Objektovo orientované programovanie

Učiteľ:
Ing. Jozef Wagner, PhD.

Učebnica:
https://oop.wagjo.com/

OPG
Teória 22

1. JavaFX udalosti – eventy
2. Typy udalostí a ich spracovanie
3. Capturing, Bubbling, Consuming
4. Udalosti v FXML

JavaFX udalosti

Udalosť – **event** – kliknutie myšou, stlačenie klávesy, zmena veľkosti okna, ...

Udalosť je objekt dediaci z triedy `javafx.event.Event`

Príklady tried: `DragEvent`, `KeyEvent`, `MouseEvent` a `ScrollEvent`

JavaFX udalosti

Každá udalosť obsahuje:

- **Typ udalosti** ktorá nastala, napr. `MouseEvent.MOUSE_CLICKED`
- **Source** (zdroj) - Kde sa práve v procese propagácie udalosti nachádzame
- **Target** (cieľ) - ktorého ovládacieho prvku alebo iného JavaFX objektu sa udalosť týka

Trieda udalosti	Typy udalosti	Detaily
`ActionEvent`	`ACTION`	UI akcia (kliknutie na tlačidlo, výber menu, stlačenie enter v TextField). Signalizuje, že nastala nejaká akcia.
`MouseEvent`	`MOUSE_CLICKED, MOUSE_PRESSED, MOUSE_RELEASED, MOUSE_MOVED, MOUSE_DRAGGED, MOUSE_ENTERED, MOUSE_EXITED`	Interakcia myšou. Obsahuje dodatočné informácie o polohe a tlačidlách: `getX(), getButton(), getClickCount(), isShiftDown()`
`KeyEvent`	`KEY_PRESSED, KEY_RELEASED, KEY_TYPED`	Interakcia klávesnicou. Obsahuje informácie o stlačenej klávese a znaku: `getCode(), getCharacter(), getText(), isControlDown()`
`ScrollEvent`	`SCROLL, SCROLL_STARTED, SCROLL_FINISHED`	Užívateľ scrolluje.
`TouchEvent`	`TOUCH_PRESSED, TOUCH_MOVED, TOUCH_RELEASED`	Interakcia pomocou dotykovej obrazovky. Obsahuje pozíciu dotyku, počet dotykových bodov a iné.
`WindowEvent`	`WINDOW_SHOWING, WINDOW_SHOWN, WINDOW_HIDING, WINDOW_HIDDEN, WINDOW_CLOSE_REQUEST`	Udalosti na úrovni života aplikácie, napríklad zobrazenie alebo zatvorenie okna.

Spracovanie udalosti

Spracovanie udalosti má 4 fázy

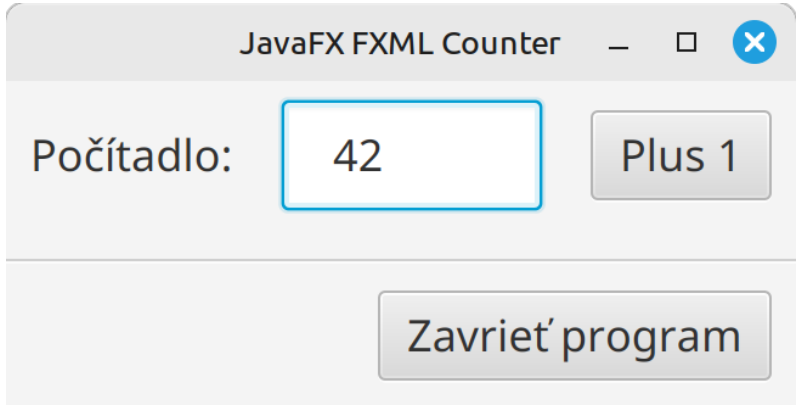
1. **Nájdenie cieľa**
2. **Vytvorenie cesty** ku komponentu v stromovej štruktúre scény - scene graphu
3. Zachytávanie udalosti po ceste smerom od stage ku cieľu - **capturing**
4. Ošetrenie udalosti po ceste od cieľa ku stage - tzv. vybublenie - **bubbling**

