知识点列表

编号	名称	描述	级别
1	字符串的序列化(文字的编码方案)	理解字符串序列化过程中的编码问题，认识并理解常见编码规范	*
2	认识文本和文本文件	理解文本和文件	*
3	字符流的基本实现 Reader Writer	理解并掌握字符流的用法，以及字符流和字节流的区别	*
4	字符流的过滤器 BufferedReader	重点掌握过滤流的用法，理解其原理	***
5	字符流的过滤器 PrintWriter	重点掌握过滤流的用法，理解其原理	***
6	对象的序列化	理解并掌握对象序列化和反序列化的原理和编程步骤	**
7	浅层复制与深层复制	能够理解浅层复制与深层复制的内存原理	*
8	流的过滤器	对流的过滤器有初步认识，能够在使用时通过学习 API 帮助文档灵活应用	*

注： "*"理解级别 "**"掌握级别 "***"应用级别

目录

1. 字符串的序列化（文字的编码方案）* 3

2. 认识文本和文本文件 * 6

3. 字符流(Reader Writer) 6

3.1. 字符流的基本实现 Reader Writer * 6

3.2. 字符流的过滤器 BufferedReader *** 9

3.3. 字符流的过滤器 PrintWriter *** 10

4. 对象的序列化 ** 11

5. 浅层复制与深层复制 * 13

6. 流的过滤器 * 14

1. 字符串的序列化（文字的编码方案）*

1) Stirng 字符串本质上是 char[]
   - 将 char[] 转换成 byte 序列，就是字符串的编码，就是字符串的序列化问题
   - char 类型是 16 位无符号整数，值是 unicode 编码
2) **UTF-16BE** 编码方案
   - UTF-16BE 编码方案，是将 16 位 char 从中间切开为 2 个 byte
   - UTF-16BE 是将 unicode 编码的 char[] 序列化为 byte[] 的编码方案

     如: char[] = ['A','B','中']

     byte[] = [00, 41, 00, 42, 4e, 2d]
   - UTF-16BE 编码能够支持 65535 个字符编码
3) **UTF-8** 编码方案
   - 采用变长编码 1~N 方案，其中英文占 1 个 byte，中文占 3 个 byte
4) 较常用的编码
   - GBK 中国国标，支持 20000+中日英韩字符，英文 1 位编码，中文 2 位
     - 与 Unicode 编码不兼容，需要码表转换
     - GB2312 简体中文编码
   - ISO8859-1 西欧常用字符
   - UTF-8 Unicode 的一种边长字符编码，又称“万国码”
   - ……

【案例 1】编码方案演示

案例描述

输出“ABC 中”，比较采用不同编码 UTF-16BE、UTF-8、GBK 输出的结果

参考代码

- **IOUtils.java**

IOUtils.java

```
1 package corejava.day09.ch08;
2 /** IO 工具类 */
3 public class IOUtils {
4     public static void printHex(byte[] ary){
5         int i=1;
6         for (byte b : ary) {
7           //将byte转换为int，正数保留符号位不变,负数补1
8           int x = b;
9           x &= 0xff;  //即x = x & 0xff
```

```
10              if(x<=0xf){
11                System.out.print("0");
12              }
13              System.out.print(Integer.toHexString(x)+" ");
14              if(i++%10==0)
15                System.out.println();
16            }
17            System.out.println();
18          }
19  }
```

注：

✓ **掩码（mask） 0xff**

int x = 0xfffffffd； //负数

x = x & 0xff； //与掩码做运算

一个负数与掩码 **0xff** 进行**按位与运算**（即不进位的乘运算，不同得 0，相同得 1）后，得到了新的操作数 **0xfd**

按位与运算 &	11111111 11111111 11111111 11111101	0xfffffffd
	00000000 00000000 00000000 11111111	0xff 掩码（mask）
	00000000 00000000 00000000 11111101	0xfd

