* 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, getCononcialPath

* exisits,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

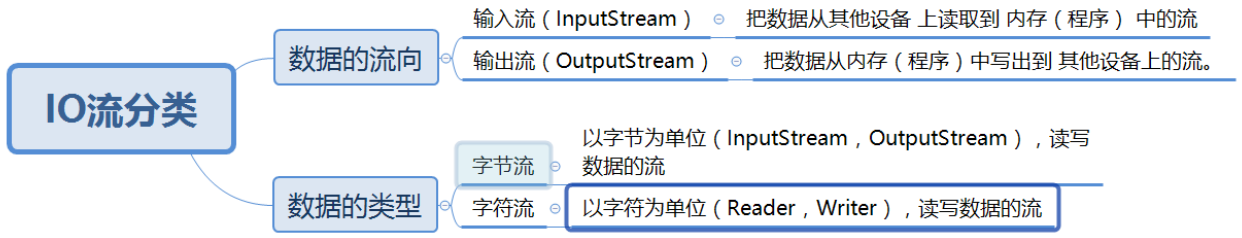
* 在Java中，通过不同输入输出设备之间的数据传输抽象的表述为“流”，程序允许通过流的方式与输入输出设备进行数据传输。

* 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。

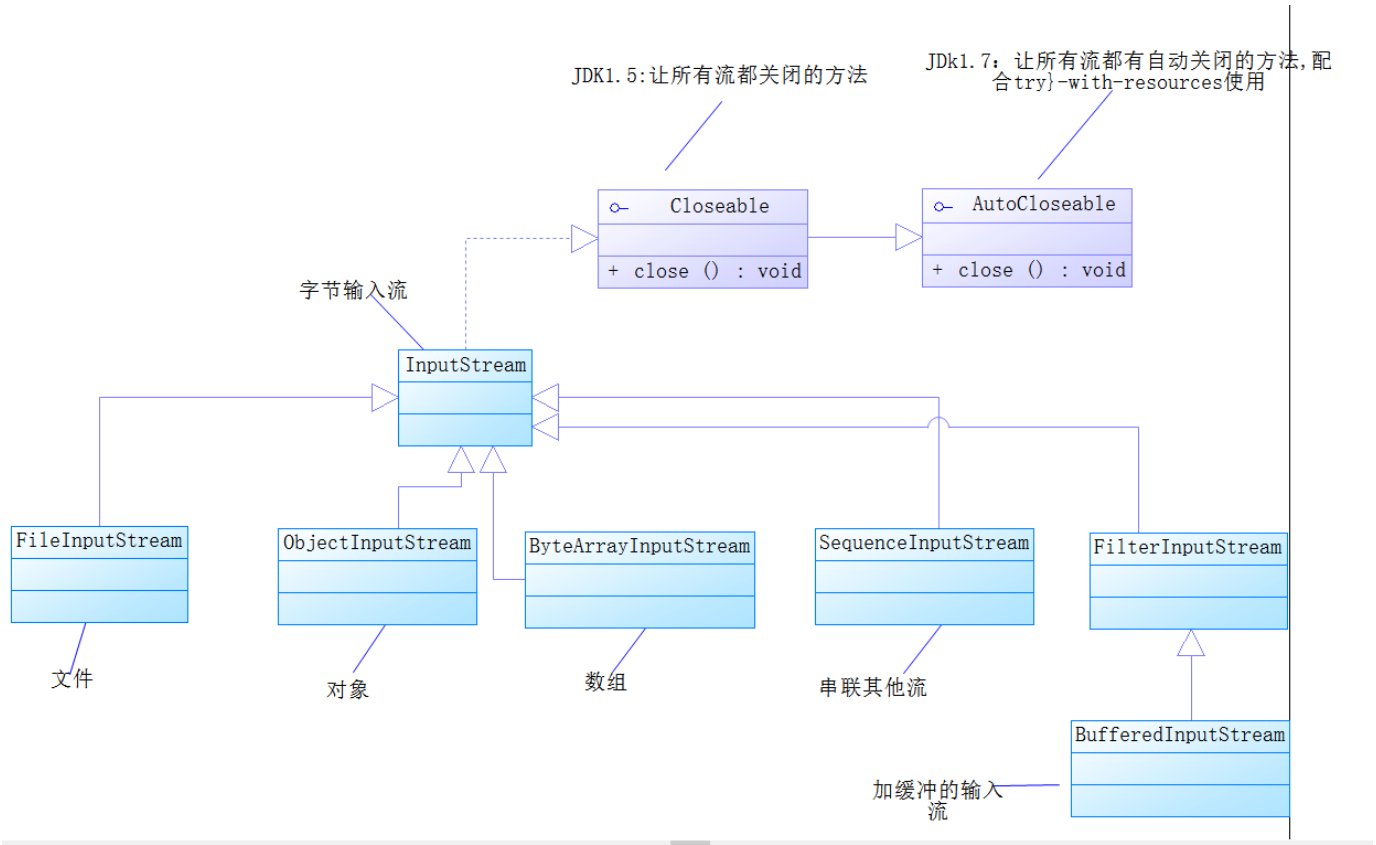
* Java中的“流”主要位于java.io包中，称之为IO（输入输出）流。输入也叫做读取数据，输出也叫做作写出数据。

