java高级特性

****

chapter 1　集合框架

1.集合框架

接口　　Collection　　Map

子接口　　List　Set

实现类　　ArrayList　HashSet　　HashMap　TreeMap

`　　LinkedList　　TreeSet

2.Collection 不唯一　无序　对象

Set　唯一　　无序

List 不唯一　有序

Map　键值对　key/value　　key:唯一，无序　　value:无序　可重复

3.ArrayList

长度可变的数组　遍历和随机访问效率比较高

使用:　List dogs=new ArrayList();

```
dogs.add(dog1);     //添加dog1 到集合dogs中
dogs.add(0,dog2);   //添加到指定位置　位置不能大于集合的元素个数
dogs.add(dog3);
dogs.size();        //集合的元素个数
```

遍历集合

```
for (int i = 0; i < dogs.size(); i++) {
                    Dog dog = (Dog) dogs.get(i); //得到单个对象　类型为
Object 需要强转
    System.out.println(dog.getName() + "\t"
            + dog.getStrain());
}
```

```
dogs.remove(0);          //删除 指定位置的对象
dogs.remove(dog1);       //删除 指定名称的对象
dogs.contains(dog1);     //false 集合是否包含该对象
```

```
dogs.contains(dog3);    //true
```

4.LinkedList

链表式结构　插入和删除操作 用LinkedList
添加　删除 和 获取 指定位置

```
LinkedList dogs=new LinkedList();
        dogs.add(dog1);          //添加dog1 到集合dogs中
        dogs.add(0,dog2);        //添加到指定位置　位置不能大于集合的元素个数
        dogs.addLast(dog3);      //添加到末尾位置
        dogs.addFirst(dog3);     //添加到开始位置
        dogs.size();             //集合的元素个数

Dog dogFirst=(Dog)dogs.getFirst();  //获取第一个对象
Dog dogLast=(Dog)dogs.getLast();    //获取最后一个对象

dogs.removeFirst();              //删除第一个对象
dogs.removeLast();               //删除最后一个对象
```

5.HashSet

HashSet是set集合的常用实现类　set接口存储的是唯一　无序

```
Set set=new HashSet();
```

如何保证唯一　equals();方法

```
set.add(dog1);          //添加
set.size();             //获得对象个数
```

6.HashMap

存储的是键值对　对象
键key不能重复　值value　可以重复

```
Map games=new HashMap();
games.put("LOL","英雄联盟");            //添加　键值对
games.put("CF","穿越火线");
games.put("DOTA","DOTA");
```

```
games.put("LOL","撸啊撸");                    //添加重复键时   会进行覆盖

String game=(String)games.get("CF");     //获取指定键   对应的值
games.size();                            //获取元素个数
games.remove("CF");                      //删除指定元素
games.containsKey("CF");                 //是否包含指定元素   返回boolean
games.keySet();                          //键集
games.values();                          //值集
gemes                                    //键值对集
```

7.Iterator 迭代器

1>获取Iterator   Collection接口的iterator()方法;

2>hasNext();

next();

3>iterator 遍历Map集合

```
Set keys=games.keySet();                          //得到键集    由此可看出
```
键集为Set集合   不能重复

```
Iterator it=keys.iterator();                      //得到iterator对象
while(it.hasNext()){                              //是否有下一个可访问的元
```
素

```
        String key=(String)it.next();             //得到键
        System.out.println(value);
        String value=(String)games.get(key);     //得到值
        System.out.println(value);
}
```

4>iterator 遍历List集合

```
Iterator it=dogs.iterator();
while(it.hasNext()){                              //是否有下一个可访问的元
```
素

```
        Dog dog=(Dog)it.next();                   //得到一个元素
        System.out.println(dog.name);
}
```

5>增强for循环

```
for(元素类型 元素变量:数组或集合对象){
        使用元素变量
}
eg:
for(Object o:dogs){
        Dog dog=(Dog)o;
        System.out.println(dog.name);
}
```

8.泛型

解决强制类型装换

1>给List添加泛型

```
List<Dog> dogs=new ArrayList<Dog>();
        dogs.add(dog1);
        dogs.add(0,dog2);
        dogs.add(dog3);
        dogs.add("狗狗");   //报错 无法添加
```

