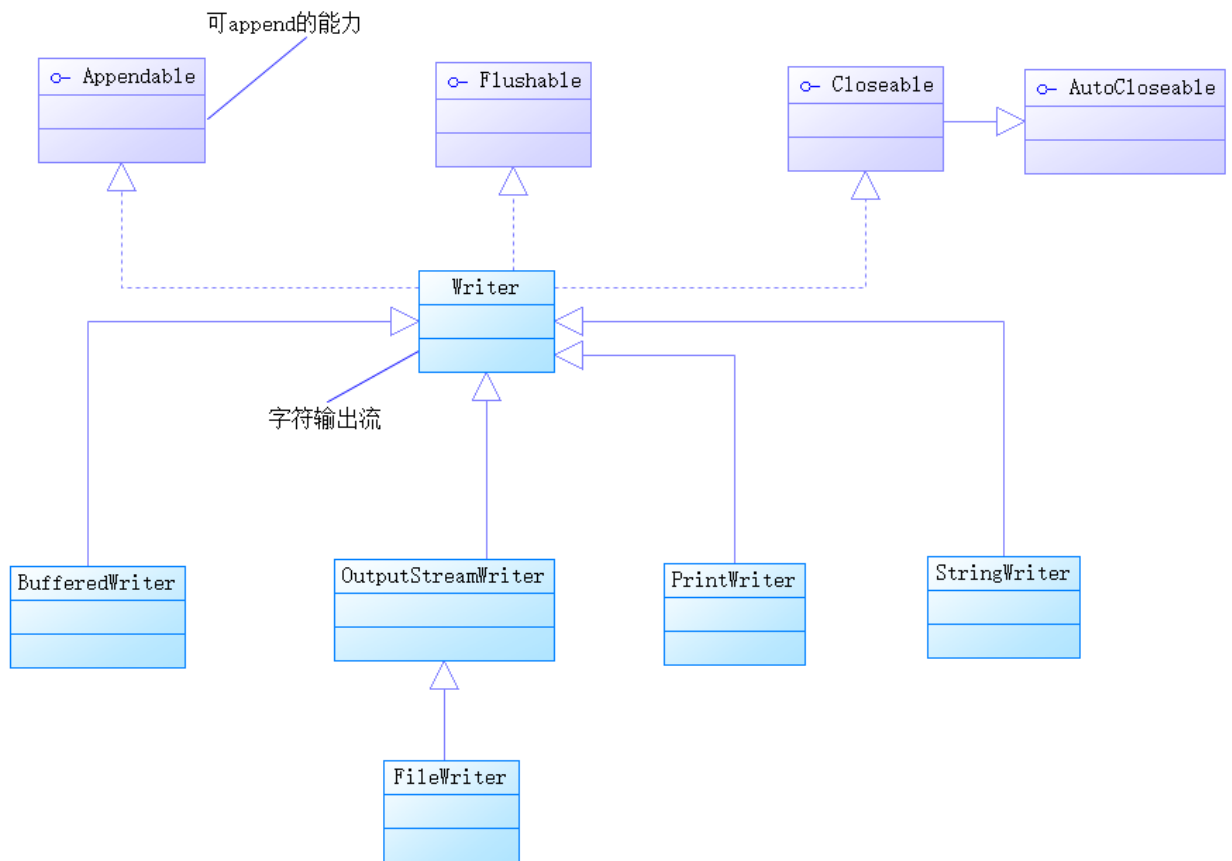
* IO流测试继承结构



数据通过write, 是写到缓冲区（内存），要刷一下，才可以写到硬盘（其他设备）







* InputStream OutputStream

* FileInputStream FileOutputStream

* ObjectInputStream ObjectOutputStream

* BufferedInputStream BufferedOutputStream

* ByteArrayInputStream ByteArrayOutputStream

* Closable--close AutoClosable--close

* SequenceInputStream, PrintStream

* Reader Writer

* BufferedReader BufferedWriter

* InputStreamReader OutputStreamWriter

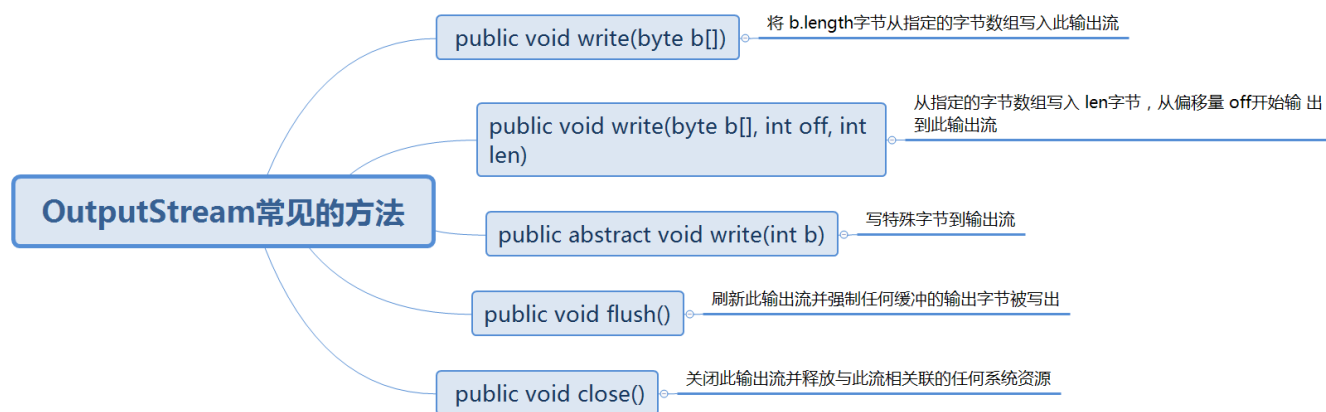
* FileReader FileWriter

* StringReader StringWriter

* PrintWriter

* 能够使用字节输出流 (OutputStream) 写出数据到文件

* OutputStream 常见的方法



```
1 * FileOutputStream: 用于将数据写到文件
2 * FileOutputStream的两种构造方式
3 public static void main(String[] args) throws IOException {
4     // 使用File对象创建流对象
5     File file=new File("test1.txt");
6     FileOutputStream fos=new FileOutputStream(file);
7     // 查看ASCII码表
8     fos.write(65);
9     fos.write(66);
10    fos.close();
11
12    // 使用文件名称创建流对象
13    FileOutputStream fos1=new FileOutputStream("test2.txt");
14    fos1.write(67);
15    fos1.write(68);
16    fos1.close();
17    System.out.println("程序结束。。。");
18 }
19 * 温馨提醒:
20 * 当你创建一个流对象时，传入一个文件路径。
21 * 该路径下，如果没有这个文件，会创建该文件。如果有这个文件，会清空这个文件的数据。
22
23 public static void main(String[] args) throws IOException {
24     // 使用文件名称创建流对象
25     FileOutputStream fos1=new FileOutputStream("test3.txt");
