```
1  import java.awt.Graphics;
2  import java.awt.Color;
3  import java.awt.Font;
4  import java.awt.FontMetrics;
5  import java.awt.Image;
6  import java.awt.Event;
7  import java.net.URL;
8  import java.net.MalformedURLException;
9  import java.io.*;
10
11 public class scroller extends java.applet.Applet implements Runnable
12     {
13         //***************//
14         // declarations: //
15         //***************//
16
17         Image       OffScreenImage; // reduce flicker
18         Graphics    OffScreen;      // reduce flicker
19
20         Color       BackColor;      // background color
21         Color       ForeColor;      // foreground color
22         Font        UsedFont;       // font to be used
23         int         TextLeft;       // left space to frame
24
25         Thread      Runner;         // use multithreading
26         int         SpeedDelay;     // speed settings
27         int         MsgDelay;       // speed settings
28         int         CurrentY;       // holds current Y-position of scrolling text
29         int         CurrentMsg;     // holds item index for current item
30         boolean     Up = true;      // direction Up if true, else direction is down
31
32         String      ErrorMsg="";    // contains ErrorMessage, if an error occurs
33
34         String[]    Items;
35
36         //*************//
37         // functions:  //
38         //*************//
39
40         //*******************************//
41         // gets the specified parameter  //
42         // and decodes it to a Color     //
43         //*******************************//
44         Color DecodeColor(String param)
45             {
46                 String pString = getParameter(param);
47                 Color Result = new Color(Integer.parseInt(pString.substring(0,2), 16),
48                                          Integer.parseInt(pString.substring(2,4), 16),
49                                          Integer.parseInt(pString.substring(4,6), 16));
```

```
50                  return Result;
51              }
52
53          //************************************//
54          // checks, if string contains substring //
55          //************************************//
56          boolean SubStringExists(String MainString, String SubString)
57              {
58                  int index = MainString.indexOf(SubString);
59                  boolean Result;
60                  if ((index >= 0) & (index < MainString.length())) Result = true;
61                      else
62                          Result = false;
63                  return Result;
64              }
65
66          //*****************************//
67          // get the specified parameter //
68          // and decodes it to a Font    //
69          //*****************************//
70          Font DecodeFont(String param)
71              {
72                  int    FontStyle;
73                  int    FontSize;
74                  String FontName;
75                  Font   Result;
76
77                  String pString = getParameter(param);
78                  if (pString == null) Result = new Font("Arial", Font.PLAIN, 12);
79                      else
80                          {
81                              int ende = pString.indexOf(",");
82                              FontSize = Integer.parseInt(pString.substring(0, ende));
83                              int ende1 = pString.indexOf(",", ende+1);
84                              FontName = pString.substring(ende+2, ende1);
85                              pString = pString.toUpperCase();
86                              FontStyle = Font.PLAIN;
87                              if (SubStringExists(pString, "ITALIC"))
88                                  FontStyle += Font.ITALIC;
89                              if (SubStringExists(pString, "BOLD"))
90                                  FontStyle += Font.BOLD;
91                              Result = new Font(FontName, FontStyle, FontSize);
92                          }
93                  return Result;
94              }
95
96          //***************************//
97          // gets specified parameter   //
98          // and converts it to integer //
```

```
 99          //****************************//
100          int DecodeInteger(String param)
101              {
102                  String pString = getParameter(param);
103                  return Integer.parseInt(pString);
104              }
105
106          //**************************//
107          // checks, if param exists  //
108          //**************************//
109          boolean ParamExists(String param)
110              {
111                  String pString = getParameter(param);
112                  boolean Result;
113                  if (pString == null) Result = false;
114                      else
115                          Result = true;
116                  return Result;
117              }
118
119          //***********************//
120          // resizes Items[]-array //
121          //***********************//
122          String[] CreateItemArray(int count)
123              {
124                  String ItemArray[] = new String[count];
125                  for (int i = 0; i <= ItemArray.length; i++)
126                      {
127                          ItemArray[0] = "";
128                      }
129                  return ItemArray;
130              }
131
132          //************************************//
133          // encodes a byte value to a character //
134          //************************************//
135          String EncodeByte(byte Byte)
136              {
137                  String Result;
138                  switch (Byte)
139                      {
140                          case -28: Result = "ä";
141                                    break;
142                          case -10: Result = "ö";
143                                    break;
144                          case  -4: Result = "ü";
145                                    break;
146                          case -42: Result = "Ö";
