SIMULADOS & TUTORIAIS

TUTORIAIS FLASH

O Componente TextArea

O Componente TextArea

Introdução

Esse componente é praticamente um recipiente para o objeto **TextField**. Você poderá usá-lo para mostrar texto e também editar e receber entrada de dados, caso a propriedade **editable** esteja configurada para **true**. Ele é utilizado com muita freqüência em Web sites, principalmente em formulários de cadastramento de currículos, em fóruns e opiniões sobre algum produto, entre outros usos, onde o usuário escreve com mais liberdade e com um maior limite de caracteres, e praticamente sem regras pré-estabelecidas de preenchimento (dependendo do caso).
O componente **TextArea** é nada mais nada menos do que uma caixa de texto com maiores recursos do que uma caixa de texto convencional. O texto tanto pode ser digitado como importado de um arquivo externo, como também poderá ser formatado em HTML, ou ainda como um campo para entrada de senha. Ele poderá mostrar ou receber múltiplas linhas como também quebrar longas linhas caso a propriedade **wordWrap** esteja configurada para **true**. A propriedade **restrict** permite-lhe restringir os caracteres que o usuário deverá ou não utilizar, e a propriedade **maxChars** permite especificar o número máximo de caracteres que o usuário poderá digitar. Se o texto exceder as bordas da caixa de texto, barras de rolamento horizontal ou vertical, conforme o caso, aparecerão automaticamente, a menos que as propriedades associadas **horizontalScrollPolicy** e **verticalScrollPolicy** estejam configuradas para **off**.

Utilize esse componente somente quando necessitar de um campo de texto com mais de uma linha. Para textos com apenas uma linha, utilize o campo de texto apropriado, tal como o componente **TextInput**.

Você poderá configurar o estilo **textFormat** usando o método **setStyle()** para alterar o estilo do texto que aparecerá em uma instância de um componente **TextArea**. Você também poderá formatar um componente **TextArea** com HTML usando a propriedade **htmlText** em ActionScript, e configurar a propriedade **displayAsPassword** para **true** para mascarar o texto com asteriscos. Se você configurar a propriedade **condenseWhite** para **true**, o Flash removerá os espaços extras em branco, quebras de linhas, e assim por diante.

Interação do usuário com o componente TextArea

Um componente **TextArea** tanto poderá estar habilitado quanto desabilitado em uma aplicação. Quando desabilitado, ele não poderá receber entrada nem do mouse nem do teclado. Quando habilitado, ele obedecerá as mesmas regras de foco, seleção e navegação de um objeto **TextField** do ActionScript. Quando uma instância de um componente **TextArea** estiver com o foco, você poderá utilizar as seguintes teclas para controlá-lo:

Tecla	Descrição
Teclas das setas	Move o ponto de inserção para cima, para baixo, para a esquerda ou para a direita se o texto for editável.
Page Down	Move o ponto de inserção para o final do texto, se o texto for editável.
Page Up	Move o ponto de inserção para o início do texto, se o texto for editável.
Shift + Tab	Move o foco para o objeto anterior.
Tab	Move o foco para o próximo objeto.

Parâmetros do componente TextArea

Para conhecer melhor esse componente e utilizá-lo da maneira correta, será necessário saber que recursos ele oferece e como usá-los para que o resultado retornado seja o esperado. Para isso, veja a seguir quais os parâmetros que poderão ser aplicados a ele diretamente no painel **Component Inspector** durante a fase de projeto:

Parâmetro	Descrição
condenseWhite	É um valor booleano que indica se os espaços extras em branco de um componente **TextArea** que contenha texto formatado em HTML deverão ser removidos ou não. O valor padrão é: **false**.
editable	Indica se o componente **TextArea** poderá ser editado (**true**), ou não (**false**). O valor padrão é **true**. Isso significa que você poderá copiar o seu conteúdo, apagar ou modificar.
enabled	Esse parâmetro é um valor booleano que indica se o componente deverá estar habilitado ou desabilitado quando a aplicação for executada. O valor padrão é **true**.
horizontalScrollPolicy	Permite que a barra de rolagem horizontal de um componente **TextArea** apareça automaticamente ou não quando o texto ultrapassar os limites da área. Você poderá selecionar uma das seguintes opções: **auto**, **on** ou **off**. O valor padrão é: **auto**.
htmlText	Permite configurar a representação **HTML** da string que o campo de texto contém. O valor padrão é uma string vazia: **""**.
maxChars	Define o número máximo de caracteres que a área de texto pode conter. O valor padrão é **null**, ou seja, quantidade ilimitada de caracteres.
restrict	Indica o conjunto de caracteres que o usuário poderá digitar na área de texto. O valor padrão é **undefined**. Significa que podem ser digitados quaisquer caracteres.
text	Indica o conteúdo do componente **TextArea**. Não é permitido usar o retorno de carro (**Enter**) no painel **Component Inspector**. O valor padrão é uma string vazia: "".
verticalScrollPolicy	Permite que a barra de rolagem vertical de um componente **TextArea** apareça automaticamente ou não quando o texto ultrapassar os limites da área. Você poderá selecionar uma das seguintes opções: **auto**, **on** ou **off**. O valor padrão é: **auto**.
visible	Esse parâmetro é um valor booleano que indica se o componente deverá aparecer visível ou oculto quando a aplicação for executada. O valor padrão é: **true**.
wordWrap	É um valor booleano que indica se o texto deverá quebrar a linha no final da margem (**true**), ou não (**false**). O valor padrão é **true**.

Cada um desses parâmetros possui uma propriedade equivalente em ActionScript com o mesmo nome, que podem ser utilizadas juntamente com outras propriedades, métodos e eventos disponíveis para esse componente, que permitem-lhe criar aplicações mais interativas, conforme descrevemos abaixo:

As propriedades do componente TextArea

São as seguintes as propriedades disponíveis para o componente **TextArea**:

Propriedade	Descrição
alwaysShowSelection	É um valor booleano que indica se o Flash Player deverá destacar uma seleção no campo de texto quando esse campo não possuir o foco. O valor padrão é: **false**.
displayAsPassword	É um valor booleano que indica se o conteúdo de uma instância do componente **TextArea** será um campo de texto utilizado como se fosse uma senha ou não. O valor padrão é: **false**.
horizontalScrollBar	É uma referência à barra de rolagem horizontal.
horizontalScrollPosition	É um número utilizado para configurar a alteração na posição do controle de uma barra de rolagem, em pixels, após o usuário rolar o campo de texto horizontalmente. O valor padrão é: **0**.
length	É um número que informa a quantidade de caracteres que um componente **TextArea** deverá conter. O valor padrão é: **0**.

maxHorizontalScrollPosition	É um inteiro que determina o valor máximo da propriedade **horizontalScrollPosition**. O valor padrão é: **0**.
maxVerticalScrollPosition	É um inteiro que determina o valor máximo da propriedade **verticalScrollPosition**. O valor padrão é: **1**.
selectionBeginIndex	É um inteiro que determina a posição do índice do primeiro caractere em uma seleção de um ou mais caracteres. O valor padrão é: **0**.
selectionEndIndex	É um inteiro que determina a posição do índice do último caractere em uma seleção de um ou mais caracteres. O valor padrão é: **0**.
textField	É uma referência ao campo de texto interno do componente **TextArea**.
textHeight	É um número que determina a altura do texto, em pixels. O valor padrão é: **0**. Propriedade somente de leitura.
textWidth	É um número que determina a largura do texto, em pixels. O valor padrão é: **0**. Propriedade somente de leitura.
verticalScrollBar	É uma referência à barra de rolagem vertical.
verticalScrollPosition	É um número utilizado para configurar a alteração na posição do controle de uma barra de rolagem, em pixels, após o usuário rolar o campo de texto verticalmente. O valor padrão é: **1**.

