目录

[Day14.    Java](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586520)

[练习1   文件加密复制](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586521)

[1     内部类](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586522)

[1.1       非静态内部类、属于实例的内部类](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586523)

[1.2       静态内部类](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586524)

[1.3       局部内部类](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586525)

[1.4       匿名内部类](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586526)

[练习2   内部类](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586527)

[2     for-each循环](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586528)

[练习3   手写双向链表](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586529)

[3     字符编码](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586530)

[3.1   ASC-II](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586531)

[3.2   iso-8859-1，Latin-1](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586532)

[3.3   CJK编码](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586533)

[3.4   GBK](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586534)

[3.5   Unicode](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586535)

[3.6   UTF-8](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586536)

[3.7   Java的char类型是 Unicode](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586537)

[练习4   编码转换](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586538)

[4     Reader，Writer](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586539)

[5     InputStreamReader,OutputStreamWriter](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586540)

[练习5   编码转换流](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586541)

[6     作业](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day14.htm#_Toc6586542)

**Day14.  Java**

**练习1  文件加密复制**

f1

---------------

10010101 11101101 01101110

---------------

f2

---------------

11011010 11001001 00100101

---------------

10010101

00011101 ^

10001000

项目：day1401\_文件加密复制

类：day1401.Test1

**package** day1401;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.util.Scanner;

**public** **class** Test1 {

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

    System.***out***.println("原文件：");

    String s1 = **new** Scanner(System.***in***).nextLine();

    File from = **new** File(s1);

**if**(! from.isFile()) {

        System.***out***.println("请输入正确的文件");

**return**;

    }

    System.***out***.println("目标文件：");

    String s2 = **new** Scanner(System.***in***).nextLine();

    File to = **new** File(s2);

**if** (to.isDirectory()) {

        System.***out***.println(

         "请具体到文件，不能是文件夹");

**return**;

    }

    System.***out***.print("KEY："); //0到255，超出255只有后面8位起作用

**int** key = **new** Scanner(System.***in***).nextInt();

**try** {

*copy*(from, to, key);

        System.***out***.println("完成");

    } **catch** (Exception e) {

        System.***out***.println("失败");

        e.printStackTrace();

    }

}

**private** **static** **void** copy(

        File from, File to, **int** key) **throws** Exception {

    FileInputStream in = **new** FileInputStream(from);

    FileOutputStream out = **new** FileOutputStream(to);

**byte**[] buff = **new** **byte**[8192];

**int** n;

**while**((n = in.read(buff)) != -1) {

        /\* 先把buff数组中的前n个字节值加密

         \* 加密之后再向目标文件输出 \*/

**for** (**int** i = 0; i < n; i++) {

           buff[i] ^= key;

        }

        out.write(buff, 0, n);

    }

    in.close();

    out.close();

}

}

**1   内部类**

定义在类内部、方法内部或局部代码块内部的类

**1.1     非静态内部类、属于实例的内部类**

class A {

class Inner {

}

}

非静态内部类实例，必须依赖于一个外部类的实例才能存在

A a1 = new A();

Inner i = a1.new Inner();

用来辅助外部实例运算，封装局部数据，或局部的运算逻辑

**1.2     静态内部类**

class A {

static class Inner {

}

}

Inner i = new Inner();

静态内部类，与普通的类没有区别

静态内部类，只是一个设计的选项，可以选择嵌套定义，也可以选择独立定义

**1.3     局部内部类**

class A {

Weapon f() {

    class Inner implements Weapon{

}

Inner i = new Inner();

return i;

}

}

A a = new A();

Weapon w = a.f();

局部定义的类型，类似于局部变量，有作用范围，只能在局部代码块内使用这种类型

**1.4     匿名内部类**

Weapon w = new Weapon() {...};

  {} - 匿名类

  new - 新建匿名类的实例

  Weapon - 父类型

  () - super()，可传参数super(1,2,3)

**练习2  内部类**

项目：day1402\_内部类

类：day1402.Test1

**package** day1402;

**import** day1402.A.Inner1;

**import** day1402.A.Inner2;

**public** **class** Test1 {

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

    Inner1 i1 = **new** Inner1();

    System.***out***.println(i1);

    A a = **new** A();

    Inner2 i2 = a.**new** Inner2();

    System.***out***.println(i2);

}

}

**class** A {

**static** **class** Inner1 {

}

**class** Inner2 {

}

}

Test2

**package** day1402;

**public** **class** Test2 {

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

    // 得到的是 Weapon 的子类实例

    Weapon w1 = *f1*();

    w1.kill();

