

Java 的 16 进制与字符串的相互转换函数

```
1. /**
2.  * 将指定 byte 数组以 16 进制的形式打印到控制台
3.  * @param hint String
4.  * @param b byte[]
5.  * @return void
6.  */
7. public static void printHexString(String hint, byte[] b) {
8.     System.out.print(hint);
9.     for (int i = 0; i < b.length; i++) {
10.         String hex = Integer.toHexString(b[i] & 0xFF);
11.         if (hex.length() == 1) {
12.             hex = '0' + hex;
13.         }
14.         System.out.print(hex.toUpperCase() + " ");
15.     }
16.     System.out.println("");
17. }
```

```
1. /**
2.  *
3.  * @param b byte[]
4.  * @return String
5.  */
6. public static String Bytes2HexString(byte[] b) {
7.     String ret = "";
8.     for (int i = 0; i < b.length; i++) {
9.         String hex = Integer.toHexString(b[i] & 0xFF);
10.         if (hex.length() == 1) {
11.             hex = '0' + hex;
12.         }
13.         ret += hex.toUpperCase();
14.     }
15.     return ret;
16. }
```

```

1. /**
2.  * 将两个 ASCII 字符合成一个字节;
3.  * 如: "EF"--> 0xEF
4.  * @param src0 byte
5.  * @param src1 byte
6.  * @return byte
7.  */
8. public static byte uniteBytes(byte src0, byte src1) {
9.     byte _b0 = Byte.decode("0x" + new String(new byte[]{src0})).byteValue();
10.    _b0 = (byte)(_b0 << 4);
11.    byte _b1 = Byte.decode("0x" + new String(new byte[]{src1})).byteValue();
12.    byte ret = (byte)(_b0 ^ _b1);
13.    return ret;
14. }

```

```

1. /**
2.  * 将指定字符串 src, 以每两个字符分割转换为 16 进制形式
3.  * 如: "2B44EFD9" --> byte[]{0x2B, 0x44, 0xEF, 0xD9}
4.  * @param src String
5.  * @return byte[]
6.  */
7. public static byte[] HexString2Bytes(String src){
8.     byte[] ret = new byte[8];
9.     byte[] tmp = src.getBytes();
10.    for(int i=0; i<8; i++){
11.        ret[i] = uniteBytes(tmp[i*2], tmp[i*2+1]);
12.    }
13.    return ret;
14. }

```

CRC16Util

```
package com.sunwei.sim4xian;
```

```
import sun.misc.CRC16;
```

```

public class Crc16Util {
    private static final byte[] hex = "0123456789ABCDEF".getBytes();
    public static int getCRC16(byte[] data) {

```

```

int CRCTABLE[] = { 0xF078, 0xE1F1, 0xD36A, 0xC2E3, 0xB65C, 0xA7D5,
    0x954E, 0x84C7, 0x7C30, 0x6DB9, 0x5F22, 0x4EAB, 0x3A14, 0x2B9D,
    0x1906, 0x088F, 0xE0F9, 0xF170, 0xC3EB, 0xD262, 0xA6DD, 0xB754,
    0x85CF, 0x9446, 0x6CB1, 0x7D38, 0x4FA3, 0x5E2A, 0x2A95, 0x3B1C,
    0x0987, 0x180E, 0xD17A, 0xC0F3, 0xF268, 0xE3E1, 0x975E, 0x86D7,
    0xB44C, 0xA5C5, 0x5D32, 0x4CBB, 0x7E20, 0x6FA9, 0x1B16, 0x0A9F,
    0x3804, 0x298D, 0xC1FB, 0xD072, 0xE2E9, 0xF360, 0x87DF, 0x9656,
    0xA4CD, 0xB544, 0x4DB3, 0x5C3A, 0x6EA1, 0x7F28, 0x0B97, 0x1A1E,
    0x2885, 0x390C, 0xB27C, 0xA3F5, 0x916E, 0x80E7, 0xF458, 0xE5D1,
    0xD74A, 0xC6C3, 0x3E34, 0x2FBD, 0x1D26, 0x0CAF, 0x7810, 0x6999,
    0x5B02, 0x4A8B, 0xA2FD, 0xB374, 0x81EF, 0x9066, 0xE4D9, 0xF550,
    0xC7CB, 0xD642, 0x2EB5, 0x3F3C, 0x0DA7, 0x1C2E, 0x6891, 0x7918,
    0x4B83, 0x5A0A, 0x937E, 0x82F7, 0xB06C, 0xA1E5, 0xD55A, 0xC4D3,
    0xF648, 0xE7C1, 0x1F36, 0x0EBF, 0x3C24, 0x2DAD, 0x5912, 0x489B,
    0x7A00, 0x6B89, 0x83FF, 0x9276, 0xA0ED, 0xB164, 0xC5DB, 0xD452,
    0xE6C9, 0xF740, 0x0FB7, 0x1E3E, 0x2CA5, 0x3D2C, 0x4993, 0x581A,
    0x6A81, 0x7B08, 0x7470, 0x65F9, 0x5762, 0x46EB, 0x3254, 0x23DD,
    0x1146, 0x00CF, 0xF838, 0xE9B1, 0xDB2A, 0xCA3, 0xBE1C, 0xAF95,
    0x9D0E, 0x8C87, 0x64F1, 0x7578, 0x47E3, 0x566A, 0x22D5, 0x335C,
    0x01C7, 0x104E, 0xE8B9, 0xF930, 0xCBAB, 0xDA22, 0xAE9D, 0xBF14,
    0x8D8F, 0x9C06, 0x5572, 0x44FB, 0x7660, 0x67E9, 0x1356, 0x02DF,
    0x3044, 0x21CD, 0xD93A, 0xC8B3, 0xFA28, 0xEBA1, 0x9F1E, 0x8E97,
    0xBC0C, 0xAD85, 0x45F3, 0x547A, 0x66E1, 0x7768, 0x03D7, 0x125E,
    0x20C5, 0x314C, 0xC9BB, 0xD832, 0xEAA9, 0xFB20, 0x8F9F, 0x9E16,
    0xAC8D, 0xBD04, 0x3674, 0x27FD, 0x1566, 0x04EF, 0x7050, 0x61D9,
    0x5342, 0x42CB, 0xBA3C, 0xABB5, 0x992E, 0x88A7, 0xFC18, 0xED91,
    0xDF0A, 0xCE83, 0x26F5, 0x377C, 0x05E7, 0x146E, 0x60D1, 0x7158,
    0x43C3, 0x524A, 0xAABD, 0xBB34, 0x89AF, 0x9826, 0xEC99, 0xFD10,
    0xCF8B, 0xDE02, 0x1776, 0x06FF, 0x3464, 0x25ED, 0x5152, 0x40DB,
    0x7240, 0x63C9, 0x9B3E, 0x8AB7, 0xB82C, 0xA9A5, 0xDD1A, 0xCC93,
    0xFE08, 0xEF81, 0x07F7, 0x167E, 0x24E5, 0x356C, 0x41D3, 0x505A,
    0x62C1, 0x7348, 0x8BBF, 0x9A36, 0xA8AD, 0xB924, 0xCD9B, 0xDC12,
    0xEE89, 0xFF00 };

```

```
int CRCVal = 0;
```

```
int i = 0;
```

```

for (i = 0; i < data.length; i++) {
    CRCVal = CRCTABLE[(CRCVal ^= ((data[i]) & 0xFF)) & 0xFF]
        ^ (CRCVal >> 8);
}

