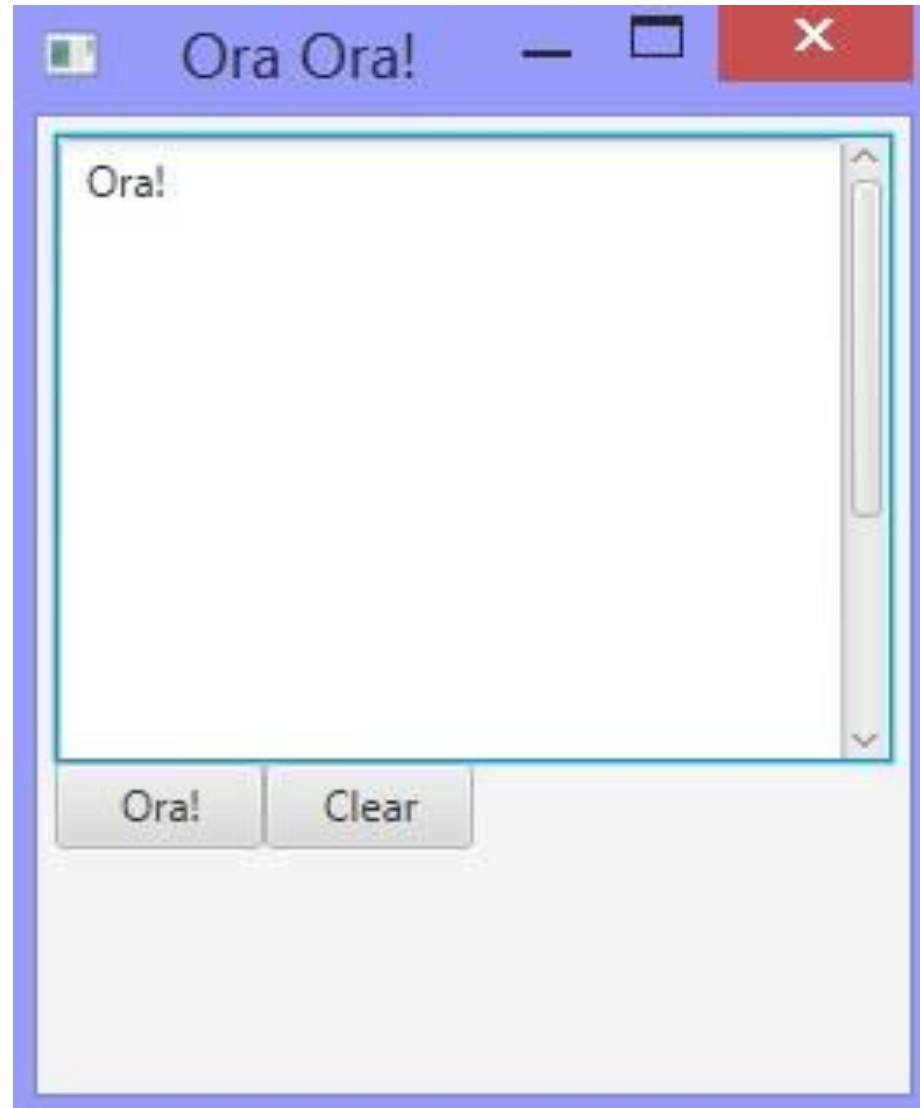


More User Interface Examples

Trying Key press
and Mouse click
[KeyPress.java]