Os métodos do componente TextArea

São os seguintes os métodos disponíveis para o componente **TextArea**:

Método	Descrição
TextArea	Cria uma nova instância do componente **TextArea**.
appendText	Anexa a string especificada após o último caractere que o componente **TextArea** contém.
drawFocus	É um valor booleano que indica se o foco de um componente **TextArea** deve ser mostrado ou não. O valor padrão é: **false**.
getLineMetrics	Recupera a informação de uma linha específica do texto.
getStyleDefinition	Recupera o estilo de mapa padrão para o componente atual.
setSelection	Define a faixa de uma seleção feita em uma área de texto que possui o foco.

Os eventos do componente TextArea

Eventos são ações utilizadas pelos componentes para executar tarefas específicas, de acordo com a finalidade de cada situação. Veja a seguir os eventos disponíveis para esse componente e para que servem.

Evento	Descrição
change	É executado quando o usuário altera o texto do componente **TextArea**.
enter	É executado quando o usuário pressiona a tecla **Enter** enquanto estiver no componente.
scroll	É executado quando o conteúdo é rolado.
textInput	É executado quando o usuário entra, apaga ou cola texto no componente **TextArea**.

Veja como utilizar esses recursos nos exemplos mais adiante neste capítulo e no DVD, que acompanha a presente obra.

Criando aplicações com o componente TextArea

Para entender melhor como funciona esse componente, mostraremos a seguir alguns exemplos práticos, inclusive em conjunto com outros componentes.

Exemplo 1

Nesse exemplo utilizaremos um componente **TextArea** e preencheremos com um texto manualmente diretamente no painel **Component Inspector**. Esse é o método mais simples e prático de preencher esse componente, mas, em contrapartida, muito limitado, já que não é possível editar o texto e fazer as alterações necessárias. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**), e arraste um componente **TextArea** para o palco. Veja a aparência original desse componente na **Figura 15.1** a seguir:

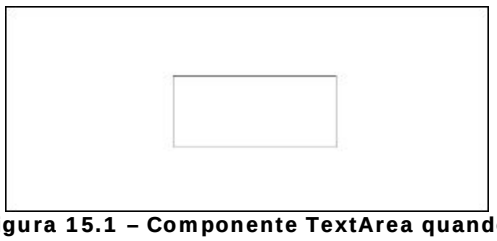


Figura 15.1 – Componente TextArea quando arrastado para o palco.

3. Redimensione-o para o tamanho: **300x100** pixels;
4. Copie um texto qualquer e cole-o no parâmetro **text** do painel **Component Inspector**;
5. Nomeie a camada atual para: **componentes**;
6. Execute a aplicação e confira o resultado com o mostrado na **Figura 15.2** abaixo:

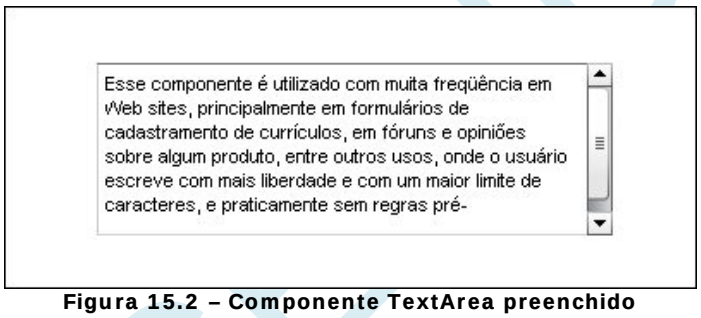


Figura 15.2 – Componente TextArea preenchido com texto manualmente.

Observe que na caixa de texto do componente será mostrada uma barra de rolagem vertical, tendo em vista que o texto ultrapassa a altura da caixa. Nesse caso, embora o texto seja editável, isto é, você possa modificá-lo, ele sempre será o mesmo toda vez que a aplicação for executada, e não será mostrado com as novas mudanças feitas, pois ele não é um texto dinâmico, ele já foi inserido previamente. Experimente.

