Java中主要包含两种流：输入流与输出流，其中输入、输出流又可以分为字节流和字符流

1）File类

File类是java.io包下代表与平台无关的文件和目录，也就是说，如果希望在程序中操作文件和目录，都可以通过File类来完成，值得指出的是，不管是文件还是目录都是使用File来才做的，File能新建、删除、重命名文件和目录，File不能访问文件本身的内容，如果需要访问文件内容本身，则需要使用输入输出流

**public** **class** file {

**public** **static** **void** main(String[] args) **throws** Exception{

File file = **new** File(".");

System.*out*.println(file.getName());

System.*out*.println(file.getAbsolutePath());

System.*out*.println(file.getAbsoluteFile());

System.*out*.println(file.getAbsoluteFile().getParent());

File tmpFile = File.*createTempFile*("aaaa", ".txt", file);

tmpFile.deleteOnExit();//指定当JVM退出时删除该文件

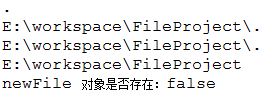
File newFile = **new** File(System.*currentTimeMillis*() + "");

System.*out*.println("newFile 对象是否存在：" + newFile.exists());

newFile.createNewFile();

}

}



2）文件过滤器

在File的list方法中可以接受一个Filenameilter参数，通过该参数可以只列出符合条件的文件

**public** **class** file {

**public** **static** **void** main(String[] args) **throws** Exception{

File file = **new** File(".");

String[] nameList = file.list(**new** MyFileNameFilter());

**for**(String name: nameList){

System.*out*.println(name);

}

}

}

**class** MyFileNameFilter **implements** FilenameFilter{

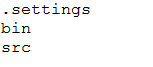
@Override

**public** **boolean** accept(File dir, String name) {

**return** name.endsWith(".java") || **new** File(name).isDirectory();

}

}



3）输入流与输出流

Java的输入流主要由InputStream和Reader作为基类，而输出流则主要由OutputStream与Writer作为基类，它们都是一些抽象基类，无法直接创建实例，其中InputStream与OutputStream是字节流，Reader与Writer是字符流，字节流操作的数据单元是8位的字节，而字符流操作的数据单元是16为的字符

**FileInputStream示范：**

**public** **class** file {

**public** **static** **void** main(String[] args) **throws** Exception{

FileInputStream inputStream = **new** FileInputStream("E:\\workspace\\FileProject\\src\\file.java");//创建字节输入流

**int** hasRead = 0;//当前已经读取的字节

**byte**[] buffer = **new** **byte**[1024];

**while**((hasRead = inputStream.read(buffer)) > 0){

//取出字节并将字节转化为字符串

System.*out*.println(**new** String(buffer, 0, hasRead));

}

inputStream.close();

}

}

**FileReader示范：**

**public** **class** file {

**public** **static** **void** main(String[] args) **throws** Exception{

FileReader fileReader = **new** FileReader("E:\\workspace\\FileProject\\src\\file.java");

**int** hasRead = 0;//当前已经读取字符

**char**[] buffer = **new** **char**[32];

**while**((hasRead = fileReader.read(buffer)) > 0){

System.*out*.println(**new** String(buffer, 0, hasRead));

}

fileReader.close();

}

}

**输出流示范：**

**public** **class** file {

**public** **static** **void** main(String[] args) **throws** Exception{

FileReader fileReader = **new** FileReader("E:\\workspace\\FileProject\\src\\file.java");

FileWriter fileWriter = **new** FileWriter("E:\\wuqianhu.txt");

**int** hasRead = 0;

**char**[] buffer = **new** **char**[1024];

**while**((hasRead = fileReader.read(buffer)) > 0){

fileWriter.write(**new** String(buffer, 0, hasRead));

}

fileReader.close();

fileWriter.close();

}

}

通过上面的四个基类的用法比较繁琐，可以采用处理流来进行处理：处理流隐藏了底层设备上节点流的差异，并对外提供了更加方便的输入/输出方法，因此我们使用处理流的方法是，使用处理流来包装节点流，程序通过处理流来执行输入/输出功能，让节点流于底层的I/O设备、文件进行交互。

**使用PrintStream处理流示范：**

PrintStream是字节打印流，PrintWriter是字符打印流

FileOutputStream fileOutputStream = **new** FileOutputStream("E:\\file.txt");

PrintStream ps = **new** PrintStream(fileOutputStream);

ps.println("hello world");

ps.println(**new** file());

ps.close();

输出结果：



在使用处理流包装了底层节点流后，关闭输入输出流资源时，只要关闭最上层的处理流即可，关闭最上层的处理流时，系统会自动关闭被该处理流包装的节点流

**通常来说，如果进行输入输出的内容是文本内容，则应该考虑使用字符流，如果进行输入输出的内容是二进制内容，则应该考虑使用字节流**

**try**{

InputStreamReader inputStreamReader = **new** InputStreamReader(System.*in*);

BufferedReader bufferReader = **new** BufferedReader(inputStreamReader);

String buffer;

**while**((buffer = bufferReader.readLine()) != **null**){

**if**(buffer.equals("exit")){

System.*exit*(1);

}

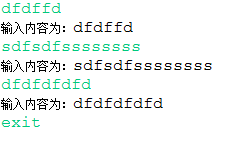
System.*out*.println("输入内容为：" + buffer);

}

}**catch**(Exception e){

e.printStackTrace();

}



Java中使用System.in代表标准输入，即键盘输入，大米这个标准输入流是InputStream实例，使用不太方便，而且键盘输入内容都是文本内容，所以可以采用InputStreamReader将其转换为字符输入流，普通的Reader读取输入内容时仍然不太方便，我们可以将普通的Reader包装为BufferedReader，利用BufferedReader的readLine()方法可以一次读取一行内容。