

```
1 import java.awt.Graphics;
2 import java.awt.Polygon;
3 import java.awt.Color;
4 import java.awt.Font;
5 import java.awt.FontMetrics;
6 import java.awt.Image;
7 import java.awt.Event;
8 import java.net.URL;
9 import java.net.MalformedURLException;
10 import java.io.InputStream;
11 import java.io.BufferedReader;
12 import java.io.IOException;
13 import java.io.InputStreamReader;
14
15 public class special_scroller extends java.applet.Applet
16 {
17     //*****//
18     // declarations: //
19     //*****//
20
21     Image      OffScreenImage; // reduce flicker
22     Graphics    OffScreen;      // reduce flicker
23
24     //hold strings ... for use when turning over
25     Image      Image1;
26     Image      Image2;
27     Image      Image3;
28     Graphics    Temp;
29
30     Color      BackColor;      // background color
31     Color      ForeColor;      // foreground color
32     Font        UsedFont;      // font to be used
33     Font        UsedFontBold;  // same as UsedFont, but bold
34     Font        SmallFont;     // small font
35     Font        BigFont;       // big font
36
37     int         CurrentImage = 1; // holds current image number
38
39     String      ErrorMsg="";    // contains ErrorMessage, if an error occurs
40     String      s1;
41
42     boolean     OverLeft = false;
43     boolean     OverRight = false;
44
45     String[]    RawItems;
46     String[]    Items;
47     String[]    Items1;
48     String[]    Items2;
49
```

```
50     boolean    Items1Exist = false;
51     boolean    Items2Exist = false;
52
53     int l1 = 5;
54     int t1 = 5;
55     int r1 = 50;
56     int b1 = 30;
57
58     //*****//
59     // functions: //
60     //*****//
61
62     //*****//
63     // gets the specified parameter //
64     // and decodes it to a Color //
65     //*****//
66     Color DecodeColor(String param)
67     {
68         String pString = getParameter(param);
69         Color Result = new Color(Integer.parseInt(pString.substring(0,2), 16),
70                                 Integer.parseInt(pString.substring(2,4), 16),
71                                 Integer.parseInt(pString.substring(4,6), 16));
72         return Result;
73     }
74
75     //*****//
76     // checks, if string contains substring //
77     //*****//
78     boolean SubStringExists(String MainString, String SubString)
79     {
80         int index = MainString.indexOf(SubString);
81         boolean Result;
82         if ((index >= 0) & (index < MainString.length())) Result = true;
83         else
84             Result = false;
85         return Result;
86     }
87
88     //*****//
89     // get the specified parameter //
90     // and decodes it to a Font //
91     //*****//
92     Font DecodeFont(String param)
93     {
94         int    FontStyle;
95         int    FontSize;
96         String FontName;
97         Font    Result;
98     }
```

```
99         String pString = getParameter(param);
100         if (pString == null) Result = new Font("Arial", Font.PLAIN, 8);
101         else
102             {
103                 int ende = pString.indexOf(",");
104                 FontSize = Integer.parseInt(pString.substring(0, ende));
105                 int ende1 = pString.indexOf(",", ende+1);
106                 FontName = pString.substring(ende+2, ende1);
107                 pString = pString.toUpperCase();
108                 FontStyle = Font.PLAIN;
109                 if (SubStringExists(pString, "ITALIC"))
110                     FontStyle += Font.ITALIC;
111                 if (SubStringExists(pString, "BOLD"))
112                     FontStyle += Font.BOLD;
113                 Result = new Font(FontName, FontStyle, FontSize);
114             }
115         return Result;
116     }
117
118     //*****//
119     // gets specified parameter //
120     // and converts it to integer //
121     //*****//
122     int DecodeInteger(String param)
123     {
124         String pString = getParameter(param);
125         return Integer.parseInt(pString);
126     }
127
128     //*****//
129     // checks, if param exists //
130     //*****//
131     boolean ParamExists(String param)
132     {
133         String pString = getParameter(param);
134         boolean Result;
135         if (pString == null) Result = false;
136         else
137             Result = true;
138         return Result;
139     }
140
141     //*****//
142     // resizes Items[]-array //
143     //*****//
