TI-220 Java Orientado a Objetos

ANTONIO CARVALHO - TREINAMENTOS

Eventos

Eventos

Os eventos são ações percebidas pelo sistema operacional, as quais são ignoradas, consumidas ou despachadas.

Dentre estas ações existem as ações de Mouse, Teclado, TouchScreen, Swipe, Joysticks, Movimentação e alteração das janelas, e várias outras.

Quando o usuário move o **mouse**, o sistema operacional detecta o movimento, e notifica a aplicação que está sob o ponteiro mouse de maneira que esta aplicação possa tomar uma atitude com base neste movimento.

Eventos

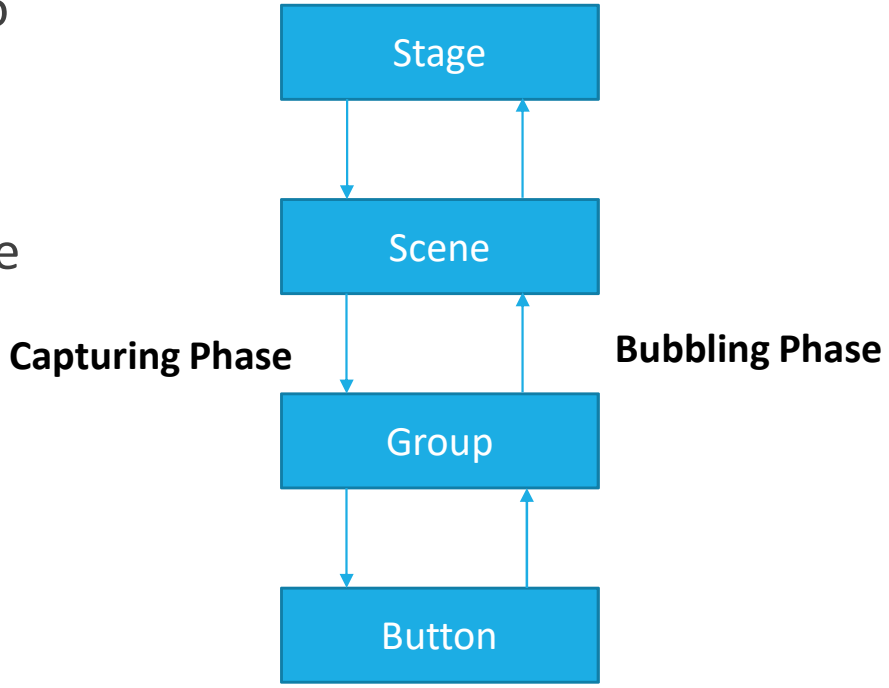
Todo evento possui 3 informações básicas:

- **Type**, tipo do evento, como sendo ACTION, MOUSEPRESSED, MOUSERELEASED, KEYPRESSED, KEYRELEASE, etc.
- **Source**, origem do evento, em relação ao local em que ele foi gerado
- **Target**, componente o qual o evento está dirigido

O sistema de eventos com base em uma cadeia (**chain**), de forma que quando um evento é ativado ele é passado de componente a componente, desde o componente de origem (**source**) até o qual ele é dirigido (**target**)

Eventos

O evento transita desde a origem até o destino em um trajeto chamado de Fase de Captura (**Capturing Phase**) e o outro trajeto de volta chamado de Fase de Bolhas (**Bubling Phase**)



Eventos

Por exemplo, ao criar uma janela contendo os seguintes componentes **Stage, Scene, Group, Button** e quando o botão é clicado, as informações do evento são:

Type: ACTION

Source: Stage

Target: Button

```
public class ExemploApp1 extends Application {
   class ManipuladorMouse implements EventHandler<ActionEvent> {
      public void handle(ActionEvent e) {
         System.out.println("Action : " + e.getEventType());
          System.out.println("Source : " + e.getSource().getClass().getName());
          System.out.println("Target : " + e.getTarget().getClass().getName());
       }
    }

   public void start(Stage stage) throws Exception {
      Pane panel = new Pane();
      Button btn = new Button("Ok");
      EventHandler<ActionEvent> manipulador = new ManipuladorMouse();
      stage.addEventFilter(ActionEvent.ACTION, manipulador);
      panel.getChildren().add(btn);
      Scene scn = new Scene(panel);
      stage.setScene(scn);
      stage.setTitle("Teste de Eventos");
      stage.show();
    }
}
```