26     fos1.write("亮哥教育".getBytes());
```

```

27      // 测试 只有\n 两种加一起  \n\r 发现光标的位置不一样
28      fos1.write("\r\n".getBytes());
29      fos1.write("做教育，我们是认真的.".getBytes());
30      fos1.close();
31      System.out.println("程序结束。。。");
32  }
33  * 温馨提醒：
34  * 回车符 \r 和换行符 \n :
35      * 回车符：回到一行的开头（return）。
36      * 换行符：下一行（newline）。
37  * 系统中的换行：
38      * Windows系统里，每行结尾是 回车+换行 ，即 \r\n
39      * Linux系统里，每行结尾只有 换行 ，即 \n
40
41      public static void main(String[] args) throws IOException {
42          // 使用文件名称创建流对象
43          FileOutputStream fos1=new FileOutputStream("test4.txt");
44          // 写出从索引1开始，3个字节。索引1是b，3个字节，也就是bcd。
45          fos1.write("abcde".getBytes(),1,3);
46          fos1.close();
47          System.out.println("程序结束。。。");
48      }
49  * 结果只有
50      bcd
51      public static void main(String[] args) throws IOException {
52          // 使用文件名称创建流对象
53          FileOutputStream fos1=new FileOutputStream("test4.txt");
54          //一个汉字正常是两个字节
55          fos1.write("亮哥教育".getBytes(),0,4);
56          fos1.close();
57          System.out.println("程序结束。。。");
58      }
59
60  * 结果：
61      * 亮哥
62
63  * 追加的形式（默认是不追加的）
64      public static void main(String[] args) throws IOException {
65          // 使用文件名称创建流对象
66          FileOutputStream fos1=new FileOutputStream("test3.txt",true);