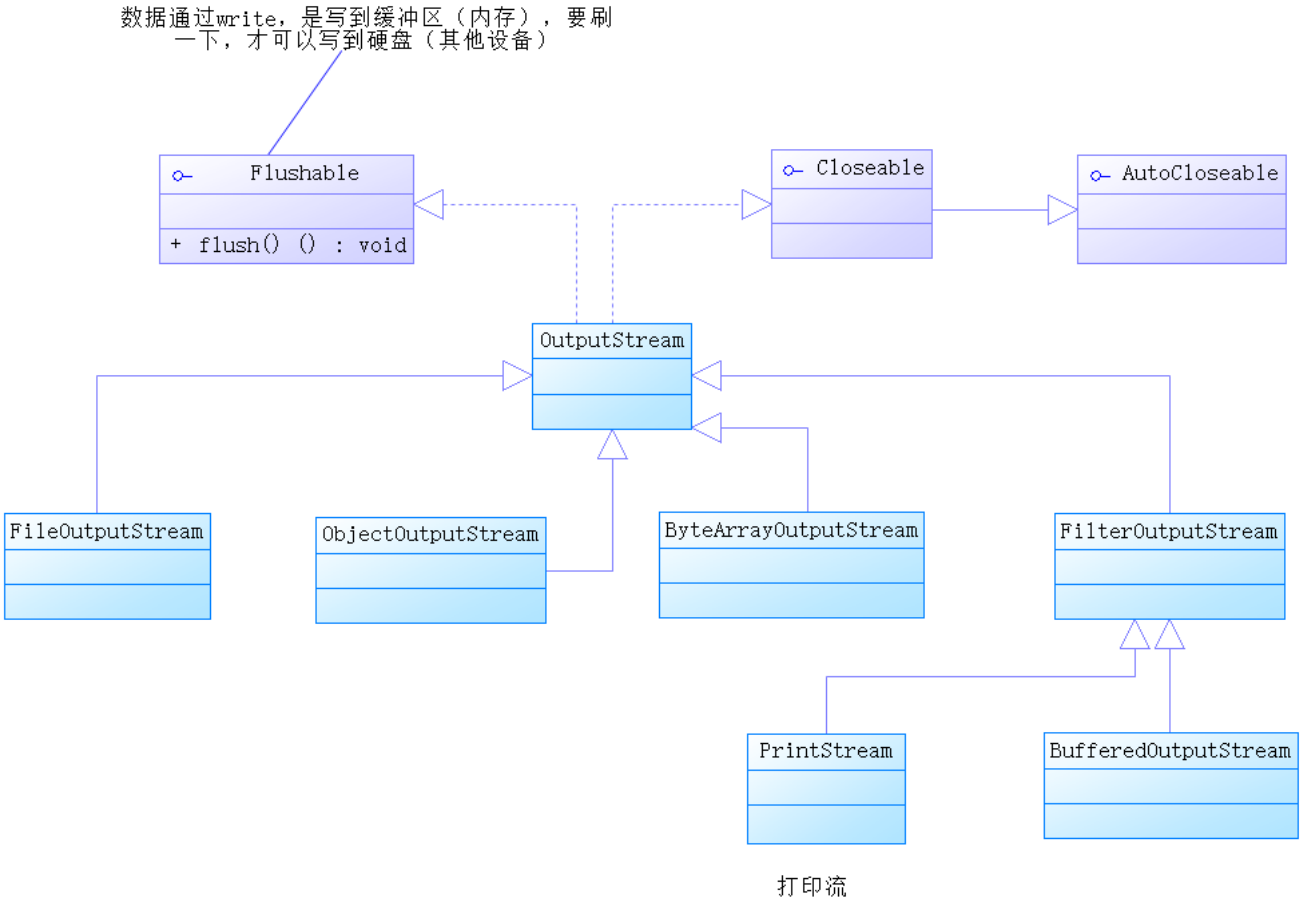
* IO流的分类



