LCP - Aula 07

Prof. Lucas Guerreiro

Aula anterior

- GUIs em Java
- Swing
  - JFrame
  - JLabel
  - JTextField
  - JButton
  - Tratamento de eventos
  - etc…

Aula 07

Objetivos: GUI - Parte 2. Eventos com mouse. Eventos com teclado. Mais opções de GUI.

Eventos com mouse

- Temos duas interfaces ouvintes de eventos com Mouse:
  - MouseListener
  - MouseMotionListener
- O pacote javax.swing.event contém a interface MouseInputListener que estende as interfaces MouseListener e MouseMotionListener para criar uma única interface com todos os métodos de ações do mouse.
- Com isso, componentes registrados com objetos que implementam métodos de MouseListener e MouseMotionListener serão disparados em eventos com o mouse dentro destes componentes.

Métodos da Interface **MouseListener**

`public void` **`mousePressed`**`(MouseEvent event)`
Chamado quando um botão do mouse é pressionado enquanto o cursor de mouse estiver sobre um componente.

`public void` **`mouseClicked`**`(MouseEvent event)`
Chamado quando um botão do mouse é pressionado e liberado enquanto o cursor do mouse pairar sobre um componente. Sempre precedido por uma chamada a mousePressed e mouseReleased.

`public void` **`mouseReleased`**`(MouseEvent event)`
Chamado quando um botão do mouse é liberado depois de ser pressionado. Sempre precedido por uma chamada a mousePressed e uma ou mais chamadas a mouseDragged.

`public void` **`mouseEntered`**`(MouseEvent event)`
Chamado quando o cursor do mouse entra nos limites de um componente.

`public void` **`mouseExited`**`(MouseEvent event)`
Chamado quando o cursor do mouse deixa os limites de um componente.

Métodos da Interface **MouseMotionListener**

`public void` **`mouseDragged`**`(MouseEvent event)`

Chamado quando o botão do mouse é pressionado enquanto o cursor de mouse estiver sobre um componente e o mouse é movido enquanto o botão do mouse permanecer pressionado. Sempre precedido por uma chamada a mousePressed. Todos os eventos de arrastar são enviados para o componente em que o usuário começou a arrastar o mouse.

`public void` **`mouseMoved`**`(MouseEvent event)`

Chamado quando o mouse é movido (sem botões do mouse pressionados) quando o cursor do mouse está sobre um componente. Todos os eventos de movimento são enviados para o componente sobre o qual o mouse atualmente está posicionado.

```
import java.awt.Color;
import java.awt.BorderLayout;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.awt.event.MouseEvent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
public class MouseTrackerFrame extends JFrame{
    private final JPanel mousePanel;
    private final JLabel statusBar;
    public MouseTrackerFrame(){
        super("Analisando uso de Eventos do Mouse");
        mousePanel = new JPanel();
        mousePanel.setBackground(Color.WHITE);
        add(mousePanel, BorderLayout.CENTER);
        statusBar = new JLabel("Mouse fora do JPanel");
        add(statusBar, BorderLayout.SOUTH);
        MouseHandler handler = new MouseHandler();
        mousePanel.addMouseListener(handler);
        mousePanel.addMouseMotionListener(handler);
    }
```

MouseTrackerFrame.java

```
private class MouseHandler implements MouseListener, MouseMotionListener{
      @Override
      public void mouseClicked(MouseEvent event){
            statusBar.setText(String.format("Clique em [%d, %d]", event.getX(), event.getY()));
      }
      @Override
      public void mousePressed(MouseEvent event) {
            statusBar.setText(String.format("Pressionado  em [%d, %d]", event.getX(),event.getY()));
      }
      @Override
      public void mouseReleased(MouseEvent event){
            statusBar.setText(String.format("Solto em [%d, %d]", event.getX(),event.getY()));
      }
      @Override
      public void mouseEntered(MouseEvent event){
            statusBar.setText(String.format("Mouse entrou em [%d, %d]", event.getX(),event.getY()));
            mousePanel.setBackground(Color.GREEN);
      }
      @Override
      public void mouseExited(MouseEvent event){
            statusBar.setText("Mouse fora do JPanel");
            mousePanel.setBackground(Color.WHITE);
       }
      @Override
      public void mouseDragged(MouseEvent event){
            statusBar.setText(String.format("Arrastado em [%d, %d]", event.getX(),event.getY()));
       }
      @Override
      public void mouseMoved(MouseEvent event){
            statusBar.setText(String.format("Movido em [%d, %d]", event.getX(),event.getY()));
      }
}
```

MouseTrackerFrame.java

```
import javax.swing.JFrame;
public class MouseTracker {
	public static void main(String[] args){
		MouseTrackerFrame mouseTrackerFrame = new MouseTrackerFrame();
		mouseTrackerFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		mouseTrackerFrame.setSize(300, 100);
		mouseTrackerFrame.setVisible(true);
	}
}
```

MouseTracker.java

Exercício

Criar uma tela com um botão.

Ao mover o mouse sobre ele, o botão deve trocar sua posição para uma posição aleatória da tela, impedindo que o usuário clique no botão.