147                                    break;
```

```
148                         case -60: Result = "Ä";
149                                   break;
150                         case -36: Result = "Ü";
151                                   break;
152                         case -33: Result = "ß";
153                                   break;
154                         case -27: Result = "å";
155                                   break;
156                         case -59: Result = "Å";
157                                   break;
158                         default:  String[] List = {  "",  "",  "",  "",  "",  "",  "",  "",  "",  "",
159                                                      "",  "",  "",  "",  "",  "",  "",  "",  "",  "",
160                                                      "",  "",  "",  "",  "",  "",  "",  "",  "",  "",
161                                                      "",  " ",  "!",  "'",  "#",  "$",  "%",  "&",  "'",  "(",
162                                                      ")",  "*",  "+",  ",",  "-",  ".",  "/",  "0",  "1",  "2",
163                                                      "3",  "4",  "5",  "6",  "7",  "8",  "9",  ":",  ";",  "<",
164                                                      "=",  ">",  "?",  "@",  "A",  "B",  "C",  "D",  "E",  "F",
165                                                      "G",  "H",  "I",  "J",  "K",  "L",  "M",  "N",  "O",  "P",
166                                                      "Q",  "R",  "S",  "T",  "U",  "V",  "W",  "X",  "Y",  "Z",
167                                                      "  ",  "_",  "  ",  "^",  "_",  "  ",  "a",  "b",  "c",  "d",
168                                                      "e",  "f",  "g",  "h",  "i",  "j",  "k",  "l",  "m",  "n",
169                                                      "o",  "p",  "q",  "r",  "s",  "t",  "u",  "v",  "w",  "x",
170                                                      "y",  "z",  "{",  "|",  "}",  "~",  " ",  "€",  " ",  ",",
171                                                      "ƒ",  „",  "…",  "†",  "‡",  "ˆ",  "‰",  "Š",  "‹",  "Œ",
172                                                      " ",  "Ž",  " ",  " ",  "‘",  "’",  "“",  "”",  "•",  "–",
173                                                      "—",  "˜",  "™",  "š",  "›",  "œ",  " ",  "ž",  "Ÿ",  " ",
174                                                      "¡",  "¢",  "£",  "¤",  "¥",  "¦",  "§",  "¨",  "©",  "ª",
175                                                      "«",  "¬",  "-",  "®",  "¯",  "°",  "±",  "²",  "³",  "´",
176                                                      "µ",  "¶",  "·",  "¸",  "¹",  "º",  "»",  "¼",  "½",  "¾",
177                                                      "¿",  "À",  "Á",  "Â",  "Ã",  "Ä",  "Å",  "Æ",  "Ç",  "È",
178                                                      "É",  "Ê",  "Ë",  "Ì",  "Í",  "Î",  "Ï",  "Ð",  "Ñ",  "Ò",
179                                                      "Ó",  "Ô",  "Õ",  "Ö",  "×",  "Ø",  "Ù",  "Ú",  "Û",  "Ü",
180                                                      "Ý",  "Þ",  "ß",  "à",  "á",  "â",  "ã",  "ä",  "å",  "æ",
181                                                      "ç",  "è",  "é",  "ê",  "ë",  "ì",  "í",  "î",  "ï",  "ð",
182                                                      "ñ",  "ò",  "ó",  "ô",  "õ",  "ö",  "÷",  "ø",  "ù",  "ú",
183                                                      "û",  "ü",  "ý",  "þ",  "ÿ",  " " };
184                                   Result = List[Byte-1];
185                                   break;
186                     }
187                 return Result;
188             }
189
190         //*********************//
191         // applet functions:  //
192         //*********************//
193
194         public void init()
195             {
196                 //initialize input stream:
```

```
197                 InputStream File = null;
198                 try
199                     {
200                         File = new URL(getDocumentBase(), getParameter("file")).openStream();
201                         byte b[] = new byte[1];
202                         byte cr[] = new byte[4];
203                         File.read(cr);
204                         int CR = -1;
205                         int Error = 0;
206                         int ItemCount = 0;
207                         if ((cr[0]==0x2A) && (cr[2]==0x2A)) CR = 1;
208                         if ((cr[0]==0x2A) && (cr[3]==0x2A)) CR = 2;
209                         File.close();
210                         File = new URL(getDocumentBase(), getParameter("file")).openStream();
211                         switch (CR)
212                             {
213                                 case -1:
214                                     ErrorMsg = "internal error: file is empty!";
215                                     break;
216                                 case 1:
217                                     //carriage return only #13 OR #10
218                                     {
219                                     for (int i=0;i<=3;i++)
220                                         {
221                                             File.read(b);
222                                         }
223                                     while (Error != -1)
224                                         {
225                                             Error = File.read(b);
226                                             if ((b[0]==13) || (b[0]==10))
227                                                 ItemCount += 1;
228                                         }
229                                     Items = CreateItemArray(ItemCount);
230                                     for (int i=0; i <= Items.length-1; i++)
231                                         Items[i] = "";
232                                     //String Items[] = new String[ItemCount];
233                                     File.close();