- **EncodingDemo.java**

EncodingDemo.java

```
1  package corejava.day09.ch08;
2  import java.util.Arrays;
3  /** 编码方案演示 */
4  public class EncodingDemo {
5    public static void main(String[] args)
6      throws Exception{
7      String s = "AB中";
8      byte[] utf16be = s.getBytes("utf-16be");
9      System.out.println(Arrays.toString(utf16be));
10     IOUtils.printHex(utf16be);
11     byte[] utf8 = s.getBytes("utf-8");
12     IOUtils.printHex(utf8);
13     byte[] gbk = s.getBytes("gbk");
14     IOUtils.printHex(gbk);
15
16     //将utf-8编码的byte[]转换为字符串
17     String str = new String(utf8 ,"utf-8");
18     System.out.println(str);
19   }
20 }
```

```
Console
<terminated> EncodingDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Nov 10, 2011 3:
[0, 65, 0, 66, 78, 45]
00 41 00 42 4e 2d
41 42 e4 b8 ad
41 42 d6 d0
AB中
```

【案例 2】写入文本和读取文本的编码问题

BasicCharIODemo.java

```
1  package corejava.day09.ch09;
2  import java.io.FileOutputStream;
3  import java.io.IOException;
4  /** 字符串 IO */
5  public class BasicCharIODemo {
6    public static void main(String[] args)
7      throws IOException{
8      //写出文本
9      String file = "demo/str.txt";
10     FileOutputStream out = new FileOutputStream(file);
11     String str = "abcde中华人民";
12     byte[] utf8 = str.getBytes("utf-8");
13     out.write(utf8);
14     out.close();
15     IOUtils.printHex(file);
16     //读取文本
17     byte[] buf = IOUtils.read(file);
18     String s = new String(buf, "utf-8");
19     String s2 = new String(buf, "gbk");
20     System.out.println(s);
21     System.out.println(s2); //乱码
22   }
23 }
```

```
Console
<terminated> BasicCharIODemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Nov 10, 2011
61 62 63 64 65 e4 b8 ad e5 8d
8e e4 ba ba e6 b0 91
abcde中华人民
abcde涓    崖浜烘皯
```

2. 认识文本和文本文件 *

1) java 的**文本**（char）是 16 位无符号整数，是字符的 unicode 编码
2) 文件是 byte by byte … 的数据序列
3) **文本文件**是文本（char）序列按照某种编码方案（utf-8, utf-16be, gbk）序列化为 byte 的存储结果

3. 字符流(Reader Writer)

1) 字符的处理，一次处理一个字符（unicode 编码）
2) 字符的底层仍然是基本的字节流
3) 字符流的基本实现
   - ✓ **InputStreamReader** 完成 byte 流解析为 char 流，按照编码解析
   - ✓ **OutputStreamWriter** 提供 char 流到 byte 流，按照编码处理
4) 字符流的**过滤器**
   是字符读写的功能扩展，极大的方便了文本的读写操作
   BufferedReader：**readLine**() 一次读取一行
   PrintWriter: println() 一次打印一行
5) 读取一个文本文件

```
InputStream is = new FileInputStream("gbk.txt");
Reader in = new InputStreamReader(is);
BufferedReader reader = new BufferedReader(in);
or
BufferedReader in = new BufferedReader(new FileReader(filename));
```

6) 写出一个文本文件

```
PrintWriter out = new PrintWtirer(new FileWriter(filename));
or
PrintWriter out = new PrintWtirer(
                    new OutputStreamWriter(
                        new FileOutputStream(filename)));
```

7) 系统的默认编码，中文一般是 GBK

```
String encoding=System.getProperty("file.encoding");
```

3.1. 字符流的基本实现 Reader Writer *

【案例 1】Reader Writer 演示

ReaderWriterDemo.java

```
1  package corejava.day10.ch01;
2  import java.io.FileInputStream;
7  /** Reader Writer Demo */
8  public class ReaderWriterDemo {
9    public static void main(String[] args)
10     throws IOException{
11     //1. 写入文本
12     String file = "demo/reader_writer.txt";
13     FileOutputStream fos = new FileOutputStream(file);
14     //过滤流 将字符按照utf-8编码处理
15     OutputStreamWriter out=
16                 new OutputStreamWriter(fos, "utf-8");
17
18     out.write("abcde中国好,大家好"); //写一个字符串
19     out.write('好');//写一个字符
20     out.close();
21
22     IOUtils.printHex(file); //验证结果
23
24     //2. 文本读取
25     FileInputStream fis = new FileInputStream(file);
26     InputStreamReader in =
27                 new InputStreamReader(fis, "utf-8");
28     int c;
29     //in.read() 读取一个字符, 无符号填充到c的低16位
30     while((c=in.read())!=-1){
31       System.out.print((char)c);
32     }
33     in.close();
34   }
35 }
```