Eventos

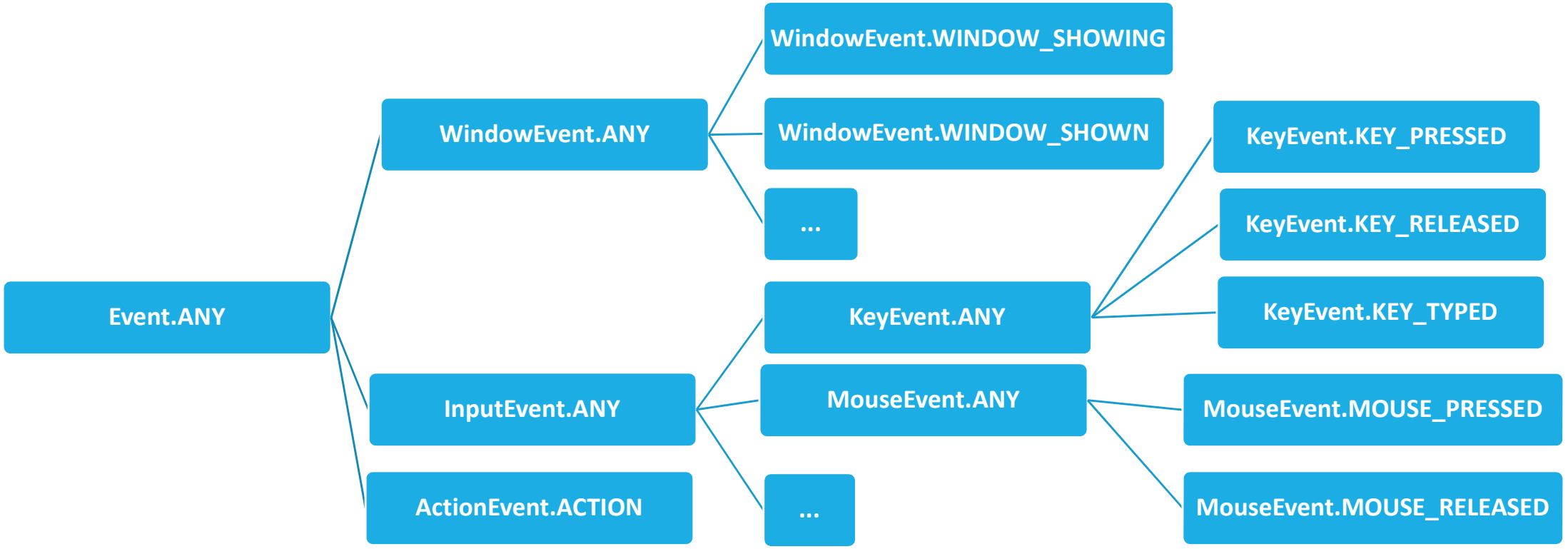
Já o tipo do evento pode obedecer uma hierarquia, portanto ao capturar o evento do tipo **Event.ANY,** todos os eventos irão acionar o **handler** (manipulador), se capturar o **MouseEvent.MOUSE_PRESSED** apenas os eventos de pressionamento do mouse acionarão o **handler**

Eventos

Os eventos que herdam da classe **Event** são:

ActionEvent	**InputEvent**	**TableColumn.CellEditEvent**
MediaMarkerEvent	**ContextMenuEvent**	**TransformChangedEvent**
CheckBoxTreeItem.TreeModificationEvent	**DragEvent**	**TreeItem.TreeModificationEvent**
DialogEvent	**GestureEvent**	**TreeTableColumn.CellEditEvent**
ListView.EditEvent	**InputMethodEvent**	**TreeTableView.EditEvent**
MediaErrorEvent	**KeyEvent**	**TreeView.EditEvent**
ScrollToEvent	**MouseEvent**	**WebErrorEvent**
SortEvent	**TouchEvent**	**WebEvent**
		WindowEvent
		WorkerStateEvent

Eventos – Tipos (Hierarquia)



Eventos – Handler (Manipulador)

O **handler** para capturar o evento deve ser criado com base na interface **EventHandler** associada ao evento que será manipulado.

- Criar EventHandler
  - Sintaxe:

```
class <Nome da Classe> implements EventHandler<T> {
        @Override
        public void handle(T event) { ... }
}
```

  - Exemplo:

```
class ManipuladorMouse implements EventHandler<MouseEvent> {
        @Override
        public void handle(MouseEvent e) {
                System.out.println(e.getTarget());
        }
}
```

Eventos - Handler

Com o **handler** criado é possível associar ele a um Node para que sirva como manipulador (assinante) de um tipo específico do **Evento** associado

- Associar EventHandler
  - Sintaxe:

```
EventHandler<T> <objeto manipulador> = new <Nome da Classe EventHandler>();
<objeto node>.addEventHandler(<Event Type>, <objeto manipulador>);
```

  - Exemplo:

```
Label lblHello = new Label("Hello");
EventHandler<MouseEvent> manipulador = new ManipuladorMouse();
lblHello.addEventHandler(MouseEvent.MOUSE_PRESSED, manipulador);
```

Eventos - Handler

Ao associar a um nó é possível associar em um dos caminhos do fluxo, se no caminho de **Capturing** ou no **Bubbling**

Capturing ocorre quando o objeto **EventHandler** é associado ao Node pelo método **addEventFilter**

Bubbling ocorre quando o objeto **EventHandler** é associado ao Node pelo método **addEventHandler**

- Associar **Capturing**

```
lblHello.addEventFilter(MouseEvent.MOUSE_PRESSED, manipulador);
```

- Associar **Bubbling**

```
lblHello.addEventHandler(MouseEvent.MOUSE_PRESSED, manipulador);
```

Eventos - Consuming

Qualquer evento pode ser consumido no seu fluxo de acionamento, no momento que é consumido ele deixa de ser propagado para o próximo **Node** na hierarquia.

Para consumir o evento é preciso acionar o método **consume()** do objeto do tipo **Event**

- Consumir o evento

```
class ManipuladorMouse implements EventHandler<MouseEvent> {
        @Override
        public void handle(MouseEvent e) {
                System.out.println(e.getTarget());
                e.consume();
        }
}
```

Bibliografia

BARNES, DAVID J. Programação Orientada a Objetos com Java

DEITEL, Java - Como Programar - 6ª Edição, Pearson Education

SIERRA, KATHY e BATES BERT, Use a Cabeça Java, Alta Books

SIERRA, KATHY e BATES BERT, OCA/OCP Java SE 7 Programmer I & II Study Guide