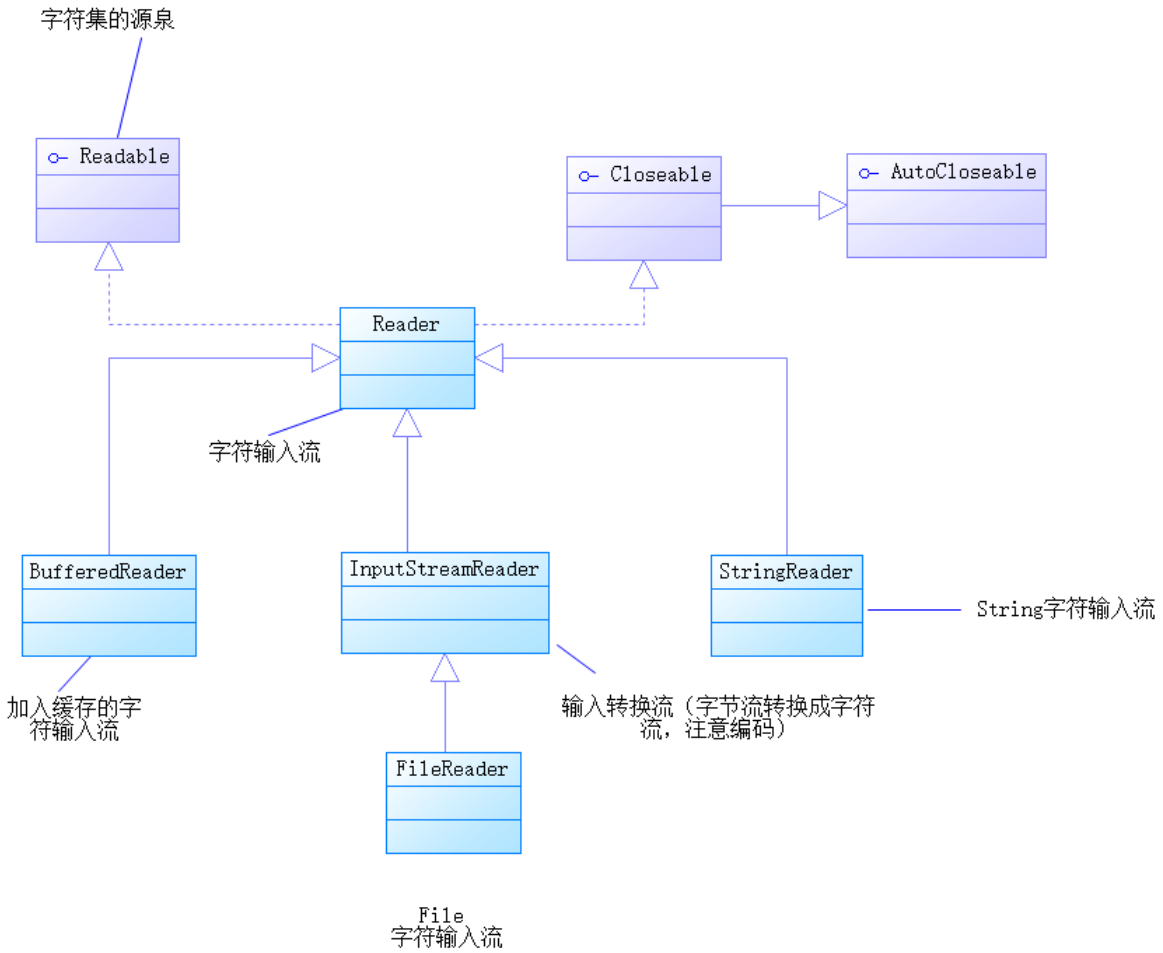
* 温馨提醒： 字节流可以传输任意文件数据（图片，视频，文本...），底层传输的始终为二进制数据。