144     String[] CreateItemArray(int count)
145     {
146         String ItemArray[] = new String[count];
147         for (int i = 0; i <= ItemArray.length; i++)
```

```
148         {
149             ItemArray[0] = "";
150         }
151         return ItemArray;
152     }
153
154     //*****//
155     // identify various font tokens //
156     //*****//
157     int FoundIdent(String Line)
158     {
159         int found = -1;
160         int index = -1;
161         index = Line.indexOf("<r>");
162         if (index != -1) found = 3;
163         index = Line.indexOf("<fBIG>");
164         if (index != -1) found = 1;
165         index = Line.indexOf("<fSMALL>");
166         if (index != -1) found = 2;
167         index = Line.indexOf("<b>");
168         if (index != -1) found = 4;
169         return found;
170     }
171
172     //*****//
173     // extract string from tokens //
174     //*****//
175     String ExtractString(String str, int Token)
176     {
177         switch (Token)
178         {
179             case 1:
180                 str = str.substring(6);
181                 break;
182             case 2:
183                 str = str.substring(8);
184                 break;
185             case 3:
186                 str = str.substring(3);
187                 break;
188             case 4:
189                 {
190                     int i1 = str.indexOf("<b>")+3;
191                     int i2 = str.indexOf("</b>", i1);
192                     str = str.substring(i1, i2);
193                     break;
194                 }
195             case 10:
196                 {
```

```
197         int i1 = str.indexOf("</b>")+4;
198         str = str.substring(i1, str.length());
199         break;
200     }
201 }
202 return str;
203 }
204
205 //*****//
206 // division returning rounded value //
207 //*****//
208 int div(float x, float y)
209 {
210     return Math.round(x/y);
211 }
212
213 //*****//
214 // lightens or darkens a color //
215 // by the given parameter //
216 //*****//
217 Color ModifyColor(Color color, int amount)
218 {
219     int r = color.getRed();
220     int g = color.getGreen();
221     int b = color.getBlue();
222     r += amount;
223     if (r > 255) r = 255;
224     else
225         if (r < 0) r = 0;
226     g += amount;
227     if (g > 255) g = 255;
228     else
229         if (g < 0) g = 0;
230     b += amount;
231     if (b > 255) b = 255;
232     else
233         if (b < 0) b = 0;
234     Color Result = new Color(r, g, b);
235     return Result;
236 }
237
238 //*****//
239 // applet functions: //
240 //*****//
241
242 public void init()
243 {
244     if (ParamExists("file1"))
245     {
```

```
246         //initialize input stream:
247         InputStream File = null;
248         BufferedReader in = null;
249         try
250         {
251             URL FileURL = new URL(getDocumentBase(), getParameter("file1"));
252             //open a buffered reader
253             in = new BufferedReader(new InputStreamReader(FileURL.openStream()));
254             s1 = in.readLine();
255             int ItemCount = Integer.parseInt(s1);
256             Items = CreateItemArray(ItemCount);
257             //now read all the items
258             for (int i=0;i<ItemCount;i++)
259             {
260                 Items[i] = in.readLine();
261             }
262             try
263             {
264                 in.close();
265             } catch (IOException e) {}
266         } catch (Exception e)
267         {
268             ErrorMsg = e.toString();
269         }
270     }
271     if (ParamExists("file2"))
272     {
273         Items1Exist = true;
274         InputStream File = null;
275         BufferedReader in = null;
276         try
277         {
278             URL FileURL = new URL(getDocumentBase(), getParameter("file2"));
279             //open a buffered reader
280             in = new BufferedReader(new InputStreamReader(FileURL.openStream()));
281             s1 = in.readLine();
282             int ItemCount = Integer.parseInt(s1);
283             Items1 = CreateItemArray(ItemCount);
284             //now read all the items
285             for (int i=0;i<ItemCount;i++)
286             {
287                 Items1[i] = in.readLine();
288             }
289             try
290             {
291                 in.close();
292             } catch (IOException e) {}
293         } catch (Exception e)
294         {
```

```