2>给Map集合添加泛型

```
Map<String,String> games=new HashMap<String,String>();
Set<String> keys=games.keySet();                    //得到键集    由此可看出键集为Set集合  不能重复
Iterator<String> it=keys.iterator();                //得到iterator对象
while(it.hasNext()){                                //是否有下一个可访问的元素
        String key=it.next();           //得到键
        System.out.println(value);
        String value=games.get(key);    //得到值
        System.out.println(value);
}

for(Dog dog:dogs){
        System.out.println(dog.name);
```

}

****

chapter 2 实用类

1.Java API

JAVA Application Programming Interface java应用程序编程接口

2.枚举 Enum

1>. 定义枚举

```
public enum Gender{
        Male,Female
}
```

2>使用枚举

```
stu.sex=Gender.Male;
```

3>好处:易于维护.易于输入,直观方便

3.包装类: 把基本数据类型包装成对象

1>每个基本数据类型都在lang包下有一个相应的包装类

作用:提供一些使用的方法 集合不能存储基本数据类型 存放数字时,要用包装类型

Byte Short Integer Long Float Double

Boolean Number Character

Object

2>构造方法 基本数据类型转化为包装类

```
public Type(type value){}
public Type(String value){}    //Character除外
```

注意:Boolean时 若字符串为true不计大小写 则为true 否则false

字符串不能为null,且必须可解析为相应类型的数据,否则报异常

```
Integer intvalue=new Integer(51);
Integer intvalue=new Integer("51");
```

3>常用方法

```
Integer intObject=Integer.valueOf(51);
Integer intObject=Integer.valueOf("51");  //Character除外

String sex=Character.toString('男');      //其他数据类型转化为String类型
String sex='男'+"";

int num=Integer.parse("12345");           //把字符串类型变为其他类型
```

4>包装类转化为基本数据类型

```
int intid=intvalue.intValue();
```

5>自动类型转换

```
Integer intObject=5;
int intValue=intObject;
```

6>特点;

final类型 不能创建子类

JDK1.5后,混合运算

where:集合存储

4.Math类

```
Math.abs(-3.5);                          //得到绝对值    3.5
Math.max(2.5,38);                        //得到最大值    38
int random=(int)(Math.random()*10);      //得到随机数   0-9
```

5.String类

不可变的对象,每次修改都是新建了一个新对象

1>创建字符串

```
String s="hello";
String s1=new String("hello");
```

2>常用方法

```
length();                          //得到字符串的长度
```

equals(); //比较两个字符串内容是否相等 == 比较的是内存中的地址

equalsIgnoreCase(); //比较时不计大小写

toLowerCase(); //将字符串转化为小写

toUpperCase(); //将字符串转化为大写

concat(); //字符串连接 或+

indexOf(String value); //得到字符开始出现的下标 无则, -1

lastIndexOf(String value); //得到字符最后出现的下标 无则, -1

substring(int num1); //字符串的截取 从下标位置以后的字符串

substring(int num1,int num2); //下标之间的字符串 包前 不包后

trim(); //删除字符串首尾的空格

split(String str); //以string来分割字符串 得到一个字符串数组

6.StringBuffer util包

操作的对象为字符串,可变对象, 效率比String高

使用:

```
StringBuffer sb=new StringBuffer("哈沙尅");

//追加
sb.append("一库");

//转化为字符串
sb.toString();

//插入
sb.insert(2,"背对疾风吧");    //插入位置 插入内容

//长度
sb.length();
```

7.日期时间类

1>Date类

```
Date date=new Date();   //当前时间对象
```

2>SimpleDateFormat 时间格式转换器　　java.text包

```
SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
String  dateStr=sdf.format(date);
```

3.Calendar 日历

```
Calendar cal=Calendar.getInstence();
cal.get(Calendar.YEAR);          //当前年
cal.get(Calendar.MONTH)+1;       //当前月     0-11
cal.get(Calendar.DAY_OF_MONTH); //当前日期
cal.get(Calendar.DAY_OF_WEEK);  //当前星期几    星期天是0  其他正常
cal.set(2017,4,19);              //设置到指定日期  2017 5 19
```

8.Random类 随机数

```
Random random=new Random();
int num=random.nextInt(10); //产生0-9 随机数
```

