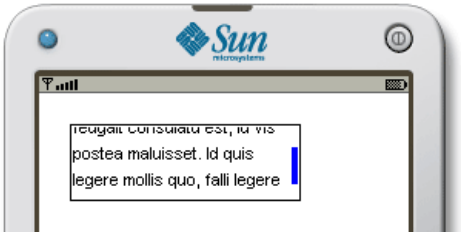
Texto com barra de rolagem, usando Canvas, em Java ME

Aqui está um exemplo em J2ME mostrando como pode ser implementda, a exibição de texto com uma barra de rolagem. (Postarei o código do MIDlet completo em breve)



Este é um simples exemplo mostrando o seguinte código em ação: Media:ScrollableTextField.zip, o qual você pode testar também com o emulador nesta página

Nós iniciaremos definindo algumas variáveis, para definir o layout do nosso elemento de texto:

```
static final int SCROLL_STEP = 25;

int scrollbarWidth = 4;
int scrollbarHeight = 0;
int scrollbarTop = 0;
int scrollbarColor = 0x0000ff;

int borderWidth = 1;
int borderColor = 0x000000;
int bgColor = 0xffffff;

Font textFont = Font.getDefaultFont();
int textColor = 0x000000;

int padding = 1;
int interline = 2;
```

Nós definiremos algumas variáveis que serão usadas internamente:

```
static final String VOID_STRING = "";
static final char SPACE_CHAR = ' ';

int width = 0;
int height = 0;
int innerWidth = 0;
int innerHeight = 0;

int currentY = 0;
int textHeight = 0;

String[] textRows = null;
```

Agora, vamos definir um construtor simples que aceite como parâmetros, largura e altura:

```
public ScrollableTextFieldExt(int width, int height)
```


(C) Copyright Nokia 2014. All rights reserved.

```
{
this.width = width;
this.height = height;

this.innerWidth = width - 2 * borderWidth - 2 * padding - scrollbarWidth;
this.innerHeight = height - 2 * borderWidth - 2 * padding;
}
```

Agora, é a hora de configurar algum texto dentro deste elemento, você não acha? Aqui está o método:

```
public void setText(String text)
{
 this.textRows = getTextRows(text, textFont, innerWidth);

 this.textHeight = textRows.length * (interline + textFont.getHeight());

 scrollbarHeight = Math.min(innerHeight, innerHeight * innerHeight / textHeight);

 scrollbarTop = 0;

 currentY = 0;
}
```

E, vamos gerenciar a barra de rolágem deste elemento, com estes métodos:

```
public void scrollDown()
{
 scroll(SCROLL_STEP);
}
public void scrollUp()
{
 scroll(- SCROLL_STEP);
}
private void scroll(int delta)
{
 currentY += delta;

 if(currentY < 0)
 {
  currentY = 0;
 }
 else if(currentY > textHeight - innerHeight)
 {
  currentY = Math.max(0, textHeight - innerHeight);
 }

 scrollbarTop = innerHeight * currentY / textHeight;
}
```

O método **getTextRows()** retornará as linhas de texto, divididas de acordo com as dadas **Font(fonte)** e **width(largura)**. Uma possível implementação é a seguinte: (note que esta implementação não dividirá palavras simples, mesmo que estas sejam maior do que a largura determinada):

```
public static String[] getTextRows(String text, Font font, int width) {
```


(C) Copyright Nokia 2014. All rights reserved.

```
 char spaceChar = ' ';

 //conterá as linhas de texto
 Vector rowsVector = new Vector();

 //conterá a linha de texto corrente
 StringBuffer currentRowText = new StringBuffer();

 //índices utilizados para quebrar as palavras
 int prevIndex = 0;
 int currIndex = text.indexOf(spaceChar);

 if(currIndex == -1)
      currIndex = text.length();

 //conterá as larguras da linha corrente e do token
 int rowWidth = 0;
 int tokenWidth = 0;

 //largura de um espaço simples
 int whitespaceWidth = font.stringWidth(" ");

 //token corrente do texto
 String currentToken = null;

 while (currIndex != -1) {
  //obtém o token corrente
  currentToken = text.substring(prevIndex, currIndex);

  //obtém a largura do token corrente
  tokenWidth = font.stringWidth(currentToken);

  //..e atualiza a largura da linha
  rowWidth += tokenWidth;

  //se a linha não está vazia, adiciona também o espaço em branco
  if (currentRowText.length() > 0) {
   rowWidth += whitespaceWidth;
  }

  //se o nova largura da linha é maior que o tamanho máximo da largura, e a linha
anterior não está vazia
  if (currentRowText.length() > 0 && rowWidth > width) {
   //adiciona o texto da linha corrente para o vetor de linhas
   rowsVector.addElement(currentRowText.toString());

   //reinicializa a linha corrente com o token corrente
   currentRowText.setLength(0);
   currentRowText.append(currentToken);

   //e atualiza a largura da linha corrente
   rowWidth = tokenWidth;
  } else {
   //se a linha corrente não é vazia, adiciona um espaço
   if (currentRowText.length() > 0)
    currentRowText.append(spaceChar);
```


(C) Copyright Nokia 2014. All rights reserved.

```
    //e então adiciona o token corrente
    currentRowText.append(currentToken);
   }

   //checa se o texto está terminado
   if (currIndex == text.length())
    break;

   //atualiza índices
   prevIndex = currIndex + 1;

   currIndex = text.indexOf(spaceChar, prevIndex);

   if (currIndex == -1)
    currIndex = text.length();
  }

  //Finalmente, adiciona a linha corrente, se não for vazia
  if (currentRowText.length() > 0) {
   rowsVector.addElement(currentRowText.toString());
  }

  //Converte o vetor de linhas num array de String
  String[] rowsArray = new String[rowsVector.size()];

  rowsVector.copyInto(rowsArray);

  return rowsArray;
 }
```

E finalmente, nós deveremos desenhar este elemento :)

```
 public void paint(Graphics g)
 {
  g.setColor(borderColor);
  g.fillRect(0, 0, width, height);

  g.setColor(bgColor);
  g.fillRect(borderWidth, borderWidth, width - 2 * borderWidth, height - 2 *
 borderWidth);

  g.setColor(textColor);
  g.setFont(textFont);

  g.translate(borderWidth + padding, borderWidth + padding);

  g.setClip(0, 0, innerWidth, innerHeight);

  if(textRows != null)
  {
   for(int i = 0; i < textRows.length; i++)
   {
    g.drawString(textRows[i], 0, i * (textFont.getHeight() + interline) - currentY,
 Graphics.TOP | Graphics.LEFT);
   }
  }
```


(C) Copyright Nokia 2014. All rights reserved.

```
 g.setClip(0, 0, width, height);

 g.setColor(scrollbarColor);
 g.fillRect(innerWidth, scrollbarTop, scrollbarWidth, scrollbarHeight);

 g.translate(- (borderWidth + padding), - (borderWidth + padding));
}
```


(C) Copyright Nokia 2014. All rights reserved.