295         errorMsg = e.toString();
296     }
297 }
298     else
299     {}
300     if (ParamExists("file3"))
301     {
302         Items2Exist = true;
303         InputStream File = null;
304         BufferedReader in = null;
305         try
306         {
307             URL FileURL = new URL(getDocumentBase(), getParameter("file3"));
308             //open a buffered reader
309             in = new BufferedReader(new InputStreamReader(FileURL.openStream()));
310             s1 = in.readLine();
311             int ItemCount = Integer.parseInt(s1);
312             Items2 = CreateItemArray(ItemCount);
313             //now read all the items
314             for (int i=0;i<ItemCount;i++)
315             {
316                 Items2[i] = in.readLine();
317             }
318             try
319             {
320                 in.close();
321             } catch (IOException e) {}
322         } catch (Exception e)
323         {
324             errorMsg = e.toString();
325         }
326     }
327     else
328     {}
329     //get colors:
330     if (ParamExists("background"))
331         BackColor = DecodeColor("background");
332     else
333         BackColor = Color.white;
334     if (ParamExists("foreground"))
335         ForeColor = DecodeColor("foreground");
336     else
337         ForeColor = Color.black;
338     //get font and its style:
339     UsedFont = DecodeFont("font");
340     SmallFont = DecodeFont("smallfont");
341     BigFont = DecodeFont("bigfont");
342     UsedFontBold = DecodeFont("fontbold");
343     //initialize OffScreen Images:
```

```
344         int w = getSize().width;
345         int h = getSize().height;
346         OffScreenImage = createImage(w, h);
347         OffScreen = OffScreenImage.getGraphics();
348         OffScreen.setFont(UsedFont);
349         //init FontMetrics':
350         FontMetrics Normal      = getFontMetrics(UsedFont);
351         FontMetrics Big         = getFontMetrics(BigFont);
352         FontMetrics Small       = getFontMetrics(SmallFont);
353         FontMetrics NormalBold  = getFontMetrics(UsedFontBold);
354         //initialize other images:
355         Image1 = createImage(w, h);
356         Temp = Image1.getGraphics();
357         Temp.setColor(BackColor);
358         Temp.fillRect(0, 0, w, h);
359         Temp.setColor(ForeColor);
360         setBackground(BackColor);
361         int top = 5;
362         String s = "";
363         int fi = -1;
364         int dh = 0;
365         int str_w = 0;
366         for (int i=0;i<Items.length;i++)
367         {
368             s = Items[i];
369             fi = FoundIdent(s);
370             switch (fi)
371             {
372                 case -1: //normal font, left aligned
373                     s = ExtractString(s, fi);
374                     Temp.setFont(UsedFont);
375                     dh = Normal.getHeight();
376                     str_w = Normal.stringWidth(s);
377                     top += dh;
378                     Temp.drawString(s, 2, top);
379                     break;
380                 case 1: //big font
381                     s = ExtractString(s, fi);
382                     Temp.setFont(BigFont);
383                     dh = Big.getHeight();
384                     str_w = Big.stringWidth(s);
385                     top += dh;
386                     if (FoundIdent(s) == 3)
387                     {
388                         Temp.drawString(s, w-5-str_w, top);
389                     }
390                     else
391                     {
392                         Temp.drawString(s, div(w, 2)-div(str_w, 2), top);
```

```
393         }
394         break;
395     case 2: //small font
396         s = ExtractString(s, fi);
397         Temp.setFont(SmallFont);
398         dh = Small.getHeight();
399         str_w = Small.stringWidth(s);
400         top += dh;
401         if (FoundIdent(s) == 3)
402         {
403             Temp.drawString(s, w-5-str_w, top);
404         }
405         else
406         {
407             Temp.drawString(s, div(w, 2)-div(str_w, 2), top);
408         }
409         break;
410     case 3: //normal font bold, right aligned
411         s = ExtractString(s, fi);
412         Temp.setFont(UsedFontBold);
413         dh = NormalBold.getHeight();
414         str_w = NormalBold.stringWidth(s);
415         top += dh;
416         Temp.drawString(s, w-5-str_w, top);
417         break;
418     case 4: //normal font, left aligned, bold:
419         String s1 = ExtractString(s, fi);
420         dh = NormalBold.getHeight();
421         str_w = NormalBold.stringWidth(s1);
422         String s2 = ExtractString(s, 10);
423         int str_w1 = Normal.stringWidth(s2);
424         top += dh;
425         Temp.setFont(UsedFontBold);
426         Temp.drawString(s1, 2, top);
427         Temp.setFont(UsedFont);
428         Temp.drawString(s2, 2+str_w, top);
429         break;
430     }
431 }
432 //initialize other images:
433 Image2 = createImage(w, h);
434 Temp = Image2.getGraphics();
435 Temp.setColor(BackColor);
436 Temp.fillRect(0, 0, w, h);
437 Temp.setColor(ForeColor);
438 setBackground(BackColor);
439 top = 5;
440 s = "";
441 fi = -1;
```

```
442         dh = 0;
443         str_w = 0;
444         for (int i=0;i<Items1.length;i++)
445         {
446             s = Items1[i];
447             fi = FoundIdent(s);
448             switch (fi)
449             {
450                 case -1: //normal font, left aligned
451                     s = ExtractString(s, fi);
452                     Temp.setFont(UsedFont);
453                     dh = Normal.getHeight();
454                     str_w = Normal.stringWidth(s);
455                     top += dh;
456                     Temp.drawString(s, 2, top);
457                     break;
458                 case 1: //big font
459                     s = ExtractString(s, fi);
460                     Temp.setFont(BigFont);
461                     dh = Big.getHeight();
462                     str_w = Big.stringWidth(s);
463                     top += dh;
464                     if (FoundIdent(s) == 3)
465                     {
466                         Temp.drawString(s, w-5-str_w, top);
467                     }
468                     else
469                     {
470                         Temp.drawString(s, div(w, 2)-div(str_w, 2), top);
471                     }
472                     break;
473                 case 2: //small font
474                     s = ExtractString(s, fi);
475                     Temp.setFont(SmallFont);
476                     dh = Small.getHeight();
477                     str_w = Small.stringWidth(s);
478                     top += dh;
479                     if (FoundIdent(s) == 3)
480                     {
481                         Temp.drawString(s, w-5-str_w, top);
482                     }
483                     else
484                     {
485                         Temp.drawString(s, div(w, 2)-div(str_w, 2), top);
486                     }
487                     break;
488                 case 3: //normal font bold, right aligned
489                     s = ExtractString(s, fi);
490                     Temp.setFont(UsedFontBold);
```

```
491         dh = NormalBold.getHeight();
492         str_w = NormalBold.stringWidth(s);
493         top += dh;
494         Temp.drawString(s, w-5-str_w, top);
495         break;
496     case 4: //normal font, left aligned, bold:
497         String s1 = ExtractString(s, fi);
498         dh = NormalBold.getHeight();
499         str_w = NormalBold.stringWidth(s1);
500         String s2 = ExtractString(s, 10);
501         int str_w1 = Normal.stringWidth(s2);
502         top += dh;
503         Temp.setFont(UsedFontBold);
504         Temp.drawString(s1, 2, top);
505         Temp.setFont(UsedFont);
506         Temp.drawString(s2, 2+str_w, top);
507         break;
508     }
509 }
510 //initialize other images:
511 if (Items2Exist)
512 {
513     Image3 = createImage(w, h);
514     Temp = Image3.getGraphics();
515     Temp.setColor(BackColor);
516     Temp.fillRect(0, 0, w, h);
517     Temp.setColor(ForeColor);
518     setBackground(BackColor);
519     top = 5;
520     s = "";
521     fi = -1;
522     dh = 0;
523     str_w = 0;
524     for (int i=0; i<Items2.length; i++)
525     {
526         s = Items2[i];
527         fi = FoundIdent(s);
528         switch (fi)
529         {
530             case -1: //normal font, left aligned
531                 s = ExtractString(s, fi);
532                 Temp.setFont(UsedFont);
533                 dh = Normal.getHeight();
534                 str_w = Normal.stringWidth(s);
535                 top += dh;
536                 Temp.drawString(s, 2, top);
537                 break;
538             case 1: //big font
539                 s = ExtractString(s, fi);
```

```
540         Temp.setFont(BigFont);
541         dh = Big.getHeight();
542         str_w = Big.stringWidth(s);
