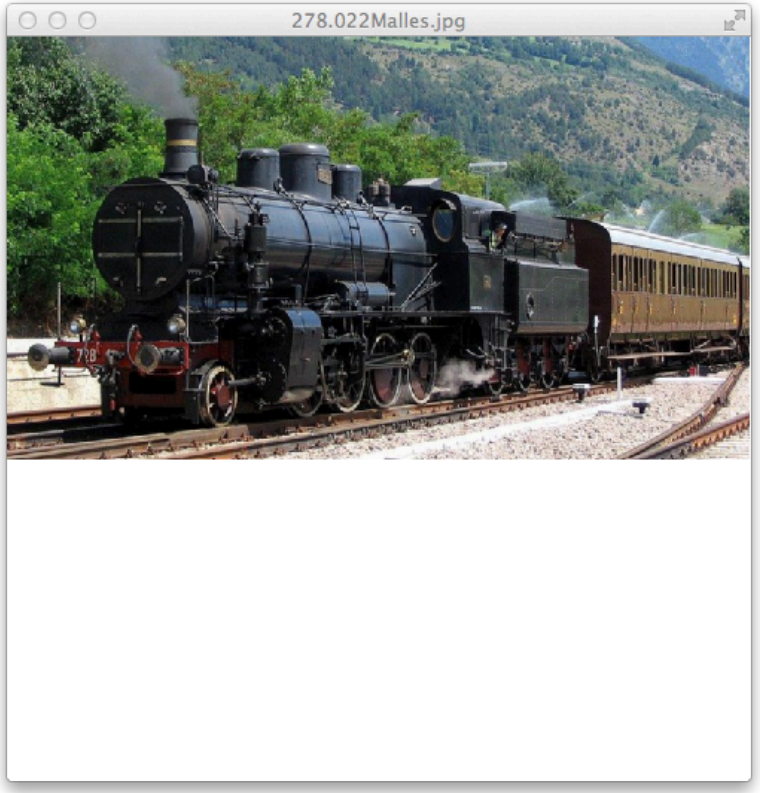
Indipendenza dal S.O.

Image and File

```
        Image image = new Image("file://" +
                    file.getAbsolutePath(), 500, 500, true, true);
        ImageView iw = new ImageView(image);
        Group root = new Group(iw);
        Scene scene = new Scene(root, 500,500);
        stage.setTitle(file.getName());
        stage.setScene(scene);
        stage.sizeToScene();
        stage.show();
    }

    public static void main(String[] args) {
        Application.launch(args);
    }
}
```



```
Su unix/mac
image = new Image("file://" +
                  file.getAbsolutePath(), 500, 500, true, true);

su windows
image = new Image("file:\\" +
                  file.getAbsolutePath(), 500, 500, true, true);
```

System.properties

Key	Meaning
"file.separator"	Character that separates components of a file path. This is "/" on UNIX and "\" on Windows.
"java.class.path"	Path used to find directories and JAR archives containing class files. Elements of the class path are separated by a platform-specific character specified in the path.separator property.
"java.home"	Installation directory for Java Runtime Environment (JRE)
"java.vendor"	JRE vendor name
"java.vendor.url"	JRE vendor URL
"java.version"	JRE version number
"line.separator"	Sequence used by operating system to separate lines in text files
"os.arch"	Operating system architecture
"os.name"	Operating system name
"os.version"	Operating system version
"path.separator"	Path separator character used in java.class.path
"user.dir"	User working directory
"user.home"	User home directory
"user.name"	User account name

https://docs.oracle.com/javase/tutorial/essential/environment/sysprop.html

Su unix/mac

```
image = new Image("file://" +
                  file.getAbsolutePath(), 500, 500, true, true);
```

su windows

```
image = new Image("file:\\" +
                  file.getAbsolutePath(), 500, 500, true, true);
```

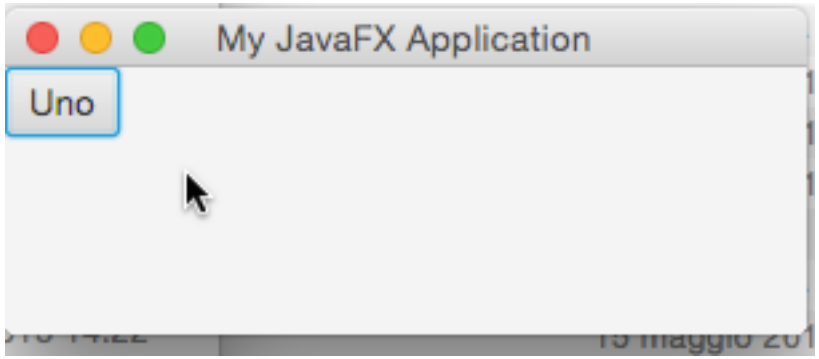
Soluzione generale:

```
String fs=System.getProperty("file.separator");
image = new Image("file:" + fs +fs +
                  file.getAbsolutePath(), 500, 500, true, true);
```

