8、IO

8.1 其他对象API

8.1.1 System类

System类中的字段和方法都是静态的。

常见方法：

long currentTimeMillis();获取当前时间的毫秒值，可以通过此方法检测程序的执行时间。

示例：

```java
public class SystemDemo{
    public static void main(String[] args){
        long l1 = System.currentTimeMillis();
        System.out.println(l1);

        code...

        long l2 = System.currentTimeMillis();
        System.out.println(l2-l1);
    }
}
```
复制代码

Properties getProperties();确定当前的系统属性。

Properties集合中存储的都是String类型的键和值。

最好使用它自己的存储和取出的方法来完成元素的操作。

示例1：

```java
import java.util.Properties;
import java.util.Set;

public class SystemDemo{
      public static void main(String[] args){
           //获取系统的属性信息，并存储到了Properties集合中
          Properties prop = System.getProperties();

          Set<String> nameSet = prop.stringPropertyNames();

           for(String name : nameSet){
                String value = prop.getProperty(name);
                System.out.println(name + " = " + value);
          }
      }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day20>javac SystemDemo.java

D:\code\day20>java SystemDemo
java.runtime.name = Java(TM) SE Runtime Environment
sun.boot.library.path = D:\Java\jdk1.6.0_21\jre\bin
java.vm.version = 17.0-b16
java.vm.vendor = Sun Microsystems Inc.
java.vendor.url = http://java.sun.com/
path.separator = ;
java.vm.name = Java HotSpot(TM) Client VM
file.encoding.pkg = sun.io
user.country = CN
sun.java.launcher = SUN_STANDARD
sun.os.patch.level = Service Pack 1
```

Windows系统中换行为\r\n两个转义字符，Linux只有一个\n。

示例2：

```java
import java.util.Properties;
import java.util.Set;

public class SystemDemo{
      public static void main(String[] args){
          System.out.println( "hello-\r\n world");

           final String LINE_SEPARATOR = System.getProperty("line.separator" );
          System.out.println( "hello" + LINE_SEPARATOR  + "word" );
      }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day20>javac SystemDemo.java

D:\code\day20>java SystemDemo
hello-
 world
hello
word
```

P.S.

给属性设置一些属性信息，这些信息是全局的，其他程序都可以使用。例：System.setPeroperty("myclasspath","c:\myclass");。

8.1.2 Runtime类

每个 Java 应用程序都有一个 Runtime 类实例，使应用程序能够与其运行的环境相连接。应用程序不能创建自己的 Runtime 类实例。

P.S.

Runtime：没有构造方法摘要，说明该类不可以创建对象。又发现还有非静态的方法，说明该类应该提供静态的返回该类对象的方法。而且只有一个，说明Runtime类使用了单例设计模式。

示例1：

```java
public class RuntimeDemo{
      public static void main(String[] args) throws Exception{
          Runtime r = Runtime.getRuntime();

          Process p = r.exec( "notepad.exe D:\\code\\day20\\RuntimeDemo.java");
          Thread.sleep(5000);
          p.destroy();
      }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day20>javac RuntimeDemo.java

D:\code\day20>java RuntimeDemo
```

```
RuntimeDemo.java - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
public class RuntimeDemo{
      public static void main(String[] args) throws Exception{
          Runtime r = Runtime.getRuntime();

          Process p = r.exec( "notepad.exe D:\\code\\day20\\RuntimeDemo.java");
          Thread.sleep(5000);
          p.destroy();
      }
}
```

8.1.3 Math类

Math：提供了操作数学运算的方法，都是静态的。

常用方法：

ceil():返回大于参数的最小整数。

floor():返回小于参数的最大整数。

round():返回四舍五入的整数。

pow(a,b):a的b次方。

示例1：

```java
import java.util.Random;

public class MathDemo{
      public static void main(String[] args){
           double d1 = Math.ceil(12.56);
           double d2 = Math.floor(12.56);
           double d3 = Math.round(12.56);

          System.out.println(d1);
          System.out.println(d2);
          System.out.println(d3);

           double d4 = Math.pow(10,2);
          System.out.println(d4);

          Random r = new Random();

           for(int i = 0; i < 5; i++){
                double d5 = Math.ceil(Math.random() * 10);
               System.out.println( "d5 = " + d5);
           }

           for(int i = 0; i < 5; i++){
                double d6 = (int)(r.nextDouble() * 6 + 1);
               System.out.println( "d6 = " + d6);
           }
      }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day20>javac MathDemo.java

D:\code\day20>java MathDemo
13.0
12.0
13.0
100.0
d5 = 8.0
d5 = 7.0
d5 = 10.0
d5 = 3.0
d5 = 7.0
d6 = 5.0
d6 = 2.0
d6 = 3.0
d6 = 1.0
d6 = 2.0
```

