Desenvolvimento Desktop

Aula 10 - Eventos e Listeners

Apresentação

Nesta aula, você verá um assunto diferente. Trata-se dos eventos e dos escutadores de eventos (listeners) utilizados em Java. Eventos são ações que associamos aos componentes para que os mesmos possam executar alguma tarefa. Por exemplo, clicar o botão do mouse, girar a rodinha do mouse, passar o mouse sobre um objeto, arrastar um componente para outro lugar, utilizar as teclas para dar entrada nas informações, entre outros. Todos os componentes que você conheceu nas aulas anteriores não servem de nada se você não implementar uma ação para eles, ou seja, criar um evento para eles. Você deve se lembrar que nós utilizamos alguns eventos em alguns componentes de aulas anteriores, como o **Controle deslizante** e o **Controle giratório**. Entretanto, nesta aula, você conhecerá outros eventos e sua utilização em alguns componentes já conhecidos seus.

Vídeo 01 - Apresentação

Objetivos

Ao final desta aula, você será capaz de:

- Descrever Eventos e Listeners e aprender como utilizá-los com alguns componentes já conhecidos.
- Utilizar alguns Eventos do mouse em diversas situações.
- Aplicar os Eventos **Focus** e saber o que fazer quando um componente recebe ou perde o foco.
- Reconhecer o Evento certo no componente certo.

O que são Eventos

Nas aulas sobre os componentes, você descobriu como criar frames no NetBeans e neles incluir painéis, rótulos, botões, combo boxes, entre outros. Entretanto, esses componentes não fazem nada se você não disser a eles o que fazer. Eles são completamente inúteis, mesmo ao serem clicados.

Nesta aula, você aprenderá como criar o código necessário para que um componente possa fazer alguma coisa quando o usuário clicar em um botão, por exemplo. Essa técnica chama-se **event listening** (traduzido grosseiramente como: escutador de eventos), que é um dos mais importantes aspectos de um programa Java. Pode parecer um pouco complicado inicialmente entender como isso funciona, mas você se adaptará.

Embora um **event listening** seja utilizado com maior frequência para responder a um clique do mouse, ele também pode ser utilizado para responder outros tipos de interações do usuário. Por exemplo, você poderá utilizar um **event listening** para executar alguma ação quando o usuário selecionar um item de um combo box, mover o mouse sobre um rótulo ou pressionar uma tecla do teclado, como **TAB** ou **ENTER**. Você saberá como alguns eventos funcionam e aprenderá a utilizá-los de acordo com a necessidade de cada situação.

Examinando os Eventos

Um evento é uma ação que é gerada quando o usuário faz alguma coisa em um componente. Por exemplo, se o usuário clica em um botão, arrasta o mouse sobre um rótulo ou seleciona um item de um list box, uma ação é gerada. Esse evento (ação) é então passado a um método especial que você criou chamado **event listener**.

Um **event listener** pode examinar um evento, determinar exatamente que tipo de evento ocorreu e responder de acordo com o solicitado. Por exemplo, se o usuário clica em um botão, o **event listener** poderia mostrar uma janela de diálogo

com alguma informação ou desabilitar um outro componente. Se o usuário passar o mouse sobre um rótulo, o **event listener** poderia alterar o texto desse mesmo rótulo ou a cor de fundo. E se o usuário selecionar um item em um combo box ou list box, o **event listener** poderia utilizar o valor que foi selecionado para procurar a informação em um banco de dados. As possibilidades são infinitas.

Há vários tipos de eventos para cada tipo de situação. Veja, no **Quadro 1**, os eventos mais utilizados, suas classes e suas respectivas finalidades:

Evento	Classe	Descrição
ActionEvent	ActionListener	É criado quando o usuário executa uma ação (clique) com um botão ou outro componente.
AdjustmentEvent	AdjustmentListener	É gerado quando um componente é ajustado. Por exemplo, quando uma barra de rolagem é movida.
DocumentEvent	DocumentListener	Esse evento ocorre quando o usuário altera o conteúdo de um componente do tipo texto, tal como um TextField (campo de texto).
FocusEvent	FocusListener	É criado quando um componente recebe ou perde o foco.
ItemEvent	ItemListener	É criado quando o item selecionado em uma lista, tal como um combo box ou list box, é alterado.
KeyEvent	KeyListener	É gerado quando o usuário pressiona uma tecla no teclado. Esse evento pode ser usado para verificar que tecla foi pressionada pelo usuário, se uma tecla de texto ou de controle, e para tomar a decisão correta.

Evento	Classe	Descrição
MouseEvent	MouseListener	Esse evento ocorre quando o usuário faz algo com o mouse, como clicar um botão, arrastar ou simplesmente mover sobre um outro objeto.
WindowEvent	WindowListener	É gerado quando uma janela é movida, minimizada, maximizada ou fechada.

Quadro 1 - Eventos mais utilizados e suas respectivas classes e finalidades

Observação

A maioria dessas classes de eventos está contida no pacote **java.awt** do Java, com exceção de **DocumentEvent**, por se tratar de uma interface e não de uma classe, e se encontra no pacote **javax.swing.event**.