Eventi di tastiera: il fuoco

Un app con un bottone…

```
public class Keyboard1 extends Application {
      int counter=0;
      public void start(Stage stage) {
         TilePane box=new TilePane();
         box.setHgap(50);
         final Button b1=new Button("Uno");
          box.getChildren().addAll(b1,b2);
           EventHandler actionHandler=new EventHandler(){
              public void handle(Event t) {
                  System.out.println((counter++)+
                        ((Button)(t.getTarget())).getText());
              }};
          b1.addEventHandler(ActionEvent.ACTION, actionHandler);
          Scene scene = new Scene(box, 400, 300);
          stage.setTitle("My JavaFX Application");
          stage.setScene(scene); stage.show();
      }
      public static void main(String[] args){Application.launch(args);}
}
```



0Uno
1Uno
2Uno
3Uno

...catturiamo gli eventi di keyboard

```
// inseriamo nello start() il seguente codice:
        EventHandler<KeyEvent> keyEventHandler =
                        new EventHandler<KeyEvent>() {
                            public void handle(KeyEvent keyEvent) {
                                if (keyEvent.getCode() == KeyCode.U) {
                                    b1.fireEvent(new ActionEvent());
                                    System.out.println(keyEvent.getSource()
                                          +" => "+keyEvent.getTarget());
                                }
                            }
                        };
        b1.addEventHandler(KeyEvent.KEY_PRESSED, keyEventHandler);
```

Button[id=null, styleClass=button] => Button[id=null, styleClass=button]

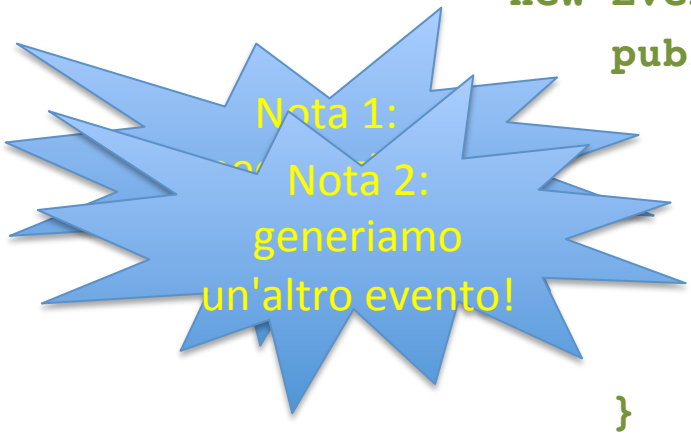
...catturiamo gli eventi di keyboard

```
// inseriamo nello start() il seguente codice:
        EventHandler<KeyEvent> keyEventHandler =
                        new EventHandler<KeyEvent>() {
                            public void handle(KeyEvent keyEvent) {
                                if (keyEvent.getCode() == KeyCode.U) {
                                    b1.fireEvent(new ActionEvent());
                                    System.out.println(keyEvent.getSource()
                                          +" => "+keyEvent.getTarget());
                                }
                            }
                        };
        b1.addEventHandler(KeyEvent.KEY_PRESSED, keyEventHandler);
```

Button[id=null, styleClass=button] => Button[id=null, styleClass=button]

...catturiamo gli eventi di keyboard

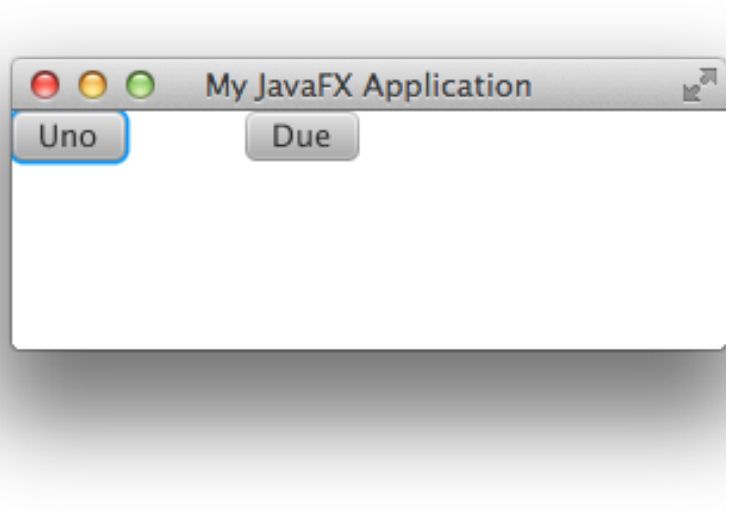
```
// inseriamo nello start() il seguente codice:
        EventHandler<KeyEvent> keyEventHandler =
                    new EventHandler<KeyEvent>() {
                        public void handle(KeyEvent keyEvent) {
                            if (keyEvent.getCode() == KeyCode.U) {
                                b1.fireEvent(new ActionEvent());
                                System.out.println(keyEvent.getSource()
                                        +" => "+keyEvent.getTarget());
                            }
                        }
                    };
        b1.addEventHandler(KeyEvent.KEY_PRESSED, keyEventHandler);
```