Interest Points: Text Wrap and Scroll Pane

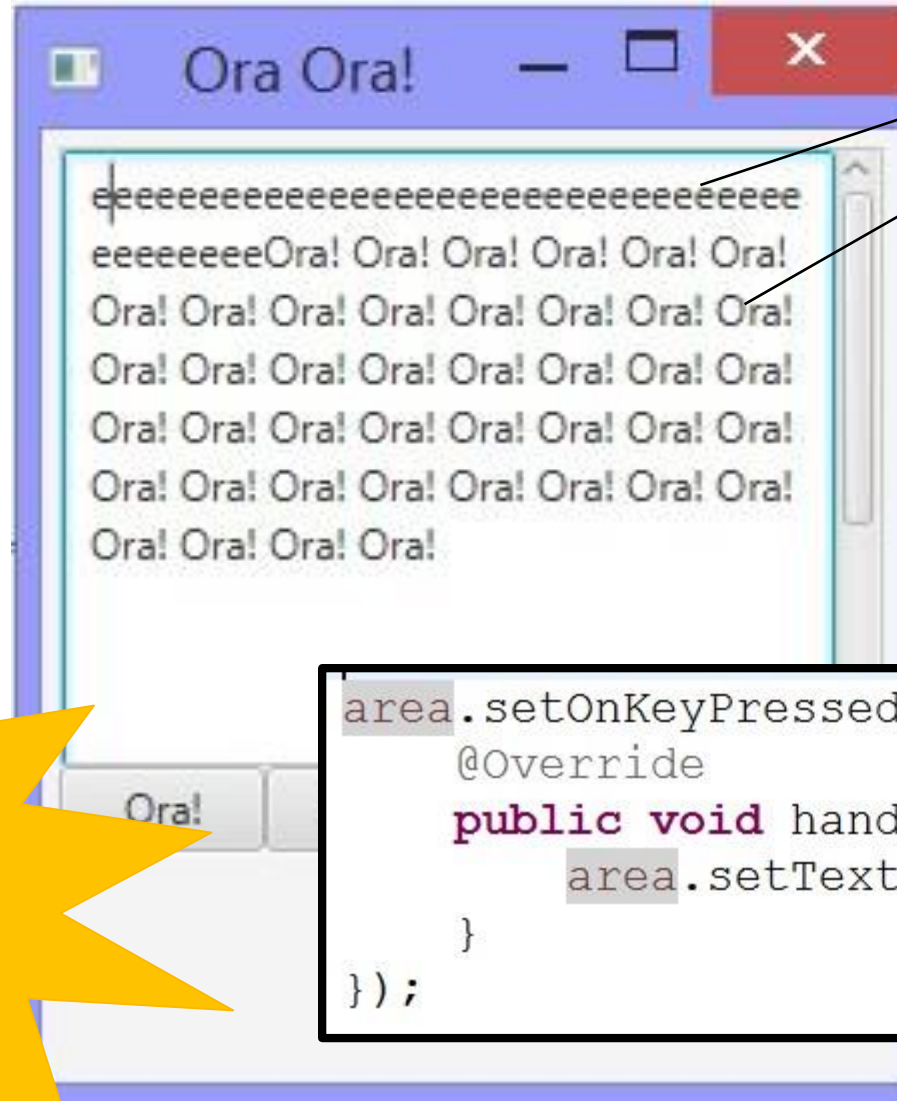


Scroll Pane

```
TextArea area = new TextArea();  
area.setWrapText(true); // important prop  
area.setPrefSize(250, 280);  
  
ScrollPane scrollPane = new ScrollPane();  
scrollPane.setContent(area);  
scrollPane.setFitToWidth(true);  
scrollPane.setPrefWidth(200);  
scrollPane.setPrefHeight(180);
```

KeyPress in TextArea

**What if we want
the event to fire
just 1 time!? (see
later)**



Pressed Key and
"Ora!" keep on
growing as we
keep pressing!

```
area.setOnKeyPressed(new EventHandler<KeyEvent>() {  
    @Override  
    public void handle(KeyEvent event) {  
        area.setText(area.getText()+ "Ora! ");  
    }  
});
```