8.1.4 Date、DateFormat类

示例1：

```java
import java.util.Date;

public class DateDemo{
      public static void main(String[] args){
           long time = System.currentTimeMillis();
          System.out.println(time);

           //将当前日期和时间封装成Date对象
          Date date1 = new Date();
          System.out.println(date1);

           //将指定毫秒值封装成Date对象
          Date date2 = new Date(1405244787235l);
          System.out.println(date2);
      }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day20>javac DateDemo.java

D:\code\day20>java DateDemo
1433656405298
Sun Jun 07 13:53:25 CST 2015
Sun Jul 13 17:46:27 CST 2014
```

日期对象和毫秒值之间的转换

毫秒值-->日期对象：

1. 通过Date对象的构造方法 new Date(timeMillis);
2. 还可以通过setTime设置。

因为可以通过Date对象的方法对该日期中的各个字段（年月日等）进行操作。

日期对象-->毫秒值：

1. getTime方法。

因为可以通过具体的数值进行运算。

对日期对象进行格式化：

将日期对象-->日期格式的字符串。

使用的是DateFormat类中的format方法。

示例2：

```java
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateDemo{
      public static void main(String[] args){
          Date date = new Date();

           //获取日期格式对象，具备着默认的风格。也可以指定为FULL、LONG风格。
          DateFormat df = DateFormat.getDateInstance(DateFormat.FULL);
          String str_date1 = df.format(date);
          System.out.println(str_date1);

          df = DateFormat.getDateTimeInstance(DateFormat.FULL,DateFormat.LONG);
          String str_date2 = df.format(date);
          System.out.println(str_date2);

           //如果风格是自定义的如何解决呢？
          df = new SimpleDateFormat("yyyy--MM--dd" );
          String str_date3 = df.format(date);
          System.out.println(str_date3);
      }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day20>javac DateDemo.java

D:\code\day20>java DateDemo
2015年6月7日 星期日
2015年6月7日 星期日 下午01时54分57秒
2015--06--07
```

将日期格式的字符串-->日期对象。

使用的是DateFormat类中的prase方法。

示例3：

```java
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateDemo{
      public static void main(String[] args) throws Exception {
          String str_date1 = "2012年4月19日" ;
          String str_date2 = "2012--4--19";

          DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.LONG);
          Date date1 = dateFormat.parse(str_date1);
          System.out.println(date1);

          dateFormat= new SimpleDateFormat("yyyy--MM--dd" );
          Date date2 = dateFormat.parse(str_date2);
          System.out.println(date2);
      }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day20>javac DateDemo.java

D:\code\day20>java DateDemo
Thu Apr 19 00:00:00 CST 2012
Thu Apr 19 00:00:00 CST 2012
```

练习：

"2012-3-17"到"2012-4-6"中间有多少天？

思路：

两个日期相减就可以了。

必须有两个可以进行减法运算的数。

能减可以是毫秒值，如何获取毫秒值？通过date对象。

如何获取date对象呢？可以将字符串转成date对象。

1. 将日期格式的字符串转成Date对象。
2. 将Date对象转成毫秒值。
3. 相减，编程天数。

代码：

```java
import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;

public class Test{
      public static void main(String[] args) throws Exception {
          String str_date1 = "2014-3-17";
          String str_date2 = "2014-4-6";
          test(str_date1,str_date2);
      }

      public static void test(String str_date1,String str_date2) throws Exception {
           //1、将日期字符串转成日期对象
           DateFormat dateFormat = DateFormat.getDateInstance();
           dateFormat = new SimpleDateFormat("yyyy-MM-dd" );

           Date date1 = dateFormat.parse(str_date1);
           Date date2 = dateFormat.parse(str_date2);

           long time1 = date1.getTime();
           long time2 = date2.getTime();

           long time = Math.abs(time2-time1);
           System.out.println(time);

           int day = getDay(time);
           System.out.println(day);
      }

      private static int getDay(long time){
           int day = (int)(time/1000/60/60/24);
           return day;
      }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day20>javac Test.java

D:\code\day20>java Test
1728000000
20
```