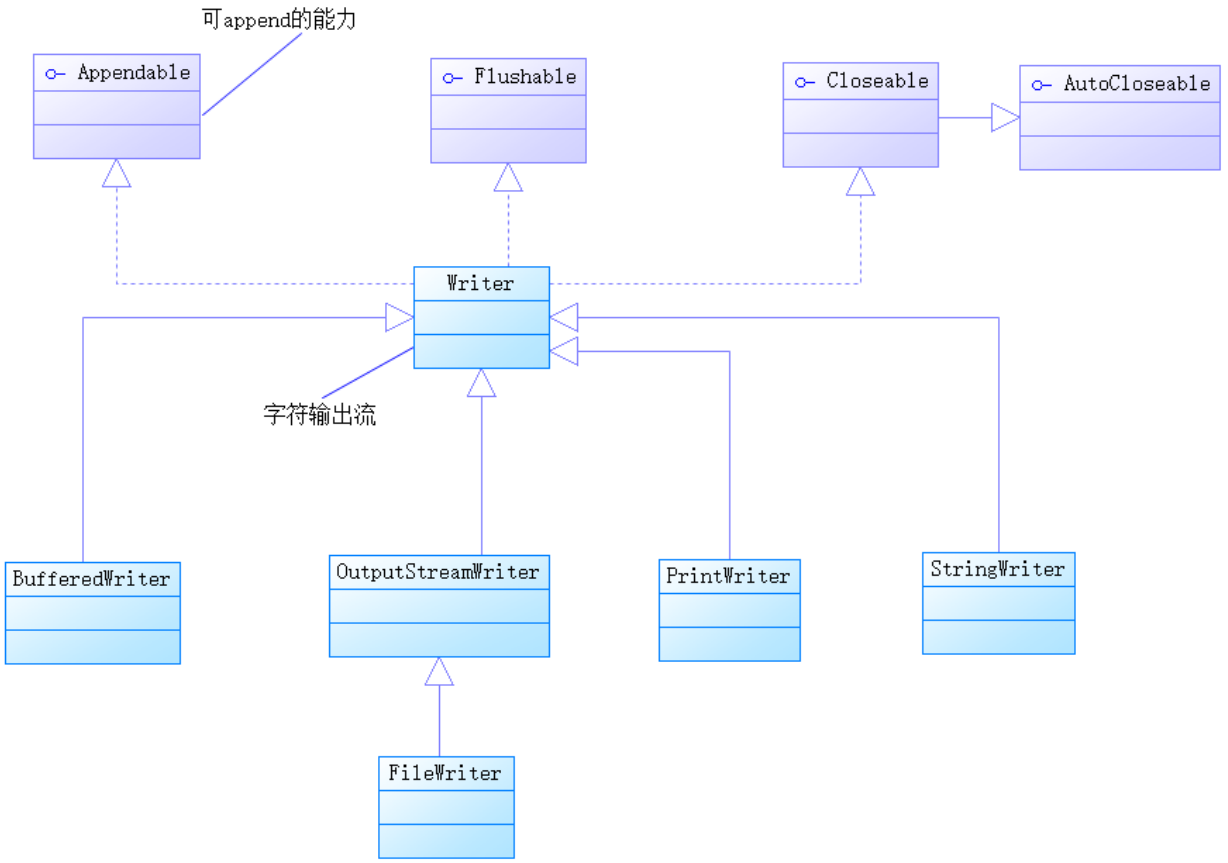
* IO流测继承结构




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）
Flushable
+ flush() () : void
Closeable
AutoCloseable
OutputStream
FileOutputStream
ObjectOutputStream
ByteArrayOutputStream
FilterOutputStream
PrintStream
BufferedOutputStream