*****

Exemplo 2

Nesse exemplo utilizaremos um componente **TextArea** cujo conteúdo será preenchido pelo usuário somente após a execução da aplicação, e o mesmo ficará totalmente escondido dos bisbilhoteiros, uma vez que configuramos a propriedade **displayAsPassword** para **true**. Além disso, limitamos o número de caracteres a serem digitados. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**), e arraste um componente **TextArea** para o palco.
3. Redimensione-o para o tamanho: **300x100** pixels;
4. Crie uma instância para o componente, por exemplo: **texto**;
5. Nomeie essa camada para: **componentes**;
6. Crie uma nova camada e chame-a de: **ações**;
7. No primeiro frame dessa camada insira o seguinte código:

```
stop();
texto.displayAsPassword = true;
texto.maxChars=200;
```

Vejamos os comentários do código acima:

OBS.: Em todos os exemplos do presente capítulo incluiremos o comando **stop()** na primeira linha do código. Esse comando tem como objetivo interromper a aplicação quando a mesma for executada, mesmo que a aplicação não possua mais de um frame na sua linha do tempo. Consideramos um bom hábito essa prática. Portanto, não comentaremos mais essa linha nos próximos exemplos.

Na linha:

```
texto.displayAsPassword = true;
```

Configuramos a propriedade **displayAsPassword** para **true**, de forma que qualquer caractere que for digitado na área de texto será convertido para asterisco, inclusive os espaços, ocultando dessa forma a verdadeira mensagem digitada.

Na linha seguinte:

```
texto.maxChars=200;
```

Limitamos o conteúdo da área de texto para um máximo de 200 caracteres a serem digitados.

8. Execute a aplicação e confira o resultado com o apresentado na **Figura 15.3** abaixo.

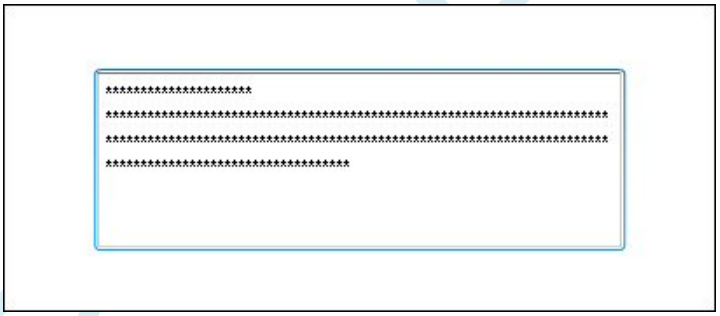


Figura 15.3 – Componente TextArea preenchido pelo usuário e substituído por asteriscos.

*****

Exemplo 3

Nesse exemplo, o texto será digitado pelo usuário, mas com algumas restrições também, como o limite no número de caracteres a serem digitados e a permissão somente de alguns caracteres. Veja como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**), e arraste um componente **TextArea** para o palco.
3. Redimensione-o para o tamanho: **300x100** pixels;
4. Crie uma instância para o componente, por exemplo: **texto**;
5. Nomeie a camada atual para: **componentes**;
6. Crie uma nova camada e chame-a de: **ações**;
7. No primeiro frame dessa camada insira o seguinte código:

```
stop();
```

```
texto.maxChars = 50;
texto.restrict = "A-Z 0-9";
```

Vejamos os comentários do código acima:

Na linha:

```
texto.maxChars = 50;
```

Definimos o número máximo de caracteres (**50**) que poderão ser digitados pelo usuário.

Na linha seguinte:

```
texto.restrict = "A-Z 0-9";
```

Restringimos os caracteres que poderão ser digitados pelo usuário. Nesse caso, só poderão ser utilizadas as letras de A a Z (elas aparecerão em maiúsculas, mesmo que sejam digitadas em minúsculas) e os números de 0 a 9. Quaisquer outros caracteres não serão permitidos.