8.1.5 Calendar类

Calendar 类是一个抽象类，它为特定瞬间与一组诸如YEAR、MONTH、DAY_OF_MONTH、HOUR等日历字段之间的转换提供了一些方法，并为操作日历字段（例如获得下星期的日期）提供了一些方法。

示例一：

```java
import java.util.Calendar;

public class CalendarDemo{

       public static void main(String[] args){
             //GregorianCalendar
            Calendar c = Calendar.getInstance();
            showDate(c);
      }

       public static void showDate(Calendar c){
             int year = c.get(Calendar.YEAR);
             int month = c.get(Calendar.MONTH) + 1;
             int day = c.get(Calendar.DAY_OF_MONTH);
             int week = c.get(Calendar.DAY_OF_WEEK);

             System.out.println(year + "年" + month + "月" + day + "日" + getWeek(week));
      }

       public static String getWeek(int i){
             String[] weeks = { "","星期日" ,"星期一" ,"星期二" ,"星期三" ,"星期四" ,"星期五" ,"星期六" };
             return weeks[i];
      }
}
```
复制代码

运行结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day20>javac CalendarDemo.java

D:\code\day20>java CalendarDemo
2015年6月7日星期日
```

示例二：

```java
import java.util.Calendar;

public class CalendarDemo{

       public static void main(String[] args){
           Calendar c = Calendar.getInstance();
           c.set(2014,3,20);
           c.add(Calendar.YEAR,-2);
           showDate(c);
      }

       public static void showDate(Calendar c){
            int year = c.get(Calendar.YEAR);
            int month = c.get(Calendar.MONTH) + 1;
            int day = c.get(Calendar.DAY_OF_MONTH);
            int week = c.get(Calendar.DAY_OF_WEEK);

           System.out.println(year + "年" + month + "月" + day + "日" + getWeek(week));
      }

       public static String getWeek(int i){
           String[] weeks = { "","星期日" ,"星期一" ,"星期二" ,"星期三" ,"星期四" ,"星期五" ,"星期六" };
            return weeks[i];
      }
}
```
复制代码

运算结果：

```
管理员: C:\Windows\system32\cmd.exe
D:\code\day20>javac CalendarDemo.java

D:\code\day20>java CalendarDemo
2012年4月20日星期五
```

练习：

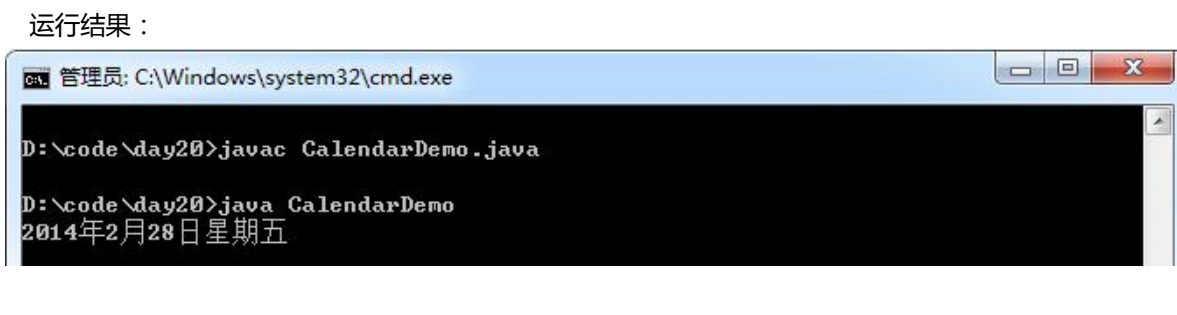
打印每年2月有多少天

代码：