打印流


字符集的源泉
Readable
Closeable
AutoCloseable
Reader
字符输入流
BufferedReader
InputStreamReader
StringReader
String字符输入流
加入缓存的字符输入流
输入转换流（字节流转换成字符流，注意编码）
FileReader
File
字符输入流



* InputStream　OutputStream

  * FileInputStream　FileOutputStream

  * ObjectInputStream ObjectOutputStream

  * BufferedInputStream BufferedOutputStream

  * ByteArrayInputStream ByteArrayOutputStream

  * Closable--close AutoClosable--close

  * SequenceInputStream, PrintStream

* Reader Writer

  * BufferedReader　BufferedWriter

  * InputStreamReader OutputStreamWriter

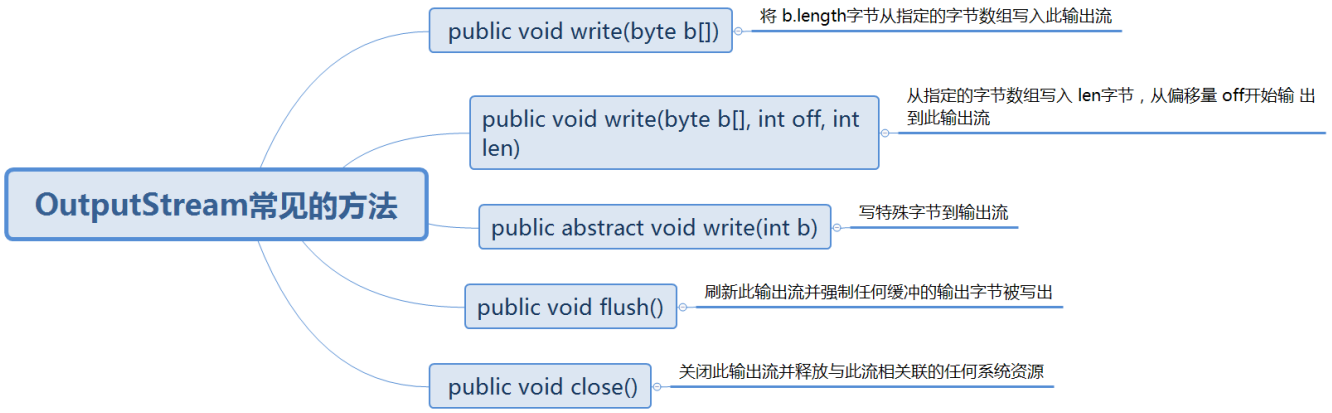
  * FileReader　FileWriter

  * 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




```
* FileOutputStream：用于将数据写到文件
  * FileOutputStream的两种构造方式
  public static void main(String[] args) throws IOException {
        // 使用File对象创建流对象
        File file=new File("test1.txt");
        FileOutputStream fos=new FileOutputStream(file);
        // 查看ASCII码表
        fos.write(65);
        fos.write(66);
        fos.close();

        // 使用文件名称创建流对象
        FileOutputStream fos1=new FileOutputStream("test2.txt");
        fos1.write(67);
        fos1.write(68);
        fos1.close();
        System.out.println("程序结束。。。");
    }
 * 温馨提醒：
    * 当你创建一个流对象时，传入一个文件路径。
    * 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。

public static void main(String[] args) throws IOException {
        // 使用文件名称创建流对象
        FileOutputStream fos1=new FileOutputStream("test3.txt");
        fos1.write("亮哥教育".getBytes());
```

```
        // 测试 只有\n 两种加一起  \n\r 发现光标的位置不一样
        fos1.write("\r\n".getBytes());
        fos1.write("做教育，我们是认真的。".getBytes());
        fos1.close();
        System.out.println("程序结束。。。");
    }
* 温馨提醒：
 * 回车符 \r 和换行符 \n ：
    * 回车符：回到一行的开头（return）。
    * 换行符：下一行（newline）。
 * 系统中的换行：
    * Windows系统里，每行结尾是 回车+换行 ，即 \r\n
    * Linux系统里，每行结尾只有 换行 ，即 \n

    public static void main(String[] args) throws IOException {
        // 使用文件名称创建流对象
        FileOutputStream fos1=new FileOutputStream("test4.txt");
        // 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。
        fos1.write("abcde".getBytes(),1,3);