****

chapter 3 file i/o

1.File类

1>导包:java.io.*;

2>创建file对象

```
File file=new File("c:/a.text");       //     ("c:\\a.text")  转义字符
```

\t 制表位

3>常用方法

```
exists();       //判断文件或目录是否存在
isFile();       //判断是否是文件
isDirectory();  //判断是否是目录
getPath();      //得到相对路径
getAbsolute();  //得到绝对路径
getName();      //得到文件或目录名称
delete();       //删除文件或目录
```

createNewFile();//创建空文件,不能创建文件夹

length();　　//得到文件的长度　单位为字节　否则,01

2.IO流分类

输入流　InputStream　Reader

流向

输出流　OutputStream　Writer

字节流　InputStream　OutputStream

单元

字符流　Reader　　Writer

全是抽象类

3.字节流读取文本文件

1>导包

```
import java.io.*;
```

2>创建文件流对象

```
FileInputStream fis=new FileInputStream("F:/a.txt");
```

3>读取文本文件

fis.available();　　//文件内容的大小　　一个中文　　一个英文　　一个换行

fis.read();　　//读取一个字节 类型的ASCII码值

```
byte[]  b= new  byte[1024];  //创建一个字节数组
fis.read(b);                 //将数据读取到数组去
```

```
int data;
```

//循环读数据

```
while((data=fis.read())!=-1){
        System.out.print( (char)data+"" );
}
```

4>关闭流对象

```
fis.close();
```

4.字节流书写文本文件

1>导包

```
import java.io.*;
```

2>创建文件流对象　　3种方式

```
FileOutputStream fos=new FileOutputStream("F:/b.txt");          //覆盖
FileOutputStream fos=new FileOutputStream("F:/b.txt",true);     //true表示追加    默认覆盖
FileOutputStream fos=new FileOutputStream(new File("F:/b.txt"));//file对象作参
```

3>读取文本文件

```
String str="死亡如风hasaki";    //要写的内容

byte[]  b=str.getBytes[];       //转换为字节数组

fos.write(b,0,b.length);        //写
```

4>关闭流对象

```
fos.close();
```

where:可完成文本文件得拷贝复制

5.字符流读取文本文件

1>导包

```
import java.io.*;
```

2>创建文件流对象

```
FileReader fr=new FileReader("F:/a.txt");
```

3>读取文本文件

```
fr.read();                //读取单个字符

char[] a=new char[1024]; //字符数组中转站

int length=fr.read(ch);
StringBuffer sb=new StringBuffer();

while( length!=-1){
        sb.append(ch) ;
        length=fr.read(ch);
}

System.out.println(sb.toString);
```

4>关闭流对象

```
fr.close();
```

BufferedReader 是Reader的子类 带有缓冲区

6.字符流读取文本文件 BufferedReader 是Reader的子类 带有缓冲区

步骤：

1>导包

```
import java.io.*;
```

2>创建流对象

```
FileReader fr=new FileReader("F:/a.txt");
BufferedReader br=new BufferedReader(fr);
```

3>读取

```
String str=br.readLine();
```

```
while(str=!null){
        System.out.println(str);
        str=br.readLine();
}
```

4>关闭

```
br.colse();
fr.colse();
```

7.字符流读取文本文件

1>导包

```
import java.io.*;
```

2>创建文件流对象

```
FileWriter fw=new FileWriter("F:/a.txt");
```

3>读取文本文件

```
fw.write("热爱我的热爱");
fw.flush();              //刷新缓冲区
```

4>关闭流对象

```
fw.close();
```

6.字符流读写文本文件 BufferedWriter 是Writer的子类

步骤:

1>导包

```
import java.io.*;
```

2>创建流对象

```
FileWriter fw=new FileWriter("F:/a.txt");
BufferedWriter bw=new BufferedWriter(fw);
```

3>读取

```
bw.
```

```
while(str=!null){
        System.out.println(str);
        str=br.readLine();
}
```

4>关闭

```
br.colse();
fr.colse();
```

7.二进制文件的读写　.class文件

1>导包

2>创建流对象

```
//创建输出流对象
        FileInputStream fis = new
FileInputStream("D:\\myDoc\\FileCopy.class");
        DataInputStream dis = new DataInputStream(fis);
//创建输入流对象
        FileOutputStream outFile = new
FileOutputStream("D:\\myDoc\\temp.class");
        DataOutputStream out = new DataOutputStream(outFile);
```