```

```
// return Integer.toHexString(CRCVal);
```

```
// return String.valueOf(CRCVal);
```

```
    return CRCVal;
}
```

```
public static String crcTable(byte[] bytes) {
    int[] table = { 0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280,
        0xC241, 0xC601, 0x06C0, 0x0780, 0xC741, 0x0500, 0xC5C1, 0xC481,
        0x0440, 0xCC01, 0x0CC0, 0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81,
        0x0E40, 0x0A00, 0xCAC1, 0xCB81, 0x0B40, 0xC901, 0x09C0, 0x0880,
        0xC841, 0xD801, 0x18C0, 0x1980, 0xD941, 0x1B00, 0xDBC1, 0xDA81,
        0x1A40, 0x1E00, 0xDEC1, 0xDF81, 0x1F40, 0xDD01, 0x1DC0, 0x1C80,
        0xDC41, 0x1400, 0xD4C1, 0xD581, 0x1540, 0xD701, 0x17C0, 0x1680,
        0xD641, 0xD201, 0x12C0, 0x1380, 0xD341, 0x1100, 0xD1C1, 0xD081,
        0x1040, 0xF001, 0x30C0, 0x3180, 0xF141, 0x3300, 0xF3C1, 0xF281,
        0x3240, 0x3600, 0xF6C1, 0xF781, 0x3740, 0xF501, 0x35C0, 0x3480,
        0xF441, 0x3C00, 0xFCC1, 0xFD81, 0x3D40, 0xFF01, 0x3FC0, 0x3E80,
        0xFE41, 0xFA01, 0x3AC0, 0x3B80, 0xFB41, 0x3900, 0xF9C1, 0xF881,
        0x3840, 0x2800, 0xE8C1, 0xE981, 0x2940, 0xEB01, 0x2BC0, 0x2A80,
        0xEA41, 0xEE01, 0x2EC0, 0x2F80, 0xEF41, 0x2D00, 0xEDC1, 0xEC81,
        0x2C40, 0xE401, 0x24C0, 0x2580, 0xE541, 0x2700, 0xE7C1, 0xE681,
        0x2640, 0x2200, 0xE2C1, 0xE381, 0x2340, 0xE101, 0x21C0, 0x2080,
        0xE041, 0xA001, 0x60C0, 0x6180, 0xA141, 0x6300, 0xA3C1, 0xA281,
        0x6240, 0x6600, 0xA6C1, 0xA781, 0x6740, 0xA501, 0x65C0, 0x6480,
        0xA441, 0x6C00, 0xACC1, 0xAD81, 0x6D40, 0xAF01, 0x6FC0, 0x6E80,
        0xAE41, 0xAA01, 0x6AC0, 0x6B80, 0xAB41, 0x6900, 0xA9C1, 0xA881,
        0x6840, 0x7800, 0xB8C1, 0xB981, 0x7940, 0xBB01, 0x7BC0, 0x7A80,
        0xBA41, 0xBE01, 0x7EC0, 0x7F80, 0xBF41, 0x7D00, 0xBDC1, 0xBC81,
        0x7C40, 0xB401, 0x74C0, 0x7580, 0xB541, 0x7700, 0xB7C1, 0xB681,
        0x7640, 0x7200, 0xB2C1, 0xB381, 0x7340, 0xB101, 0x71C0, 0x7080,
        0xB041, 0x5000, 0x90C1, 0x9181, 0x5140, 0x9301, 0x53C0, 0x5280,
        0x9241, 0x9601, 0x56C0, 0x5780, 0x9741, 0x5500, 0x95C1, 0x9481,
        0x5440, 0x9C01, 0x5CC0, 0x5D80, 0x9D41, 0x5F00, 0x9FC1, 0x9E81,
        0x5E40, 0x5A00, 0x9AC1, 0x9B81, 0x5B40, 0x9901, 0x59C0, 0x5880,
        0x9841, 0x8801, 0x48C0, 0x4980, 0x8941, 0x4B00, 0x8BC1, 0x8A81,
        0x4A40, 0x4E00, 0x8EC1, 0x8F81, 0x4F40, 0x8D01, 0x4DC0, 0x4C80,
        0x8C41, 0x4400, 0x84C1, 0x8581, 0x4540, 0x8701, 0x47C0, 0x4680,
        0x8641, 0x8201, 0x42C0, 0x4380, 0x8341, 0x4100, 0x81C1, 0x8081,
        0x4040, };

    int crc = 0x0000;

    for (byte b : bytes) {
        crc = (crc >>> 8) ^ table[(crc ^ b) & 0xff];
    }
}
```

```

        return Integer.toHexString(crc);
    }

    public static String mkCrc16(byte[] b) {
        CRC16 crc16 = new CRC16();

        for (int i = 0; i < b.length; i++) {
            crc16.update(b[i]);
        }

        return Integer.toHexString(crc16.value);
    }

    public static final String evalCRC16(byte[] data) {
        int crc = 0xFFFF;

        for (int i = 0; i < data.length; i++) {
            crc = (data[i] << 8) ^ crc;

            for (int j = 0; j < 8; ++j) {
                if ((crc & 0x8000) != 0)
                    crc = (crc << 1) ^ 0x1021;
                else
                    crc <<= 1;
            }
        }

        return Integer.toHexString((crc ^ 0xFFFF) & 0xFFFF);
    }

    private static int parse(char c) {
        if (c >= 'a') {
            return (c - 'a' + 10) & 0x0f;
        }

        if (c >= 'A') {
            return (c - 'A' + 10) & 0x0f;
        }

        return (c - '0') & 0x0f;
    }

    public static byte[] HexString2Bytes(String hexstr) {
        byte[] b = new byte[hexstr.length() / 2];
    }

```

```

int j = 0;

for (int i = 0; i < b.length; i++) {
    char c0 = hexstr.charAt(j++);
    char c1 = hexstr.charAt(j++);

    b[i] = (byte) ((parse(c0) << 4) | parse(c1));
}

return b;
}

public static void main(String[] args) {
    //byte[] test = Crc16Util.HexString2Bytes("0200fb000130");
    byte[] test = {(byte)0x21, (byte)0x02, (byte)0x64,
        (byte)0x80, (byte)0x00, (byte)0x00,
        (byte)0x80, (byte)0x00, (byte)0x9c, (byte)0x47,
        (byte)0x00, (byte)0x37,
        (byte)0x00, (byte)0x00,
        (byte)0x0C,
        (byte)0x31,
        (byte)0x30, (byte)0x30, (byte)0x34, (byte)0x30, (byte)0x30,
        (byte)0x39, (byte)0x39, (byte)0x71,
        (byte)0x30, (byte)0x30, (byte)0x34, (byte)0x33, (byte)0x30, (byte)0x30, (byte)0x30, (byte)0x31,
        (byte)0x33,
        (byte)0x30, (byte)0x39, (byte)0x34, (byte)0x39, (byte)0x32, (byte)0x39,
        (byte)0x33, (byte)0x34, (byte)0x31, (byte)0x35, (byte)0x35, (byte)0x39,
        (byte)0x4E,
        (byte)0x31, (byte)0x30, (byte)0x39, (byte)0x30, (byte)0x30, (byte)0x30, (byte)0x30,
        (byte)0x45,
        (byte)0x00
    };
    System.out.println(Crc16Util.mkCrc16(test));
    System.out.println(Crc16Util.evalCRC16(test));
    System.out.println(Crc16Util.crcTable(test));
    System.out.println(Crc16Util.getCRC16(test));
}
}

```

JAVA 时间格式化处理

```

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

```

```

class dayTime
{
public static void main(String args[])
{
Date nowTime=new Date();
System.out.println(nowTime);
SimpleDateFormat time=new SimpleDateFormat("yyyy MM dd HH mm ss");
System.out.println(time.format(nowTime));
}
}

```

将毫秒转化为日期

```

import java.awt.BorderLayout;
import java.awt.Frame;
import java.awt.TextArea;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.text.SimpleDateFormat;
import java.util.Date;
public class ConvertLong2Date extends Frame{

    TextField tf = new TextField();
    TextArea ta = new TextArea();

    public static void main(String[] args) {
        new ConvertLong2Date().launchFrame();
    }

    public String convertL2D(long l) {
        long _l = 0L;
        Date _d = null;
        SimpleDateFormat _sdf = null;
        String _s = null;

        _l = l;
        _d = new Date(_l);
        _sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        _s = _sdf.format(_d);
    }
}

```

```

    return _s;
}

public void launchFrame() {
    setLocation(300, 300);
    setSize(480, 320);
    setResizable(false);
    add(tf,BorderLayout.SOUTH);
    add(ta,BorderLayout.NORTH);
    pack();
    tf.addActionListener(new tfActionListener());
    this.setVisible(true);
    this.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    });
}

public class tfActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        long l = Long.parseLong(tf.getText());
        ta.setText(new ConvertLong2Date().convertL2D(l));
        tf.setText("");
    }
}
}