Mouse Drag in TextArea

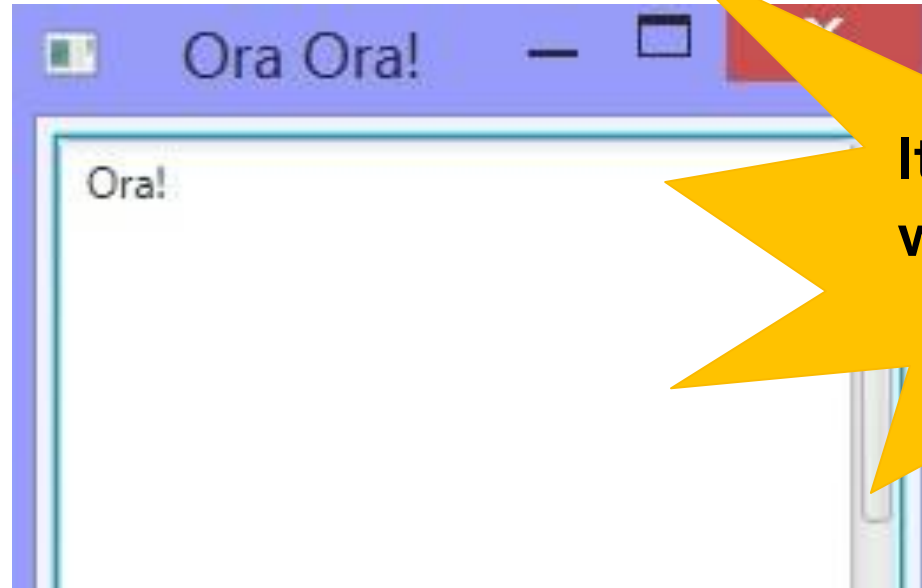
**Make sure you
understand the actual
meaning of events
inside a component!**

**Nothing
happens!**

**But it works when
you are dragging
the left most side
of the area!**

```
area.setOnMouseDragged(new EventListener() {  
    //does not make much sense  
    //does not do anything.  
    @Override  
    public void handle(MouseEvent e) {  
        area.setText(area.getText() + " ");  
    }  
});
```