3>读写

```
int temp;
//读取文件并写入文件
while ( (temp = dis.read()) != -1) {
        out.write(temp);
}
```

4>关流

```
out.close();
dis.close();
```

8.序列化和反序列化

1>概念:序列化就是将对象的状态存储到特定存储介质中的过程中;

所存储的对象要实现序列化　implements　Serializable;

2>步骤: a: 导包

b: 创建流对象 ObjectOutputStream oos=new ObjectOutputStream(new FileOutputStream("f:stu.txt"));

c: 写入 Student stu1=new Student("张三",20);
oos.write(stu);

d: 关闭 oos.close();

3>概念: 反序列化是将存储在中的数据重新构建为对象的过程.

4>步骤: a: 导包

b: 创建流对象 ObjectInputStream ois=new ObjectInputStream(new FileInputStream("f:stu.txt"));

c: 读取 Student stu1=(Student) ois.readObject();

d: 关闭 ois.close();

****

chapter 4 多线程

1.进程 是程序的一次动态执行过程

2.线程 是进程中执行运算的最小单位 ,可完成一个独立的顺序控制流程

3.多线程的好处 充分利用Cpu资源 简化编程模型 良好的用户体验

4.常用方法: Thread.currentThread(); //获得当前线程
Thread.currentThread().getName(); //获得当前线程的名字
t1.setName(); //设置线程的名字
main()是主线程的入口
用于创建子线程 和 最后完成执行

5.实现多线程的方式

1>继承Thread类创建线程

2>实现Runnable接口创建线程

6.继承Thread类创建线程

1>:继承　　public class MyThread extends Thread{}

2>:重写run()方法

```
public void run(){
    for(int i=1;i<100;i++){

System.out.println(Thread.currentThread().getName()+":"+i);
    }
}
```

3>:创建线程对象　　MyThread my=new MyThread();

4>:运行　　my.start();

7.实现Runnable接口创建线程

1>:实现　　class MyRunnable implements Runnable{

2>:实现run()方法

```
public void run(){
    for(int i=1;i<100;i++){

System.out.println(Thread.currentThread().getName()+":"+i);
    }
}
```

3>:创建线程对象

```
MyRunnable my=new MyRunnable();
Thread  th=new Thread(my);
```

4>:运行　th.start();

8.线程的状态

创建　就绪　运行　阻塞　死亡

9.线程的调度

1>优先级　Priority　　0-10　　默认5

设置优先级　t1.setPriority(Thread.MAX.PRIORITY);

`　　t1.setPriority(Thread.MIN.PRIORITY);

t1.setPriority(10);

t1.setPriority(0);

获得优先级　t1.getPriority();

2>休眠 sleep();　　毫秒值　　1秒=1000毫秒;　　处理异常

3>强制执行　join();

调用者强制执行

4>礼让　yield();

调用者暂停执行　处于就绪状态　　允许具有相同优先级的线程 仅仅提供一种可能

10.线程的同步 synchronized

when:多线程共享数据引发的问题　买票

how：同步代码块　　和　　同步方法

1.同步方法

```
public synchronized void  sale(){}
```

1.同步代码块

```
public void run(){
        synchronized(this){
        }
}
```

11.线程安全

ArrayList　非线程安全

Hashtable　线程安全

HashMap　　线程非安全　重速度

StringBuffer　线程安全
StringBuilder　　线程非安全　重速度
****
chapter 5　网络编程

1.ip地址

唯一标识网络中的一台计算机　　eg:192.168.1.1　　0-255
ip地址=网络标识+主机地址
网络地址:标识计算机或网络设备所在的网段
主机地址:标识特定主机或网络设备
A类:　网络 主机 主机 主机　　1-126
B类:　网络 网络 主机 主机　　128-191
C类:　网络 网络 网络 主机　　192-233
D类:　通信　　224-239
E类:　科研　　240-255