234                                     File = new URL(getDocumentBase(), getParameter("file")).openStream();
235                                     //jump over *#13#10*#13#10:
236                                     for (int i=0;i<=5;i++)
237                                         {
238                                             File.read(b);
239                                         }
240                                     Error = 0;
241                                     int CurrentItem = 0;
242                                     while (Error != -1)
243                                         {
244                                             Error = File.read(b);
245                                             if ((b[0] != 13) && (b[0] != 10))
```

```
246                                         {
247                                             if (Error != -1)
248                                                 Items[CurrentItem] += EncodeByte(b[0]);
249                                         }
250                                             else
251                                                 {
252                                                     CurrentItem = CurrentItem + 1;
253                                                 }
254                                     }
255                                 }
256                                 break;
257                             case 2:
258                                 //carriage return plus line feed #13#10
259                                 {
260                                 // * #13 #10 * #13 #10
261                                 // 0  1   2  3  4   5
262                                 for (int i=0;i<=4;i++)
263                                     {
264                                         File.read(b);
265                                     }
266                                 //determine ItemCount:
267                                 while (Error != -1)
268                                     {
269                                         Error = File.read(b);
270                                         if ((b[0]==13) || (b[0]==10))
271                                             {
272                                                 Error = File.read(b);
273                                                 ItemCount += 1;
274                                             }
275                                     }
276                                 Items = CreateItemArray(ItemCount);
277                                 for (int i=0; i <= Items.length-1; i++)
278                                     Items[i] = "";
279                                 //String Items[] = new String[ItemCount];
280                                 File.close();
281                                 File = new URL(getDocumentBase(), getParameter("file")).openStream();
282                                 //jump over *#13#10*#13#10:
283                                 for (int i=0;i<=5;i++)
284                                     {
285                                         File.read(b);
286                                     }
287                                 Error = 0;
288                                 int CurrentItem = 0;
289                                 while (Error != -1)
290                                     {
291                                         Error = File.read(b);
292                                         if ((b[0] != 13) && (b[0] != 10))
293                                             {
294                                                 if (Error != -1)
```

```
                                                    Items[CurrentItem] += EncodeByte(b[0]);
                                                }
                                                else
                                                    {
                                                        Error = File.read(b);
                                                        CurrentItem = CurrentItem + 1;
                                                    }
                                                }
                                            }
                                            break;
                                        }
                                } catch (Exception e)
                                    {
                                        ErrorMsg = e.toString();
                                    }
                            //free stream if operations finished:
                            try
                                {
                                    if (File != null) File.close();
                                } catch (Exception e)
                                    {
                                        ErrorMsg = e.toString();
                                    }
                            //get colors:
                            if (ParamExists("background"))
                                BackColor = DecodeColor("background");
                                    else
                                        BackColor = Color.white;
                            if (ParamExists("foreground"))
                                ForeColor = DecodeColor("foreground");
                                    else
                                        ForeColor = Color.black;
                            //get font and its style:
                            UsedFont = DecodeFont("font");
                            //get the timer parameters:
                            if (ParamExists("speeddelay"))
                                SpeedDelay = DecodeInteger("speeddelay");
                                    else
                                        SpeedDelay = 10;
                            if (ParamExists("msgdelay"))
                                MsgDelay = DecodeInteger("msgdelay");
                                    else
                                        MsgDelay = 1500;
                            //left frame:
                            if (ParamExists("left"))
                                TextLeft = DecodeInteger("left");