    System.***out***.println(w1);

    System.***out***.println("--------------------");

    Weapon w2 = *f2*("方天画戟");

    w2.kill();

    System.***out***.println(w2);

}

**private** **static** Weapon f2(String name) {

    /\*

     \* 局部内部类中，使用外面的局部变量，

     \* 必须加 final，jdk1.8，缺省

     \*/

    Weapon w = **new** Weapon() {

        @Override

**public** **void** kill() {

           System.***out***.println(

            "使用"+name+"进攻");

        }

    };

**return** w;

}

**private** **static** Weapon f1() {

**class** AK47 **implements** Weapon {

        @Override

**public** **void** kill() {

           System.***out***.println(

            "使用AK47进攻");

        }

    }

    AK47 a = **new** AK47();

**return** a;

}

}

**2   for-each循环**

数组遍历、集合迭代遍历的语法简化

数组

for(int i=0; i<a.length; i++) {

String s = a[i];

//处理s

}

for-each 简化语法：

for(String s : a) {

//处理s

}

集合迭代遍历

for(Iterator<String> it=list.iterator(); it.hasNext(); ) {

String s = it.next();

//处理s

}

for-each 语法简化

for(String s : list) {

//处理

}

**练习3  手写双向链表**

模仿 LinkedList，实现双向链表数据结构

项目：day1403\_手写双向链表

类：day1403.Test1

             MyList

**package** day1403;

/\*

 \* {

 \*      {...}-{...}-{...}

 \* }

 \*

 \* {

 \*      123{ ... }

 \*

 \*      {

 \*         prev: 123,

 \*         value: 8,

 \*         next: 456

 \*      }

 \*

 \*      456{ ... }

 \* }

 \*/

**public** **class** MyList {

//首尾节点

**private** Node first; //null

**private** Node last;  //null

//数据的数量

**private** **int** size;   //数据的数量，默认0

**public** **void** add(Object value) {

    //如果添加的是第一个节点

    Node n = **new** Node();

    n.value = value;

    //

**if** (size == 0) {

        //添加第一个

        n.prev = n;

        n.next = n;

        //

        first = n;

        last = n;

    } **else** {

        //在尾部新加数据

        //n和last连接

        n.prev = last;

        last.next = n;

        //n和first连接

        n.next = first;

        first.prev = n;

        //新的节点n，变成尾部节点

        last = n;

    }

    size++;

}

**public** Object get(**int** i) {

    //越界

**if**(i<0 || i>=size) {

**throw**

**new** IndexOutOfBoundsException(""+i);

    }

    //访问 i 位置的节点对象

    Node n = getNode(i);

    //返回节点中封装的数据

**return** n.value;

}

**private** Node getNode(**int** i) {

**if** (i==0) {

**return** first;

    } **else** **if**(i==size-1){

**return** last;

    }

**if** (i<size/2) {//i<中间位置

        Node n = first;

        //从下标1到下标i

**for** (**int** j = 1; j <= i; j++) {

           n = n.next;

        }

**return** n;

    } **else** {

        Node n = last;

        //从size-2到i

**for** (**int** j = size-2; j >= i; j--) {

           n = n.prev;

        }

**return** n;

    }

}

**public** **int** size() {

**return** size;

}

//内部类，辅助双向链表，

//封装局部的数据：保存的值和前后引用

**private** **class** Node {

    Node prev;

    Node next;

    Object value;

}

}

Test2

**package** day1403;

**public** **class** Test1 {

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

    MyList list = **new** MyList();

    list.add("aaa");

    list.add("rrr");

    list.add("hhh");

    list.add("qqq");

    list.add("jjj");

    list.add("iii");

    list.add("ggg");

    System.***out***.println(list.size());

    System.***out***.println(list.get(0));

    System.***out***.println(list.get(list.size()-1));

    System.***out***.println(list.get(3));

**for** (**int** i = 0; i < list.size(); i++) {

        System.***out***.println(list.get(i));

    }

}

}

**3   字符编码**

**3.1     ASC-II**

0到 127，英文、指令字符

**3.2     iso-8859-1，Latin-1**

西欧编码

ASC-II扩展到255

**3.3     CJK编码**

亚洲编码，中日韩

**3.4     GBK**

国标码

英文单字节

中文双字节

**3.5     Unicode**

统一码，万国码

100万+

常用表，双字节

生僻字符表，三字节、四字节...

