LPII	Formatando do JTextField	Módulo III

Limitando caracteres com JTextField

Controlar a entrada de texto num componente *JTextField* não é tão simples e ainda causa muitas dúvidas para os desenvolvedores que trabalham com Swing em java. A classe *javax.swing.JTextField* é uma subclasse da classe abstrata JTextComponent e o modelo de dados padrão é um objeto da classe *PlainDocument*, subclasse de *AbstractDocument* que implementa a interface *Document*. Um dos métodos da classe *PlainDocument* é o *insertString()* que é herdado da superclasse abstrata *AbstractDocument*. A assinatura do método é:

```
 public void insertString(int offset, String str,
AttributeSet attr) throws BadLocationException
```

O argumento *offset* indica o deslocamento (a posição) inicial onde o objeto String *str* deverá ser inserido. O *AttributeSet* seta os atributos do *JTextField,* como os estilos.

Sempre que o conteúdo de um *JTextField* é modificado, o método *insertString()* é invocado.

Segue na **Listagem 1** um exemplo de como poderíamos utilizar o método *insertString* para criar um novo modelos de dados que seja capaz de restringir a quantidade de caracteres aceitos no modelo de dados de qualquer *JTextField*.

Listagem 1. Criando um novo modelo de dados para restringir o número de caracteres.

```
 package com.exemplo;
import javax.swing.text.AttributeSet;
import javax.swing.text.BadLocationException;
import javax.swing.text.PlainDocument;

public class TamanhoFixoJTextField extends PlainDocument {
  private int tamMax;
 public TamanhoFixoJTextField(int tamMax) {
```

```
                super();
                this.tamMax = tamMax;
        }

   public void insertString(int offset, String str, AttributeSet
attr)
                        throws BadLocationException {

                if (str == null)
                        return;

                //Define a condição para aceitar qualquer número
de caracteres
          if (tamMax <= 0)
          {
              super.insertString(offset, str, attr);
              return;
          }

          int tam = (getLength() + str.length());

          //Se o tamanho final for menor, chama insertString()
aceitando a String
          if (tam <= tamMax)
              super.insertString(offset, str, attr);
          }

  }
```

O argumento recebido pelo construtor define o número máximo de caracteres aceitos. A chamada a *super()* no construtor permite que a classe ancestral faça todas as inicializações necessárias. A variável de instância *tamMax* recebe o valor do argumento do construtor. O método *insertString()* nos permite sobrepor o método definido na superclasse para que possamos defini-lo como quisermos. A exceção *BadLocationException* é disparada quando a posição de inserção na String é inválida. No corpo do método *insertString* primeiramente garantimos que não estamos recebendo uma String nula. Após isso verificamos o valor máximo de caracteres a serem aceitos, se este valor for menor ou igual a zero, tem-se que essa classe terá um comportamento igual a sua ancestral *PlainDocument*, chamando o método *insertString()* da superclasse e simplesmente a retornando. Dessa forma, se o

máximo de caracteres for menor ou igual zero, o número de caracteres aceitos será indeterminado. A classe *AbstractDocument* ainda possui alguns métodos que permitem o acesso aos dados que formam o conteúdo do *JTextField*, são eles: *getLength()*ou ainda *getText()*. O método *getLength()* permite que possamos acessar o número de caracteres atual do conteúdo. O conteúdo pode ser acessado diretamente através de *getText()*. Portanto, no código acima temos "*int tam = (getLength() + str.length())*" em que *tam* é a soma do comprimento atual do conteúdo mais o comprimento da String *str* que está por ser inserida. Se a soma for menor ou igual ao número máximo de caracteres, aceitaremos *str* através da chamada ao método *insertString* da superclasse, caso contrário não ocorrerá a chamada e o conteúdo a ser inserido será ignorado.

Para testar o código presente na **Listagem 1,** vamos fazer um exemplo utilizando um *JTextField* que segue esse modelo. Na **Listagem 2** segue um exemplo.