8. Execute a aplicação e confira o resultado com o apresentado na **Figura 15.4** abaixo.

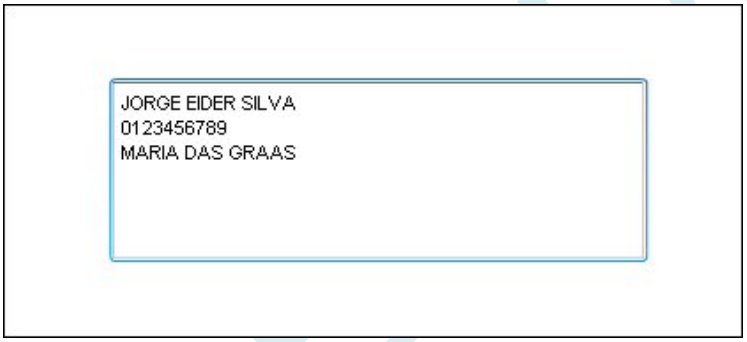


Figura 15.4 – Componente TextArea preenchido pelo usuário com restrições na digitação.

Observe que não é exibido nenhum caractere minúsculo nem caracteres especiais. Mesmo que você copie o texto de algum lugar e cole na caixa de texto, essas regras serão obedecidas. Se o texto carregado for maior que o número de caracteres estabelecido, o texto será truncado. Experimente criar exemplos com outras restrições.

*****

Exemplo 4

Esse exemplo é bem interessante. Utilizaremos para isso dois componentes **TextArea**, de forma que, o que for digitado na instância do primeiro componente (**texto1**) será copiado automaticamente na instância do segundo componente (**texto2**), como se fosse um clone, mas a operação inversa não é verdadeira. Além disso, restringimos os caracteres a serem digitados: somente caracteres maiúsculos, minúsculos e o espaço. Números não serão permitidos. Vejamos com fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**), e arraste dois componentes **TextArea** para o palco.
3. Redimensione-os para o tamanho: **200x100** pixels;
4. Crie para cada um uma instância diferente, por exemplo: **texto1** e **texto2**;
5. Nomeie a camada atual para: **componentes**;
6. Crie uma nova camada e chame-a de: **ações**;
7. Selecione o primeiro frame dessa camada e insira o seguinte código:

```
stop();
//
texto1.restrict = "a-z,A-Z, ";
texto1.addEventListener(Event.CHANGE,duplicaTexto);
//
function duplicaTexto(evento:Event):void {
    texto2.text = texto1.text;
}
```

Vejamos os comentários sobre o código acima:

Na linha:

```
texto1.restrict = "a-z,A-Z, ";
```

Utilizamos a propriedade **restrict** na primeira instância (**texto1**) do componente **TextArea** para definirmos os caracteres que poderão ser digitados, como sendo: todos os caracteres minúsculos, todos os caracteres maiúsculos (com exceção dos caracteres especiais e acentuados), e a barra de espaços.

Na linha seguinte:

```
texto1.addEventListener(Event.CHANGE,duplicaTexto);
```

Utilizamos o evento **CHANGE** vinculado à instância **texto1** do componente **TextArea**, de forma que quando o conteúdo for alterado a função **duplicaTexto** deverá ser executada.

E finalmente no bloco de código a seguir:

```
function duplicaTexto(evento:Event):void {
texto2.text = texto1.text;
}
```

Criamos a função **duplicaTexto** de forma que quando a mesma for executada, o texto digitado na instância **texto1** seja automaticamente copiado para a instância **texto2**.