Quando você for criar um evento para um componente, atente para o seguinte:

- **Event** — um objeto que é criado quando o usuário faz alguma coisa com um componente, tal como um clique.
- **Event Source** — o componente que gerou o objeto evento. Geralmente, o **event source** éum botão ou outro componente que o usuário pode clicar, mas qualquer componente Swing pode ser um **event source**.
- **Event Listener** — o objeto que **escuta** os eventos e manipula-os quando eles ocorrem. O objeto **event listener** deve implementar a interface apropriada para o evento. Essas interfaces **listeners** definem o método ou métodos que o **event source** chama quando o evento ocorre.

Tanto você pode criar um evento para um componente e este executar uma ação de imediato, como também criar um evento para um componente executar um método e esse método executar o que se deseja. Você também poderá criar tantos eventos quantos forem necessários para um único componente, e não apenas um evento para cada componente.

Vídeo 02 - Eventos

Como Utilizar os Eventos em uma Aplicação

Nesta aula, veremos alguns eventos utilizados pelo mouse, tais como clicar, arrastar, passar sobre um objeto e fora dele. Veja, no **Quadro 2**, que eventos estão disponíveis para o mouse:

Evento	Descrição
mouseClicked	Esse evento ocorre quando o mouse é clicado sobre um componente.
mouseEntered	Esse evento ocorre quando o mouse é passado sobre um componente.
mouseExited	Esse evento ocorre quando o mouse deixa a área do componente.
mousePressed	Esse evento ocorre quando o mouse é pressionado sobre um componente.
mouseReleased	Esse evento ocorre quando o mouse é liberado de um componente.

Evento	Descrição
mouseMotion	Existem duas opções para esse evento: **mouseDragged** (quando o botão do mouse é clicado e o mouse é arrastado para outra posição) e **mouseMoved** (quando o mouse é simplesmente arrastado, sem necessidade de clicar nenhum botão).
mouseWheel	Esse evento ocorre quando a rodinha do mouse é girada, ou seja, é movida para qualquer uma das direções. Para utilizar esse evento, você deve usar a opção **mouseWheelMoved**.

Quadro 2 - Listagem dos eventos aplicados ao mouse

Evento	Descrição
keyPressed	Esse evento ocorre quando uma tecla é pressionada.
keyReleased	Esse evento ocorre quando uma tecla é liberada.
keyTyped	Esse evento ocorre quando uma tecla é digitada.

Quadro 3 - Listagem dos eventos aplicados ao teclado

Vejamos, então, os exemplos

Exemplo 1

Nesse exemplo, utilizaremos um botão, um rótulo e o evento **mouseClicked**, de forma que quando o usuário clicar no botão, a cor de fundo e da fonte do rótulo sejam alteradas para amarelo e vermelho respectivamente, como também o próprio botão ficará desabilitado.

1. Execute o NetBeans e crie um novo projeto (**Arquivo -> Novo projeto**).

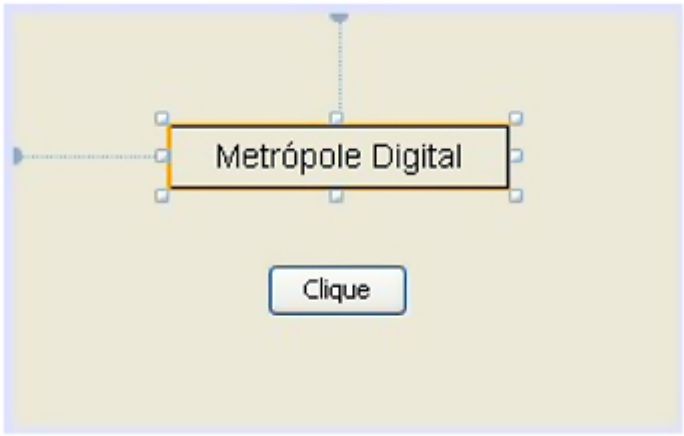
2. Salve seu projeto com o nome: **Eventos e Listeners — Exercicio1**.

3. Crie um novo **Formulário JFrame**.

4. Clique na propriedade **title** do **JFrame** e digite um titulo para ele, por exemplo: **Eventos do Mouse — Exercício 1**.

5. Arraste um componente botão (**JButton**) e um **Rótulo (JLabel)**. Na propriedade **text** de cada um, digite as respectivas informações de acordo com a **Figura 1**. Sendo que, para a informação contida no rótulo, altere a fonte para **Arial**, tamanho **14**, centralize o texto, marque o check box da opção **opaque** e crie uma borda para o componente.