```

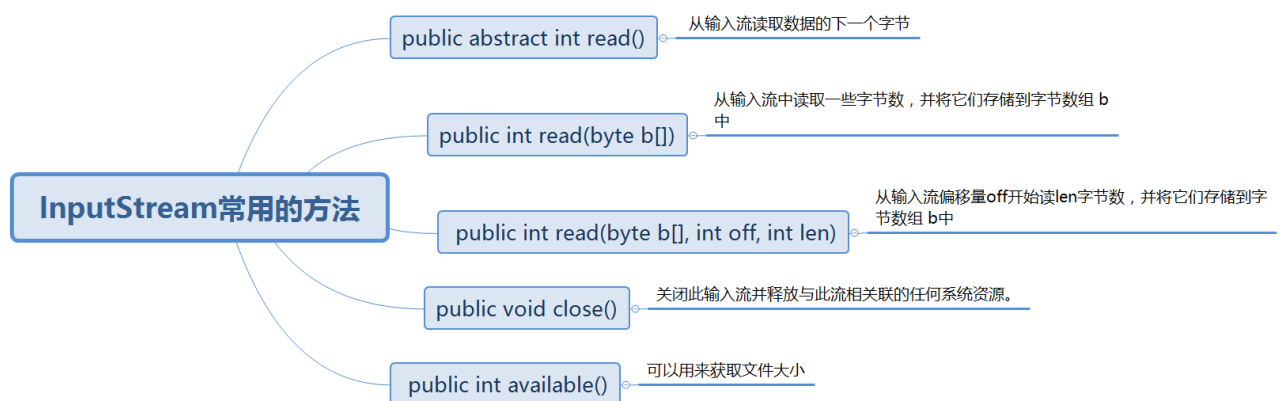


```

67     fos1.write("\r\n".getBytes());
68     fos1.write("亮哥教育".getBytes());
69     // 测试 只有\n 两种加一起  \n\r 发现光标的位置不一样
70     fos1.write("\r\n".getBytes());
71     fos1.write("做教育，我们是认真的.".getBytes());
72     fos1.close();
73     System.out.println("程序结束。。。");
74 }
75
76
77

```

* 能够使用字节输入流(InputStream)读取数据到程序



```

1  * FileInputStream
2  * 构建文件输入流的两种方式
3  public static void main(String[] args) throws IOException {
4      // 构建文件输入流的两种方式
5      File file=new File("test1.txt");
6      FileInputStream fis=new FileInputStream(file);
7
8      FileInputStream fis1=new FileInputStream("test1.txt");
9      fis.close();
10     fis1.close();
11 }

```

12 * 温馨提醒：假如文件不存在会报异常：java.io.FileNotFoundException

13

14 * 获得文件输入流可读的大小

```
15 public static void main(String[] args) throws IOException {  
16     FileInputStream fis=new FileInputStream("test1.txt");  
17     System.out.println("文件可读的大小: "+fis.available());  
18     fis.close();  
19 }
```

20 * 结果

21 文件可读的大小: 2

22

23 * read():读下一个字节，一个字节一个字节的读

```
24 public static void main(String[] args) throws IOException {  
25     FileInputStream fis=new FileInputStream("test1.txt");  
26     System.out.println("文件可读的大小: "+fis.available());  
27     char c1=(char)fis.read();  
28     System.out.println(c1);  
29     char c2=(char)fis.read();  
30     System.out.println(c2);  
31     // 读取到末尾,返回-1  
32     int c3=fis.read();  
33     System.out.println(c3);  
34     int c4=fis.read();  
35     System.out.println(c4);  
36     fis.close();  
37 }
```