8. Execute a aplicação e confira o seu resultado com o mostrado na **Figura 15.5** a seguir:

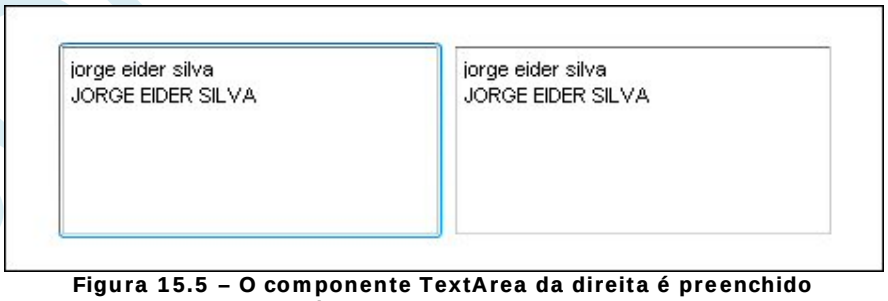


Figura 15.5 – O componente TextArea da direita é preenchido com o mesmo conteúdo digitado no componente da esquerda.

Observe que, se você digitar alguma coisa no componente da direita nada acontecerá no componente da esquerda, como também não existe nenhuma restrição para ele. Entretanto, se você voltar a digitar alguma coisa no primeiro componente, o que foi digitado no segundo será descartado. Experimente.

*****

Exemplo 5

Nesse exemplo utilizaremos um componente **TextArea** juntamente com um campo de texto dinâmico, de forma que à medida que o usuário for digitando na área de texto, o número de caracteres digitados será informado no campo de texto dinâmico, para que o usuário saiba quantos caracteres já digitou. Além disso, restringimos o número máximo de caracteres para 200. Quando o número de caracteres digitados atingir essa marca, o componente ficará desabilitado, não sendo mais possível acrescentar qualquer texto.
Esse recurso é muito utilizado em formulários encontrados em Web sites para limitar o número de caracteres digitados pelos usuários. Isso evita a digitação de textos muito extensos sem necessidade, ocupando espaços desnecessários nos servidores. Veja a seguir como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**), e arraste um componente **TextArea** para o palco.
3. Redimensione-o para o tamanho: **300x100** pixels;
4. Crie uma instância para o componente, por exemplo: **texto**;
5. Crie um texto estático logo abaixo da caixa de texto **TextArea**, e digite: **Quantidade de caracteres digitados:**;
6. Crie um texto dinâmico e digite crie uma instância para ele: **nr_caracteres**, por exemplo;
7. Nomeie a camada atual para: **componentes**;
8. Crie uma nova camada e chame-a de: **ações**;
9. No primeiro frame dessa camada insira o seguinte código:

```
stop();
//
texto.addEventListener(Event.CHANGE, nCaracteres);
//
function nCaracteres(event:Event):void {
    var caracteres=texto.length;
    nr_caracteres.text = caracteres;
    if (caracteres==200) {
            texto.enabled=false;
    }
}
```

Vejamos como funciona o código:

Na linha:

```
texto.addEventListener(Event.CHANGE, nCaracteres);
```

Utilizamos o evento **CHANGE** vinculado à instância **texto** do componente **TextArea**, de forma que quando algum caractere for digitado na caixa de texto do componente a função **nCaracteres** seja executada.

No bloco de código a seguir:

```
function nCaracteres(event:Event):void {
    var caracteres=texto.length;
    nr_caracteres.text = caracteres;
    if (caracteres==200) {
            texto.enabled=false;
    }
}
```

Criamos a função **nCaracteres** de forma que quando a mesma for executada, faça o seguinte:

- Armazene na variável **caracteres** a quantidade de caracteres digitados (**var caracteres = texto.length**).
- Mostre no campo dinâmico **nr_caracteres** o valor da variável **caracteres** (**nr_caracteres.text = caracteres**);
- A condição **IF** testa se a quantidade de caracteres digitados chegou à marca estabelecida, ou seja, 200. Em caso positivo, o componente **TextArea** ficará desabilitado encerrando a digitação, caso contrário, a digitação poderá ser continuada.