【案例 2】统计小说的字符

CharCountDemo.java

```
1  package corejava.day10.ch02;
2  import java.io.FileInputStream;
12
13 /** 统计小说的字符 */
14 public class CharCountDemo {
15   public static void main(String[] args)
16     throws IOException{
17
```

```
18      String encoding = "utf-8";
19      String file = "demo/山楂树之恋.txt";
20      //1. 打开文件
21      FileInputStream fis = new FileInputStream(file);
22      //2. 转换为指定的编码
23      InputStreamReader in =
24        new InputStreamReader(fis, encoding);
25
26      //3. 定义一个Map，存放字符、统计数据
27      Map<Character, Integer> map =
28        new HashMap<Character, Integer>();
29      int c;
30      int all = 0;//全部字符数量
31      //3.1 迭代 统计字符
32      while((c=in.read())!=-1){
33        //3.1.1 强转为字符
34        char ch = (char)c;
35        //3.1.2 忽略掉标点符号：tab、空格、\r、\n、逗号
36        if(ch=='\t' || ch==' ' || ch=='\r' ||
37                                  ch=='\n' || ch ==', '){
38          continue;
39        }
40        //3.1.3 获得该字符已统计的数据
41        Integer count = map.get(ch);
42        //3.1.4 如果该字符从没统计过count记为1，否则+1
43        map.put(ch, count==null?1:count+1);
44        //3.1.5  全部统计字符数量+1
45        all++;
46      }
47      in.close();
48
49      //4. 排序统计结果
50      //4.1 将获得的Map键值对(Entry)，放入ArrayList
51      ArrayList<Entry<Character, Integer>> list =
52              new ArrayList
53                  <Entry<Character,Integer>>(map.entrySet());
54      //4.2 指定排序规则：按键值对(Entry)的value排序
55      Collections.sort(
56          list, new Comparator<Entry<Character, Integer>>(){
57            public int compare(Entry<Character, Integer>o1,
58                      Entry<Character, Integer> o2) {
59              return o2.getValue()-o1.getValue();
60            }
61          });
62
```

```
63      //5. 输出统计结果
64      System.out.println("文字数量:" + list.size());
65      System.out.println("字数:" + all);
66      int i=1;
67      //格式化
68      DecimalFormat fmt = new DecimalFormat("00.##%");
69      for (Entry<Character, Integer> entry : list) {
70        System.out.println(
71                entry.getKey() + "\t " +
72                entry.getValue() + "\t" +
73                fmt.format(entry.getValue()/(double)all));
74        //只统计list集合中前3个字符
75        if(i++==3)
76          break;
77      }
78    }
79  }
```

```
Console
<terminated> CharCountDemo (1) [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Nov 10, 2011 5:13
文字数量:2147
字数:67550
的        2077    03.07%
        1674    02.48%
。        1577    02.33%
```

注：

✓ **中文的空格**没有在程序中做处理

3.2. 字符流的过滤器 BufferedReader ***

【案例】读取文本文件（标准文本文件读取方式）

TextFileReadDemo.java

```
1 package corejava.day10.ch02;
2 import java.io.BufferedReader;
6
7 /** 文本文件读取 */
8 public class TextFileReadDemo {
9   public static void main(String[] args)
10    throws IOException{
11    String encoding = "gbk";
12    String file = "src/corejava/day10/java_day10.txt";
13
14    FileInputStream fis = new FileInputStream(file);
```

```
14      FileInputStream fis = new FileInputStream(file);
15      InputStreamReader reader =
16        new InputStreamReader(fis, encoding);
17      BufferedReader in = new BufferedReader(reader);
18
19      String str;
20      //in.readLine()返回字符串对象，如果到文件末尾返回null
21      while((str=in.readLine())!=null){//读取一行
22        System.out.println(str);
23      }
24
25      in.close();
26    }
```