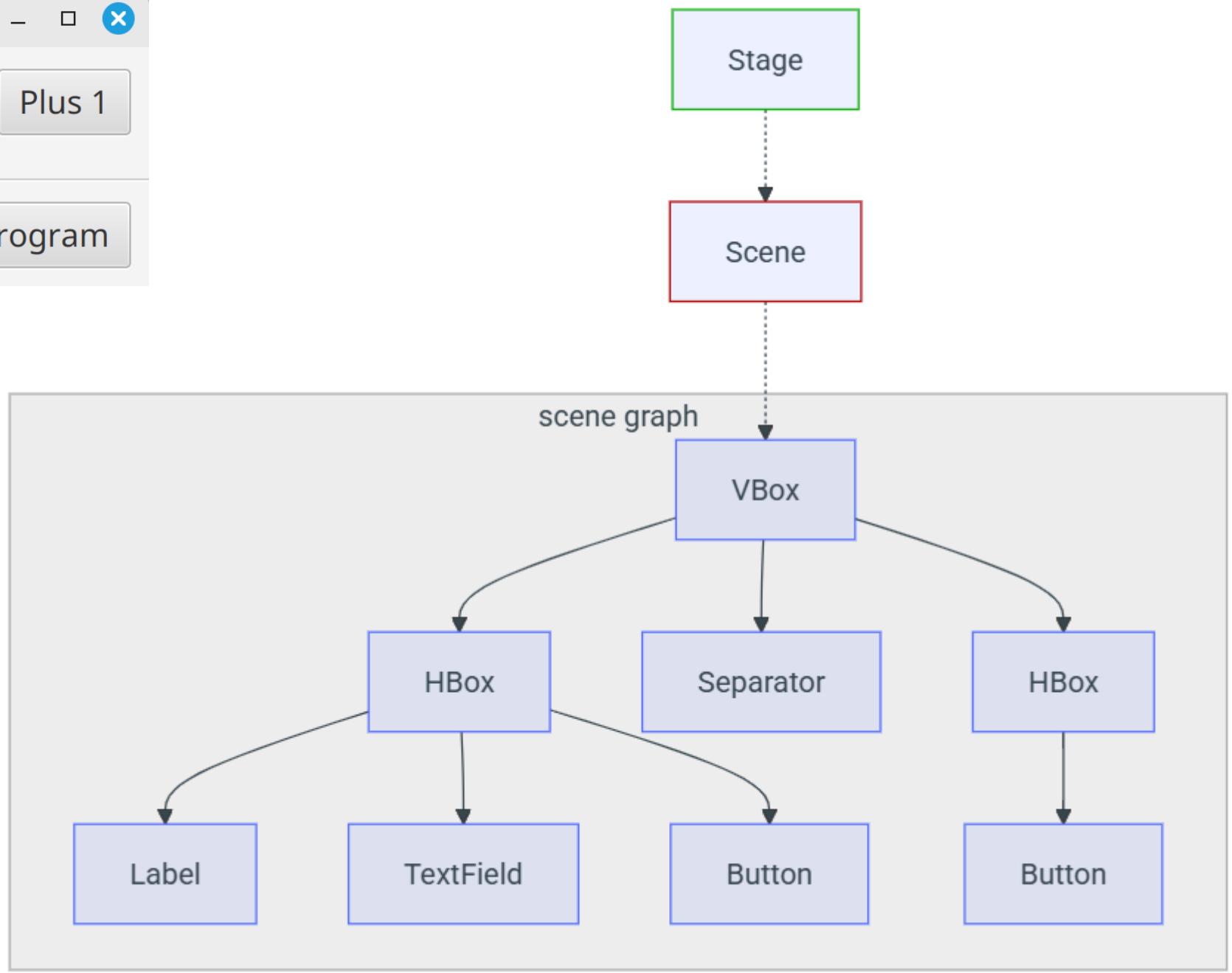
Nájdenie cieľa a vytvorenie cesty

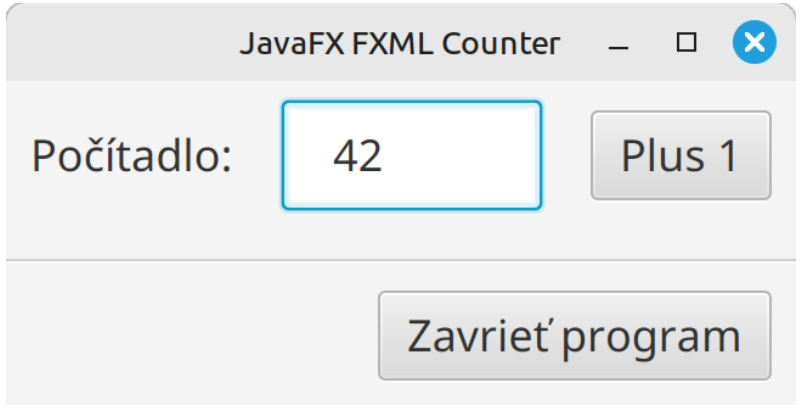
Cieľ je ovládací prvok, napr. tlačidlo, kde udalosť nastala