543         top += dh;
544         if (FoundIdent(s) == 3)
545         {
546             Temp.drawString(s, w-5-str_w, top);
547         }
548         else
549         {
550             Temp.drawString(s, div(w, 2)-div(str_w, 2), top);
551         }
552         break;
553     case 2: //small font
554         s = ExtractString(s, fi);
555         Temp.setFont(SmallFont);
556         dh = Small.getHeight();
557         str_w = Small.stringWidth(s);
558         top += dh;
559         if (FoundIdent(s) == 3)
560         {
561             Temp.drawString(s, w-5-str_w, top);
562         }
563         else
564         {
565             Temp.drawString(s, div(w, 2)-div(str_w, 2), top);
566         }
567         break;
568     case 3: //normal font bold, right aligned
569         s = ExtractString(s, fi);
570         Temp.setFont(UsedFontBold);
571         dh = NormalBold.getHeight();
572         str_w = NormalBold.stringWidth(s);
573         top += dh;
574         Temp.drawString(s, w-5-str_w, top);
575         break;
576     case 4: //normal font, left aligned, bold:
577         String s1 = ExtractString(s, fi);
578         dh = NormalBold.getHeight();
579         str_w = NormalBold.stringWidth(s1);
580         String s2 = ExtractString(s, 10);
581         int str_w1 = Normal.stringWidth(s2);
582         top += dh;
583         Temp.setFont(UsedFontBold);
584         Temp.drawString(s1, 2, top);
585         Temp.setFont(UsedFont);
586         Temp.drawString(s2, 2+str_w, top);
587         break;
588 }
```

```
589     }
590     }
591 }
592
593 public void update(Graphics g)
594 {
595     paint(g);
596 }
597
598 public void destroy()
599 {
600     OffScreen.dispose();
601 }
602
603 public boolean mouseMove(Event event, int i, int j)
604 {
605     if (((i >= l1) && (i <= r1)) && ((j >= t1) && (j <= b1)))
606     {
607         if (OverLeft != true)
608         {
609             OverLeft = true;
610             repaint();
611         }
612     }
613     else
614     {
615         if (OverLeft)
616         {
617             OverLeft = false;
618             repaint();
619         }
620     }
621     int w = getSize().width-1;
622     if (((i <= w-l1) && (i >= w-r1)) && ((j >= t1) && (j <= b1)))
623     {
624         if (OverRight != true)
625         {
626             OverRight = true;
627             repaint();
628         }
629     }
630     else
631     {
632         if (OverRight)
633         {
634             OverRight = false;
635             repaint();
636         }
637     }
```

```
638         return true;
639     }
640
641     public boolean mouseExit(Event event, int i, int j)
642     {
643         if (OverRight)
644         {
645             OverRight = false;
646             repaint();
647         }
648         if (OverLeft)
649         {
650             OverLeft = false;
651             repaint();
652         }
653         return true;
654     }
655
656     public boolean mouseDown(Event event, int i, int j)
657     {
658         if (OverLeft)
659         {
660             CurrentImage--;
661             if (CurrentImage == 0)
662             {
663                 if (Items2Exist)
664                     CurrentImage = 3;
665                 else
666                     CurrentImage = 2;
667             }
668         }
669         if (OverRight)
670         {
671             CurrentImage++;
672             if ((Items2Exist == false) && (CurrentImage == 3))
673                 CurrentImage = 1;
674             else
675             {
676                 if (CurrentImage == 4)
677                     CurrentImage = 1;
678             }
679         }
680         repaint();
681         return true;
682     }
683
684     public void paint(Graphics canvas)
685     {
686         //fill rectangel with background color:
```

```
687         OffScreen.setColor(BackColor);
688         OffScreen.fillRect(0, 0, getSize().width, getSize().height);
689         OffScreen.setColor(ForeColor);
690         setBackground(BackColor);
691         //draw on OffScreenImage:
692         switch (CurrentImage)
693         {
694             case 1:
695                 OffScreen.drawImage(Image1, 0, 0, this);
696                 break;
697             case 2:
698                 OffScreen.drawImage(Image2, 0, 0, this);
699                 break;
700             case 3:
701                 OffScreen.drawImage(Image3, 0, 0, this);
702                 break;
703         }