3.3. 字符流的过滤器 PrintWriter ***

【案例】写一个文本文件

TextWriterDemo.java

```
1  package corejava.day10.ch02;
2  import java.io.FileOutputStream;
6
7  /** 写一个文本文件 */
8  public class TextWriterDemo {
9    public static void main(String[] args)
10     throws IOException {
11     String file = "demo/out.txt";
12     String encoding = "gbk";
13
14     FileOutputStream fos= new FileOutputStream(file);
15     OutputStreamWriter osw =
16       new OutputStreamWriter(fos, encoding);
17     PrintWriter out = new PrintWriter(osw);
18
19     //println()是PW提供的方法,可以向流里写入一行
20     out.println("HI,"); //向目标流写一行文本
21     out.println("  I want to kill u!");
22     out.println("我是Jerry!");
23     out.close();
24     IOUtils.printHex(file);
25   }
26 }
```

```
Console
<terminated> TextWriterDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Nov 11, 2011
48 49 2c 0d 0a 20 20 49 20 77
61 6e 74 20 74 6f 20 6b 69 6c
6c 20 75 21 0d 0a ce d2 ca c7
4a 65 72 72 79 21 0d 0a
```

注：

- ✓ 控制台输出结果中“0d 0a”表示回车（Windows 操作系统），了解即可
  - Linux 操作系统下“0d”表示回车
  - Mac OS 下“0a”表示回车

4. 对象的序列化 **

对象序列化，就是将 Object 转换为 byte 序列，反之叫对象的反序列化。

1) 序列化流（**ObjectOutputStream**），是过滤流
   - ObjectOutputStream **writeObject**(Object)　序列化对象
   - ObjectInputStream **readObject**()　对象的反序列化
2) **序列化接口（Serializable）**
   - 对象必须实现“序列化接口”才能进行序列化，否则将出现不能序列化的异常！
   - Serializable 是一个空的接口，没有任何方法，仅作为序列化的一个标识
3) **JavaBean 规范**规定，Java 类必须实现 Serializable 接口
   Java API 中的类大多是符合 Java Bean 规范的，基本都实现了 Serializable
4) 对象的序列化和反序列化可以变相实现**对象的深层复制**

【案例 1】对象序列化和反序列化演示

ObjectIODemo.java

```
1  package corejava.day10.ch03;
2  import java.io.FileInputStream;
7
8  /** 对象IO */
9  public class ObjectIODemo {
10   public static void main(String[] args)
11     throws Exception{
12     String file = "demo/obj.dat";
13     //1. 对象序列化
14     FileOutputStream fos = new FileOutputStream(file);
15     ObjectOutputStream out =
16       new ObjectOutputStream(fos);
```

```
17
18      Foo foo= new Foo();
19      Foo f2= new Foo();
20      out.writeObject(foo);//序列化
21      out.writeObject(f2);//将对象写入到流中
22
23      out.close();
24      IOUtils.printHex(file);
25
26      //2. 对象的反序列化
27      FileInputStream fis = new FileInputStream(file);
28      ObjectInputStream in = new ObjectInputStream(fis);
29      Foo f = (Foo)in.readObject(); //反序列化
30      Foo ff = (Foo)in.readObject();
31      System.out.println(f.a + "," + ff.a);
32      //f是foo的深层复制
33      System.out.println(f==foo);//false
34      in.close();
35    }
36  }
37  class Foo implements Serializable{
38    int a = 1;
39  }
```

【案例 2】对象序列化演示 2

- **Card.java**

Card.java

```
1  package corejava.day10.ch04;
2  import java.io.Serializable;
3  /** 扑克牌 Card extends Object
4   *  Card是东西的一种 啥都是东西
5   */
6  public class Card implements Serializable{
7    /** 花色 */
8    private int suit;//实例变量，每个扑克牌实例都有自己的
9    /** 点数 0代表3，1代表4，2代表5 */
10   private int rank;//实例变量，每个扑克牌实例都有自己的
```