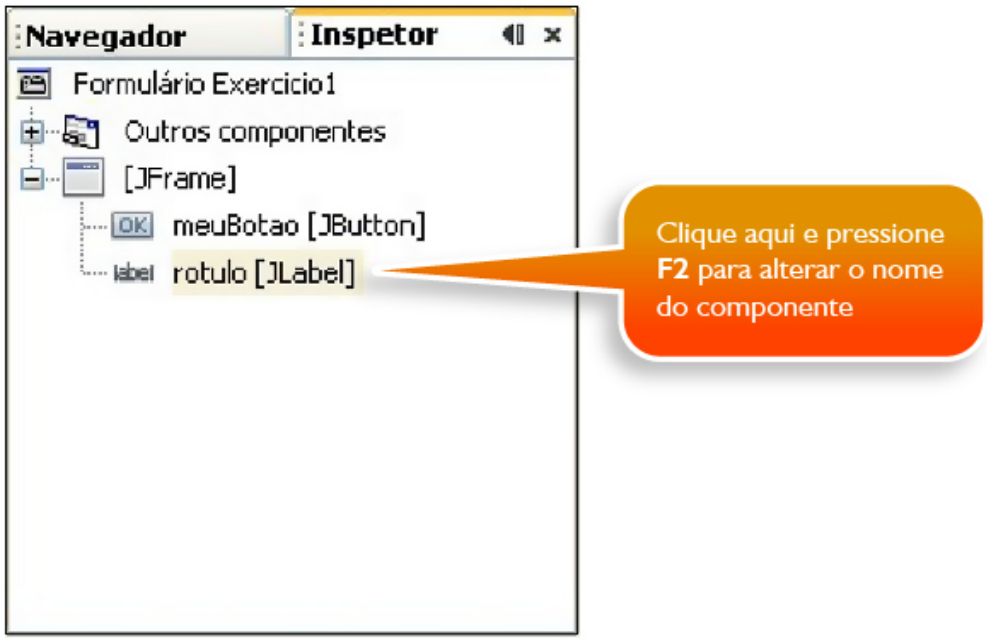
Figura 01 - Aplicação para demonstração de evento em tempo de projeto



Antes de criarmos o evento para nossa aplicação, renomearemos os componentes para nomes que nos pareça mais práticos e fáceis de lembrar. Para isso:

6. No painel **Inspetor**, clique em cada um dos componentes, pressione a tecla **F2** e altere os seus nomes para: **meuBotao** e **rotulo**, respectivamente, ou outros nomes que você achar mais conveniente. Veja o exemplo na **Figura 2**:

Figura 02 - Painel para alteração dos nomes dos componentes



Observação

Esses nomes podem ser nomes quaisquer, porém é aconselhável que você nomeie os componentes de acordo com a finalidade de cada um para facilitar a identificação deles no código. Procure não usar espaços, nem caracteres especiais ou acentuados.

Essa aplicação, se executada agora, nada fará, mostrará apenas os componentes da forma que foram configurados. Para que essa aplicação seja funcional, é preciso "dar vida" aos componentes; ou seja, criarmos alguma ação para eles executarem alguma tarefa. No nosso caso, criaremos um evento no botão de forma que quando o usuário clicar nele altere algumas propriedades do rótulo e que o próprio botão fique desabilitado. É o que vamos fazer agora:

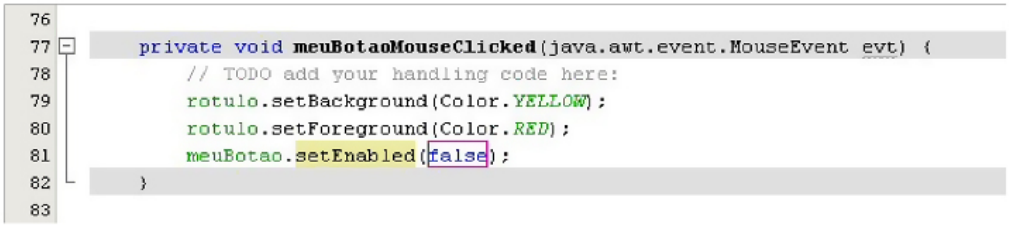
7. Clique com o botão direito do **mouse sobre o componente meuBotao.**
8. No menu apresentado, selecione **Eventos -> Mouse -> mouseClicked.**
9. Feito isso, será mostrado o código referente a esse evento, conforme mostra a **Figura 3**:

Figura 03 - Código gerado automaticamente para o evento selecionado

Mesmo se você executasse a aplicação agora, ela não faria nada ainda. Apesar de criarmos o evento, não dissemos ainda o que deverá ser feito após o botão ser clicado. É como se você girasse a chave de ignição do carro e ficasse esperando que ele saísse do lugar. Para ele sair do lugar, seria necessário você passar a marcha adequada e acelerar. Da mesma forma funcionam os eventos criados para os componentes. Uma vez criado o evento, temos que dizer o que fazer quando o botão do mouse for clicado. Você pode mandar fazer quase tudo. Para o nosso exemplo, vamos alterar algumas propriedades do rótulo e, ao mesmo tempo, desabilitar o botão após ser clicado. Para isso:

10. No final da linha 78, tecle **ENTER** para criar uma nova linha e digite as linhas mostradas na **Figura 4**:

Figura 04 - Métodos criados para alterar as propriedades do rótulo e do botão



Nesse código, utilizamos três métodos. Vejamos o que eles fazem:

A linha:

```
1 rotulo.setBackground(Color.YELLOW);
```

Utiliza o método **setBackground** para definir a cor de fundo do rótulo através do parâmetro **Color**.

A linha:

```
1 rotulo.setForeground(Color.RED);
```

Utiliza o método **setForeground** para definir a cor da fonte do rótulo através do parâmetro **Color**.