38

39 * 结果:

40 文件可读的大小: 2

41 A

42 B

43 -1

44 -1

45

46 * 循环的读

```
47 public static void main(String[] args) throws IOException {  
48     FileInputStream fis=new FileInputStream("test1.txt");  
49     System.out.println("文件可读的大小: "+fis.available());  
50     int len=-1;  
51     while((len=fis.read())!=-1) {
```

```
52         System.out.println((char)len);
53     }
54     fis.close();
55 }
56 * 结果:
57 文件可读的大小: 2
58 A
59 B
60 public static void main(String[] args) throws IOException {
61     FileInputStream fis=new FileInputStream("test3.txt");
62     System.out.println("文件可读的大小: "+fis.available());
63     byte[] buffer=new byte[2];
64     while((fis.read(buffer))!=-1) {
65         String v=new String(buffer);
66         System.out.printf("%s",v);
67     }
68     fis.close();
69 }
70 * 结果:
71 文件可读的大小: 66
72 亮哥教育
73 做教育，我们是认真的。
74 亮哥教育
75 做教育，我们是认真的。
76 public static void main(String[] args) throws IOException {
77     FileInputStream fis=new FileInputStream("test3.txt");
78     System.out.println("文件可读的大小: "+fis.available());
79     byte[] b=new byte[2];
80     fis.read(b, 0, 2);
81     System.out.println(new String(b));
82     fis.read(b, 0, 2);
83     System.out.println(new String(b));
84     fis.read(b, 0, 2);
85     System.out.println(new String(b));
86     fis.read(b, 0, 2);
87     System.out.println(new String(b));
88     fis.close();
89 }
90 结果:
91 文件可读的大小: 66
```

亮
哥
教
育

96

```
public static void main(String[] args) throws IOException {  
    FileInputStream fis=new FileInputStream("test3.txt");  
    System.out.println("文件可读的大小: "+fis.available());  
    byte[] buffer=new byte[2];  
    while((fis.read(buffer,0,2))!=-1) {  
        String v=new String(buffer);  
        System.out.printf("%s",v);  
    }  
    fis.close();  
}
```

结果:

文件可读的大小: 66

亮哥教育

做教育,我们是认真的。

亮哥教育

做教育,我们是认真的。

113

* 图片的复制 (使用缓冲数组)

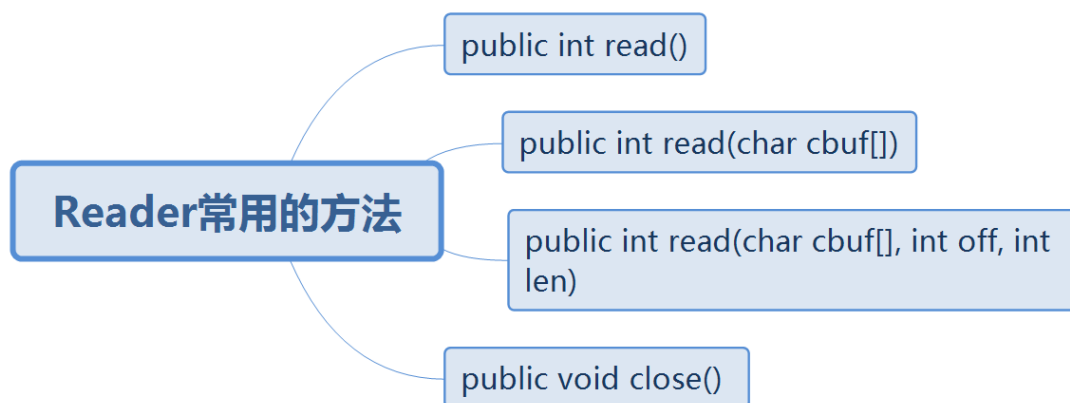
```
public static void main(String[] args) throws IOException {  
    // 1.创建流对象  
    // 1.1 指定数据源  
    FileInputStream fis = new FileInputStream("pic/abc.jpg");  
    // 1.2 指定目的地  
    FileOutputStream fos = new FileOutputStream("pic1/test.jpg");  
    // 2.读写数据  
    // 2.1 定义数组  
    byte[] buffer = new byte[1024];  
    // 2.2 定义长度  
    int len;  
    // 2.3 循环读取  
    while((len=fis.read(buffer))!=-1) {  
        fos.write(buffer, 0, len);  
    }  
    // 3.关闭资源  
    fos.close();
```

```

132         fis.close();
133     }
134
135
136 * 温馨提醒:
137 * 流的关闭原则: 先开后关, 后开先关。