        fos1.close();
        System.out.println("程序结束。。。");
    }
 * 结果只有
   bcd
 public static void main(String[] args) throws IOException {
        // 使用文件名称创建流对象
        FileOutputStream fos1=new FileOutputStream("test4.txt");
        //一个汉字正常是两个字节
        fos1.write("亮哥教育".getBytes(),0,4);
        fos1.close();
        System.out.println("程序结束。。。");
    }

 * 结果：
  * 亮哥

 * 追加的形式（默认是不追加的）
 public static void main(String[] args) throws IOException {
        // 使用文件名称创建流对象
        FileOutputStream fos1=new FileOutputStream("test3.txt",true);
```

```
        fos1.write("\r\n".getBytes());
        fos1.write("亮哥教育".getBytes());
        // 测试 只有\n 两种加一起  \n\r 发现光标的位置不一样
        fos1.write("\r\n".getBytes());
        fos1.write("做教育，我们是认真的。".getBytes());
        fos1.close();
        System.out.println("程序结束。。。");
    }
```

* 能够使用字节输入流(InputStream)读取数据到程序


```
* FileInputStream
  * 构建文件输入流的两种方式
   public static void main(String[] args) throws IOException {
        // 构建文件输入流的两种方式
        File file=new File("test1.txt");
        FileInputStream fis=new FileInputStream(file);

        FileInputStream fis1=new FileInputStream("test1.txt");
        fis.close();
        fis1.close();
    }
```

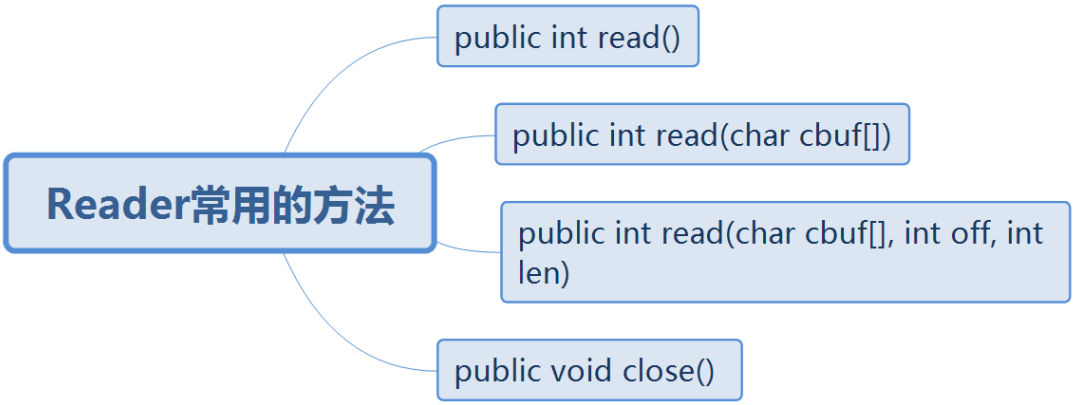
```
   * 温馨提醒：假如文件不存在会报异常：java.io.FileNotFoundException

* 获得文件输入流可读的大小
   public static void main(String[] args) throws IOException {
        FileInputStream fis=new FileInputStream("test1.txt");
        System.out.println("文件可读的大小："+fis.available());
        fis.close();
    }
  * 结果
  文件可读的大小：2

 * read():读下一个字节，一个字节一个字节的读
public static void main(String[] args) throws IOException {
        FileInputStream fis=new FileInputStream("test1.txt");
        System.out.println("文件可读的大小："+fis.available());
        char c1=(char)fis.read();
        System.out.println(c1);
        char c2=(char)fis.read();
        System.out.println(c2);
        // 读取到末尾,返回-1
        int c3=fis.read();
        System.out.println(c3);
        int c4=fis.read();
        System.out.println(c4);
        fis.close();
    }

* 结果：
文件可读的大小：2
A
B
-1
-1

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