10. Execute a aplicação e confira o resultado com o mostrado na **Figura 15.6** a seguir:

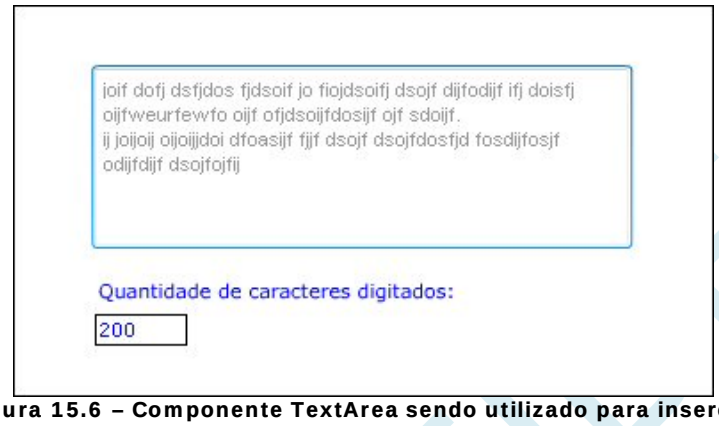


Figura 15.6 – Componente TextArea sendo utilizado para inserção de texto com restrições na quantidade de caracteres a serem digitados.

Durante a digitação você poderá utilizar os recursos de edição presentes em qualquer editor de textos convencional, como por exemplo: **delete**, **backspace**, as teclas das **setas** para navegar no texto, **Shift**, **Ctrl**, etc.

*****

Exercícios de Fixação

1. O parâmetro HTML de um componente **TextArea** indica que o texto pode ser formatado com as tags do HTML:

   a) Verdadeiro
   b) Falso

2. Que parâmetro do componente **TextArea** indica o número máximo de caracteres que pode ser digitado?

   a) charsMaximum
   b) maxChars
   c) chars.length
   d) charsLength
   e) max.Chars

3. Se a propriedade **displayAsPassword** for configurada para **true**, indica que o texto de um componente **TextArea** será substituído por:

   a) barras
   b) asteriscos
   c) pontos
   d) hífens
   e) espaços

4. O comando:

   texto.restrict = "0-9";

   Utilizado em uma instância de um componente **TextArea** chamada **texto**, indica que:

   a) Só poderão ser digitados os números de 0 a 9.
   b) Não poderão ser digitados os números de 0 a 9.
   c) Só poderão ser digitados os números 0 e 9.
   d) Não poderão ser digitados os números 0 e 9.
   e) Só poderão ser digitados os números 0 a 9 e o hífen.

5. Para sabermos a quantidade de caracteres de um componente **TextArea**, utilizamos a seguinte propriedade:

   a) maxChars
   b) numberChars
   c) charsLength
   d) charsSelected
   e) length

6. Que método determina se o foco de um componente **TextArea** deve ser mostrado ou oculto?

   a) drawFocus()
   b) set.Focus()
   c) getFocus()
   d) draw.Focus()
   e) get.Focus()

7. Considerando a linha de código abaixo, vinculada a um componente **TextArea**, cuja instância se chama **texto**:

   texto.restrict = "A-Z^QX";

   Indica que:

   a) Só poderão ser digitados os caracteres QX.
   b) Só poderão ser digitados os caracteres A e Z com exceção de Q e X.
   c) Só poderão ser digitados os caracteres de A a Z com exceção de Q e X.
   d) Só poderão ser digitados os caracteres A, Z, Q e X.
   e) Só poderão ser digitados os caracteres de A a Z com exceção de Q a X.

8. O parâmetro **wordWrap** de um componente **TextArea** quando configurado para **true** indica que o texto:

   a) Será mostrado em uma única linha.
   b) Será quebrado quando atingir a margem direita da área de texto.
   c) Ficará desabilitado.
   d) Poderá ser formatado para HTML.
   e) Será hifenizado.