```java
import java.util.Calendar;

public class CalendarDemo{

```

```
    public static void main(String[] args){
        int year = 2014;
        showDays(year);
    }

    public static void showDays(int year){
        Calendar c = Calendar.getInstance();
        //将日期设置为3月1日，然后减一天，打印出2月份的最后一天日期，即可知2月有多少天
        c.set(year,2,1);
        c.add(Calendar.DAY_OF_MONTH,-1);
        showDate(c);
    }

    public static void showDate(Calendar c){
        int year = c.get(Calendar.YEAR);
        int month = c.get(Calendar.MONTH) + 1;
        int day = c.get(Calendar.DAY_OF_MONTH);
        int week = c.get(Calendar.DAY_OF_WEEK);

        System.out.println(year + "年" + month + "月" + day + "日" + getWeek(week));
    }

    public static String getWeek(int i){
        String[] weeks = { "","星期日","星期一","星期二","星期三","星期四","星期五","星期六" };
        return weeks[i];
    }
}
```
复制代码

运行结果：



8.2 IO流

8.2.1 IO流

IO流用来处理设备之间的数据传输。Java对数据的操作是通过流的方式。Java用于操作流的对象都在IO包中。
输入流和输出流相对于内存设备而言。

将外设中的数据读取到内存中：输入。
将内存的数写入到外设中：输出。

流按操作数据分为两种：字节流与字符流。

字符流的由来：
其实就是：字节流读取文字字节数据后，不直接操作而是先查指定的编码表，获取对应的文字。再对这个文字进行操作。简单说：字节流+编码表。

8.2.2 IO流常用基类-字符流

字节流的抽象基类：InputStream，OutputStream。

字符流的抽象基类：Reader，Writer。

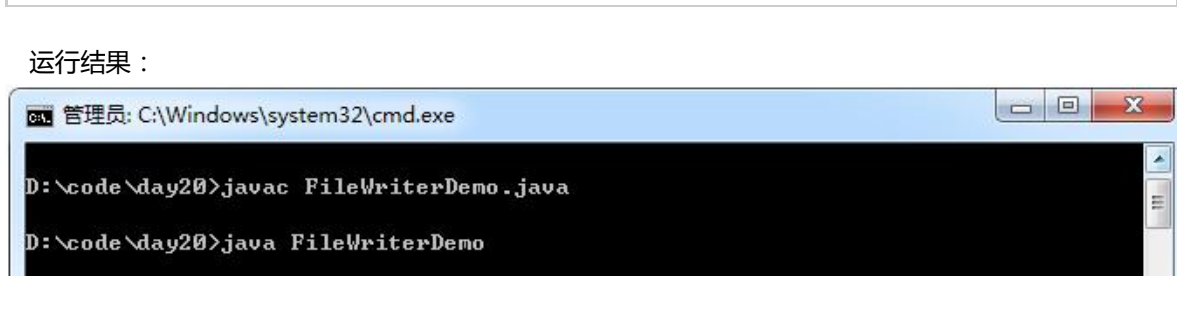
P.S.
由这四个类派生出来的子类名称都是以其父类名作为子类名的后缀。
如：InputStream的子类FileInputStream。
如：Reader的子类FileReader。

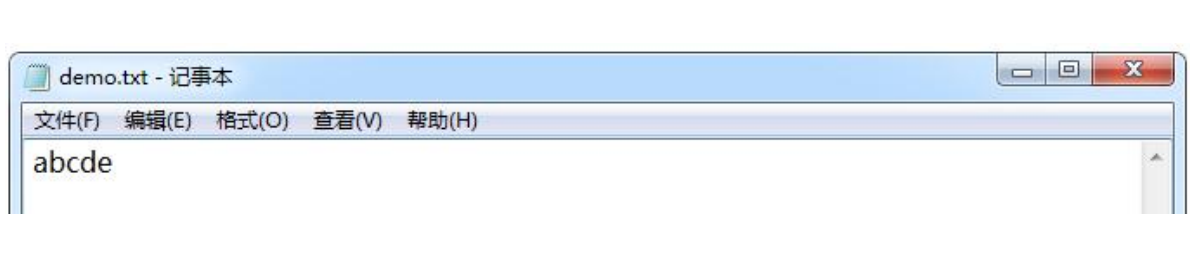
需求：将一些文字存储到硬盘一个文件中。
记住：如果要操作文字数据，建议优先考虑字符流。
而且要将数据从内存写到硬盘上，要使用字符流中的输出流：Writer。
硬盘的数据基本体现是文件，希望找到一个可以操作文件的Writer：FileWriter。

示例1：

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

public class FileWriterDemo{
    public static void main(String[] args) throws IOException{
        //创建一个可以往文件中写入字符数据的字符输出流对象
        //既然是往一个文件中写入文字数据，那么在创建对象时，就必须明确该文件（用于存储数据的目的地）
        //如果文件不存在，则会自动创建
        //如果文件存在，则会被覆盖
        FileWriter fw = new FileWriter("demo.txt" );

        //调用Writer对象中的write(string)方法，写入数据
        //其实数据写入到临时存储缓冲区中
        fw.write( "abcde");

        //进行刷新，将数据直接写入到目的地中
        fw.flush();

        //关闭流，关闭资源，在关闭前会先调用flush刷新缓冲中的数据到目的地。
        fw.close();
    }
}
```
复制代码