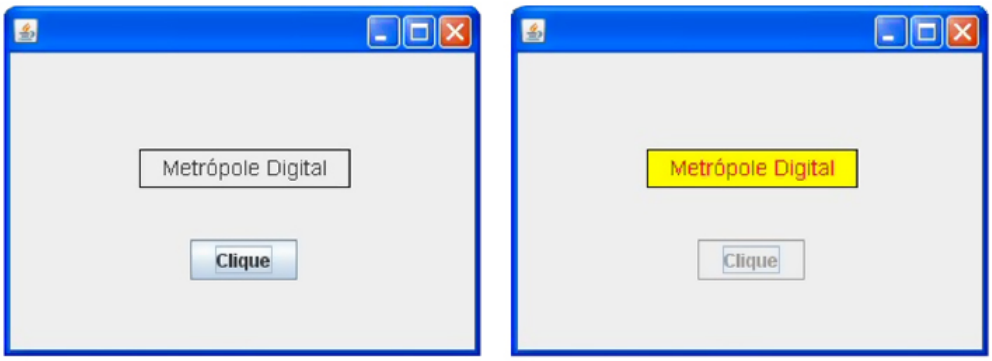
A linha:

```
1 meuBotao.setEnabled(false);
```

Utiliza o método **setEnabled** para desabilitar o botão através do parâmetro **false**, após o mesmo ser clicado.

11. Execute a aplicação e confira o resultado com a **Figura 5**:

Figura 05 - Aplicação antes e depois de executada



Antes da execução | Após a execução

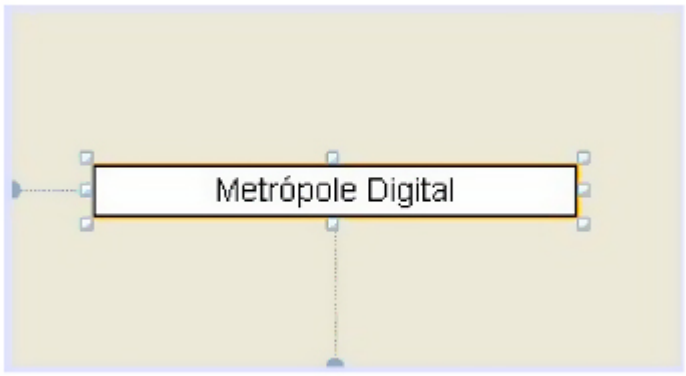
Exemplo 2

Nesse exemplo, utilizaremos um rótulo e os eventos **mouseEntered** e **mouseExited**, de forma que quando o usuário passar o mouse sobre esse rótulo uma mensagem será mostrada, a cor de fundo será alterada para amarelo e a cor da fonte para azul; e quando o cursor do mouse sair da área do rótulo, outra mensagem será mostrada, a cor de fundo será alterada para preto e a cor da fonte para branco.

1. Execute o NetBeans e crie um novo projeto (**Arquivo -> Novo projeto**).
2. Salve seu projeto com o nome: **Eventos e Listeners – Exercicio2**.
3. Crie um novo **Formulário JFrame**.
4. Clique na propriedade **title** do **JFrame** e digite um titulo para ele, por exemplo: **Eventos do Mouse — Exercício 2**.

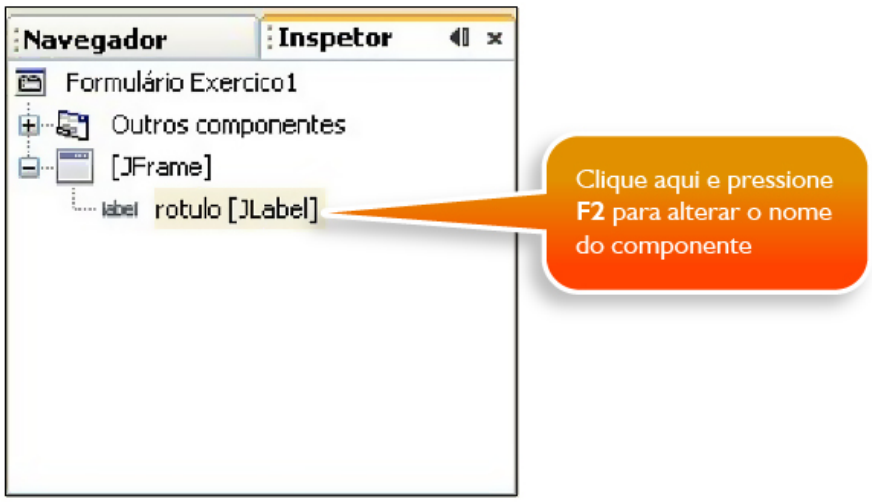
5. Arraste para o projeto um componente **Rótulo (JLabel)**. Na propriedade **text** desse componente, digite: **Metrópole Digital**. Altere a fonte para **Arial**, tamanho **14**, cor de fundo branca, centralize o texto, marque o check box da opção **opaque** e crie uma borda para o componente, conforme mostra a **Figura 6**:

Figura 06 - Componente Rótulo a ser utilizado para demonstração de evento



6. No painel **Inspetor**, clique no componente **jLabel**, pressione a tecla **F2** e altere o seu nome para: **rotulo**, ou outro nome que você achar mais conveniente. Veja o exemplo na **Figura 7**:

Figura 07 - Painel para alteração dos nomes do componente



Da mesma forma que a aplicação do exemplo anterior, se essa aplicação for executada agora, nada fará. Para que essa aplicação seja funcional, é preciso adicionar a ação que queremos ao componente. Nesse caso, criaremos um evento no rótulo de forma que quando o usuário passar o mouse sobre ele sejam alteradas algumas de suas propriedades, e, quando o mouse sair dos limites do componente, as mesmas propriedades sejam novamente alteradas. Vejamos como fazer isso:

7. Clique com o botão direito sobre o componente **Rótulo**. No menu apresentado, selecione **Eventos -> Mouse -> mouseEntered**.

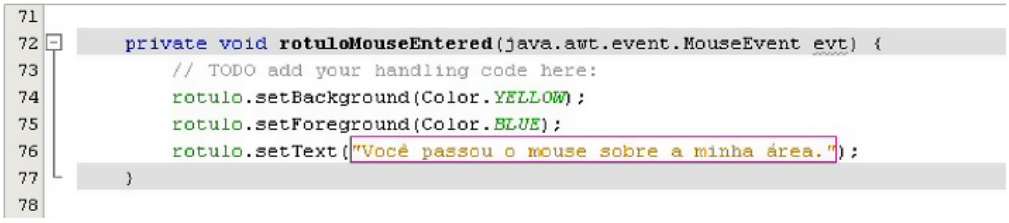
8. Feito isso, será mostrado o código referente a esse evento, conforme mostra a **Figura 8**:

Figura 08 - Código gerado automaticamente para o evento selecionado

Mesmo com esse evento criado, a aplicação ainda nada faria se fosse executada, porque não dissemos ainda o que deverá ser feito quando o evento for solicitado. O que vamos querer que o evento faça é alterar algumas propriedades do componente quando o mouse for passado sobre os limites de sua área. Então, vejamos como fazer isso:

9. No final da linha **70** (do código acima) tecle **ENTER** para criar uma nova linha, e digite as linhas mostradas na **Figura 9**:

Figura 09 - Métodos criados para alterar as propriedades do componente Rótulo



Para criarmos o segundo evento, volte -> janela de projeto (aba **Projeto**) e faça o seguinte:

1. Clique com o botão direito sobre o componente **Rótulo**. No menu apresentado, selecione **Eventos -> Mouse -> mouseExited**.

2. Feito isso, será mostrado o código referente a esse evento logo abaixo do primeiro evento, conforme mostra a **Figura 10:**

Figura 10 - Código gerado automaticamente para o segundo evento

Quando esse evento ocorrer, queremos que as mesmas propriedades do componente sejam novamente alteradas com outros atributos, quando o usuário mover o mouse para fora da área do componente. Vejamos como fazer isso:

No final da linha **83**, tecle **ENTER** para criar uma nova linha e digite as linhas mostradas na **Figura 11**:

Figura 11 - Métodos criados para alterar as propriedades do componente rotulo

```
81
82      private void rotuloMouseExited(java.awt.event.MouseEvent evt) {
83          // TODO add your handling code here:
84          rotulo.setBackground(Color.BLACK);
85          rotulo.setForeground(Color.WHITE);
86          rotulo.setText("Você tirou o mouse da minha área.");
87      }
88
```

Em cada um dos eventos, utilizamos três métodos. Os dois primeiros já foram apresentados no primeiro exemplo. A seguir, temos o terceiro:

```
1 rotulo.setText("Você tirou o mouse da minha área.");
```

Do primeiro evento, altera-se a propriedade **text** (com **setText**) do componente **rotulo** para a mensagem entre parênteses, quando o mouse for passado sobre ele.

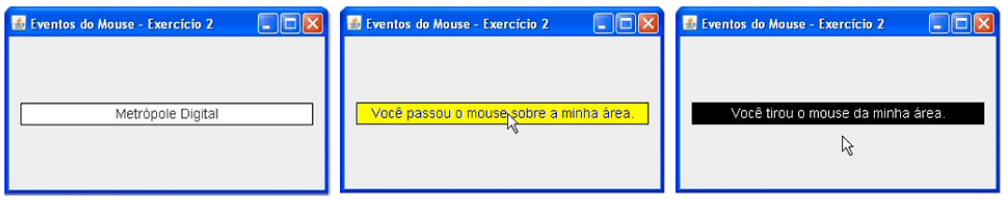
E o método:

```
1 rotulo.setText("Você tirou o mouse da minha área.");
```

Do segundo evento, altera a propriedade **text** do componente **rotulo** para a mensagem entre parênteses, quando o mouse for retirado da área do componente.