```

文本的倒序输出

文件 before:

Hello
World

要求输出文件 after:

World
Hello

代码如下:

```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

```



```

import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.ListIterator;

public class ReverseOrder extends ArrayList {
    public static String read(String fileName) throws IOException {
        StringBuffer sb = new StringBuffer();
        LinkedList lines = new LinkedList();
        BufferedReader in = new BufferedReader(new FileReader(fileName));
        String s;
        while ((s = in.readLine()) != null)
            lines.add(s);
        in.close();
        ListIterator it = lines.listIterator(lines.size());
        while (it.hasPrevious()) {
            sb.append(it.previous());
            sb.append("\n");
        }
        return sb.toString();
    }

    public static void write(String fileName, String text) throws IOException {
        PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(
            fileName)));
        out.print(text);
        out.close();
    }

    public ReverseOrder(String fileName) throws IOException {
        super(Arrays.asList(read(fileName).split("\n")));
    }

    public void write(String fileName) throws IOException {
        PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(
            fileName)));
        for (int i = 0; i < size(); i++)
            out.println(get(i));
        out.close();
    }

    public static void main(String[] args) throws Exception {
        String fileName = "e:\\1124\\before.txt";
    }
}

```

```

ReverseOrder text = new ReverseOrder(fileName);
text.write("e:\\1124\\after.txt");
}
/*
 * 最后会多一个空行，手工删除一下
 */
}

```

判断一个数字是奇数还是偶数

判断一个数是否是奇数：

```

public static boolean isOdd(int i) {
    return (i&1) != 0;
}

```

判断一个数是否是偶数

```

public static boolean isEven(int i) {
    return (i&1) == 0;
}

```

//位运算符说明在 java 文件夹里面

用 Hibernate 实现分页

```

public List queryByStatus(int status, int currentPage, int lineSize)
    throws Exception {
    List all = null;
    String hql = "FROM Question AS q WHERE q.status=? ORDER BY q.questiontime desc";
    Query q = super.getSession().createQuery(hql);
    q.setInteger(0, status);
    q.setFirstResult((currentPage - 1) * lineSize);
    q.setMaxResults(lineSize);
    all = q.list();
    return all;
}

```

35 选 7 彩票程序

```

public class caipiao
{

```

```

static void generate()
{
    int a[]=new int[7];
    int i,m,j;
    fan:for(j=0;j <7;j++){//外循环实现随机生成每组 7 个数
        a[j]=(int)(Math.random()*35+1);
        m=a[j];
        if(j>=1){
            for(i=0;i <j;i++){//内循环实现无重复
                if(a[i]==m){
                    j--;
                    continue fan;
                }
            }
        }
        if(a[j] <10)
            System.out.print("0"+a[j]+" ");
        else
            System.out.print(a[j]+" ");
    }
}

public static void main (String args[]){
    int n=Integer.parseInt(args[0]);
    System.out.println("中国福利彩票 35 选 7");
    for(int i=0;i <n;i++){//循环调用方法实现输出 n 组数
        generate();
        System.out.println();
    }
}
}

```

获取 GMT8 时间

```

/**
 * Description: 获取 GMT8 时间
 * @return 将当前时间转换为 GMT8 时区后的 Date
 */
public static Date getGMT8Time(){
    Date gmt8 = null;
    try {
        Calendar cal = Calendar.getInstance(TimeZone.getTimeZone("GMT+8"),Locale.CHINESE); Calendar day =
        Calendar.getInstance();
        day.set(Calendar.YEAR, cal.get(Calendar.YEAR));
        day.set(Calendar.MONTH, cal.get(Calendar.MONTH));
    }
}

```

```

        day.set(Calendar.DATE, cal.get(Calendar.DATE));
        day.set(Calendar.HOUR_OF_DAY, cal.get(Calendar.HOUR_OF_DAY));
        day.set(Calendar.MINUTE, cal.get(Calendar.MINUTE));
        day.set(Calendar.SECOND, cal.get(Calendar.SECOND));
        gmt8 = day.getTime();
    } catch (Exception e) {
        System.out.println("获取 GMT8 时间 getGMT8Time() error !");
        e.printStackTrace();
        gmt8 = null;
    }
    return gmt8;
}

```

中文乱码转换

```

public String china(String args)
{
    String s=null;
    String s=new String(args.getBytes("ISO-8859-1"),"gb2312");
    return s;
}

```

小标签

```

import java.io.IOException;
import java.util.List;

import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.TagSupport;

import com.formcontent.show.ShowFormTypeOperateDb;
import com.forum.hibernatePrj.Space;
public class OutPrintForumType extends TagSupport{
    public int doStartTag() throws JspException
    {
        String printStr="";
        ShowFormTypeOperateDb showtype=new ShowFormTypeOperateDb();
        List list=showtype.getForumType();
        if(list!=null&&list.size()>0)
        {

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

```

```

{
    Space space=(Space)list.get(i);
    if(space!=null)
    {
        printStr+=" <tr> <td>"+ " <div align='left' class='TypeCss'>"+

            space.getSpaceName()+" "+space.getSpaceDescription()+" <br/> 目前登陆总人数:"+i+"    人访问数:"+i+"人 </div>
        </td> </tr>"
        +" <tr> <td> </td> </tr>";
    }
}
}
try {
    pageContext.getOut().write(printStr);
} catch (IOException e) {
    e.printStackTrace();
}
return super.doStartTag();
}
}

```

Big5 字与 Unicode 的互换

```

/**
 * Big5 字与 Unicode 的互换
 * 转换后的正常字型
 */

import java.io.*;

public class MyUtil{
    public static String big5ToUnicode(String s){
        try{
            return new String(s.getBytes("ISO8859_1"), "Big5");
        }
        catch (UnsupportedEncodingException uee){
            return s;
        }
    }

    public static String UnicodeTobig5(String s){
        try{

```

```

        return new String(s.getBytes("Big5"), "ISO8859_1");
    }
    catch (UnsupportedEncodingException uee){
        return s;
    }
}

public static String toHexString(String s){
    String str="";
    for (int i=0; i<s.length(); i++){
        int ch=(int)s.charAt(i);
        String s4="0000"+Integer.toHexString(ch);
        str=str+s4.substring(s4.length()-4)+" ";
    }
    return str;
}
}

```

取得服务器当前的各种具体时间

```

/**
 * 取得服务器当前的各种具体时间
 * 回车：日期时间
 */

import java.util.*;

public class GetNowDate{
    Calendar  calendar = null;

    public GetNowDate(){
        calendar = Calendar.getInstance();
        calendar.setTime(new Date());
    }

    public int getYear(){
        return calendar.get(Calendar.YEAR);
    }

    public int getMonth(){
        return 1 + calendar.get(Calendar.MONTH);
    }
}

```

```

public int getDay(){
    return calendar.get(Calendar.DAY_OF_MONTH); } public int getHour(){
    return calendar.get(Calendar.HOUR_OF_DAY); } public int getMinute(){
    return calendar.get(Calendar.MINUTE);
}

public int getSecond(){
    return calendar.get(Calendar.SECOND);
}

public String getDate(){
    return getMonth()+"/"+getDay()+"/"+getYear();
}

public String getTime(){
    return getHour()+":"+getMinute()+":"+getSecond();
}

public String getDate2(){
    String yyyy="0000", mm="00", dd="00";
    yyyy = yyyy + getYear();
    mm  = mm  + getMonth();
    dd  = dd + getDay();
    yyyy = yyyy.substring(yyyy.length()-4);
    mm  = mm.substring(mm.length()-2);
    dd  = dd.substring(dd.length()-2);
    return yyyy + "/" + mm + "/" + dd;
}

public String getTime2(){
    String hh="00", mm="00", ss="00";
    hh = hh + getHour();
    mm = mm + getMinute();
    ss = ss + getSecond();
    hh = hh.substring(hh.length()-2, hh.length());
    mm = mm.substring(mm.length()-2, mm.length());
    ss = ss.substring(ss.length()-2, ss.length());
    return hh + ":" + mm + ":" + ss;
}
}