Button[id=null, styleClass=button] => Button[id=null, styleClass=button]

Un app con due bottoni…

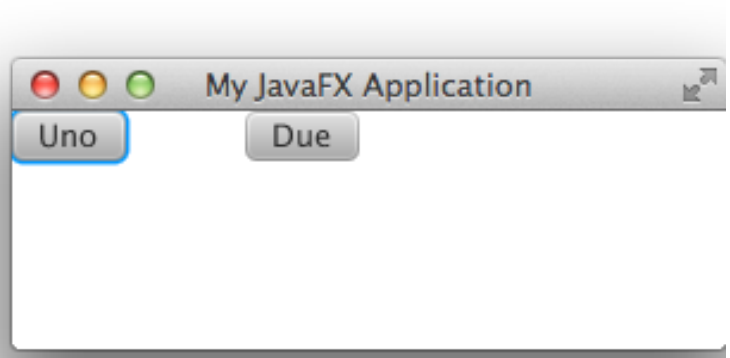
```
public class Keyboard1 extends Application {
      int counter=0;
      public void start(Stage stage) {
         TilePane box=new TilePane();
         box.setHgap(50);
         final Button b1=new Button("Uno");
         final Button b2=new Button("Due");
          box.getChildren().addAll(b1,b2);
           EventHandler actionHandler=new EventHandler(){
              public void handle(Event t) {
                  System.out.println((counter++)+
                         ((Button)(t.getTarget())).getText());
              }
          };
        b1.addEventHandler(ActionEvent.ACTION, actionHandler);
         b2.addEventHandler(ActionEvent.ACTION, actionHandler);
```



0Uno
1Uno
2Uno
3Uno

Aggiungiamo un altro bottone

```
public class Keyboard1 extends Application {
      int counter=0;
      public void start(Stage stage) {
         TilePane box=new TilePane();
         box.setHgap(50);
         final Button b1=new Button("Uno");
         final Button b2=new Button("Due");
          box.getChildren().addAll(b1,b2);
          EventHandler actionHandler=new EventHandler(){
             public void handle(Event t) {
                 System.out.println((counter++)+
                        ((Button)(t.getTarget())).getText());
             }
         };
        b1.addEventHandler(ActionEvent.ACTION, actionHandler);
         b2.addEventHandler(ActionEvent.ACTION, actionHandler);
```



Nota:
riuso lo stesso
ascoltatore

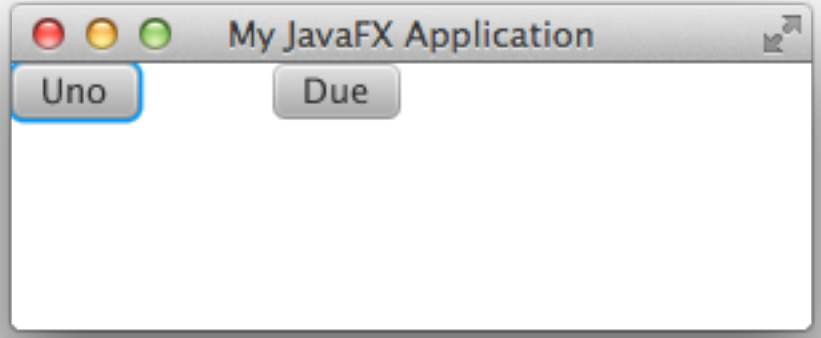
0Uno
1Uno
2Uno
3Uno

...catturiamo gli eventi di keyboard

```
// inseriamo nello start() il seguente codice:
        EventHandler<KeyEvent> keyEventHandler =
                        new EventHandler<KeyEvent>() {
                            public void handle(KeyEvent keyEvent) {
                                if (keyEvent.getCode() == KeyCode.U) {
                                    b1.fireEvent(new ActionEvent());
                                    System.out.println(keyEvent.getSource()
                                          +" => "+keyEvent.getTarget());
                                }
                            }
                        };
        b1.addEventHandler(KeyEvent.KEY_PRESSED, keyEventHandler);
```