3. Execute a aplicação e confira o resultado com a **Figura 12**:

Figura 12 - Etapas executadas pelos eventos utilizados no componente rotulo



Execução inicial da aplicação | Execução do primeiro evento, quando o mouse passar sobre o componente | Execução do segundo evento, quando o mouse sair da área do componente

Atividade 01

1. O que você entende por **Eventos**? Eles são realmente necessários para o desenvolvimento de uma aplicação?
2. Podemos utilizar vários eventos para um mesmo componente? Dê um exemplo.
3. Podemos utilizar o mesmo evento em vários componentes na mesma aplicação? Justifique sua resposta.
4. Crie uma aplicação com apenas um componente **Rótulo** com a cor de fundo (vermelha), de forma que, quando o cursor do mouse for passado sobre ele, a sua cor de fundo seja alterada para preto; e quando o cursor sair da sua área de atuação, sua cor de fundo seja alterada para azul. Utilize as propriedades **mouseEntered** e **mouseExited** para desenvolver essa aplicação.

Os Eventos mousePressed e mouseReleased

O evento **mouseClicked** visto no inicio da aula não permite criar duas situações quando o mouse é clicado e liberado, pois ele só executa a ação quando o botão do mouse é baixado. Quando ele é liberado, não acontece nada. Se você quiser criar uma situação para cada ação do mouse, você deverá utilizar o evento **mousePressed** para criar uma situação quando o usuário clicar com o botão do mouse, e outra situação para quando o botão for solto ou liberado (**mouseReleased**).

Vejamos um exemplo com esses eventos:

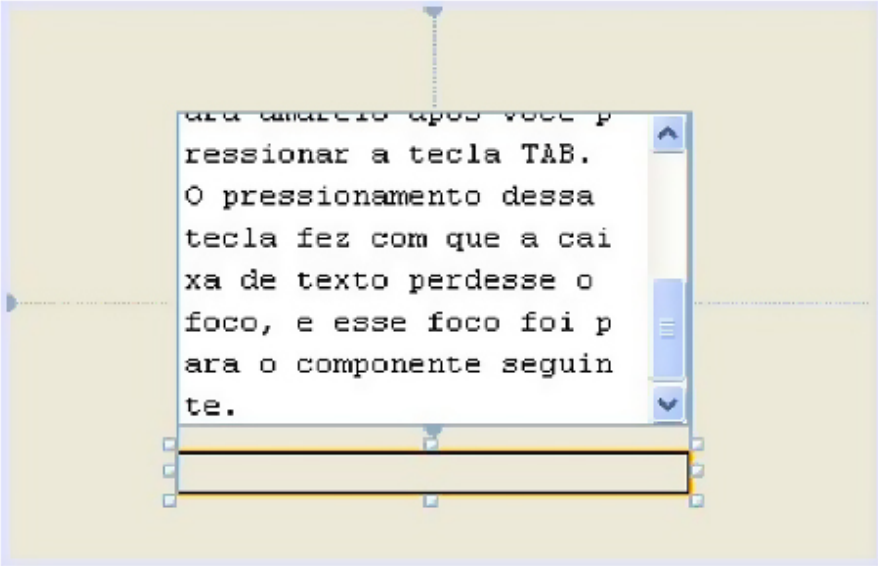
Exemplo 3

Nesse exemplo, utilizaremos um componente **Área de texto (jTextArea)**, um componente **Rótulo (jLabel)** e os eventos **mousePressed** e **mouseReleased**, de forma que, quando o usuário clicar com o mouse sobre o componente **Rótulo**, uma mensagem será mostrada no rótulo e o componente **Área de texto** ficará

desabilitado para edição. Quando o usuário liberar o botão do mouse, outra mensagem será mostrada no rótulo e o componente voltará a ser liberado para edição.

1. Execute o NetBeans e crie um novo projeto (**Arquivo -> Novo projeto**).
2. Salve seu projeto com o nome: **Eventos e Listeners — Mouse**.
3. Crie um novo **Formulário JFrame**.
4. Clique na sua propriedade **title** e digite um titulo para ele, por exemplo: **Eventos mousePressed e mouseReleased**.
5. Arraste para a área de projeto um componente **Área de texto** e um componente **Rótulo**. Na propriedade **text** do componente **Área de texto**, digite ou cole um texto qualquer. Quanto ao componente **Rótulo**, apague a propriedade **text** e crie uma borda para que ele possa ser visto quando a aplicação for executada. Disponibilize os componentes de acordo com a **Figura 13**:

Figura 13 - Componentes a serem utilizados com os eventos mousePressed e mouseReleased



6. No painel **Inspetor**, pressione a tecla **F2** para cada componente e renomei-os com os seguintes nomes:

   areaTexto
   rotulo

7. Confira com a ilustração da **Figura 14**:

Figura 14 - Painel para alteração dos nomes dos componentes.