```

用半角的特殊符号代替全角的特殊符号

```
/**
 * 用半角的特殊符号代替全角的特殊符号
 * 防止特殊字符在传输参数时出现错误
 *
 */
```

```
public class ReplaceStrE{
    public static String rightToError(String ss){
        String str;
        String str1;
        String str2;
        String str3;
        String str4;
        try{
            str = ss.replace('#','#');
        }
        catch(Exception ex){
            return ss;
        }

        try{
            str1 = str.replace(" ","");
        }
        catch(Exception ex){
            return str;
        }

        try{
            str2 = str1.replace(' ','&');
        }
        catch(Exception ex){
            return str1;
        }

        try{
            str3 = str2.replace('+','+');
        }
        catch(Exception ex){
            return str2;
        }

        try{
            str4 = str3.replace("' ','\\");
        }
```



```

        catch(Exception ex){
            return ss;
        }
        return str4;
    }
}

```

数组和数组之间的转换代码

```

import java.lang.reflect.Array;
import java.util.Date;

public class TestCast {

    /**
     * @param args
     */
    //public static void main(String[] args) {
        /** *//**
         *
         * 一般情况下数组和数组是不能直接进行转换的,例如:
         * Object[] t1={"1","2"};
         * String[] t2=(String[])t1;//这里会出现转换错误
         *
         * 下面提供了一种方式进行转换
         */

        //1.0 测试一般基础类
        /*
        Object[] t1={"1","2","3","4","5"};
        String[] m1=(String[])TestCast.cast(t1,String.class);
        for(int i=0;i<m1.length;i++)
            System.out.println(m1[i]);

        //2.0 测试复杂对象
        Object[] t2={new Date(1000),new Date(2000)};
        Date[] m2=(Date[])TestCast.cast(t2,Date.class);
        for(int i=0;i<m2.length;i++)
            System.out.println(m2[i].toString());*/
        // }

        /** *//**
         * 将数组 array 转换成 class 代表的类型后返回

```

```

* @param array 需要转换的数组
* @param class 要转换成的类型
* @return 转换后的数组
*/
public static Object cast(Object array, Class cls){
    if(null==cls)
        throw new IllegalArgumentException("argument class cannot be null");
    if(null==array)
        throw new IllegalArgumentException("argument array cannot be null");
    if(false==array.getClass().isArray())
        throw new IllegalArgumentException("argument array must be array");

    Object[] src=(Object[])array;
    Object[] dest=(Object[])Array.newInstance(cls, src.length);
    System.arraycopy(src, 0, dest, 0, src.length);
    return dest;
}
}

```

从资源文件里读取值的类

从资源文件里读取值的类，文件后缀不一定要.Properties，只要里面内容如：url=www.cnsec.net
 可通过 key（url）取得值-www.cnsec.net，简单、强大

Java code

```

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Properties;
/**
 * ReadProperties.java
 * Description: 读取操作属性配置文件
 * @author li.b
 * @version 2.0
 * Jun 26, 2008
 */
public class ReadProperties {

    /**

```

```

    * Description: 获取属性配置文件
    * @param path 资源文件路径
    * @return Properties Object
    * @throws FileNotFoundException
    * @throws IOException
    */
    public static Properties getProperties(String path) throws FileNotFoundException,
    IOException{
        Properties props = null;
        File file = new File(path);
        if(file.exists() && file.isFile()){
            props = new Properties();
            props.load(new FileInputStream(file));
        }else{
            System.out.println(file.toString() + "不存在！");
        }
        return props;
    }

    /**
    * Description: 从属性文件获取值
    * @param props Properties Object
    * @param key
    * @return 通过 key 匹配到的 value
    */
    public static String getValue(Properties props, String key, String encod) {
        String result = "";
        String en = "";
        String localEN = System.getProperty("file.encoding");
        if(encod !=null && !encod.equals("")) {
            en = encod;
        }else{
            en = localEN;
        }
        try {
            key = new String(key.getBytes(en), "ISO-8859-1");
            result = props.getProperty(key);
            if(!result.equals("")){
                result = new String(result.getBytes("ISO-8859-1"), en);
            }
        } catch (Exception e) {
        }finally{
            if(result == null)result = "";
            return result;
        }
    }

```

```

    }
}

    public static String getValue(Properties props, String key) {
        return getValue(props, key, "");
    }

}

```

一个随机类

```

/*
 * @author talent_marquis<qq' qq>
 * Email: talent_marquis@163.com
 * Copyright (C) 2007 talent_marquis<qq' qq>
 * All rights reserved.
 */
package com.dextrys.trilogy.util;

import java.util.Arrays;

import org.eclipse.swt.graphics.RGB;

public class RandomUtil
{

    /**
     * @param args
     */
    public static void main( String[] args )
    {
        //System.out.println( getRandomNormalString( 8 ) );
        int[] test = getRandomIntWithoutReduplicate( 0, 40, 39 );
        Arrays.sort( test );
        for( int i : test )
        {
            System.out.println( i );
        }
    }

    /**
     * get a integer array filled with random integer without reduplicate [min, max)
     * @param min    the minimum value

```

```

* @param max the maximum value
* @param size the capacity of the array
* @return a integer array filled with random integer without reduplicate
*/
public static int[] getRandomIntWithoutReduplicate( int min, int max, int size )
{
    int[] result = new int[size];

    int arraySize = max - min;
    int[] intArray = new int[arraySize];
    // init intArray
    for( int i = 0 ; i < intArray.length ; i++ )
    {
        intArray[i] = i + min;
    }
    // get random integer without reduplicate
    for( int i = 0 ; i < size ; i++ )
    {
        int c = getRandomInt( min, max - i );
        int index = c - min;
        swap( intArray, index, arraySize - 1 - i );
        result[i] = intArray[ arraySize - 1 - i ];
    }

    return result;
}

private static void swap( int[] array, int x, int y )
{
    int temp = array[x];
    array[x] = array[y];
    array[y] = temp;
}

/**
* get a random Integer with the range [min, max)
* @param min the minimum value
* @param max the maximum value
* @return the random Integer value
*/
public static int getRandomInt( int min, int max )
{
    // include min, exclude max
    int result = min + new Double( Math.random() * ( max - min ) ).intValue();

```

```

        return result;
    }

/**
 * get a random double with the range [min, max)
 * @param min the minimum value
 * @param max the maximum value
 * @return the random double value
 */
public static double getRandomDouble( double min, double max )
{
    // include min, exclude max
    double result = min + ( Math.random() * ( max - min ) );
    return result;
}

/**
 *
 * @return a random char with the range ASCII 33(!) to ASCII 126(~)
 */
public static char getRandomChar()
{
    // from ASCII code 33 to ASCII code 126
    int firstChar = 33; // "!"
    int lastChar = 126; // "~"
    char result = ( char ) ( getRandomInt( firstChar, lastChar + 1 ) );
    return result;
}

/**
 *
 * @return a random rgb color
 */
public static RGB getRandomRGB()
{
    int red = getRandomInt(0,256);
    int green = getRandomInt(0,256);
    int blue = getRandomInt(0,256);

    return new RGB( red, green, blue );
}

/**

```

```

*
* @return a random char with the range [0-9],[a-z],[A-Z]
*/
public static char getRandomNormalChar()
{
    // include 0-9,a-z,A-Z
    int number = getRandomInt( 0, 62 );
    int zeroChar = 48;
    int nineChar = 57;
    int aChar = 97;
    int zChar = 122;
    int AChar = 65;
    int ZChar = 90;

    char result;

    if( number < 10 )
    {
        result = ( char ) ( getRandomInt( zeroChar, nineChar + 1 ) );
        return result;
    }
    else if( number >= 10 && number < 36 )
    {
        result = ( char ) ( getRandomInt( AChar, ZChar + 1 ) );
        return result;
    }
    else if( number >= 36 && number < 62 )
    {
        result = ( char ) ( getRandomInt( aChar, zChar + 1 ) );
        return result;
    }
    else
    {
        return 0;
    }
}

/**
 *
 * @param length the length of the String
 * @return a String filled with random char
 */
public static String getRandomString( int length )