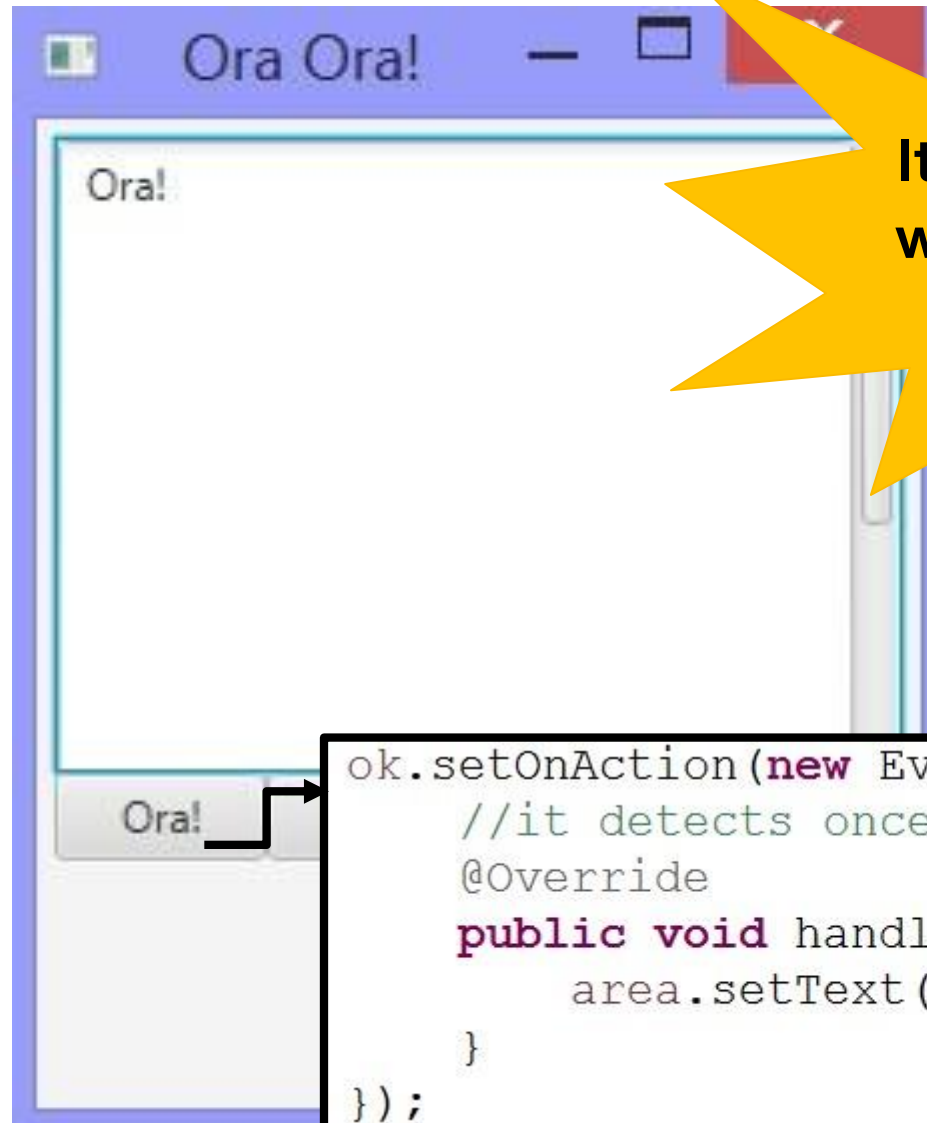
Mouse Click in TextArea



It detects **ONCE**,
when the mouse
is **RELEASED**.

```
area.setOnMouseClicked(new EventHandler<MouseEvent>() {  
    //it detects once when the mouse is released.  
    @Override  
    public void handle(MouseEvent event) {  
        area.setText(area.getText()+"Ora! ");  
    }  
});
```

Button click



```
ok.setAction(new EventHandler<ActionEvent>() {  
    //it detects once, when you release!  
    @Override  
    public void handle(ActionEvent event) {  
        area.setText(area.getText() + "Ora! ");  
    }  
});
```

KeyPress in
TextArea, how to
fire the action only
once?

Use boolean
condition to check?

Set the boolean on
key press and reset
it on key release

[KeyTrigger.java]

```
boolean trigger = false;  
area.setOnKeyPressed(new EventHandler<KeyEvent>() {  
    @Override  
    public void handle(KeyEvent event) {  
        if(!trigger) {  
            trigger = true;  
            area.setText(area.getText() + "Ora! ");  
        }  
    }  
});
```

But it's a compile
error!!

Cannot use local
variable!

Solution 1: use
a class variable
instead

[KeyTrigger01.java]



Only 1 Ora! while we
keep pressing key "d".
But "d" is still increasing
since the text area is
editable.

```
area.setOnKeyPressed(new EventHandler<KeyEvent>() {  
    @Override  
    public void handle(KeyEvent event) {  
        if(!trigger) {  
            trigger = true;  
            area.setText(area.getText() + "Ora! ");  
        }  
    }  
});  
  
area.setOnKeyReleased(new EventHandler<KeyEvent>() {  
    @Override  
    public void handle(KeyEvent event) {  
        trigger = false;  
    }  
});
```