运行结果：





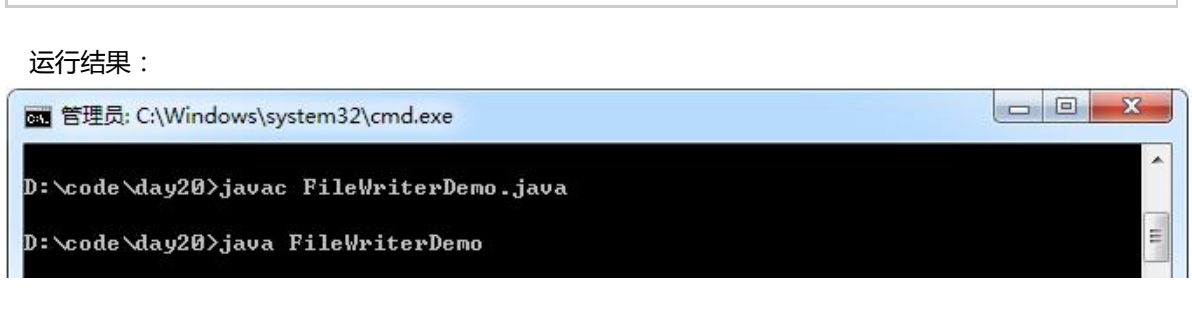
P.S.
1. close方法只能用一次。
2. 流关闭以后不能，不能再调用write方法，否则会报如下异常错误：

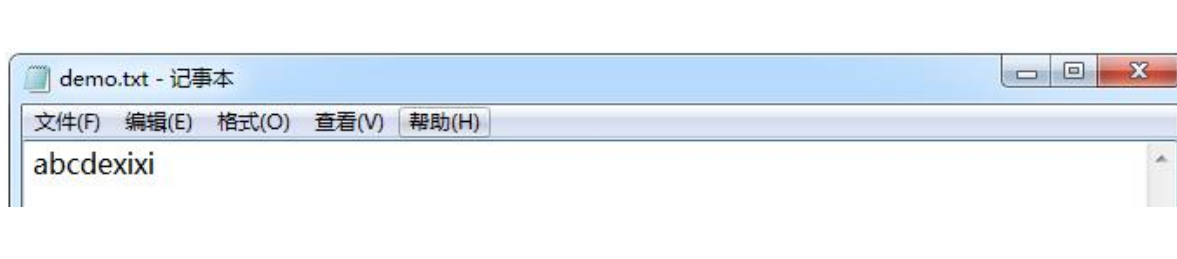
如果构造函数中加入true，可以实现对文件进行续写。

示例2：

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

public class FileWriterDemo{
    public static void main(String[] args) throws IOException{
        FileWriter fw = new FileWriter("demo.txt" ,true);