```

```

{
    // include ASCII code from 33 to 126
    StringBuffer result = new StringBuffer();
    for( int i = 0; i < length; i++ )
    {
        result.append( getRandomChar() );
    }
    return result.toString();
}

/**
 *
 * @param length the length of the String
 * @return a String filled with normal random char
 */
public static String getRandomNormalString( int length )
{
    // include 0-9,a-z,A-Z
    StringBuffer result = new StringBuffer();
    for( int i = 0; i < length; i++ )
    {
        result.append( getRandomNormalChar() );
    }
    return result.toString();
}
}

```

计算传入值是否星期六

```

/**
 * 计算传入值是否星期六
 * 回车: true or false
 */

import java.util.*;

public class Week6 {
    public boolean checkWeek6(String str){
        boolean flag=false;
        int week6=0;
        str.replace('/', '-');
        Calendar cal=Calendar.getInstance();
        cal.setTime(java.sql.Date.valueOf(str.substring(0,10)));
    }
}

```



```

week6=cal.get(Calendar.DAY_OF_WEEK);

if(week6==7){
    flag=true;
}

return flag;
}
}

```

为 RootPaneContainer 组件添加键盘事件

```

/**
 * 为 RootPaneContainer 组件添加键盘事件
 * @param rpc RootPaneContainer 组件
 * @param action 需要执行的动作
 * @param keyName 键的名称
 * @param keyCode 键的数字代码
 * @param modifiers 任意修饰符的按位或组合
 */
public static void registerKeyEvent(RootPaneContainer rpc, Action action, String keyName, int keyCode, int modifiers)
{
    JRootPane rp = rpc.getRootPane();
    InputMap inputMap = rp.getInputMap(JComponent.WHEN_IN_FOCUSED_WINDOW);
    inputMap.put(KeyStroke.getKeyStroke(keyCode, modifiers), keyName);
    rp.getActionMap().put(keyName, action);
}

```

将数组转成字符串 在调试或记录日志时用到

```

/**
 * 将数组转成字符串 在调试或记录日志时用到
 *
 * @param array
 * @return
 */
public static String byte2string(byte[] array) {
    StringBuilder sb = new StringBuilder();

    sb.append("Length " + array.length + " Content ");

    for (int i = 0; i < leng; i++) {

```

```

        sb = sb.append(String.format("%02X", array[i])).append(":");
    }
    int ind = sb.lastIndexOf(":");
    sb.delete(ind, ind + 1);
    return sb.toString();
}

```

转换文件大小

```

import java.text.DecimalFormat;
import java.util.Hashtable;

/**
 * 文件大小单位转换
 * @author Administrator
 *
 */
public class UnitsConversion extends DecimalFormat {

    private static final long serialVersionUID = 3168068393840262910L;
    /**
     * 存放有效单位的数组
     */
    private static Hashtable<String, String> validUnits = new Hashtable<String, String>();
    /**
     * 限制文件大小上限为 1G
     */
    private static int GB_MAX_SIZE = 1;
    /**
     * 最大的 MB 值
     */
    private static int MB_MAX_SIZE = GB_MAX_SIZE * 1024;
    /**
     * 最大的 KB 值
     */
    private static int KB_MAX_SIZE = MB_MAX_SIZE * 1024;
    /**
     * 最大的 Bytes 值
     */
    private static int BYTES_MAX_SIZE = KB_MAX_SIZE * 1024;
    /**
     * 数字部分的值
     */
}

```

```

private Double numPart;
/**
 * 原始的单位字符串
 */
private String originalUnit;
/**
 * 标准的单位字符串
 */
private String unit;
/**
 * 转换后的结果
 */
private String result;

// 添加所有有效单位
static {
    validUnits.put("字节", "Bytes");
    validUnits.put("bytes", "Bytes");
    validUnits.put("byte", "Bytes");
    validUnits.put("kb", "KB");
    validUnits.put("k", "KB");
    validUnits.put("兆", "MB");
    validUnits.put("mb", "MB");
    validUnits.put("m", "MB");
    validUnits.put("gb", "GB");
    validUnits.put("g", "GB");
}

/**
 * 构造方法：指定了数字格式，初始所有属性为 NULL
 */
public UnitsConversion() {
    super("#####.##");
    numPart = null;
    result = null;
    unit = null;
    originalUnit = null;
}

/**
 * 根据单位、数字大小按照常用的转换原则进行转换
 *
 * @param input
 * @return 成功转换后的结果是非空字符串；若失败了，结果为空字符串

```

```

*/
public String defaultConversion(String input) {
    analyzeString(input);
    if (result != null) {
        return result;
    }
    // 单位 Bytes
    if (unit.equals("Bytes")) {
        int numPart2Int = numPart.intValue();
        // 输入大小与 1G 相差 0.5M 之内，返回 1GB
        if ((BYTES_MAX_SIZE - numPart2Int) < (1024 * 1024) / 2) {
            return "1 GB";
        }
        // (0,1KB)
        if (numPart2Int < 1024) {
            return numPart2Int + " Bytes";
        }
        // [1KB,1023KB]
        if (numPart2Int >= 1024 && numPart2Int <= (1024 - 1) * 1024) {
            return format(numPart / 1024) + " KB";
        }
        // (1023KB,1GB)
        if (numPart2Int > (1024 - 1) * 1024 && numPart2Int < BYTES_MAX_SIZE) {
            return format(numPart / (1024 * 1024)) + " MB";
        } else
            result = "";
        return result;
    }

    if (unit.equals("KB")) {
        return "还没实现....";
    }

    if (unit.equals("MB")) {
        return "还没实现....";
    }

    if (unit.equals("GB")) {
        return "还没实现....";
    }
    result = "";
    return result;
}

```

```

/** * 把字符串的数字部分与单位分离，并对数字、单位的有效性进行检验， 若有非法状况，把结果赋值为 "" ， 将其返回给用户 * *
@param input
*/
public void analyzeString(String input) {
    // 初步检验输入的字符串
    if (input == null || input.trim().length() < 2) {
        System.out.println("输入的字符串有误");
        result = "";
        return;
    }
    input = input.replaceAll(" ", "");
    int firstIndexOfUnit;// 单位的起始位置
    String strOfNum;// 数字部分的字符串
    // 从尾部开始遍历字符串
    for (int i = input.length() - 1; i >= 0; i--) {
        if (Character.isDigit(input.charAt(i))) {
            firstIndexOfUnit = i + 1;
            originalUnit = input.substring(firstIndexOfUnit,
                input.length()).toLowerCase();
            if (!isValidUnit(originalUnit)) {
                System.out.println("无效单位。");
                result = "";
                return;
            }
            unit = validUnits.get(originalUnit);
            strOfNum = input.substring(0, firstIndexOfUnit);
            numPart = Double.parseDouble(strOfNum);
            if (!isValidNum(numPart, unit)) {
                System.out.println("文件大小非法");
                result = "";
                return;
            }
        }
        if (numPart == 0) {
            result = "0 Bytes";
            return;
        }
        break;
    }
}
if (unit == null || numPart == null) {
    System.out.println("输入的字符串有误");
    result = "";
    return;
}
}

```

```
}
```

```
/**
```

```
 * 文件大小越界检查
```

```
 *
```

```
 * @param num
```

```
 * @param unit
```

```
 * @return 在 1G 范围内（包括 1G），返回 true；否则返回 false
```

```
 */
```

```
public boolean isValidNum(Double num, String unit) {  
    if (num == null || num < 0 || num > BYTES_MAX_SIZE) {  
        return false;  
    }  
    if (unit.equals("KB") && num > KB_MAX_SIZE) {  
        return false;  
    }  
    if (unit.equals("MB") && num > MB_MAX_SIZE) {  
        return false;  
    }  
    if (unit.equals("GB") && num > GB_MAX_SIZE) {  
        return false;  
    }  
    return true;  
}
```

```
/**
```

```
 * 检查原始单位 originalUnit 是否有效
```

```
 *
```

```
 * @param originalUnit
```

```
 * @return 若 originalUnit 为空，那么会给他赋默认值 bytes，并返回 true；<br>
```

```
 * 若 originalUnit 是有效单位集合中之一，返回 true。
```

```
 */
```

```
public boolean isValidUnit(String originalUnit) {  
    if (originalUnit == null || originalUnit.trim().length() < 1) {  
        originalUnit = "bytes";  
        return true;  
    }  
    for (String validUnit : validUnits.keySet()) {  
        if (validUnit.equalsIgnoreCase(originalUnit)) {  
            return true;  
        }  
    }  
    return false;  
}
```

```

}
//测试
public static void main(String[] args) {
    System.out.println("-----");
    for (int i = 1020 * 1024; i <= 1024 * 1111; i += 9) {
        String input = i + " ";
        System.out.println(input + " ---> "
            + new UnitsConversion().defaultConversion(input));
    }
}
}
}