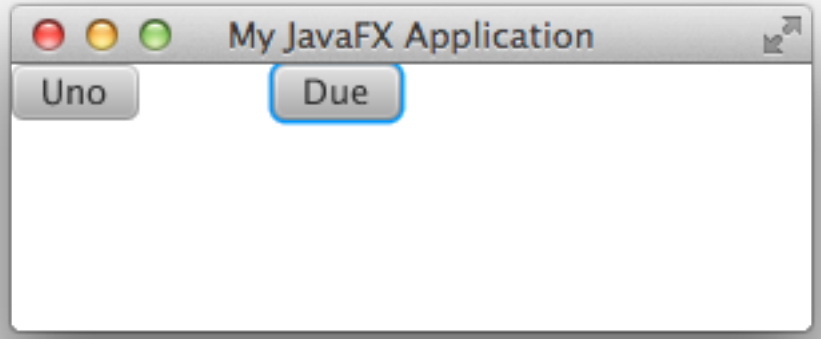
Button[id=null, styleClass=button] => Button[id=null, styleClass=button]

ma funziona?



SI!

Button[id=null, styleClass=button] => Button[id=null, styleClass=button]



NO!

Perché?

- Concetto di FUOCO

Sistemiamola

```
    EventHandler<KeyEvent> keyEventHandler =
                new EventHandler<KeyEvent>() {
                    public void handle(KeyEvent keyEvent) {
                        if (keyEvent.getCode() == KeyCode.U) {
                            b1.fireEvent(new ActionEvent());
                            System.out.println(keyEvent.getSource()
                                  +" => "+keyEvent.getTarget());
                        }
                    }
                };
     Scene scene = new Scene(box, 400, 300);
     b1.addEventHandler(KeyEvent.KEY_PRESSED, keyEventHandler);
     b2.addEventHandler(KeyEvent.KEY_PRESSED, keyEventHandler);
     stage.setTitle("My JavaFX Application");
     stage.setScene(scene);
     stage.show();
}
```

javafx.scene.Scene@68a08ca7 => Button[id=null, styleClass=button]

Sistemiamola meglio

```
    EventHandler<KeyEvent> keyEventHandler =
                new EventHandler<KeyEvent>() {
                    public void handle(KeyEvent keyEvent) {
                        if (keyEvent.getCode() == KeyCode.U) {
                            b1.fireEvent(new ActionEvent());
                            System.out.println(keyEvent.getSource()
                                  +" => "+keyEvent.getTarget());
                        }
                    }
                };
     Scene scene = new Scene(box, 400, 300);
     //b1.addEventHandler(KeyEvent.KEY_PRESSED, keyEventHandler);
     stage.addEventHandler(KeyEvent.KEY_PRESSED, keyEventHandler);
     stage.setTitle("My JavaFX Application");
     stage.setScene(scene);
     stage.show();
}
```

javafx.scene.Scene@68a08ca7 => Button[id=null, styleClass=button]

Ora gestiamo anche l'altro bottone.

```
EventHandler<KeyEvent> keyEventHandler =
                    new EventHandler<KeyEvent>() {
                        public void handle(final KeyEvent keyEvent) {
                            System.out.println(keyEvent.getSource()+"
                                  => "+keyEvent.getTarget());
                            switch (keyEvent.getCode()){
                                case U:
                                case DIGIT1:
                                    b1.fireEvent(new ActionEvent());
                                    break;
                                case D:
                                case DIGIT2:
                                    b2.fireEvent(new ActionEvent());
                                    break;
                            }
                        }
                    };
```

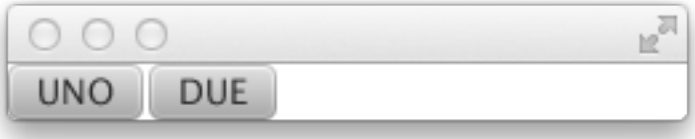
Event Chain

Events

User Action	Event Type	Class
Key on the keyboard is pressed.	KeyEvent	Node, Scene
Mouse is moved or a button on the mouse is pressed.	MouseEvent	Node, Scene
Full mouse press-drag-release action is performed.	MouseDragEvent	Node, Scene
Input from an alternate method for entering characters (typically for a foreign language) is generated, changed, removed, or committed.	InputMethodEvent	Node, Scene
Platform-supported drag and drop action is performed.	DragEvent	Node, Scene
Object is scrolled.	ScrollEvent	Node, Scene
Rotation gesture is performed on an object	RotateEvent	Node, Scene
Swipe gesture is performed on an object	SwipeEvent	Node, Scene
An object is touched	TouchEvent	Node, Scene
Zoom gesture is performed on an object	ZoomEvent	Node, Scene