---

8. Clique com o botão direito sobre o componente **rotulo**. No **menu Eventos**, selecione **Mouse -> mousePressed**.

9. Feito isso, será mostrado o código referente a esse evento, conforme mostra a **Figura 15:**

Figura 15 - Código gerado automaticamente para o evento mousePressed

```
75
76      private void rotuloMousePressed(java.awt.event.MouseEvent evt) {
77          // TODO add your handling code here:
78      }
79
```

10. No final da linha **77** (do código acima), tecle **ENTER** para criar uma nova linha e digite as linhas mostradas na **Figura 16**:

Figura 16 - Métodos criados para alterar as propriedades dos componentes rotulo e areaTexto

```
75
76      private void rotuloMousePressed(java.awt.event.MouseEvent evt) {
77          // TODO add your handling code here:
78          rotulo.setText("O texto foi desabilidado para edição.");
79          areaTexto.setEnabled(false);
80      }
81
```

Vejamos os comentários sobre esse código:

Quando o botão do mouse for pressionado, o texto entre parênteses do método **setText** será mostrado no rótulo, e o componente **areaTexto** ficará desabilitado para edição. Para isso, utilizamos o método **setEnabled(false)**.

Para criarmos o segundo evento, volte à janela de projeto e faça o seguinte:

1. Clique com o botão direito sobre o componente **rotulo**. No menu **Eventos**, selecione **Mouse -> mouseReleased**.
2. Feito isso, será mostrado o código referente a esse evento logo abaixo do primeiro evento, conforme mostra a **Figura 17**:

Figura 17 - Código gerado automaticamente para o evento mouseReleased

3. No final da linha **86**, tecle **ENTER** para criar uma nova linha e digite as linhas mostradas na **Figura 18**:

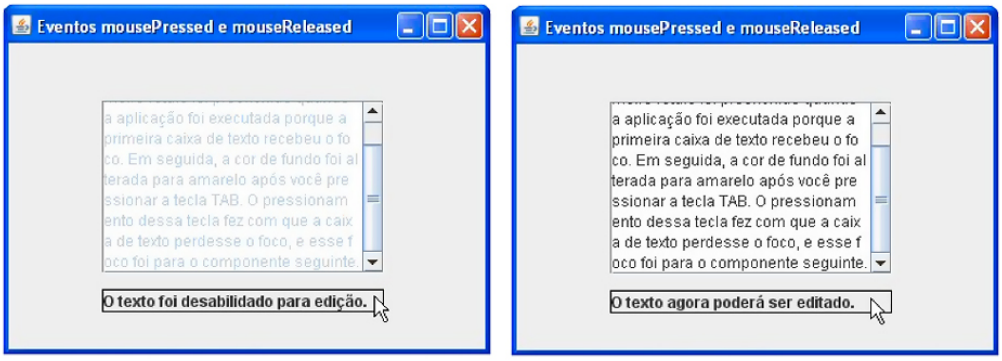
Figura 18 - Métodos criados para alterar as propriedades dos componentes rotulo e areaTexto

Vejamos os comentários sobre esse código:

Quando o botão do mouse for liberado, o texto entre parênteses do método **setText** será mostrado no rótulo, e o componente **areaTexto** voltará a ser liberado para edição. Para isso, utilizamos o método **setEnabled(true)**.

4. Execute a aplicação e confira o resultado com a **Figura 19**:

Figura 19 - Etapas executadas pelos eventos utilizados no componente rotulo



Execução do primeiro evento **(mousePressed)**. Quando o botão do mouse for clicado, o texto ficará desabilitado para edição.

Execução do segundo evento **(mouseReleased)**. Quando o botão do mouse for liberado, o texto voltará a ficar habilitado para edição.

Vídeo 03 - Eventos do Mouse

Mais Eventos

Além dos eventos referentes ao mouse vistos anteriormente, você ainda poderá utilizar os seguintes:

Os Eventos mouseDragged e mouseMoved

Esses dois eventos fazem parte do evento **mouseMotion**. O evento **mouseDragged** é utilizado para executar uma ação quando o mouse é clicado e arrastado, enquanto que o evento **mouseMoved** executa uma ação quando o mouse estiver sendo arrastado dentro da aplicação ou em uma área específica, sem a necessidade de nenhum botão ser clicado, apenas e simplesmente arrastado. Como sugestão, você poderia usar esse evento para alterar o cursor quando o mouse fosse passado por uma determinada área.

O Evento mouseWheelMoved

Esse evento faz parte do evento **mouseWheel**. O evento **mouseWheelMoved** é utilizado para executar uma ação quando a rodinha do mouse for girada, para frente ou para trás. Você poderia utilizar esse evento para alterar os valores de um controle giratório, por exemplo. Vários componentes suportam esse evento.

Os Eventos focusGained e focusLost

Quando um componente está com o foco, isso quer dizer que ele está ativo, ou seja, está na vez. É como se você estivesse em uma fila de banco e a sua vez de ser atendido chegasse. Nesse caso, diríamos que você está com o foco. Quando um componente recebe o foco, podemos utilizar o evento **focusGained** para executar qualquer tarefa que quisermos. Por exemplo, se o componente for um campo ou uma área de texto, poderíamos alterar a cor de fundo ou da fonte desse componente. Se for um botão de opção, poderíamos mostrar um rótulo dando mais alguma informação sobre a opção. Por outro lado, se for um check box, poderíamos mostrar um outro componente que estava oculto quando ele for marcado e ocultá-lo novamente quando for desmarcado, e assim por diante, dependendo do tipo de componente que estiver com o foco naquele momento.