```
            System.out.println((char)len);
        }
        fis.close();
    }
 * 结果：
 文件可读的大小：2
A
B
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))!=-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。
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[] b=new byte[2];
        fis.read(b, 0, 2);
        System.out.println(new String(b));
        fis.read(b, 0, 2);
        System.out.println(new String(b));
        fis.read(b, 0, 2);
        System.out.println(new String(b));
        fis.read(b, 0, 2);
        System.out.println(new String(b));
        fis.read(b, 0, 2);
        System.out.println(new String(b));
        fis.close();
    }
 结果：
 文件可读的大小：66
```

```
亮
哥
教
育

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。

* 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();
```

```
        fis.close();
    }


  * 温馨提醒：
    * 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

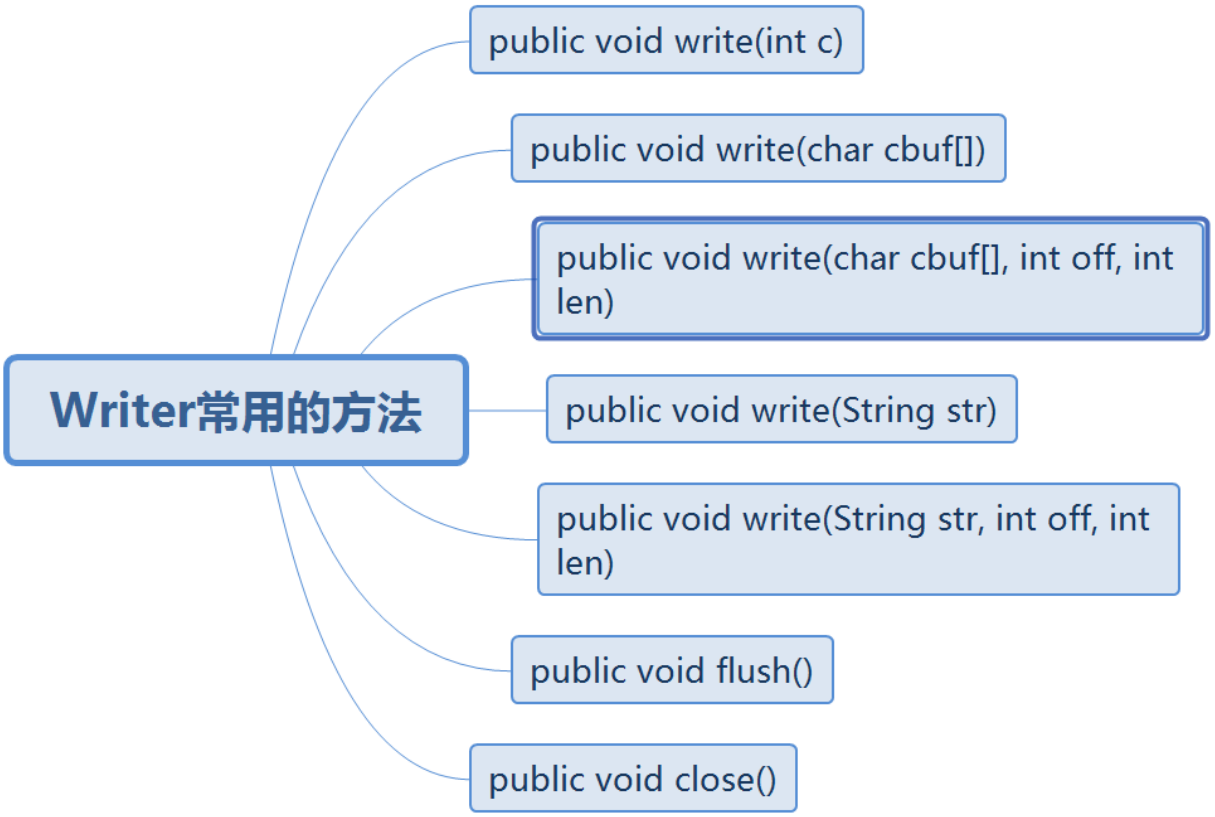



```
* FileReader
 * FileReader 的两种构建方法
    public static void main(String[] args) throws IOException {
        // 使用File对象创建流对象
        FileReader fr1=new FileReader(new File("test1.txt"));
        // 使用文件名称创建流对象
        FileReader fr=new FileReader("test1.txt");
        fr.close();
        fr1.close();
    }
  * read() 方法
  public static void main(String[] args) throws IOException {
        FileReader fr=new FileReader("test3.txt");
        char c1 = (char)fr.read();
        System.out.println(c1);
        char c2 = (char)fr.read();
```

```
        System.out.println(c2);
        fr.close();
    }
结果：
亮
哥
* 循环的读：一个字符的读一个字符的读
    public static void main(String[] args) throws IOException {
        FileReader fr=new FileReader("test3.txt");
        int c=0;
        while((c=fr.read())!=-1) {
            System.out.printf("%c",c);
        }
        fr.close();
    }
结果：
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。
 * 多个字符的读
 public static void main(String[] args) throws IOException {
        FileReader fr=new FileReader("test3.txt");
        char[] buf=new char[2];
        while((fr.read(buf))!=-1) {
            String v=new String(buf);
            System.out.printf("%s",v);
        }
        fr.close();
    }
    public static void main(String[] args) throws IOException {
        FileReader fr=new FileReader("test3.txt");
        char[] buf=new char[2];
        int len=0;
        while((len=fr.read(buf))!=-1) {
            String v=new String(buf,0,len);
            System.out.printf("%s",v);
        }
        fr.close();
    }
```

```
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