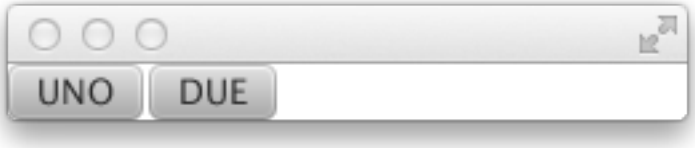
User Action	Event Type	Class
Zoom gesture is performed on an object	ZoomEvent	Node, Scene
Context menu is requested	ContextMenuEvent	Node, Scene
Button is pressed, combo box is shown or hidden, or a menu item is selected.	ActionEvent	ButtonBase,ComboBoxBase,ContextMenu,MenuItem, TextField
Item in a list, table, or tree is edited.	ListView.EditEvent TableColumn.CellEditEvent TreeView.EditEvent	ListView TableColumn TreeView
Media player encounters an error.	MediaErrorEvent	MediaView
Menu is either shown or hidden.	Event	Menu
Popup window is hidden.	Event	PopupWindow
Tab is selected or closed.	Event	Tab
Window is closed, shown, or hidden.	WindowEvent	Window

Event chain v.1- 1



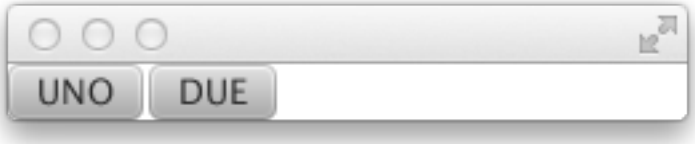

```
public class EventFilterDemo extends Application {
   public void start(final Stage stage) {
      EventHandler handler=new EventHandler<ActionEvent>() {
         @Override
         public void handle(ActionEvent t) {
            EventTarget target=t.getTarget();
            Object source=t.getSource();
            String id=null;
            if (source instanceof Node {
                id=((Node)source).getId();
            } else if (source instanceof Stage) {
                id="STAGE";
            } else if (source instanceof Scene) {
                id="SCENE";
            } else {
                System.out.println("Unrecognized Object"+source);
            }
            System.out.println("HANDLER:"+id+" "+source+" ==> "
                                          +target);
      }
   };
```

Event chain v.1- 2




```
EventHandler filter=new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent t) {
       EventTarget target=t.getTarget();
       Object source=t.getSource();
       String id=null;
       if (source instanceof Node {
           id=((Node)source).getId();
       } else if (source instanceof Stage) {
           id="STAGE";
       } else if (source instanceof Scene) {
           id="SCENE";
       } else {
           System.out.println("Unrecognized Object"+source);
       }
       System.out.println("FILTER:"+id+" "+source+" ==> "
                                        +target);
   }
};
```

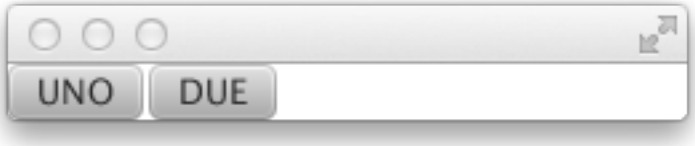
Event chain v.1- 3




```
TilePane layout=new TilePane();
Button button=new Button("Uno");
Button button2=new Button("DUE");
layout.getChildren().addAll(button,button2);
Scene scene = new Scene(layout);
layout.setId("STACKPANE");
button.setId("BUTTON");
button2.setId("BUTTON2");
scene.addEventFilter(ActionEvent.ACTION,filter);
scene.addEventHandler(ActionEvent.ACTION,handler);
stage.addEventFilter(ActionEvent.ACTION,filter);
stage.addEventHandler(ActionEvent.ACTION,handler);
layout.addEventFilter(ActionEvent.ACTION,filter);
layout.addEventHandler(ActionEvent.ACTION,handler);
button2.addEventFilter(ActionEvent.ACTION,filter);
button2.addEventHandler(ActionEvent.ACTION,handler);
button.addEventFilter(ActionEvent.ACTION,filter);
button.addEventHandler(ActionEvent.ACTION,handler);
stage.setScene(scene);
stage.show();
}
```

```
public static void main(String[] args) {
        Application.launch(args);
    }
```