        fw.write( "xixi");
        fw.close();
    }
}
```
复制代码

运行结果：



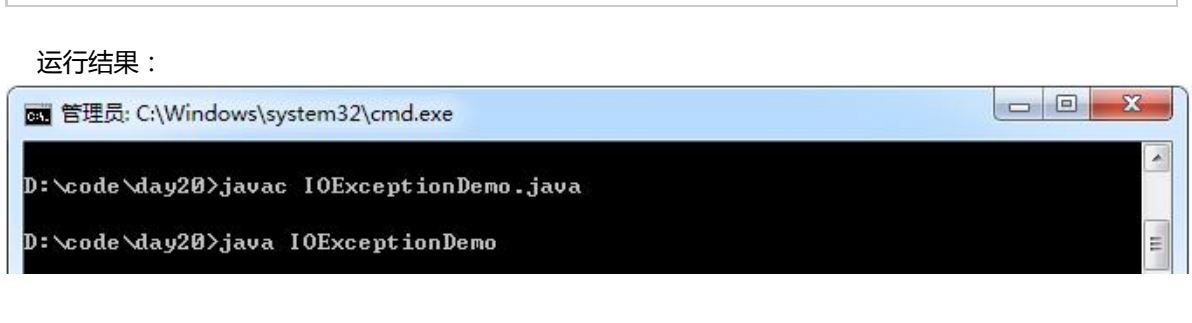


IO流的异常处理方式：为防止代码异常导致流无法关闭，因此在finally中对流进行关闭。

示例3：

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

public class IOExceptionDemo{
        public static void main(String[] args){

             FileWriter fw  = null;

              try{
                   fw = new FileWriter("demo.txt" );

                   fw.write( "abcde");
             } catch(IOException e){
                   System.out.println(e.toString());
             } finally{
                    try{
                         fw.close();
                   } catch(IOException e){
                          throw new RuntimeException("关闭失败");
                   }
             }
       }
}
```
复制代码