```

* 能够使用字符输入流Reader读取数据到程序



```

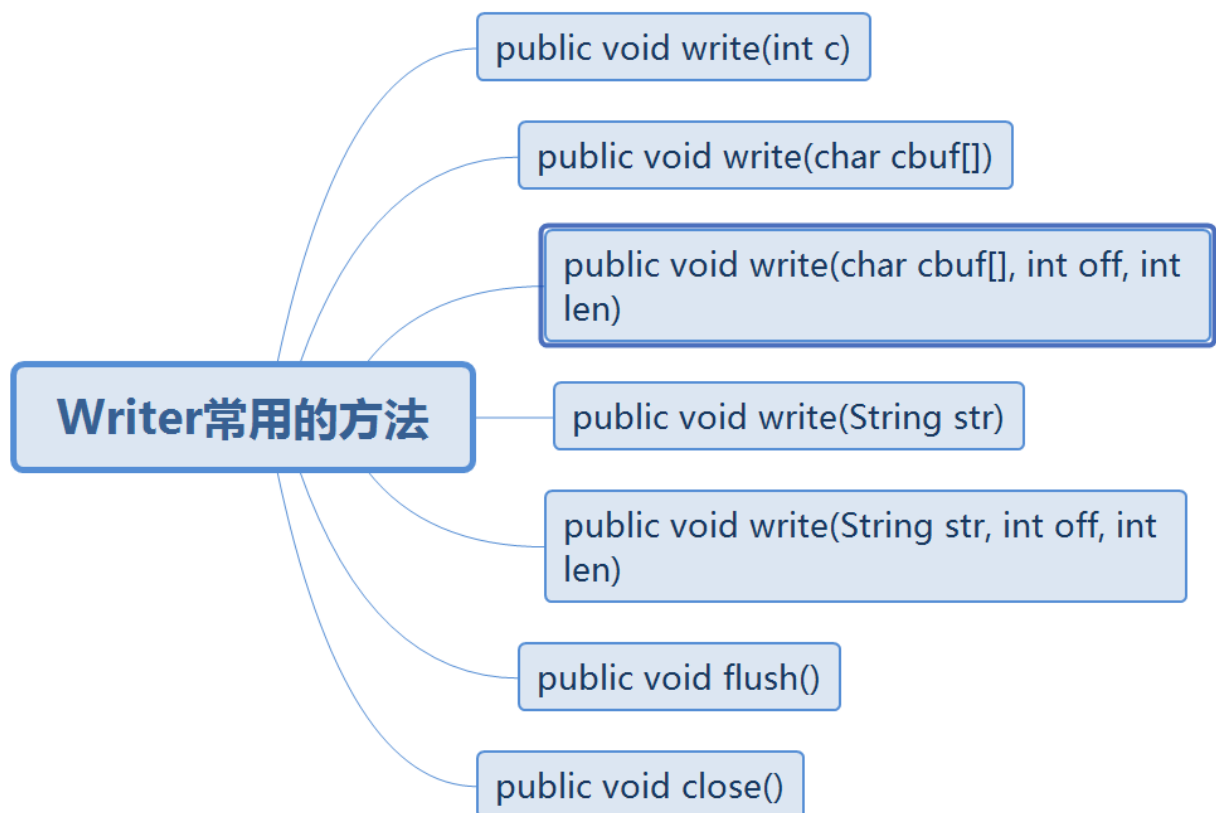
1 * FileReader
2 * FileReader 的两种构建方法
3     public static void main(String[] args) throws IOException {
4         // 使用File对象创建流对象
5         FileReader fr1=new FileReader(new File("test1.txt"));
6         // 使用文件名称创建流对象
7         FileReader fr=new FileReader("test1.txt");
8         fr.close();
9         fr1.close();
10    }
11 * read() 方法
12 public static void main(String[] args) throws IOException {
13     FileReader fr=new FileReader("test3.txt");
14     char c1 = (char)fr.read();
15     System.out.println(c1);
16     char c2 = (char)fr.read();