Output



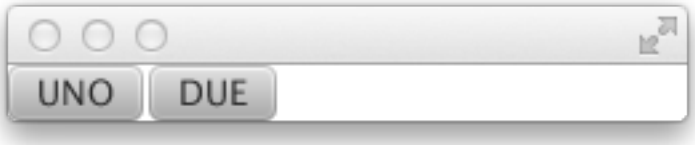

```
FILTER:STAGE javafx.stage.Stage@7c1031ba ==> Button[id=BUTTON-1, styleClass=button]
FILTER:SCENE javafx.scene.Scene@b30e9f8 ==> Button[id=BUTTON-1, styleClass=button]
FILTER:STACKPANE TilePane[id=STACKPANE, styleClass=root] ==> Button[id=BUTTON-1, styleClass=button]
FILTER:BUTTON-1 Button[id=BUTTON-1, styleClass=button] ==> Button[id=BUTTON-1, styleClass=button]
HANDLER:BUTTON-1 Button[id=BUTTON-1, styleClass=button] ==> Button[id=BUTTON-1, styleClass=button]
HANDLER:STACKPANE TilePane[id=STACKPANE, styleClass=root] ==> Button[id=BUTTON-1, styleClass=button]
HANDLER:SCENE javafx.scene.Scene@b30e9f8 ==> Button[id=BUTTON-1, styleClass=button]
HANDLER:STAGE javafx.stage.Stage@7c1031ba ==> Button[id=BUTTON-1, styleClass=button]
FILTER:STAGE javafx.stage.Stage@7c1031ba ==> Button[id=BUTTON-2, styleClass=button]
FILTER:SCENE javafx.scene.Scene@b30e9f8 ==> Button[id=BUTTON-2, styleClass=button]
FILTER:STACKPANE TilePane[id=STACKPANE, styleClass=root] ==> Button[id=BUTTON-2, styleClass=button]
FILTER:BUTTON-2 Button[id=BUTTON-2, styleClass=button] ==> Button[id=BUTTON-2, styleClass=button]
HANDLER:BUTTON-2Button[id=BUTTON-2, styleClass=button] ==> Button[id=BUTTON-2, styleClass=button]
HANDLER:STACKPANE TilePane[id=STACKPANE, styleClass=root] ==> Button[id=BUTTON-2, styleClass=button]
HANDLER:SCENE javafx.scene.Scene@b30e9f8 ==> Button[id=BUTTON-2, styleClass=button]
HANDLER:STAGE javafx.stage.Stage@7c1031ba ==> Button[id=BUTTON-2, styleClass=button]
```

Possiamo evitare la propagazione?

SI! Se ho un evento t,

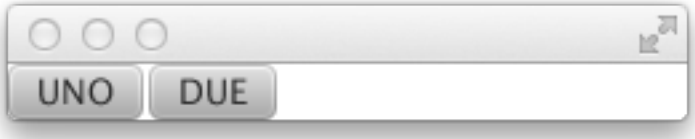
t.consume() evita che l'evento sia passato

Event chain v.2 - 1



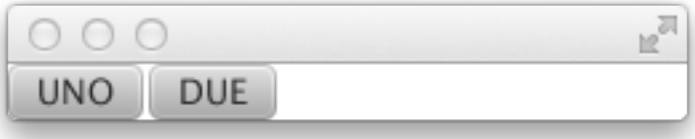

```
public void start(final Stage stage) {
        class SuperHandler<T extends Event> implements EventHandler<T>{
            SuperHandler() {  super(); }
            protected EventTarget target;
            protected Object source;
            protected String id;
            @Override
            public void handle(T t) {
                target=t.getTarget();
                source=t.getSource();
                id=null;
                if (source instanceof Node) {
                    id=((Node)source).getId();
                } else if (source instanceof Stage) {
                    id="STAGE";
                } else if (source instanceof Scene) {
                    id="SCENE";
                } else {
                    System.out.println("Unrecognized Object"+source);
                }
        };
```

Event chain v.2 – 2



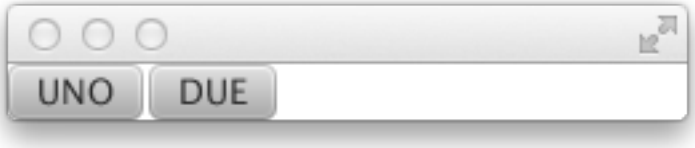

```
SuperHandler<ActionEvent> filter=new SuperHandler<ActionEvent>() {
   public void handle(ActionEvent t) {
      super.handle(t);
      System.out.println("FILTER:"+id+" "+source+" ==> "+target);
   }
};
SuperHandler<ActionEvent> handler=new SuperHandler<ActionEvent>() {
    public void handle(ActionEvent t) {
      super.handle(t);
      System.out.println("HANDLER:"+id+" "+source+" ==> "+target);
   }
};
```

Event chain – cutter 1




```
SuperHandler<ActionEvent> filter=new SuperHandler<ActionEvent>() {
   public void handle(ActionEvent t) {
      super.handle(t);
      System.out.println("FILTER:"+id+" "+source+" ==> "+target);
   }
};
SuperHandler<ActionEvent> handler=new SuperHandler<ActionEvent>() {
    public void handle(ActionEvent t) {
      super.handle(t);
      System.out.println("HANDLER:"+id+" "+source+" ==> "+target);
   }
};

SuperHandler<ActionEvent> cutter=new SuperHandler<ActionEvent>() {
    public void handle(ActionEvent t) {
      super.handle(t);
      System.out.println("CUTTER:"+id+" "+source+" ==> "+target);
      t.consume();
    }
};
```