0.0.0.0　　本机
127.0.0.1　本机回环地址　　用于检测
255.255.255.255　当前子网　广播信息

2.DNS域名系统　Domain Name　System

端口 port 计算机与外界通讯交流的出口　0-65535　2^(16-1)

3.服务器

1>概念:在网络环境下,具有较高计算能力,能够提供用户服务功能的计算机
2>邮件服务器　收发邮件　　遵守协议
3>web服务器　　网上信息浏览服务
Microsoft IIS
APACHE　　开源
Apache Tomcat　开源

4.网络通信协议

1>为在网络中不同的计算机之间进行通信而建立的规则.标准或约定的集合

2>Tcp协议　Transmission Control Protocol　传输控制协议
面向连接　可靠　集与字节流的传输通信协议

3>UDP　(user datagram protocol) 用户数据包协议
无连接的协议　在传输数据之前,客户端与服务端并不建立和维护连接

4.Socket　(插座)　通信链路的端点 套接字

5.TCP socket

客户端:

```
//1.打开socket　建立连接　指定服务器 和端口:8800
Socket socket=new Socket("192.168.25.1",8800);

//2.打开流
OutputStream os=socket.getOutputStream();
InputStream is=socket.getInputStream();


//3.发送信息
String info="登录名:焦冲冲,密码:123456";
os.write(info.getBytes());
System.out.println("发送成功");
socket.shutdownOutput();

//4.接收服务器的回应
BufferedReader br=new BufferedReader(new InputStreamReader(is));
info="";
while((info=br.readLine())!=null){
System.out.println("我是客户端，服务器的响应是："+info);
}
```

```
System.out.println("接收完毕");

//5.关闭资源
br.close();
is.close();
os.close();
socket.close();
```

服务端:

```
//1.创建一个ServerSocket对象 指定端口号
ServerSocket serversocket = new ServerSocket(8800);

//2.监听客户端请求
Socket socket = serversocket.accept();

//3.打开输入流，处理用户请求
InputStream is = socket.getInputStream();
OutputStream os = socket.getOutputStream();
BufferedReader reader = new BufferedReader(new InputStreamReader(is));
String info = null;
while((info=reader.readLine())!=null){
        System.out.println("我是服务器，客户端信息为："+info);
}
socket.shutdownInput();

//4.服务器给客户端一个响应
String reply = "欢迎您，登录成功！";
os.write(reply.getBytes());
System.out.println("服务端回复完毕");
//5.关闭资源
reader.close();
is.close();
os.close();
socket.close();
```

```
serversocket.close();
```

注意:先打开服务器端　再运行客户端

6.多用户访问　多线程　主线程负责监听　子线程负责回复

```
public class SocketThread extends Thread{
Socket socket=null;
public SocketThread(Socket socket){
        this.socket=socket;
}

public void run(){
        //3.打开输入流，处理用户请求
        InputStream is;
        try {
                is = socket.getInputStream();
                OutputStream os = socket.getOutputStream();
                BufferedReader reader = new BufferedReader(new
InputStreamReader(is));
                ObjectInputStream ois=new ObjectInputStream(is);
                User stu=(User)(ois.readObject());


                System.out.println("我是服务器，客户端信息为：");
                stu.print();
                socket.shutdownInput();
                //服务器给客户端一个响应

                String reply = "欢迎您，登录成功！";
                os.write(reply.getBytes());
                System.out.println("服务端回复完毕******");
                //4.关闭资源
                reader.close();
                is.close();
                os.close();
```

```
                socket.close();



        //1.创建一个ServerSocket对象 指定端口号
        ServerSocket serversocket = new ServerSocket(8800);
        //2.监听客户端请求
        Socket socket=null;
        while(true){
        socket= serversocket.accept();
        SocketThread st=new SocketThread(socket);
        st.start();
```

7.InetAddress类

```
InetAddress ia= InetAddress.getLocalHost();
InetAddress[] adds=InetAddress.getAllByName("www.alibaba.com");
InetAddress add=InetAddress.getByName("www.baidu.com");
```