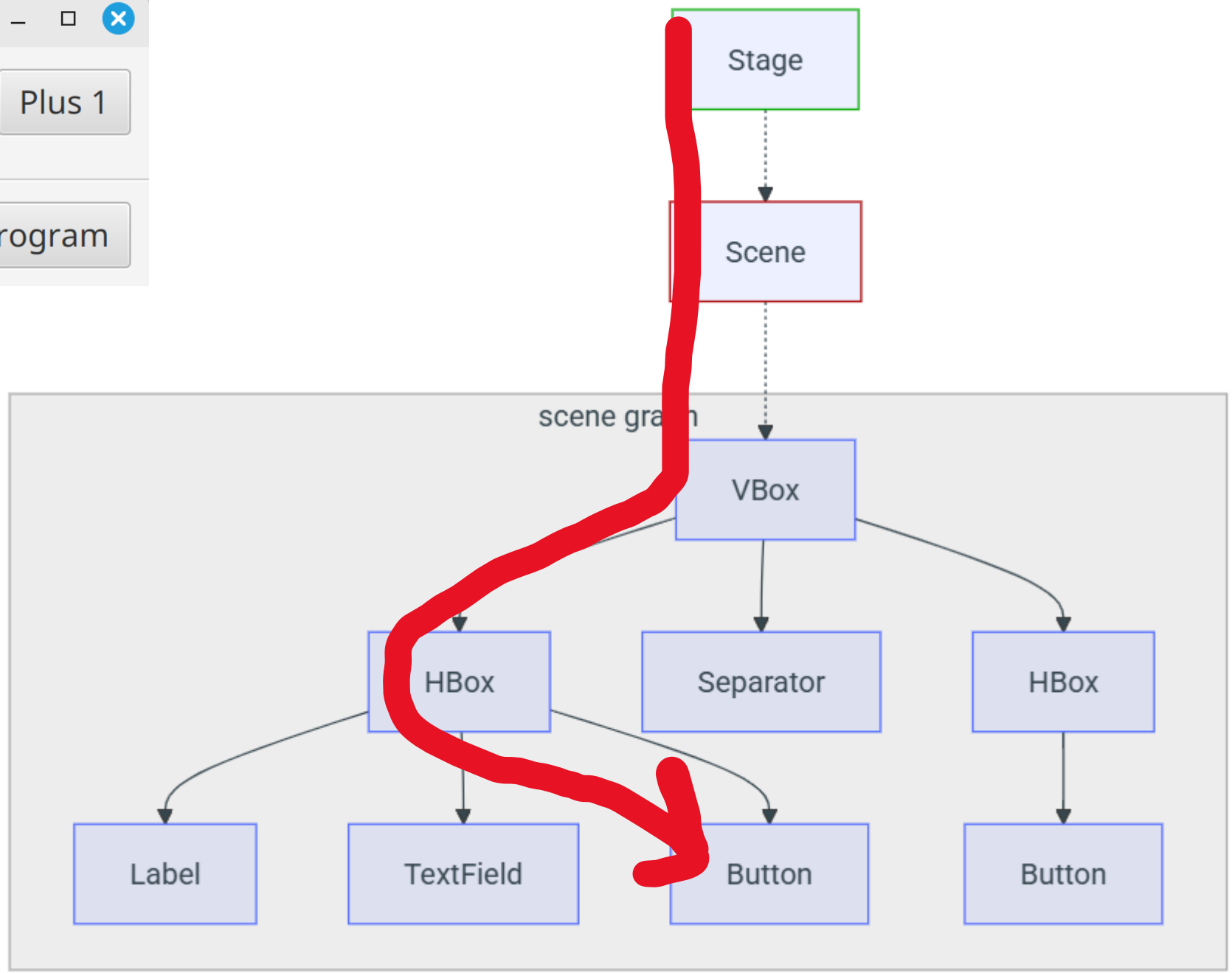
- Udalosti z klávesnice majú cieľ ovládací prvok, ktorý je práve zvolený - anglicky **focus**
- Pre udalosti z myši ide o prvok, na ktorom sa práve nachádza kurzor myši