```

多线程的世界时钟，显示巴黎，罗马，上海时间, **AWT** 界面

```

/*心得: TimeZone tz1=TimeZone.getTimeZone("Europe/Paris");
*   Calendar cld=Calendar.getInstance(tz);
*   clk.setText(cld.get(Calendar.HOUR_OF_DAY)+":"+cld.get(Calendar.MINUTE)+":"+cld.get(Calendar.SECOND));
*/

```

```

import java.awt.*;
import java.awt.event.*;
import java.util.*;

```

```

public class WorldClock{
    Frame f=new Frame("WorldClock");
    Label l1=new Label();
    Label l2=new Label();
    Label l3=new Label();
    Label cl1=new Label();
    Label cl2=new Label();
    Label cl3=new Label();

    public WorldClock(){

        l1.setFont(new Font("Arial",Font.BOLD,30));
        l2.setFont(new Font("Arial",Font.BOLD,30));
        l3.setFont(new Font("Arial",Font.BOLD,30));
        cl1.setFont(new Font("Arial",Font.BOLD,30));
        cl2.setFont(new Font("Arial",Font.BOLD,30));
        cl3.setFont(new Font("Arial",Font.BOLD,30));
        cl1.setForeground(Color.red);
    }
}

```

```
cl2.setForeground(Color.red);
cl3.setForeground(Color.red);
```

```
f.setLayout(new GridLayout(2,3));
f.add(l1);
f.add(l2);
f.add(l3);
f.add(cl1);
f.add(cl2);
f.add(cl3);
```

```
TimeZone tz1=TimeZone.getTimeZone("Europe/Paris");
clock c1=new clock(l1,cl1,tz1);
new Thread(c1).start();
```

```
TimeZone tz2=TimeZone.getTimeZone("Asia/Shanghai");
clock c2=new clock(l2,cl2,tz2);
new Thread(c2).start();
```

```
TimeZone tz3=TimeZone.getTimeZone("Europe/Rome");
clock c3=new clock(l3,cl3,tz3);
new Thread(c3).start();
```

```
f.setLocation(200,200);
f.setVisible(true);
f.pack();
```

```
}
public static void main(String[] args){
    new WorldClock();
    String[] s=TimeZone.getAvailableIDs();
    int i=0;
    while(++i<s.length){
        System.out.println (s[i]);
    }
}
}
```

```
class clock implements Runnable{
    private Label l;
    private Label clk;
    TimeZone tz;
```



```

public clock(Label l,Label clk,TimeZone tz){
    this.l=l;
    this.clk=clk;
    this.tz=tz;
}

public void run(){
    l.setText(tz.getID());
    while(true){
        Calendar cld=Calendar.getInstance(tz);
        clk.setText(cld.get(Calendar.HOUR_OF_DAY)+":"+cld.get(Calendar.MINUTE)+":"+cld.get(Calendar.SECOND));
        try{
            Thread.sleep(1000);
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
}

```

Java 日期格式化及其使用例子

1 SimpleDateFormat 担当重任,怎样格式化都行

```

import java.util.Date;
import java.text.SimpleDateFormat;
public class Demo
{
    public static void main(String[] args)
    {
        Date now=new Date();
        SimpleDateFormat f=new SimpleDateFormat("今天是"+"yyyy 年 MM 月 dd 日 E kk 点 mm 分");
        System.out.println(f.format(now));

        f=new SimpleDateFormat("a hh 点 mm 分 ss 秒");
        System.out.println(f.format(now));
    }
}

```

2 从字符串到日期类型的转换:

```

import java.util.Date;
import java.text.SimpleDateFormat;
import java.util.GregorianCalendar;
import java.text.*;

public class Demo
{
    public static void main(String[] args)
    {
        String strDate="2005 年 04 月 22 日";
        //注意: SimpleDateFormat 构造函数的样式与 strDate 的样式必须相符
        SimpleDateFormat simpleDateFormat=new SimpleDateFormat("yyyy 年 MM 月 dd 日");
        //必须捕获异常

        try
        {
            Date date=simpleDateFormat.parse(strDate);
            System.out.println(date);
        }
        catch(ParseException px)
        {
            px.printStackTrace();
        }
    }
}

```

3 将毫秒数换转成日期类型

```

import java.util.Date;
import java.text.SimpleDateFormat;
import java.util.GregorianCalendar;
import java.text.*;

public class Demo
{
    public static void main(String[] args)
    {
        long now=System.currentTimeMillis();
        System.out.println("毫秒数: "+now);
        Date dNow=new Date(now);
        System.out.println("日期类型: "+dNow);
    }
}

```

这 3 例源自 <http://blog.csdn.net/zhouljian2003/archive/2005/04/22/358363.aspx>

4 获取系统时期和时间，转换成 SQL 格式后更新到数据库

(<http://blog.csdn.net/netrope/archive/2005/11/19/532729.aspx>)

```
java.util.Date d=new java.util.Date();    //获取当前系统的时间
```

```
//格式化日期
```

```
new java.text.SimpleDateFormat s= new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
```

```
String dateStr = s.format(d); //转为字符串
```

使用 RS 更新数据库，仍然要用 rs.updateString，而不是 rs.updateDade。

```
rs.updateString("regtime",dateStr); //regtime 字段为 datetime 类型的
```

下面两例源自 <http://blog.csdn.net/kingter520/archive/2004/10/27/155435.aspx>

5 按本地时区输出当前日期

```
Date myDate = new Date();
```

```
System.out.println(myDate.toLocaleString());
```

输出结果为：

2003-5-30

6 如何格式化小数

```
DecimalFormat df = new DecimalFormat("#,###.00");
```

```
double aNumber = 33665448856.6568975;
```

```
String result = df.format(aNumber);
```

```
Sytem. out.println(result);
```

输出结果为：

33,665,448,856.66

其他：获取毫秒时间 System.currentTimeMillis();

7 在数据库里的日期只以年-月-日的方式输出

(<http://blog.csdn.net/zzsxvzzsxv/archive/2007/08/27/1761004.aspx>)

定义日期格式：SimpleDateFormat sdf = new SimpleDateFormat(yy-MM-dd);

sql 语句为：String sqlStr = "select bookDate from roomBook where bookDate between '2007-4-10' and '2007-4-25'";

输出：

```
System.out.println(df.format(rs.getDate("bookDate")));
```

几个常用方法

字符串

1、获取字符串的长度

`length()`

2、判断字符串的前缀或后缀与已知字符串是否相同

前缀 `startsWith(String s)`

后缀 `endsWith(String s)`

3、比较两个字符串

`equals(String s)`

4、把字符串转化为相应的数值

`int` 型 `Integer.parseInt(字符串)`

`long` 型 `Long.parseLong(字符串)`

`float` 型 `Float.valueOf(字符串).floatValue()`

`double` 型 `Double.valueOf(字符串).doubleValue()`

4、将数值转化为字符串

`valueOf(数值)`

5、字符串检索

`indexOf(String s)` 从头开始检索

`indexOf(String s, int startpoint)` 从 `startpoint` 处开始检索

如果没有检索到，将返回-1

6、得到字符串的子字符串

`substring(int startpoint)` 从 `startpoint` 处开始获取

`substring(int start, int end)` 从 `start` 到 `end` 中间的字符

7、替换字符串中的字符,去掉字符串前后空格

`replace(char old, char new)` 用 `new` 替换 `old`

`trim()`

8、分析字符串

`StringTokenizer(String s)` 构造一个分析器，使用默认分隔字符（空格，换行，回车，Tab，进纸符）

`StringTokenizer(String s, String delim)` `delim` 是自己定义的分隔符

`nextToken()` 逐个获取字符串中的语言符号

`boolean hasMoreTokens()` 只要字符串还有语言符号将返回 `true`，否则返回 `false`

`countTokens()` 得到一共有多少个语言符号

Java 中的鼠标和键盘事件

1、使用 `MouseListener` 接口处理鼠标事件

鼠标事件有 5 种：按下鼠标键，释放鼠标键，点击鼠标键，鼠标进入和鼠标退出

鼠标事件类型是 `MouseEvent`，主要方法有：

`getX(), getY()` 获取鼠标位置

`getModifiers()` 获取鼠标左键或者右键

`getClickCount()` 获取鼠标被点击的次数

getSource() 获取鼠标发生的事件源

事件源获得监视器的方法是 addMouseListener(), 移去监视器的方法是 removeMouseListener()