```

```
17         System.out.println(c2);
18         fr.close();
19     }
20 结果:
21 亮
22 哥
23 * 循环的读: 一个字符的读一个字符的读
24     public static void main(String[] args) throws IOException {
25         FileReader fr=new FileReader("test3.txt");
26         int c=0;
27         while((c=fr.read())!=-1) {
28             System.out.printf("%c",c);
29         }
30         fr.close();
31     }
32 结果:
33 亮哥教育
34 做教育,我们是认真的。
35 亮哥教育
36 做教育,我们是认真的。
37 * 多个字符的读
38     public static void main(String[] args) throws IOException {
39         FileReader fr=new FileReader("test3.txt");
40         char[] buf=new char[2];
41         while((fr.read(buf))!=-1) {
42             String v=new String(buf);
43             System.out.printf("%s",v);
44         }
45         fr.close();
46     }
47     public static void main(String[] args) throws IOException {
48         FileReader fr=new FileReader("test3.txt");
49         char[] buf=new char[2];
50         int len=0;
51         while((len=fr.read(buf))!=-1) {
52             String v=new String(buf,0,len);
53             System.out.printf("%s",v);
54         }
55         fr.close();
56     }
```

```
57 * 结果:
58 亮哥教育
59 做教育, 我们是认真的。
60 亮哥教育
61 做教育, 我们是认真的。
```

* 能够使用字符输出流Writer的写出数据到文件



```
1 * FileWriter
2 * 两个构造器
3 * 方法测试
4 public static void main(String[] args) throws IOException {
5     FileWriter fw=new FileWriter("test3.txt");
6     fw.write(65);
7     fw.write(20142);
8     fw.write(21733);
9     fw.write(25945);
10    fw.write(32946);
11    fw.write("亮哥教育");
```

```

12         fw.write("亮哥教育".toCharArray());
13         fw.write("亮哥教育".toCharArray(),2,2);
14         fw.close();
15     }
16     public static void main(String[] args) throws IOException {
17         //WRITE_BUFFER_SIZE = 1024;
18         FileWriter fw=new FileWriter("a.txt");
19         fw.write("xiaohei");
20         fw.flush();
21         fw.write("xiaobai");
22         fw.close();
23     }

```

* 能够处理IO的异常

```

1  * JDK1.7之前
2  public static void main(String[] args) {
3      FileWriter fw = null;
4      try {
5          fw = new FileWriter("test3.txt");
6          fw.write(65);
7          fw.write(20142);
8          fw.write(21733);
9          fw.write(25945);
10         fw.write(32946);
11         fw.write("亮哥教育");
12         fw.write("亮哥教育".toCharArray());
13         fw.write("亮哥教育".toCharArray(),2,2);
14     } catch (IOException e) {
15         e.printStackTrace();
16     }finally {
17         if(fw!=null) {
18             try {
19                 fw.close();
20             } catch (IOException e) {
21                 e.printStackTrace();
22             }
23         }
24     }

```



```
25     }
26 }
27
28 1.7 AutoClosable
29 try (创建流对象语句, 如果多个使用 ';' 隔开) {
30     // 读写数据
31 }
32 catch (IOException e)
33 {
34     e.printStackTrace();
35 }
36 public static void main(String[] args) {
37     // jdk1.7
38     try(FileWriter fw =new FileWriter("test3.txt")){
39         fw.write(65);
40         fw.write(20142);
41         fw.write(21733);
42         fw.write(25945);
43         fw.write(32946);
44         fw.write("亮哥教育");
45         fw.write("亮哥教育".toCharArray());
46         fw.write("亮哥教育".toCharArray(),2,2);
47     } catch (IOException e1) {
48         e1.printStackTrace();
49     }
50 }
51
```

* 能够掌握Properties的使用