704         //draw buttons:
705         //left button:
706         OffScreen.setColor(ModifyColor(BackColor, -20));
707         OffScreen.fillRect(l1, t1, r1-l1+1, b1-t1+1);
708         OffScreen.setColor(ModifyColor(BackColor, -40));
709         OffScreen.fillRect(l1+1, t1+1, r1-l1-1, b1-t1-1);
710         OffScreen.setColor(ModifyColor(BackColor, -60));
711         OffScreen.fillRect(l1+2, t1+2, r1-l1-3, b1-t1-3);
712         OffScreen.setColor(ModifyColor(ForeColor, 150));
713         {
714             int h = b1-t1-6;
715             int w = r1-l1-6;
716             int b = div (h, 2);
717             int px[] = {div(w,2)+l1-b, div(w,2)+l1+b, div(w,2)+l1+b};
718             int py[] = {div(h,2)+t1+3, div(h,2)+t1+3+b, div(h,2)+t1+3-b};
719             int points = px.length;
720             Polygon poly = new Polygon(px, py, points);
721             OffScreen.fillPolygon(poly);
722         }
723         //right button:
724         {
725             int wi = getSize().width-1;
726             OffScreen.setColor(ModifyColor(BackColor, -20));
727             OffScreen.fillRect(wi-r1, t1, r1-l1+1, b1-t1+1);
728             OffScreen.setColor(ModifyColor(BackColor, -40));
729             OffScreen.fillRect(wi-r1+1, t1+1, r1-l1-1, b1-t1-1);
730             OffScreen.setColor(ModifyColor(BackColor, -60));
731             OffScreen.fillRect(wi-r1+2, t1+2, r1-l1-3, b1-t1-3);
732             OffScreen.setColor(ModifyColor(ForeColor, 150));
733             int h = b1-t1-6;
734             int w = r1-l1-6;
735             int b = div (h, 2);
```

```
736     int px[] = {wi-div(w,2)-l1+b, wi-div(w,2)-l1-b, wi-div(w,2)-l1-b};
737     int py[] = {div(h,2)+t1+3, div(h,2)+t1+3+b, div(h,2)+t1+3-b};
738     int points = px.length;
739     Polygon poly = new Polygon(px, py, points);
740     OffScreen.fillPolygon(poly);
741 }
742 //draw activated button:
743 if (OverLeft)
744 {
745     OffScreen.setColor(ModifyColor(BackColor, -40));
746     OffScreen.fillRect(l1, t1, r1-l1+1, b1-t1+1);
747     OffScreen.setColor(ModifyColor(BackColor, -100));
748     OffScreen.fillRect(l1+1, t1+1, r1-l1-1, b1-t1-1);
749     OffScreen.setColor(ModifyColor(BackColor, -140));
750     OffScreen.fillRect(l1+2, t1+2, r1-l1-3, b1-t1-3);
751     OffScreen.setColor(ForeColor);
752     int h = b1-t1-6;
753     int w = r1-l1-6;
754     int b = div (h, 2);
755     int px[] = {div(w,2)+l1-b, div(w,2)+l1+b, div(w,2)+l1+b};
756     int py[] = {div(h,2)+t1+3, div(h,2)+t1+3+b, div(h,2)+t1+3-b};
757     int points = px.length;
758     Polygon poly = new Polygon(px, py, points);
759     OffScreen.fillPolygon(poly);
760 }
761 else
762 {
763 }
764 if (OverRight)
765 {
766     int wi = getSize().width-1;
767     OffScreen.setColor(ModifyColor(BackColor, -40));
768     OffScreen.fillRect(wi-r1, t1, r1-l1+1, b1-t1+1);
769     OffScreen.setColor(ModifyColor(BackColor, -100));
770     OffScreen.fillRect(wi-r1+1, t1+1, r1-l1-1, b1-t1-1);
771     OffScreen.setColor(ModifyColor(BackColor, -140));
772     OffScreen.fillRect(wi-r1+2, t1+2, r1-l1-3, b1-t1-3);
773     OffScreen.setColor(ForeColor);
774     int h = b1-t1-6;
775     int w = r1-l1-6;
776     int b = div (h, 2);
777     int px[] = {wi-div(w,2)-l1+b, wi-div(w,2)-l1-b, wi-div(w,2)-l1-b};
778     int py[] = {div(h,2)+t1+3, div(h,2)+t1+3+b, div(h,2)+t1+3-b};
779     int points = px.length;
780     Polygon poly = new Polygon(px, py, points);
781     OffScreen.fillPolygon(poly);
782 }
783 //copy OffScreenImage to Screen:
784 canvas.drawImage(OffScreenImage, 0, 0, this);
```

```
785     }
786
787     public String getAppletInfo()
788     {
789         return "Title: Menucard-Applet \nAuthor: Klaus Burgstaller \nThis applet was especially created for \nthe
Schärdinger Wurstkuchl \nlast modified on 29.12.2001 \ncopyright (c) by Klaus Burgstaller. \nwww.crosswinds.net/~burgstaller";
790     }
791
792     public String[][] getParameterInfo()
793     {
794         String[][] info = {
795             {"file1-3", "path string", "Specifies file containing data to display as relative path. \nThis parameter has no default value. The component doesn't work otherwise!"},
796             {"foreground", "hex color", "Foreground color in hex format: RRGGBB"},
797             {"background", "hex color", "Works the same way for the back ground color"},
798             {"fonts", "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 ."},
799         };
800         return info;
801     }
802 }
```