- **CardIODemo.java**

CardIODemo.java

```
1  package corejava.day10.ch04;
2  import java.io.FileInputStream;
9  /** 对象序列化 2 */
10 public class CardIODemo {
11   public static void main(String[] args)
12     throws Exception{
13     //1. 构造一副牌
14     List<Card> cards = new ArrayList<Card>();
15     for(int rank=Card.THREE;rank<=Card.DEUCE; rank++){
16       cards.add(new Card(Card.DIAMOND, rank));
17       cards.add(new Card(Card.CLUB, rank));
18       cards.add(new Card(Card.HEART, rank));
19       cards.add(new Card(Card.SPADE, rank));
20     }
21     cards.add(new Card(Card.JOKER, Card.BLACK));
22     cards.add(new Card(Card.JOKER, Card.COLOR));
23     Collections.shuffle(cards);
24
25     //2. Card对象序列化
26     String file = "demo/cards.obj";
27     FileOutputStream fos =
28                     new FileOutputStream(file);
29     ObjectOutputStream out =
30                     new ObjectOutputStream(fos);
31     out.writeObject(cards);
32     out.close();
33     IOUtils.printHex(file);
34
35     //3. Card对象反序列化
36     FileInputStream fis = new FileInputStream(file);
37     ObjectInputStream in = new ObjectInputStream(fis);
38     ArrayList<Card> others =
39       (ArrayList<Card>) in.readObject();
40     System.out.println(others);
41     System.out.println(cards);
42     System.out.println(others==cards);//false 深层复制
43     System.out.println(
44         others.get(0)==cards.get(0));//false 深层复制
45   }
46 }
```

5. 浅层复制与深层复制 *

1) Java 的默认复制规则是浅层复制，性能好，但隔离性差
浅层复制现象，只复制第一层对象

2) 利用序列化可以实现深层复制

【案例】IOUtils 工具类提供深层复制方法

IOUtils.java

```
128
129   public static Object deepClone(Object obj){
130     try{
131       //1. 对象序列化
132       //缓冲流：字节数组输出流
133       ByteArrayOutputStream buf =
134                 new ByteArrayOutputStream();
135       //对象输出流
136       ObjectOutputStream out =
137                 new ObjectOutputStream(buf);
138       out.writeObject(obj);//序列化对象到buf中
139       out.close();
140
141       //2. 对象反序列化
142       byte[] ary = buf.toByteArray();
143       ByteArrayInputStream bais =
144                 new ByteArrayInputStream(ary);
145       ObjectInputStream in =
146                 new ObjectInputStream(bais);
147       Object o = in.readObject();//从ary反序列化
148       in.close();
149
150       return o;
151     }catch(Exception e){
152       e.printStackTrace();
153       throw new RuntimeException(e);
154     }
155   }
```

6. 流的过滤器 *

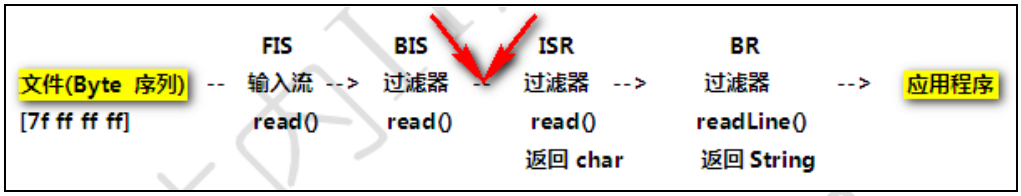
1) 流的过滤器是流的功能扩展，需要基于基本的流，更加方便的操作

2) 一类是基本 byte 数据的处理

- **BufferedInputStream**
- ZipInputStream（了解） 解压缩流，能够方便的读取 Zip 压缩文件
- JarInputStream（了解）
- CipherInputStream（了解） 解密流，提供了对输入流的加密和解密的方法

3) 一类是数据序列化功能扩展

- DataInputStream　　基本类型的 IO
- ObjectInputStream　　对象 IO
- **InputStreamReader**　　String(char[])



注：

✓ 过滤流有很多种类，根据具体业务逻辑，插入很多，比如解密流 ZipInputStream