9. O parâmetro **restrict** de um componente **TextArea** define o conjunto de caracteres que deverão ser digitados:

   a) Verdadeiro
   b) Falso

10. O valor padrão do parâmetro **text** de um componente **TextArea** é uma string vazia:

    a) Verdadeiro
    b) Falso

11. Qual o método utilizado para criarmos uma nova instância de um componente **TextArea** em tempo de execução de nome: **texto**?

    a) var texto:TextArea = new TextArea();
    b) var texto:TextArea = new.TextArea();
    c) var texto-TextArea = new TextArea();
    d) var texto:TextArea = newTextArea();
    e) var texto.TextArea = new TextArea();

12. Quais as opções disponíveis nas propriedades **horizontalScrollPolicy** e **verticalScrollPolicy** de um componente **TextArea**?

    a) auto, true e off
    b) true e false
    c) auto, true e false
    d) auto, on e off
    e) true e off

13. Quais os valores padrões das propriedades **maxHorizontalScrollPosition** e **maxVerticalScrollPosition** respectivamente de um componente **TextArea**?

    a) 0 e 1
    b) 1 e 0
    c) 0 e 10
    d) 1 e 1
    e) 10 e 10

14. Quais os valores padrões das propriedades **textHeight** e **textWidth** respectivamente de um componente **TextArea**?

    a) 0 e 1
    b) 1 e 0
    c) 0 e 10
    d) 1 e 1
    e) 0 e 0

15. Quais os valores padrões das propriedades **selectionBeginIndex** e **selectionEndIndex** respectivamente de um componente **TextArea**?

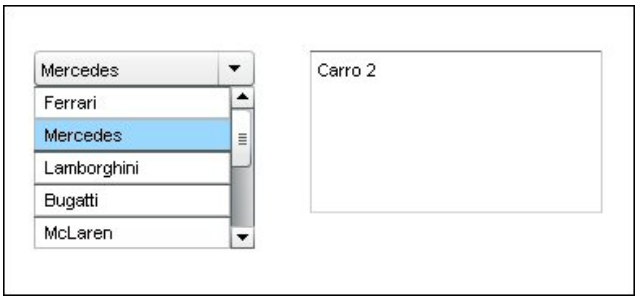
    a) 0 e 1
    b) 1 e 0
    c) 0 e 0
    d) 1 e 1
    e) 10 e 10

*****

Exercícios Propostos

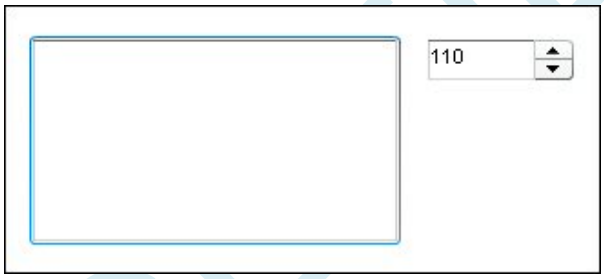
1. Desenvolva uma aplicação utilizando dois componentes, um **ComboBox** e um **TextArea**. No **ComboBox**, adicione uma lista de 10 itens, de forma que, quando um item do **ComboBox** for selecionado apareça no **TextArea** um pequeno resumo desse

item proveniente de uma matriz. Veja uma sugestão de layout para esse exercício na figura abaixo:



2. Desenvolva uma aplicação utilizando um componente **TextArea**, de forma que quando a tecla **Enter** for pressionada após um texto ser digitado, o componente **TextArea** fique desabilitado, não permitindo ao usuário digitar mais nenhum caractere. Utilize o evento **ENTER** para isso.

3. Crie uma aplicação utilizando um componente **TextArea** e um componente **NumericStepper**, de forma que quando o valor do **NumericStepper** for alterado, a altura do **TextArea** seja alterada pelo mesmo valor, em pixels. Estabeleça uma faixa entre 20 e 200 pixels para o **NumericStepper** com intervalos de 5 pixels. Veja uma sugestão de layout para esse exercício na figura abaixo:



*****