处理事件源发生的时间的事件的接口是 MouseListener 接口中有如下的方法

mousePressed(MouseEvent) 负责处理鼠标按下事件

mouseReleased(MouseEvent) 负责处理鼠标释放事件

mouseEntered(MouseEvent) 负责处理鼠标进入容器事件

mouseExited(MouseEvent) 负责处理鼠标离开事件

mouseClicked(MouseEvent) 负责处理点击事件

2、使用 MouseMotionListener 接口处理鼠标事件

事件源发生的鼠标事件有 2 种：拖动鼠标和鼠标移动

鼠标事件的类型是 MouseEvent

事件源获得监视器的方法是 addMouseMotionListener()

处理事件源发生的事件的接口是 MouseMotionListener 接口中有如下的方法

mouseDragged() 负责处理鼠标拖动事件

mouseMoved() 负责处理鼠标移动事件

3、控制鼠标的指针形状

setCursor(Cursor.getPreddfinedCursor(Cursor.鼠标形状定义)) 鼠标形状定义见（书 P 210）

4、键盘事件

键盘事件源使用 addKeyListener 方法获得监视器

键盘事件的接口是 KeyListener 接口中有 3 个方法

public void keyPressed(KeyEvent e) 按下键盘按键

public void keyReleased(KeyEvent e) 释放键盘按键

public void keyType(KeyEvent e) 按下又释放键盘按键

判断字符是否属于中文

```
public class IsChineseOrEnglish {
// GENERAL_PUNCTUATION 判断中文的“号
// CJK_SYMBOLS_AND_PUNCTUATION 判断中文的。号
// HALFWIDTH_AND_FULLWIDTH_FORMS 判断中文的，号
public static boolean isChinese(char c) {
    Character.UnicodeBlock ub = Character.UnicodeBlock.of(c);
    if (ub == Character.UnicodeBlock.CJK_UNIFIED_IDEOGRAPHS
        || ub == Character.UnicodeBlock.CJK_COMPATIBILITY_IDEOGRAPHS
        || ub == Character.UnicodeBlock.CJK_UNIFIED_IDEOGRAPHS_EXTENSION_A
        || ub == Character.UnicodeBlock.GENERAL_PUNCTUATION
        || ub == Character.UnicodeBlock.CJK_SYMBOLS_AND_PUNCTUATION
        || ub == Character.UnicodeBlock.HALFWIDTH_AND_FULLWIDTH_FORMS){
        return true;
    }
    return false;
}

public static void isChinese(String strName) {
```

```

        char[] ch = strName.toCharArray();
        for (int i = 0; i < ch.length; i++) {
            char c = ch[i];
            if(isChinese(c)==true){
                System.out.println(isChinese(c));
                return;
            }else{
                System.out.println(isChinese(c));
                return ;
            }
        }
    }

    public static void main(String[] args){

        isChinese("zhongguo");
        isChinese("中国");
    }

}

```

异常处理类

```

/**
 * (#)ThrowableManager.java    1.0    Apr 10, 2008
 *
 * Copyright 2007- wargrey , Inc. All rights are reserved.
 */
package net.wargrey.application;

import java.awt.Component;
import javax.swing.JOptionPane;

/**
 * This class <code>ExceptionHandler</code> and its subclasses are a form of
 * <code>Exception</code>. It is used to wrap all the <code>Throwable</code> instances
 * and handle them in a unified way. It will show the information which consists of
 * StackTraces and Messages by using JOptionPanel.
 *
 * @author Estelle
 * @version 1.0
 * @see java.lang.Exception
 * @since jdk 1.5

```

```

*/
public class ExceptionManager extends Exception {

    /**
     * This field <code>alerter</code> is used to show the information the Class offered.
     *
     * @see javax.swing.JOptionPane
     */
    private JOptionPane alerter;

    /**
     * This static method create an instance of the ExceptionManager by invoking the
     * constructor <code>ExceptionManager(String msg)</code>.
     *
     * @param msg    The message will pass the specified constructor
     * @return      An instance of the ExceptionManager created by invoking the constructor
     *              <code>ExceptionManager(String msg)</code>.
     */
    public static ExceptionManager wrap(String msg){
        return new ExceptionManager(msg);
    }

    /**
     * This static method create an instance of the ExceptionManager by invoking the
     * constructor <code>ExceptionManager(Throwable throwable)</code>.
     *
     * @param throwable    The cause will pass the specified constructor
     * @return      An instance of the ExceptionManager created by invoking the constructor
     *              <code>ExceptionManager(Throwable throwable)</code>.
     */
    public static ExceptionManager wrap(Throwable throwable){
        return new ExceptionManager(throwable);
    }

    /**
     * This static method create an instance of the ExceptionManager by invoking the
     * constructor <code>ExceptionManager(String msg,Throwable throwable)</code>.
     *
     * @param msg        The message will pass the specified constructor
     * @param throwable    The cause will pass the specified constructor
     * @return      An instance of the ExceptionManager created by invoking the constructor
     *              <code>ExceptionManager(String msg, Throwable throwable)</code>
     */
    public static ExceptionManager wrap(String msg,Throwable throwable){

```

```
    return new ExceptionManager(msg,throwable);  
}
```

```
/**  
 * Constructs a new instance with the specified detail message. The concrete handler  
 * is its super class. This constructor always used to construct a custom exception  
 * not wrapping the exist exception.  
 *  
 * @param msg      the detail message which is the part of the information will be  
 *                  shown.  
 */
```

```
public ExceptionManager(String msg){  
    super(msg);  
}
```

```
/**  
 * Constructs a new instance with the specified detail cause. The concrete handler  
 * is its super class. This constructor always used to wrap an exist exception.  
 *  
 * @param throwable the cause which has been caught. It's detail message and  
 *                  stacktrace are the parts the information will be shown.  
 */
```

```
public ExceptionManager(Throwable throwable){  
    super(throwable);  
}
```

```
/**  
 * Constructs a new instance with the specified detail message and cause. The  
 * concrete handler is its super class. This constructor always used to construct  
 * an exception wrapping the exist exception but requires a custom message.  
 *  
 * @param msg      the detail message which is the part of the information will  
 *                  be shown.  
 * @param throwable the cause which has been caught. It's stacktrace is the parts  
 *                  the information will be shown.  
 */
```

```
public ExceptionManager(String msg,Throwable throwable){  
    super(msg,throwable);  
}
```

```
/**  
 * Show the information with everything is default.  
 */
```

```
public synchronized void alert(){
```



```

        alert((Component)null);
    }

    /**
     * Show the information in a dialog with the specified title
     * "ThrowableManager Alerter". The dialog belongs to the given component which
     * default is the screen.
     *
     * @param parent    The component cause the exception.
     */
    public synchronized void alert(Component parent){
        alert(parent,"ThrowableManager Alerter");
    }

    /**
     * Show the information in a dialog with the specified title.
     *
     * @param title      The title of the dialog.
     */
    public synchronized void alert(String title){
        alert((Component)null,title);
    }

    /**
     * Show the information in a dialog which has the specified title and belongs to the
     * specified component.
     *
     * @param parent    The component cause the exception.
     * @param title      The title of the dialog.
     */
    public synchronized void alert(Component parent,String title){
        StringBuilder errorMessage=new StringBuilder();
        errorMessage.append(this.toString());
        for (StackTraceElement st:((this.getCause()==null)?this:this.getCause()).getStackTrace()){
            errorMessage.append("\n\t    at ");
            errorMessage.append(st.toString());
        }
        alerter.showMessageDialog(parent, errorMessage, title ,JOptionPane.ERROR_MESSAGE);
    }
}