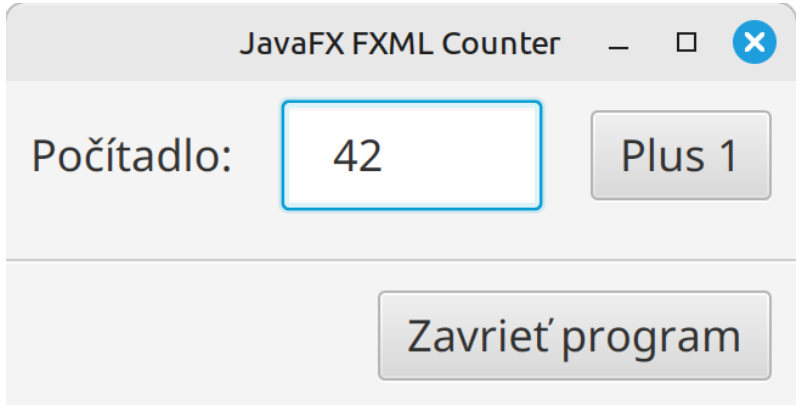
Cesta je v stromovej štruktúre scény a vedie od okna - stage objektu, až po cieľ

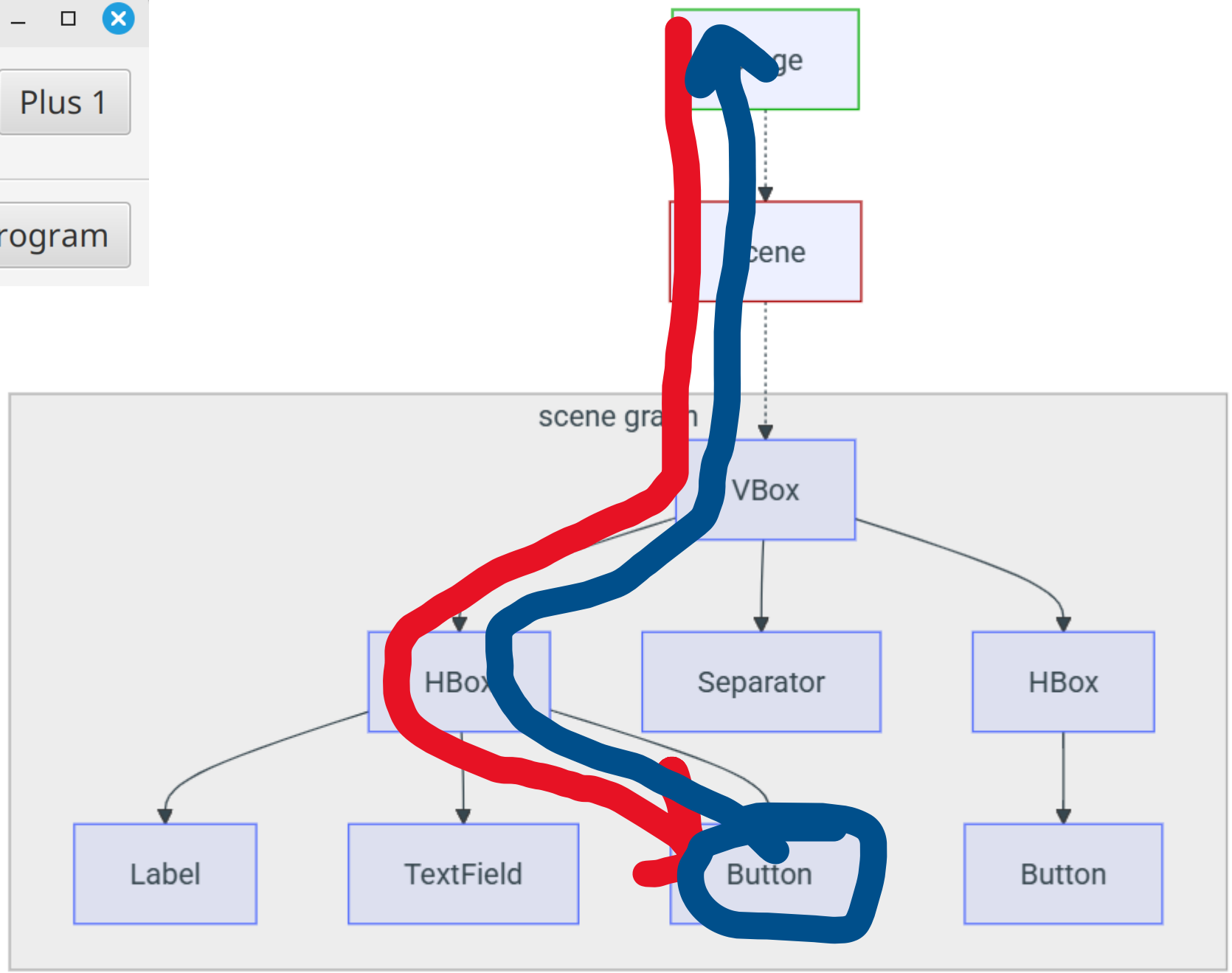

JavaFX FXML Counter
Počítadlo:
42
Plus 1
Zavrieť program


Stage
Scene
scene graph
VBox
HBox
Separator
HBox
Label
TextField
Button
Button


JavaFX FXML Counter
Počítadlo:
42
Plus 1
Zavrieť program


Stage
Scene
VBox
HBox
Separator
HBox
Label
TextField
Button
Button


JavaFX FXML Counter
Počítadlo:
42
Plus 1
Zavrieť program


ge
cene
scene gra h
VBox
HBox
Separator
HBox
Label
TextField
Button
Button

Zachytávanie udalosti - capturing

Udalosť je odoslaná **zo stage a prechádza cestou až k cieľu**.

Počas toho sa udalosť filtruje. Ak má uzol v ceste zaregistrovaný filter, tak sa vykoná.

Využitie na logovanie a blokovanie

Filter je funkcia. Zaregistrujeme ju pomocou metódy `addEventFilter` v komponente, cez ktorú udalosť prechádza

Source v udalosti ukazuje na uzol, na ktorom sa filter nachádza.

Cieľ sa nemení a je stále nastavený na cieľový objekt

Zachytávanie udalosti - capturing

```
hBox.addEventFilter(MouseEvent.MOUSE_CLICKED, e -> {
    System.out.println("Niekto klikol myšou");
});
```

Ošetrenie udalosti - bubbling

Udalosť sa začne vracať späť po ceste v opačnom smere.

Ak má uzol zaregistrovaný handler udalostí, tento handler sa vykoná.

Väčšinou sa registruje handler na **cieľovom** prvku

Registrujeme pomocou metódy `addEventHandler`

Pomocné metódy na registráciu začínajú názvom `setOn`, napr. `setOnMouseClicked`, `setOnKeyTyped` atď.

Ošetrenie udalosti - bubbling

```
plus1Button.addEventHandler(ActionEvent.ACTION, e -> {
    System.out.println("Plus 1 tlačidlo stlačené");
});

plus1Button.setOnAction(e -> {
    System.out.println("Plus 1 tlačidlo stlačené");
});
```

Skonzumovanie udalosti - consuming

```
event.consume()
```

Spracovanie udalosti sa ukončí predčasne.