Solution 2: use
a listener that
remember the
trigger status

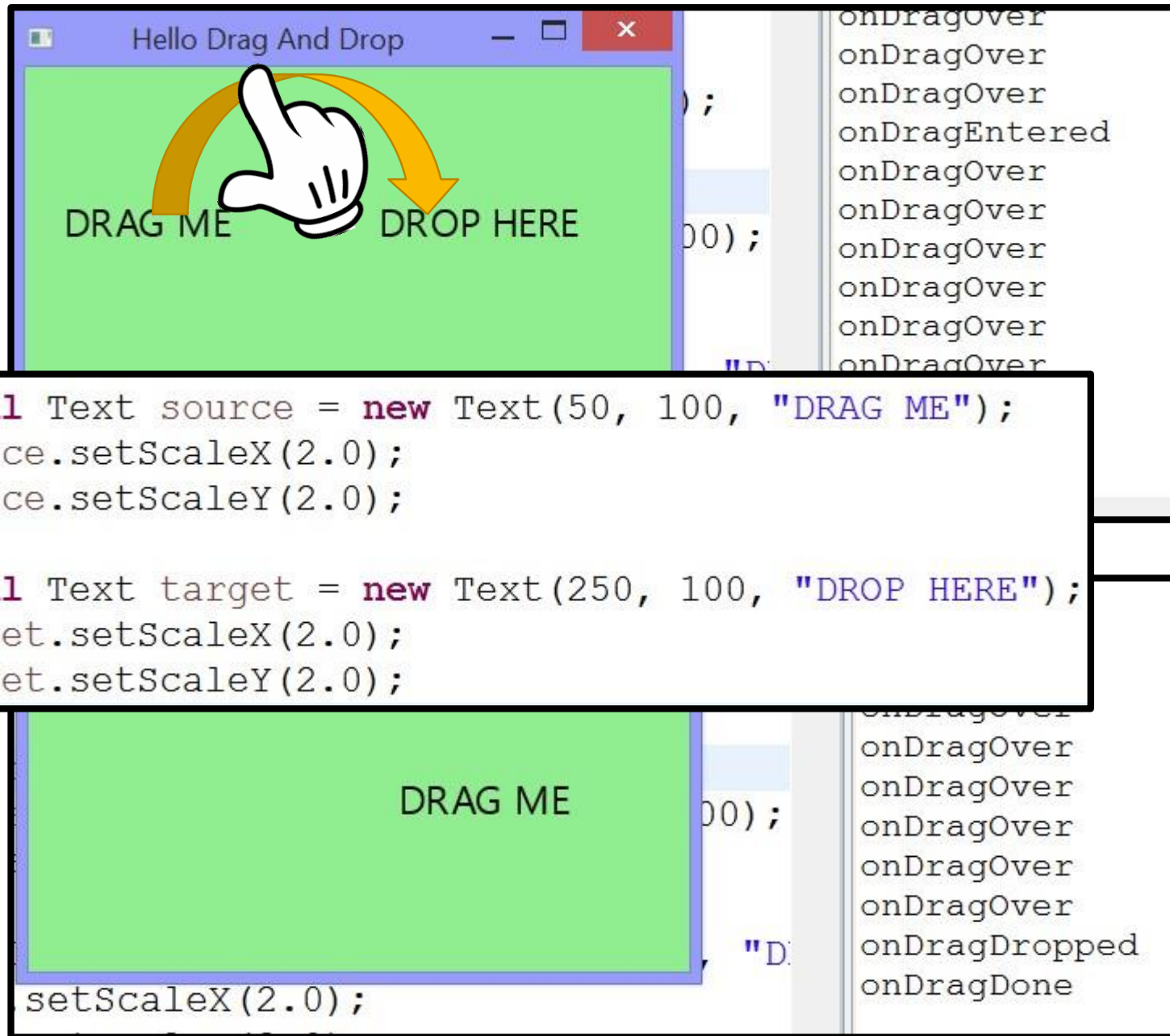
[KeyTrigger02.java]

Yes. You can
write your own
handler
(implements
from an existing
one).

```
MyKeyHandler areaKeyHandler = new MyKeyHandler(area);  
area.setOnKeyPressed(areaKeyHandler);  
|  
area.setOnKeyReleased(new EventHandler<KeyEvent>() {  
    @Override  
    public void handle(KeyEvent event) {  
        areaKeyHandler.setTrigger(false);  
    }  
});
```

```
public class MyKeyHandler implements EventHandler{  
    boolean trigger = false;  
    TextArea area;  
    public MyKeyHandler(TextArea a) {}  
  
    public void setTrigger(boolean t) {  
        trigger = t;  
    }  
  
    @Override  
    public void handle(Event event) {  
        if(trigger) {  
            //do nothing  
        } else {  
            trigger = true;  
            area.setText(area.getText() + "Ora! ");  
        }  
    }  
}
```

[HelloDragAndDrop.java] – from oracle tutorial



What you need to do.