Da mesma forma acontece quando o componente perde o foco. Poderíamos utilizar o evento **focusLost** e criar também outras situações que quisermos. Esses eventos são geralmente utilizados em formulários para validar campos após serem preenchidos. Nesses casos, poderíamos criar janelas de diálogos para informar ao usuário através de uma mensagem quando determinado campo não for preenchido corretamente, ou ativar um botão que se encontrava desabilitado, por exemplo. Você sabe quando um componente está com o foco quando ele fica com uma borda em volta, como os botões e check boxes. Se for um componente no qual seja necessário digitar algum texto, o cursor aparece piscando.

Vídeo 04 - Eventos do Teclado

Atividade 02

1. Qual a diferença entre os eventos **mouseDragged** e **mouseMoved**?
2. Descreva sucintamente os eventos **focusGained** e **focusLost**.
3. Os eventos **mousePressed** e **mouseReleased** podem ser utilizados isoladamente? Qual a diferença entre esses eventos e o evento **mouseClicked**?
4. O evento **mouseWheelMoved** pode ser utilizado com qualquer componente? Dê alguns exemplos.
5. O componente **Rótulo** pode receber foco? Justifique sua resposta.
6. Em uma aplicação com três componentes **Campos de texto**, obrigatoriamente a primeira ficará com o foco quando a aplicação for executada?
7. Numere a segunda coluna de acordo com a primeira:

a. mouseClicked	( ) É executado quando um componente recebe o foco.
b. mouseMoved	( ) É executado quando o botão do mouse é pressionado.
c. focusLost	( ) É executado quando o mouse é movido dentro da aplicação.
d. mousePressed	( ) É executado quando o botão do mouse é pressionado e liberado.
e. focusGained	( ) É executado quando um componente perde o foco.

Resumo

Nesta aula, você aprendeu o que são eventos e como utilizá-los em alguns componentes. Conheceu alguns eventos aplicados ao mouse em exemplos práticos, utilizando-os para executarem algumas tarefas. Baseados nesses exemplos, acreditamos que você será capaz de criar aplicações mais interessantes utilizando esses mesmos eventos em outros componentes. Viu também que vários eventos poderão ser aplicados a um mesmo componente. Experimente criar situações utilizando os eventos mostrados na **Tabela 2**.

Autoavaliação

1. Descreva uma situação em que podemos utilizar o evento **mouseDragged**.
2. Desenvolva uma aplicação utilizando apenas um componente **Button**, de forma que, quando o botão do mouse for pressionado, as dimensões do componente **Button** sejam alteradas, e quando o botão for liberado, suas dimensões voltem ao tamanho original.
3. Crie uma aplicação utilizando três componentes **Rótulo** em qualquer lugar do projeto, de forma que, quando o mouse for movido dentro da área da aplicação, todos os componentes fiquem alinhados horizontalmente e com cores de fundo diferentes. Para desenvolver essa aplicação, utilize o evento **mouseMotion** e os métodos **setLocation** e **setBackground**.
4. Crie uma aplicação utilizando um componente **Rótulo** e o evento **mouseWheel**, de forma que, quando a rodinha do mouse for girada, o componente seja deslocado horizontalmente somente para a direita.
5. Desenvolva uma aplicação utilizando três componentes **Botão de opção** e um componente **Button**, de forma que, quando cada botão de opção perder o foco, o rótulo de cada um seja alterado para outro nome qualquer, e quando o componente **JButton** for clicado, os rótulos de todos os botões sejam restaurados para os seus nomes originais.

6. Indique se as alternativas abaixo são verdadeiras (V) ou falsas (F):

    a. O evento **focusGained** é utilizado quando um componente perde o foco.

    b. Os eventos **focusGained** e **focusLost** não podem ser utilizados com o mesmo componente.

    c. O evento **mouseMotion** possui dois outros eventos, o **mousePressed** e o **keyPressed**.

    d. Não é possível utilizar os eventos **mousePressed** e **mouseReleased** em um mesmo componente.

    e. O evento **mouseReleased** faz o mesmo efeito que o evento **mouseClicked**.

    f. O evento **mouseWheelMoved** só pode ser utilizado em componentes que possuem barras de rolagem.

7. Crie uma aplicação utilizando um componente **Rótulo** e o evento **mouseWheel**. Na propriedade **text** do Rótulo, informe o valor zero (0). A aplicação deverá funcionar da seguinte forma: quando a rodinha do mouse for girada para a frente, o valor do Rótulo será acrescido de uma unidade, e quando a rodinha for girada para trás, o valor do Rótulo será subtraído de uma unidade.

Referências

DEITEL, Harvey M.; DEITEL, Paul J. Java: como programar. 6. ed. São Paulo: Editora Prentice-Hall, 2005.

THE JAVA tutorials. Introduction to Event Listeners. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/events/intro.html>. Acessado em: 20 abr. 2012a.

_____. How to Write a Mouse-Wheel Listener. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/events/mousewheellistener.html>. Acesso em: 20 abr. 2012b.

_____. How to Write a Focus Listener. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/events/focuslistener.html>. Acesso em: 20 abr. 2012c.

_____. How to Write a Mouse-Motion Listener. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/events/mousemotionlistener.html>. Acesso em: 20 abr. 2012d.