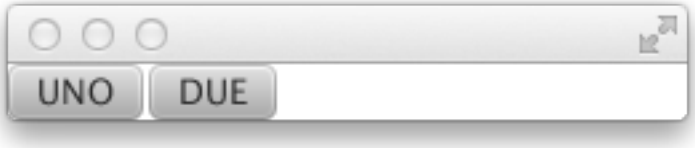
Event chain – cutter 2a




```
scene.addEventFilter(ActionEvent.ACTION,filter);
scene.addEventHandler(ActionEvent.ACTION,handler);
stage.addEventFilter(ActionEvent.ACTION,filter);
stage.addEventHandler(ActionEvent.ACTION,handler);
layout.addEventFilter(ActionEvent.ACTION,cutter);
layout.addEventHandler(ActionEvent.ACTION,handler);
button.addEventFilter(ActionEvent.ACTION,cutter);
button.addEventHandler(ActionEvent.ACTION,handler);
```

FILTER:STAGE javafx.stage.Stage@7c1031ba ==> Button[id=BUTTON-1, styleClass=button]
FILTER:SCENE javafx.scene.Scene@b30e9f8 ==> Button[id=BUTTON-1, styleClass=button]
CUTTER:STACKPANE TilePane[id=STACKPANE, styleClass=root] ==> Button[id=BUTTON-1, styleClass=button]

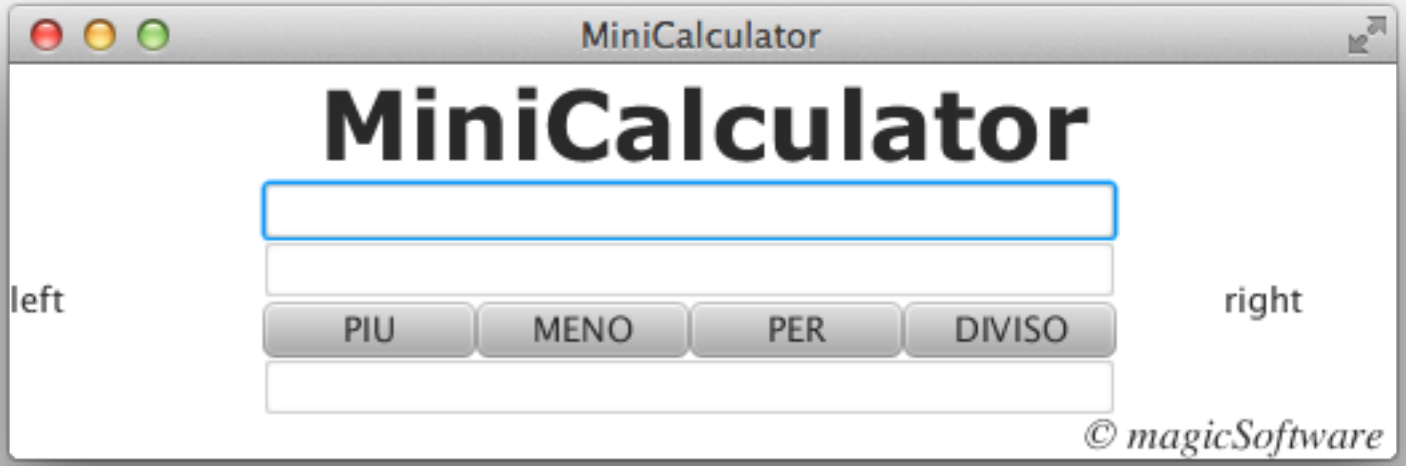
Event chain – cutter 2b




```
scene.addEventFilter(ActionEvent.ACTION,filter);
scene.addEventHandler(ActionEvent.ACTION,handler);
stage.addEventFilter(ActionEvent.ACTION,filter);
stage.addEventHandler(ActionEvent.ACTION,handler);
layout.addEventFilter(ActionEvent.ACTION, filter);
layout.addEventHandler(ActionEvent.ACTION,cutter);
button.addEventFilter(ActionEvent.ACTION, filter);
button.addEventHandler(ActionEvent.ACTION,cutter);
```