Dicas: JPanel, setLayout(null), Random

```
private final JButton botao;

    public Exercicio1() {
        super("Não clique");
        setSize(1000,1000);

        JPanel panel = new JPanel();
        panel.setLayout(null);

        botao = new JButton();
        botao.setBounds(100, 100, 100, 100);
        botao.setText("Clique aqui!");
        panel.add(botao);

        botao.addMouseListener(new EsconderBotao());

        add(panel);

    }
```

Exercicio1.java

```
private class EsconderBotao implements MouseListener{
     @Override
     public void mouseClicked(MouseEvent e) {
     }
     @Override
     public void mousePressed(MouseEvent e) {
     }
     @Override
     public void mouseReleased(MouseEvent e) {
     }
     @Override
     public void mouseEntered(MouseEvent e) {
          Random rand = new Random();
          int posx = rand.nextInt(900);
          int posy = rand.nextInt(900);
          botao.setBounds(posx, posy, 100, 100);
     }
     @Override
     public void mouseExited(MouseEvent e) {
     }
}
public static void main(String[] args) {
     Exercicio1 ex = new Exercicio1();
     ex.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     ex.setVisible(true);
}
```

Exercicio1.java

Adaptadores

- No exercício anterior precisávamos apenas do método `mouseEntered`, porém por estarmos implementando a interface `MouseListener`, foi necessário implementar cada método, mesmo que sem ações.
- Para estes casos, `java.awt.event` fornece as classes de adaptadores, de forma a implementar apenas métodos específicos para estes listeners.

Classe de adaptadores de evento em `java.awt.event`	Implementa a interface
`ComponentAdapter`	`ComponentListener`
`ContainerAdapter`	`ContainerListener`
`FocusAdapter`	`FocusListener`
`KeyAdapter`	`KeyListener`
`MouseAdapter`	`MouseListener`
`MouseMotionAdapter`	`MouseMotionListener`
`WindowAdapter`	`WindowListener`

```
import java.awt.BorderLayout;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JFrame;
import javax.swing.JLabel;
public class MouseDetailsFrame extends JFrame{
     private String details;
     private final JLabel statusBar;
     public MouseDetailsFrame(){
          super("Cliques do mouse com adaptadores");
          statusBar = new JLabel("Clique com o mouse");
          add(statusBar, BorderLayout.SOUTH);
          addMouseListener(new MouseClickHandler());
     }
     private class MouseClickHandler extends MouseAdapter{
               @Override
               public void mouseClicked(MouseEvent event){
                    int xPos = event.getX();
                    int yPos = event.getY();
                    details = String.format("%d clique(s)", event.getClickCount());
                    if (event.isMetaDown()) details += " com botão direito";
                    else if (event.isAltDown()) details += " com botão do meio";
                    else details += " com botão esquerdo";
                    statusBar.setText(details);
          }
     }
}
```

MouseDetailsFrame.java

```
import javax.swing.JFrame;

public class MouseDetails{
    public static void main(String[] args) {
        MouseDetailsFrame mouseDetailsFrame = new
    MouseDetailsFrame();
        mouseDetailsFrame.setDefaultCloseOperation(JFrame.EXIT
    _ON_CLOSE);
        mouseDetailsFrame.setSize(400, 150);
        mouseDetailsFrame.setVisible(true);
    }
}
```

MouseDetails.java

Exemplo - Painel de Desenho

```
public class PaintPanel extends JPanel {
     private final ArrayList<Point> points = new ArrayList<>();
     public PaintPanel(){
          addMouseMotionListener(
               new MouseMotionAdapter(){
                  @Override
                  public void mouseDragged(MouseEvent event)
                  {
                     points.add(event.getPoint());
                     repaint();
                  }
          }
       );
   }
   @Override
   public void paintComponent(Graphics g){
      super.paintComponent(g);

      for (Point point : points)
         g.fillOval(point.x, point.y, 4, 4);
   }
}
```

PaintPanel.java

Exemplo

```
public class Painter{
    public static void main(String[] args){
        JFrame application = new JFrame("Paint");

        PaintPanel paintPanel = new PaintPanel();
        application.add(paintPanel, BorderLayout.CENTER);

        application.add(new JLabel("Clique e arraste o mouse para
desenhar"),
         BorderLayout.SOUTH);

        application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        application.setSize(400, 200);
        application.setVisible(true);
   }
}
```

Painter.java

Eventos com teclado

- Temos a interface padrão `KeyListener` para tratar dados com teclado.
- Classes que implementem KeyListener devem apresentar implementações para os métodos `keyPressed`, `keyReleased` e `keyTyped`.
- Estes métodos recebem um argumento keyEvent (subclasse de InputEvent).
  - `keyPressed`: acionado com o pressionamento de qualquer tecla.
  - `keyTyped`: acionado com qualquer tecla que não seja de ação (seta, page up, etc)
  - `keyReleased`: acionado ao liberar uma tecla de keyPressed ou keyTyped.

Exemplo

Verificar ações do teclado.

KeyDemoFrame.java

KeyDemo.java

JSlider

- JSlider é um componente utilizado para selecionar opções incrementando ou decrementando números inteiros ao arrastar um cursor.
- Pode ser utilizado disposto de forma vertical ou horizontal.
- Adere ao ponto mais próximo quando está entre duas medidas.
- Pode ser selecionado com mouse ou teclado.

Exemplo:

OvalPanel.java

SliderFrame.java

SliderDemo.java

Menus

Podemos ter menus similares aos de aplicações comuns (Windows, Navegadores, etc).

Existem algumas implementações específicas em Swing:

- **JMenuBar** - barra de menu (aquela de Arquivo, Exibir, Editar, etc)
- **JMenu** - controle para componentes de Menu
- **JMenuItem** - componente para gerenciar itens de Menu
- **JCheckBoxMenuItem** - exibição de itens de Menu que podem ser marcados/desmarcados (CheckBox)
- **JRadioButtonMenuItem** - seleção única dentre outros itens agrupados em um ButtonGroup (RadioButton)
- **JPopupMenu** - menus de popup (ativados com o botão direito)

Exemplos

`MenuFrame` / `TestaMenu` - aplicação completa com uso de menus simples

`PopupFrame` / `TestaPopup` - uso de aplicações com botão direito do mouse

Destaques: uso de adaptadores e de listeners dos tipos de menu utilizados.