**3.6     UTF-8**

Unicode 的传输格式

Unicode Transformation format

英文，单字节

某些字符，双字节

中文，三字节

特殊符号，四字节

**3.7     Java的char类型是 Unicode**

Java的转码运算

Unicode ---> 其他编码

String s = "abc中文";

//转成系统默认编码

byte[] a = s.getBytes();

//转成指定编码

byte[] a = s.getBytes("UTF-8");

其他编码 --> Unicode

//从系统默认编码转成 Unicode

String s = new String(byte[]);

//从指定编码转成 Unicode

String s = new String(byte[], "UTF-8");

**练习4  编码转换**

项目：day1404\_编码转换

类：day1404.Test1

**package** day1404;

**import** java.io.UnsupportedEncodingException;

**import** java.util.Arrays;

**public** **class** Test1 {

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

    String s = "abc中文";

    System.***out***.println(s);

*f*(s, "UTF-16Be");

*f*(s, **null**);

*f*(s, "GBK");

*f*(s, "UTF-8");

}

/\*

 \* encoding     编码

 \* charset      字符集

 \*

 \* a 中

 \* Unicode      00 61    4e 2d

 \* UTF-8     61    e4 b8 ad

 \* GBK       61    d6 d0

 \*/

**private** **static** **void** f(

        String s, String charset) **throws** Exception {

**byte**[] a;

**if** (charset == **null**) {

        //Unicode ---> 系统默认编码

        a = s.getBytes();

    } **else** {

        //Unicode ---> 指定的编码

        a = s.getBytes(charset);

    }

    System.***out***.println(

     charset+"\t"+Arrays.*toString*(a));

    // 转回Unicode字符

**if** (charset == **null**) {

        //从系统默认编码转回Unicode

        s = **new** String(a);

    } **else** {

        //从指定编码转回Unicode

        s = **new** String(a, charset);

    }

    System.***out***.println(s);

    System.***out***.println("-------------------");

}

}

**4   Reader，Writer**

字符流的抽象父类，以字符为单位读写数据

**5   InputStreamReader,OutputStreamWriter**

字符编码转换流

OutputStreamWriter

把Java的Unicode编码字符，转成其他编码输出

InputStreamReader

读取其他编码字符，转成Unicode字符

**练习5  编码转换流**

项目：day1405\_编码转换流

类：day1405.Test1

**package** day1405;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.InputStreamReader;

**import** java.io.UnsupportedEncodingException;

**public** **class** Test1 {

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

    /\*

     \* f4   61 d6 d0

     \* f5   61 e4 b8 ad

     \*/

*f*("d:/abc/f4", "GBK");

*f*("d:/abc/f5", "UTF-8");

}

**private** **static** **void** f(

        String path, String charset) **throws** Exception {

    /\*

     \* ISR--FIS--path, charset

     \*

     \* InputStreamReader

     \* 读取其他编码，转成 Unicode 编码

     \*/

    InputStreamReader in =

**new** InputStreamReader(

**new** FileInputStream(path), charset);

    System.***out***.println((**char**)in.read());

    System.***out***.println((**char**)in.read());

    in.close();

}

}

Test2

**package** day1405;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.OutputStreamWriter;

**import** java.io.UnsupportedEncodingException;

**public** **class** Test2 {

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

    /\*

     \* f6   GBK

     \* f7   UTF-8

     \*

     \* 把Unicode编码所有的中文字符，输出到两个文件

     \* \u4e00到\u9fa5，20902个中文

     \*/

*f*("d:/abc/f6", "GBK");

*f*("d:/abc/f7", "UTF-8");

}

**private** **static** **void** f(

        String path, String charset) **throws** Exception {

    /\*

     \* OSW--FOS--path

     \*/

    OutputStreamWriter out =

**new** OutputStreamWriter(

**new** FileOutputStream(path), charset);

**int** count=0;

**for** (**char** i = '\u4e00'; i <= '\u9fa5'; i++) {

        out.write(i);

        count++;

**if**(count == 30) {

           out.write('\n');

           count=0;

        }

    }

    out.close();

}

}

**6   作业**

  重写

  day0703\_电子宠物

  day0704\_冒泡排序

  day1001\_猜游戏

  day1202\_变形金刚

  day1301\_学生对应成绩

  day1401\_文件加密复制

  加密复制文件夹中所有文件

Object

String

StrinBuilder

Integer

BigDecimal

Date

SimpelDateFormat

ArrayList

LinkedList

HashMap

Iterator

Exception

File

InputStream

FileIO

ObjectIO

Reader,Writer

ISR,OSW