Predvolené handlery skozumujú väčšinu vstupných udalostí, hlavne tie myšou

```
plus1Button.setOnAction(e -> {
    System.out.println("Plus 1 tlačidlo stlačené");
    e.consume();
  });
```

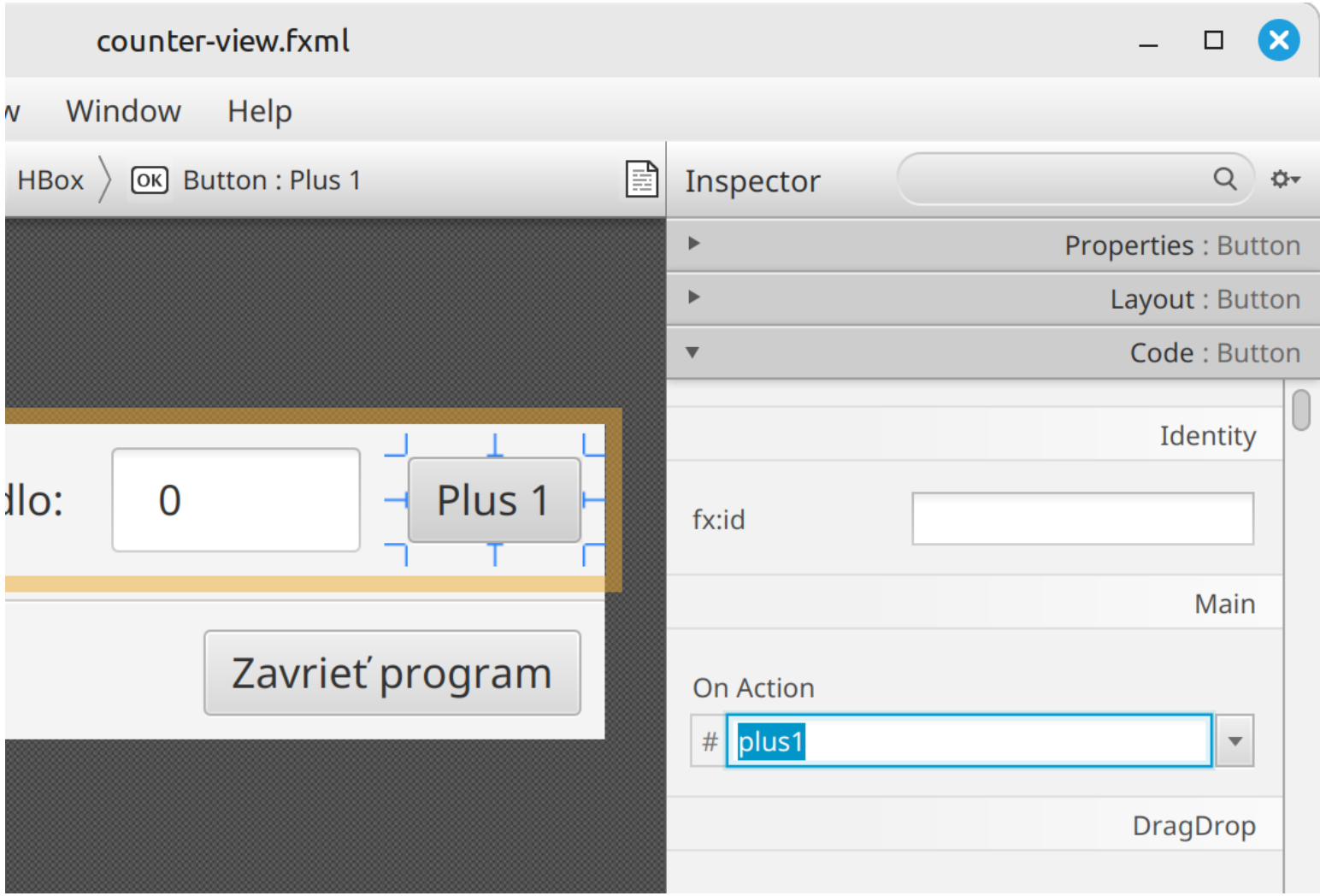
Udalosti a FXML

1. Handler funkciu vytvoríme v **Controller** triede aplikácie

```
public class Controller {
   //...
   @FXML
   public void plus1(ActionEvent actionEvent) {
      System.out.println("Plus 1 tlačidlo stlačené");
   }
}
```

Udalosti a FXML

2. V Scene builder si označíme komponentu, pre ktorú chceme zaregistrovať handler, a v paneli "Code" vyberieme handler funkciu pre udalosť, ktorú chceme ošetriť



Udalosti a FXML

3. Vo výsledom `FXML` súbore sa nám potom toto zaregistrovanie zapíše ako atribút, napr. `onAction`:

```
<Button onAction="#plus1" style="-fx-font-size: 18px;"
text="Plus 1" />
```