```
* FileWriter
 * 两个构造器
 * 方法测试
 public static void main(String[] args) throws IOException {
        FileWriter fw=new FileWriter("test3.txt");
        fw.write(65);
        fw.write(20142);
        fw.write(21733);
        fw.write(25945);
        fw.write(32946);
        fw.write("亮哥教育");
```

```
        fw.write("亮哥教育".toCharArray());
        fw.write("亮哥教育".toCharArray(),2,2);
        fw.close();
    }
public static void main(String[] args) throws IOException {
        //WRITE_BUFFER_SIZE = 1024;
        FileWriter fw=new FileWriter("a.txt");
        fw.write("xiaohei");
        fw.flush();
        fw.write("xiaobai");
        fw.close();
    }
```

* 能够处理IO的异常

```
* JDK1.7之前
public static void main(String[] args) {
        FileWriter fw = null;
        try {
            fw = new FileWriter("test3.txt");
            fw.write(65);
            fw.write(20142);
            fw.write(21733);
            fw.write(25945);
            fw.write(32946);
            fw.write("亮哥教育");
            fw.write("亮哥教育".toCharArray());
            fw.write("亮哥教育".toCharArray(),2,2);
        } catch (IOException e) {
            e.printStackTrace();
        }finally {
            if(fw!=null) {
                try {
                    fw.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
```

```
    }
}

1.7 AutoClosable
 try (创建流对象语句，如果多个使用';'隔开) {
     // 读写数据
    }
     catch (IOException e)
     {
         e.printStackTrace();
     }
public static void main(String[] args) {
        // jdk1.7
        try(FileWriter fw =new FileWriter("test3.txt")){
            fw.write(65);
            fw.write(20142);
            fw.write(21733);
            fw.write(25945);
            fw.write(32946);
            fw.write("亮哥教育");
            fw.write("亮哥教育".toCharArray());
            fw.write("亮哥教育".toCharArray(),2,2);
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }

```

* 能够掌握Properties的使用