Listagem 2. Testando o JTextField utilizando o modelo de dados criado.

```
 package com.exemplo;

import javax.swing.JFrame;
import javax.swing.JTextField;

public class TestandoJTextField extends JFrame {

        private static final long serialVersionUID = 1L;

        private JTextField txtExemplo;

    public static void main(String[] args)
    {
       TestandoJTextField field = new TestandoJTextField();
       field.testaJTextField();
    }

    private void testaJTextField() {
             this.setTitle("Exemplo");
         this.setSize(200, 180);
```

```
            //Definimos o tamanho padrão do JTextField
            txtExemplo = new JTextField(10);

            //Passamos para o construtor o número máximo de
caracteres aceitos
            txtExemplo.setDocument(new TamanhoFixoJTextField(5));

            this.getContentPane().add(txtExemplo, "North");

            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            setVisible(true);
        }

    }
```

No exemplo mostrado no código acima, tentamos adicionar uma string na caixa de texto do programa, ou seja, o nome “Rafael”, porém como o nosso componente aceita no máximo cinco caracteres não conseguimos digitar o nome inteiro. Segue na **Figura 2** a demonstração do programa funcionando e permitindo um número máximo de caracteres.

Figura 2. Executando o programa e digitando uma string.

Um pequeno problema é quando desejamos colar na nossa *JTextField* customizada uma string maior que a permitida. Se colarmos uma string que o número máximo de caracteres seja permitido não será um problema. Por exemplo, se tentarmos copiar a String “Marco” e colar no *JTextField* ele aceitará normalmente, conforme ilustra a **Figura 3.**

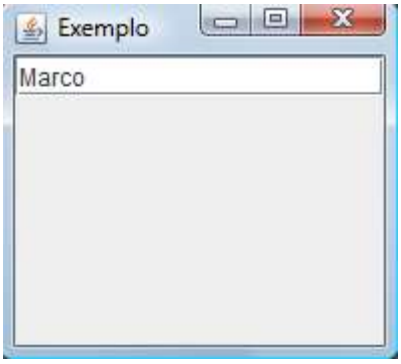

Figura 3. Colando um texto com cinco caracteres.

No entanto, se copiarmos a string “Rafael” e tentarmos colar veremos que essa string não será colada no *JTextField*, isso acontece porque esta string ultrapassa o número máximo de caracteres permitidos e portanto a string não é aceita. O ideal seria o campo de texto aceitar o número mínimo de caracteres e desprezar o restante, dessa forma, a string mostrada seria “Rafae”. Para modificar isso devemos fazer algumas modificações no código. Segue na **Listagem 3** o novo código com comentários das modificações.

Listagem 3. Modificando o código para contemplar as novas configurações.

```
package com.exemplo;

import javax.swing.text.AttributeSet;
import javax.swing.text.BadLocationException;
import javax.swing.text.PlainDocument;

public class TamanhoFixoJTextField extends PlainDocument {

        private int tamMax;

        public TamanhoFixoJTextField(int tamMax) {
              super();
              this.tamMax = tamMax;
        }

        public void insertString(int offset, String str,
AttributeSet attr)
                      throws BadLocationException {
```

```
                if (str == null)
                       return;

                //Define a condição para aceitar qualquer número
de caracteres
           if (tamMax <= 0)
           {
               super.insertString(offset, str, attr);
               return;
           }

           int tam = (getLength() + str.length());

           //Se o tamanho final for menor, chama insertString()
aceitando a String
           if (tam <= tamMax) {
               super.insertString(offset, str, attr);
           } else {
                //Caso contrário, limita a string e envia para
insertString() que aceita a string
               if (getLength() == tamMax) return;
               String novaStr = str.substring(0, (tamMax -
getLength()));
               super.insertString(offset, novaStr, attr);
           }
         }

  }
```

O bloco else adicionado no final do código acima primeiramente verifica se o comprimento atual do conteúdo é igual ao comprimento máximo, então simplesmente retornamos, caso contrário criamos um novo objeto String contendo apenas os caracteres de str que deveria caber no JTextField.

Agora se tentarmos copiar a String “Rafael” e colarmos no *JTextField* teremos como resultado a string limitada adicionada no campo, conforme mostra a **Figura 4** abaixo.

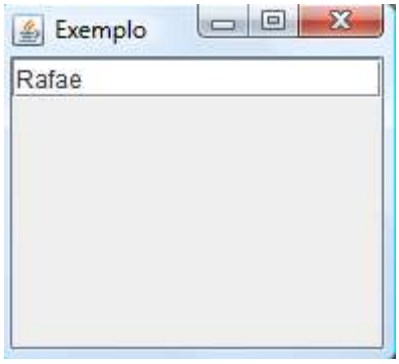

Figura 4. Colando uma string maior que o permitido no campo.

Outra situação muito comum ao trabalharmos com campos de entrada é manipularmos alguns tipos de textos com valores pré-determinados como, por exemplo, CPF, CNPJ, CEP, etc. Na próxima seção veremos como fazer isso utilizando outro componente, o *JFormattedTextField* que é semelhante ao JTextField, porém mais voltado para campos formatados.