运行结果：

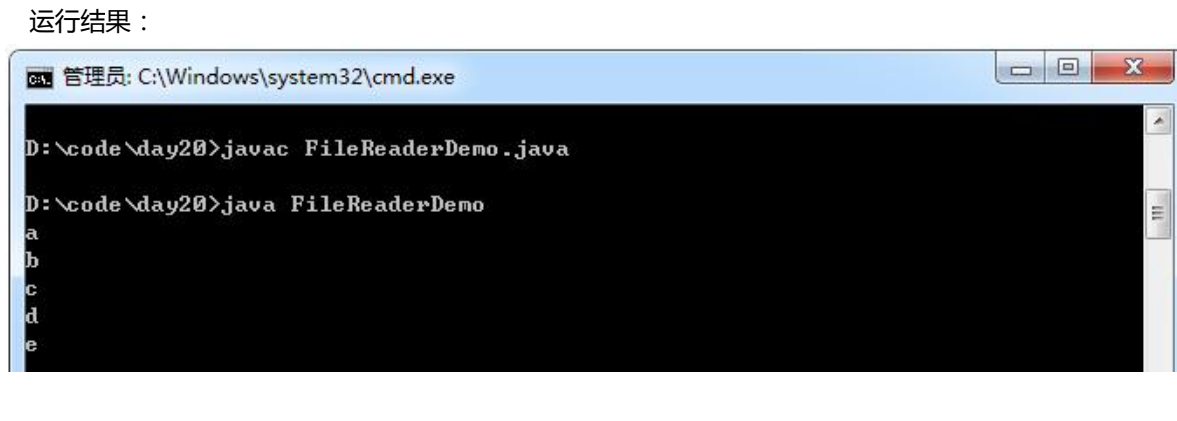


需求：读取一个文文件，将读取到的字符打印到控制台。(使用FileReader)
第一种读取方式：使用read()方法读取文本文件数据。

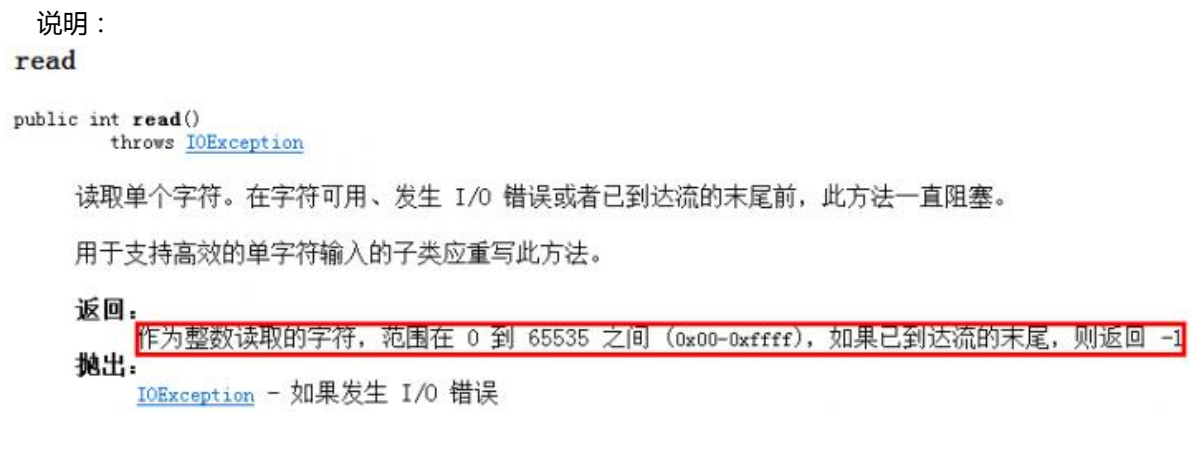
示例4：

```
import java.io.FileReader;
import java.io.IOException;

public class FileReaderDemo{
        public static void main(String[] args) throws IOException{

             FileReader fr = new FileReader("demo.txt" );

              //用Reader中的read方法读取字符
              int ch = 0;

              while((ch = fr.read()) != -1){
                   System.out.println(( char)ch);
             }

             fr.close();
       }
}
```
复制代码

运行结果：



说明：


read

public int read()
throws IOException

读取单个字符。在字符可用、发生 I/O 错误或者已到达流的末尾前，此方法一直阻塞。

用于支持高效的单字符输入的子类应重写此方法。

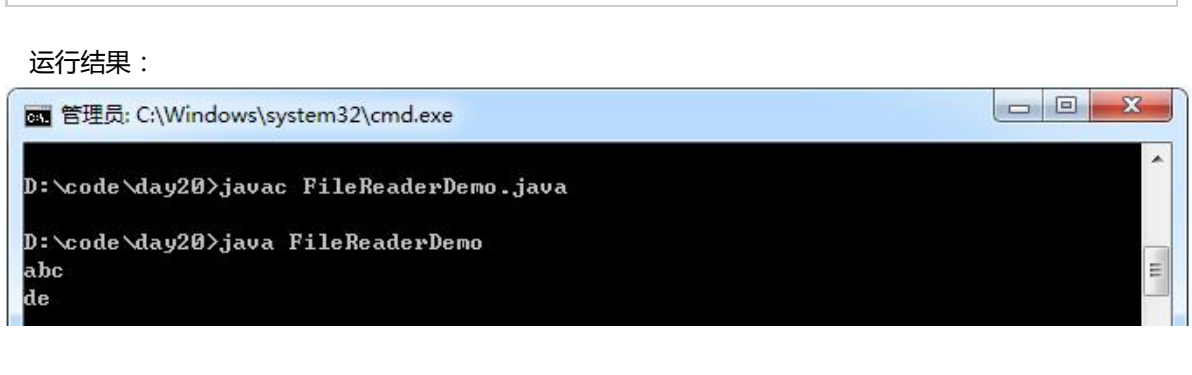
返回：
作为整数读取的字符，范围在 0 到 65535 之间 (0x00-0xffff)，如果已到达流的末尾，则返回 -1
抛出：
IOException - 如果发生 I/O 错误

第二种读取方式：使用read(char[])方法读取文本文件数据。

示例5：

```
import java.io.FileReader;
import java.io.IOException;

public class FileReaderDemo{
        public static void main(String[] args)throws IOException{
             FileReader fr = new FileReader("demo.txt" );

              //使用read(char[])读取文本文件数据
              //先创建字符数组
              char[] buf = new char[3];

              int len = 0;

              while((len = fr.read(buf)) != -1){
                   System.out.println( new String(buf,0,len));
             }

             fr.close();
       }
}
```
复制代码

运行结果：



~END~

~爱上海，爱黑马~