                                    else
                                        TextLeft = 10;
                            if (Up)
```

```
344                     CurrentY = getSize().height+UsedFont.getSize()+5;
345                         else
346                             CurrentY = -5;
347                 //initialize OffScreen Images:
348                 OffScreenImage = createImage(getSize().width, getSize().height);
349                 OffScreen = OffScreenImage.getGraphics();
350                 OffScreen.setFont(UsedFont);
351             }
352
353         public void start()
354             {
355                 if (Runner == null)
356                     {
357                         Runner = new Thread(this);
358                         Runner.start();
359                     }
360             }
361
362         public void stop()
363             {
364                 if (Runner != null)
365                     {
366                         Runner = null;
367                     }
368             }
369
370         public void run()
371             {
372                 Thread thisThread = Thread.currentThread();
373                 while (Runner == thisThread)
374                     {
375                         repaint();
376                         try
377                             {
378                                 Thread.sleep(SpeedDelay);
379                                 if (CurrentY == ((int) (getSize().height/2+UsedFont.getSize()/2-1)))
380                                     {
381                                         try
382                                             {
383                                                 Thread.sleep(MsgDelay);
384                                             }
385                                                 catch (InterruptedException e)
386                                                     {
387                                                         ErrorMsg = e.toString();
388                                                     }
389
390                                     }
391                             }
392                                 catch (InterruptedException e)
```

```
                                    {
                                        ErrorMsg = e.toString();
                                    }
                    }
            }

        public void update(Graphics g)
            {
                paint(g);
            }

        public void destroy()
            {
                OffScreen.dispose();
            }

        public void paint(Graphics canvas)
            {
                //fill rectangel with background color:
                OffScreen.setColor(BackColor);
                OffScreen.fillRect(0, 0, getSize().width, getSize().height);
                OffScreen.setColor(ForeColor);
                setBackground(BackColor);
                if (Up == true)
                    {
                        CurrentY--;
                        if (CurrentY < -5)
                            {
                                CurrentY = getSize().height+5+UsedFont.getSize();
                                CurrentMsg++;
                                if (CurrentMsg > (Items.length - 1))
                                    CurrentMsg = 0;
                            }
                        OffScreen.drawString(Items[CurrentMsg], TextLeft, CurrentY);
                    }
                        else
                            {
                                CurrentY++;
                                if (CurrentY >= getSize().height+5+UsedFont.getSize())
                                    {
                                        CurrentY = -5;
                                        CurrentMsg++;
                                        if (CurrentMsg > (Items.length - 1))
                                            CurrentMsg = 0;
                                    }
                                OffScreen.drawString(Items[CurrentMsg], TextLeft, CurrentY);
                            }
                //if (ErrorMsg != "")
                //  OffScreen.drawString("ErrorMsg: "+ErrorMsg, 0,getSize().height-1);
```

```
442                 //copy OffScreenImage to Screen:
443                 canvas.drawImage(OffScreenImage, 0, 0, this);
444             }
445
446         public String getAppletInfo()
447             {
448                 return "Title: Scroller-Applet \nAuthor: Klaus Burgstaller \nThis applet is a kind of news scroller \nlast modified on 29.11.2001 \ncopyright (c) by Klaus Burgstaller. \nwww.crosswinds.net/~burgstaller";
449             }
450
451         public String[][] getParameterInfo()
452             {
453                 String[][] info =   {
454                                         {"file", "path string", "Specifies file containing data to display as relative path. \nThis parameter has no default value. The component doesn't work otherwise!"},
455                                         {"foreground", "hex color", "Foreground color in hex format: RRGGBB"},
456                                         {"background", "hex color", "Works the same way for the back ground color"},
457                                         {"font", "font size, font name, font style", "contains description about the font style: \nfirst parameter is font size, second font name, \nadditional font stylings can be given: BOLD, ITALIC, PLAIN. \nYou may write the font stylings upper od lower case. \nDefault = Arial, plain, size:12 ."},
458                                         {"speeddelay", "int", "Motion speed in milli seconds between graphics refresh, default: 10"},
459                                         {"msgdelay", "int", "Text will stop for xxx milliseconds, then go on, default: 1500"},
460                                         {"left", "int", "Specifies the space between left border and text in pixels, default: 10"}
461                                     };
462                 return info;
463             }
464     }
```