Source

setOnDragDetected

- Call method `startDragAndDrop` and indicate its transfer mode (COPY, MOVE, or LINK)
- Copy data to transfer onto dragboard (clipboard for drag-and-drop)

Visual feedback

setOnDragDone

- Clear original data if necessary.

Destination

setOnDragOver

- Call method `acceptTransferModes` indicating its transfer mode (COPY, MOVE, or LINK). The transfer mode must match the source.

setOnDragEntered

setOnDragExited

setOnDragDropped

- Call method `setDropCompleted`



Source:
setOnDragDetect
ed to initialize.

Copy content as a
ClipboardContent
and put it in our drag
board.

```
source.setOnDragDetected(new EventHandler<MouseEvent>() {  
    @Override  
    public void handle(MouseEvent event) {  
        /* drag was detected, start drag-and-drop gesture */  
        System.out.println("onDragDetected");  
  
        /* allow MOVE transfer mode */  
        Dragboard db = source.startDragAndDrop(TransferMode.MOVE);  
  
        /* put a string on dragboard */  
        ClipboardContent content = new ClipboardContent();  
        content.putString(source.getText());  
        db.setContent(content);  
  
        event.consume();  
    }  
});
```

Consume event so
that it does not get
passed on to other
related nodes.

Destination:
setOnDragOver
to set a
matching
transfer mode.

```
target.setOnDragOver(new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        /* data is dragged over the target */
        System.out.println("onDragOver");

        /*
         * accept it only if it is not dragged from
         * the same node and if it has a string
         * data
         */
        if (event.getGestureSource() != target && event.getDragboard().hasString()) {
            /* allow for moving */
            event.acceptTransferModes(TransferMode.MOVE);
        }

        event.consume();
    }
});
```

Destination:
setOnDragEntered
and
setOnDragExited
to provide Visual
Feedback

Source
must not
come from
itself.

```
target.setOnDragEntered(new EventHandler<DragEvent>() {  
    @Override  
    public void handle(DragEvent event) {  
        /* the drag-and-drop gesture entered the target */  
        System.out.println("onDragEntered");  
        /* show to the user that it is an actual gesture target */  
        if (event.getGestureSource() != target &&  
            event.getDragboard().hasString()) {  
            target.setFill(Color.GREEN);  
        }  
        event.consume();  
    }  
});
```


Must verify
that data
has a correct
type.

```
target.setOnDragExited(new EventHandler<DragEvent>() {  
    @Override  
    public void handle(DragEvent event) {  
        /* mouse moved away, remove the graphical cues */  
        target.setFill(Color.BLACK);  
        event.consume();  
    }  
});
```

Destination: setOnDragDropped

This is what happens when the mouse is released on the destination (DRAG_OVER must work and transfer mode must also be compatible).


```
target.setOnDragDropped(new EventHandler<DragEvent>() {
    @Override
    public void handle(DragEvent event) {
        /* data dropped */
        System.out.println("onDragDropped");
        /* if there is a string data on dragboard, read it and use it */
        Dragboard db = event.getDragboard();
        boolean success = false;
        if (db.hasString()) {
            target.setText(db.getString());
            success = true;
        }
        /*
         * let the source know whether the string
         * was successfully transferred and used
         */
        event.setDropCompleted(success);
        event.consume();
    }
});
```



**Must be
called, or the
drag is never
success.**

Source: setOnDragDone

```
source.setOnDragDone(new EventHandler<DragEvent>() {  
    @Override  
    public void handle(DragEvent event) {  
        /* the drag-and-drop gesture ended */  
        System.out.println("onDragDone");  
        /* if the data was successfully moved, clear it */  
        if (event.getTransferMode() == TransferMode.MOVE) {  
            source.setText("");  
        }  
  
        event.consume();  
    }  
});
```



If this is null,
then the
drag fails.