```

去掉字符串中重复的子字符串

```

/**
 * 去掉字符串中重复的子字符串
 *
 * @param str
 * @return String
 */
private static String removeSameString(String str)
{
    Set<String> mLinkedSet = new LinkedHashSet<String>();
    String[] strArray = str.split(" ");
    StringBuffer sb = new StringBuffer();

    for (int i = 0; i < strArray.length; i++)
    {
        if (!mLinkedSet.contains(strArray[i]))
        {
            mLinkedSet.add(strArray[i]);
            sb.append(strArray[i] + " ");
        }
    }
    System.out.println(mLinkedSet);
    return sb.toString().substring(0, sb.toString().length() - 1);
}

```

将指定 **byte** 数组以 **16** 进制的形式打印到控制台

```

/**
 * 将指定 byte 数组以 16 进制的形式打印到控制台
 *
 * @param hint
 *         String
 * @param b
 *         byte[]
 * @return void
 */
public static void printHexString(String hint, byte[] b)
{
    System.out.print(hint);
    for (int i = 0; i < b.length; i++)
    {
        String hex = Integer.toHexString(b[i] & 0xFF);
        if (hex.length() == 1)
        {

```

```

        hex = '0' + hex;
    }
    System.out.print(hex.toUpperCase() + " ");
}
System.out.println("");
}

```

获得任意一个整数的阶乘，递归

```

/**
 * 获得任意一个整数的阶乘，递归
 *
 * @param n
 * @return n!
 */
public static int factorial(int n)
{
    if (n == 1)
    {
        return 1;
    }
    return n * factorial(n - 1);
}

```

拷贝一个目录或者文件到指定路径下

```

/**
 * 拷贝一个目录或者文件到指定路径下
 *
 * @param source
 * @param target
 */
public static void copy(File source, File target)
{
    File tarpath = new File(target, source.getName());
    if (source.isDirectory())
    {
        tarpath.mkdir();
        File[] dir = source.listFiles();
        for (int i = 0; i < dir.length; i++) { copy(dir[i], tarpath); } } else
    {
        try

```

```

{
    InputStream is = new FileInputStream(source);
    OutputStream os = new FileOutputStream(tarpath);
    byte[] buf = new byte[1024];
    int len = 0;
    while ((len = is.read(buf)) != -1)
    {
        os.write(buf, 0, len);
    }
    is.close();
    os.close();
}
catch (FileNotFoundException e)
{
    e.printStackTrace();
}
catch (IOException e)
{
    e.printStackTrace();
}
}
}

```

简单的 txt 转换 xml

```

package com.liu;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.StringTokenizer;

public class TxtToXml {
    private String strTxtFileName;

    private String strXmlFileName;

    public TxtToXml() {
        strTxtFileName = new String();
        strXmlFileName = new String();
    }
}

```

```

public void createXml(String strTxt, String strXml) {
    strTxtFileName = strTxt;
    strXmlFileName = strXml;
    String strTmp;
    try {
        BufferedReader inTxt = new BufferedReader(new FileReader( strTxtFileName));
        BufferedWriter outXml = new
        BufferedWriter(new FileWriter(
            strXmlFileName));
        outXml.write("<?xml version= \"1.0\" encoding=\"gb2312\"?>");
        outXml.newLine();
        outXml.write("<people>");
        while ((strTmp = inTxt.readLine()) != null) {
            StringTokenizer strToken = new StringTokenizer(strTmp, " ");
            String arrTmp[];
            arrTmp = new String[3];
            for (int i = 0; i < 3; i++)
                arrTmp[i] = new String("");
            int index = 0;
            outXml.newLine();
            outXml.write("    <students>");
            while (strToken.hasMoreElements()) {
                strTmp = (String) strToken.nextElement();
                strTmp = strTmp.trim();
                arrTmp[index++] = strTmp;
            }
            outXml.newLine();
            outXml.write("        <name>" + arrTmp[0] + "</name>");
            outXml.newLine();
            outXml.write("        <sex>" + arrTmp[1] + "</sex>");
            outXml.newLine();
            outXml.write("        <age>" + arrTmp[2] + "</age>");
            outXml.newLine();
            outXml.write("    </students>");
        }
        outXml.newLine();
        outXml.write("</people>");
        outXml.flush();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public static void main(String[] args) {
    String txtName = "testtxt.txt";

```

```
String xmlName = "testxml.xml";
TxtToXml thisClass = new TxtToXml();
thisClass.createXml(txtName, xmlName);
}
}
```

字母排序(A-Z)(先大写, 后小写)

```
import java.util.Arrays;
import java.util.Comparator;

public class SortTest
{
    public static void main(String args[])
    {
        char[] chs = {'f', 'F', 'K', 'A', 'a', 'j', 'z'};
        chs = sortChars(chs, false);

        for(char c: chs)
        {
            System.out.println(c);
        }
    }

    /**
     * 对给定的字符数组进行字典排序
     * @param chs 目标字符数组
     * @param upperFisrt 大写字母是否在前
     * @return 排序后的字符数组
     */
    public static char[] sortChars(char[] chs, final boolean upperFisrt)
    {
        Character[] srcArray = new Character[chs.length];
        char[] retArray = new char[chs.length];
        int index = 0;

        for(char ch: chs)
        {
            srcArray[index++] = ch;
        }

        Arrays.sort(srcArray, new Comparator<Character>()
        {

```

```

    public int compare(Character c1, Character c2)
    {
        char ch1 = Character.toUpperCase(c1);
        char ch2 = Character.toUpperCase(c2);

        if(ch1 == ch2)
        {
            int tempRet = c1.charValue() - c2.charValue();
            return upperFisrt? tempRet: -tempRet;
        }
        else
        {
            return ch1 - ch2;
        }
    }
});

index = 0;

for(char ch: srcArray)
{
    retArray[index++] = ch;
}

return retArray;
}
}

```

列出某文件夹及其子文件夹下面的文件，并可根据扩展名过滤

```

/**
 * 列出某文件夹及其子文件夹下面的文件，并可根据扩展名过滤
 *
 * @param path
 */
public static void list(File path)
{
    if (!path.exists())
    {
        System.out.println("文件名称不存在!");
    }
    else
    {

```

```

if (path.isFile())
{
    if (path.getName().toLowerCase().endsWith(".pdf")
        || path.getName().toLowerCase().endsWith(".doc")
        || path.getName().toLowerCase().endsWith(".html")
        || path.getName().toLowerCase().endsWith(".htm"))
    {
        System.out.println(path);
        System.out.println(path.getName());
    }
}
else
{
    File[] files = path.listFiles();
    for (int i = 0; i < files.length; i++)
    {
        list(files[i]);
    }
}
}
}

```

字符串匹配的算法.

```

public String getMaxMatch(String a,String b) {
    StringBuffer tmp = new StringBuffer();
    String maxString = "";
    int max = 0;
    int len = 0;
    char[] aArray = a.toCharArray();
    char[] bArray = b.toCharArray();
    int posA = 0;
    int posB = 0;
    while(posA<aArray.length-max) {
        posB = 0;
        while(posB<(bArray.length-max)) {
            if(aArray[posA]==bArray[posB]) {
                len = 1;
                tmp = new StringBuffer();
                tmp.append(aArray[posA]);
                while((posA+len<aArray.length)&&(posB+len<bArray.length)&&(aArray[posA+len]==bArray[posB+len]))
            {
                tmp.append(aArray[posA+len]);
            }
            }
        }
    }
}

```



```

        len++;
    }
    if(len>max) {
        max = len;
        maxString = tmp.toString();
    }
}
posB++;
}
posA++;
}
return maxString;
}

```

写入日志

```

/**
 * 写入日志
 * filePath 日志文件的路径
 * code 要写入日志文件的内容
 */
public static boolean print(String filePath,String code) {
    try {
        File tofile=new File(filePath);
        FileWriter fw=new FileWriter(tofile,true);
        BufferedWriter bw=new BufferedWriter(fw);
        PrintWriter pw=new PrintWriter(bw);

        System.out.println(getDate()+":"+code);
        pw.println(getDate()+":"+code);
        pw.close();
        bw.close();
        fw.close();
        return true;
    } catch (IOException e) {
        return false;
    }
}

```