8.UDP socket

```
        发送方:
        InetAddress add = null;
        String mess = "你这个bitch";

        // 1.获取本地主机地址
        add = InetAddress.getByName("localhost");

        // 2.创建DatagramPacket对象 封装数据
        DatagramPacket dp = new
DatagramPacket(mess.getBytes(),mess.getBytes().length, add, 8000);

        // 3.创建DatagramSocket 发送数据
        DatagramSocket ds = new DatagramSocket();
        ds.send(dp);

        //接收响应
```

```
        byte[] buf=new byte[1024];
        DatagramPacket pack=new DatagramPacket(buf, buf.length);
        ds.receive(pack);
        //显示接收到的信息
        String reply=new String(pack.getData(),0,pack.getLength());
        System.out.println(pack.getAddress().getHostAddress()+"说:"+reply);
        ds.close();
```

.....

```
        发送方:
        // 1.创建DatagramPacket对象,准备接收数据
        byte[] buf = new byte[1024];
        DatagramPacket pack = new DatagramPacket(buf, 1024);

        //2.创建DatagramSocket对象,接收数据并保存到pack中
        DatagramSocket ds = new DatagramSocket(8000);
        ds.receive(pack);

        // 3.显示接收到的信息
        String reply = new String(pack.getData(), 0, pack.getLength());
        System.out.println(pack.getAddress().getHostAddress() + "说:" + reply);

        String mess = "你才是个bitch";
        System.out.println("我说:" + mess);
        SocketAddress sa = pack.getSocketAddress();
        DatagramPacket packto = new
DatagramPacket(mess.getBytes(),mess.getBytes().length, sa);

        ds.send(packto);
```

****

chapter 6 xml

1.xml Extensible Markup Language 可扩展标记语言

规范统一 与操作系统无关 常用来做配置文件和数据交互

2.文档结构

1>xml声明

2>标签

3>根元素　包含所有其他元素

4>元素

命名规则:

1不能数字和标点符号开头

2可以包含字母 数字和其他字符

3.不能以xml开始

4.不能包含空格

5>属性　<元素　　属性名="属性值" 属性名="属性值">　起始标签

6>特殊符号　　转义字符　　<![CDATA[字符]]>

7.注释　<!--　　-->

8.注意:声明　根元素　　大小写　"属性值"　成对　嵌套

3.优势

数据存储

数据交换

数据配置

4.解析xml

DOM　　基于xml文档树结构的解析　　适用多次访问的xml文档　　比较消耗资源

SAX　　基于事件的解析　　适用于大文档　　占用资源少

Dom4J　性能优异　功能强大　　开放源代码

5.DOM解析

```
//得到Dom解析器的工厂实例
DocumentBuilderFactory dbf=DocumentBuilderFactory.newInstance();
//从dom工厂获得Dom解析器
DocumentBuilder  db=dbf.newDocumentBuilder();
//解析xml文档  得到一个document对象
Document document=db.parse("收藏信息.xml");
/得到所有brand节点列表信息
NodeList brandlist=document.getElementsByTagName("Brand");

//循环brand信息
for (int i = 0; i < brandlist.getLength(); i++) {
//获取第i个Brand元素
Node brand=brandlist.item(i);

//获得Brand元素的name属性的值
Element element=(Element) brand;
String attrValue=element.getAttribute("name");

//获得Brand  所有子元素的name 属性值
NodeList types=element.getChildNodes();

for (int j = 0; j < types.getLength(); j++) {
        Node type=types.item(j);

System.out.println("名称:"+type.getNodeName());
if (type.getNodeType()==Node.ELEMENT_NODE) {
        Element typesElement = (Element)type;
        String typeStr=typesElement.getAttribute("name");
        //System.out.println("手机:"+attrValue+typeStr);
        }
```

```
Document document=null;
public static void main(String[] args) {
        Dom4jParse1 parse=new Dom4jParse1();
        parse.loadDocument();
        parse.addElement();
        //parse.editElement();
        parse.delElement();
        parse.showElement();

}

/**
 * 获得document 对象
 */
public void loadDocument(){
        SAXReader reader=new SAXReader();
        try {
                document=reader.read("新的收藏信息.xml");
        } catch (DocumentException e) {
                e.printStackTrace();
        }
}

/**
 * 显示信息
 */
public void  showElement(){
        Element root=document.getRootElement();
        Iterator brandit=root.elementIterator();
        while (brandit.hasNext()) {
                Element brandElement=(Element) brandit.next();
                String brandStr=brandElement.attributeValue("name");

                Iterator typeIt=brandElement.elementIterator();
                while (typeIt.hasNext()) {
```

```
                                Element typeElement=(Element) typeIt.next();
                                String
typeStr=typeElement.attributeValue("name");
                                System.out.println("手机:"+brandStr+typeStr);
                        }
                }
        }

        /**
         * 添加
         */
        public void addElement(){
                Element root=document.getRootElement();
                Element newEle=root.addElement("Brand");
                newEle.addAttribute("name","小米");

                Element newEleType=newEle.addElement("Type");
                newEleType.addAttribute("name", "mi6");

                //保存
                save();

        }

        /**
         * 修改
         * 给brand添加  id 属性
         */
        public void editElement(){
                Element root=document.getRootElement();
                int id=0;
                Iterator brandit=root.elementIterator();
                while (brandit.hasNext()) {
                        id++;
                        Element brandEle=(Element) brandit.next();
```

```
                brandEle.addAttribute("id", id+"");
            }
            //保存
            save();
        }

        /**
        *删除
        */
        public void delElement(){
            Element root=document.getRootElement();

            Iterator brandit=root.elementIterator();
            while (brandit.hasNext()) {

                Element brandEle=(Element) brandit.next();
                if (brandEle.attributeValue("name").equals("小米")) {
                    brandEle.getParent().remove(brandEle);
                }
            }
            //保存
            save();
        }

        /**
         * 保存
         */
        public void save(){
            OutputFormat format = OutputFormat.createPrettyPrint();
            format.setEncoding("gb2312");
            try {
                XMLWriter writer = new XMLWriter(new FileWriter("新的收
藏信息.xml"),format);
                writer.write(document);
                writer.close();
```

```
        } catch (IOException e) {
                e.printStackTrace();
        }

}
```

6.DOM4J解析

```
Dom4jParse1 parse=new Dom4jParse1();
        parse.loadDocument();
        parse.addElement();
        //parse.editElement();
        parse.delElement();
        parse.showElement();



/**
 * 获得document 对象
 */
public void loadDocument(){
        SAXReader reader=new SAXReader();
        try {
                document=reader.read("新的收藏信息.xml");
        } catch (DocumentException e) {
                e.printStackTrace();
        }
}

/**
 * 显示信息
 */
public void  showElement(){
        Element root=document.getRootElement();
        Iterator brandit=root.elementIterator();
```

```
                while (brandit.hasNext()) {
                        Element brandElement=(Element) brandit.next();
                        String brandStr=brandElement.attributeValue("name");

                        Iterator typeIt=brandElement.elementIterator();
                        while (typeIt.hasNext()) {
                                Element typeElement=(Element) typeIt.next();
                                String
typeStr=typeElement.attributeValue("name");
                                System.out.println("手机:"+brandStr+typeStr);
                        }
                }
        }

        /**
         * 添加
         */
        public void addElement(){
                Element root=document.getRootElement();
                Element newEle=root.addElement("Brand");
                newEle.addAttribute("name","小米");

                Element newEleType=newEle.addElement("Type");
                newEleType.addAttribute("name", "mi6");

                //保存
                save();

        }

        /**
         * 修改
         * 给brand添加  id 属性
         */
        public void editElement(){
```

```
        Element root=document.getRootElement();
        int id=0;
        Iterator brandit=root.elementIterator();
        while (brandit.hasNext()) {
                id++;
                Element brandEle=(Element) brandit.next();
                brandEle.addAttribute("id", id+"");
        }
        //保存
        save();
}


public void delElement(){
        Element root=document.getRootElement();

        Iterator brandit=root.elementIterator();
        while (brandit.hasNext()) {

                Element brandEle=(Element) brandit.next();
                if (brandEle.attributeValue("name").equals("小米")) {
                        brandEle.getParent().remove(brandEle);
                }
        }
        //保存
        save();
}

/**
 * 保存
 */
public void save(){
        OutputFormat format = OutputFormat.createPrettyPrint();
        format.setEncoding("gb2312");
        try {
```

```
                        XMLWriter writer = new XMLWriter(new FileWriter("新的收
藏信息.xml"),format);
                        writer.write(document);
                        writer.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }
```

****

chapter 7　项目案例