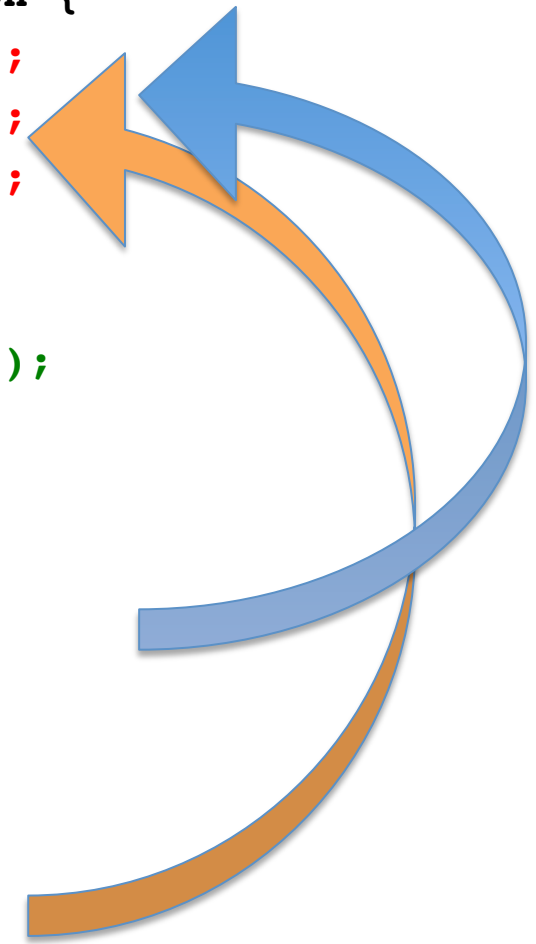
FILTER:STAGE javafx.stage.Stage@7c1031ba ==> Button[id=BUTTON-1, styleClass=button]
FILTER:SCENE javafx.scene.Scene@b30e9f8 ==> Button[id=BUTTON-1, styleClass=button]
FILTER:STACKPANE TilePane[id=STACKPANE, styleClass=root] ==> Button[id=BUTTON-1, styleClass=button]
FILTER:BUTTON-1 Button[id=BUTTON-1, styleClass=button] ==> Button[id=BUTTON-1, styleClass=button]
CUTTER:BUTTON-1 Button[id=BUTTON-1, styleClass=button] ==> Button[id=BUTTON-1, styleClass=button]

Esercizio


MiniCalculator
MiniCalculator
left
right
PIU
MENO
PER
DIVISO
© magicSoftware

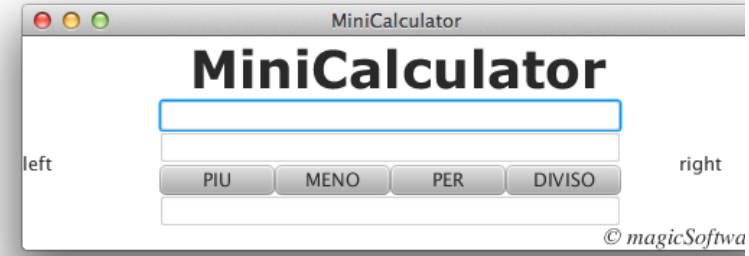
Soluzione - 1

```
public class MiniCalculator extends Application {
    final TextField input1 = new TextField("");
    final TextField input2 = new TextField("");
    final TextField output = new TextField("");

    public void start(Stage primaryStage) {
        primaryStage.setTitle("MiniCalculator");
        …
    }

    public static void main(String[] args) {
        Application.launch(args);
    }

    class MyListener implements EventHandler {
        public void handle(Event t) {
            …

        }
    }
}
```



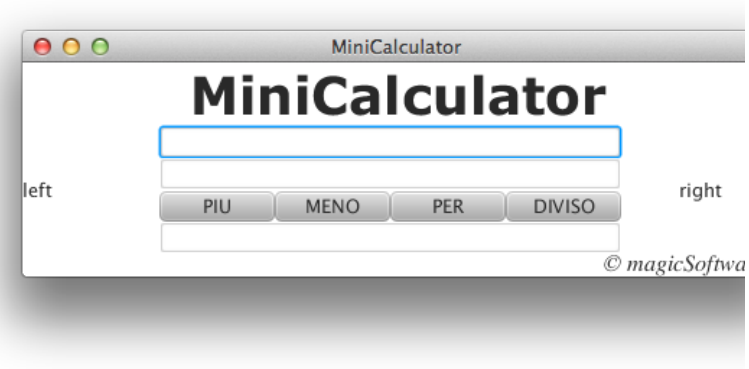
Soluzione - 2

```
public void start(Stage primaryStage) {
        primaryStage.setTitle("MiniCalculator");
        BorderPane borderP = new BorderPane();

        // ======== Top
        Label lt = new Label("MiniCalculator");
        lt.setFont(Font.font("Verdana", FontWeight.BOLD, 36));
        borderP.setTop(lt);
        BorderPane.setAlignment(lt, Pos.CENTER);

        // ======== Right
        Label labright = new Label("right");
        labright.setMinWidth(100.0);
        borderP.setRight(labright);
        BorderPane.setAlignment(labright, Pos.CENTER);

        // ======== Left
        Label lableft = new Label("left");
        lableft.setMinWidth(100.0);
        borderP.setLeft(lableft);
        BorderPane.setAlignment(lableft, Pos.CENTER_LEFT);
```



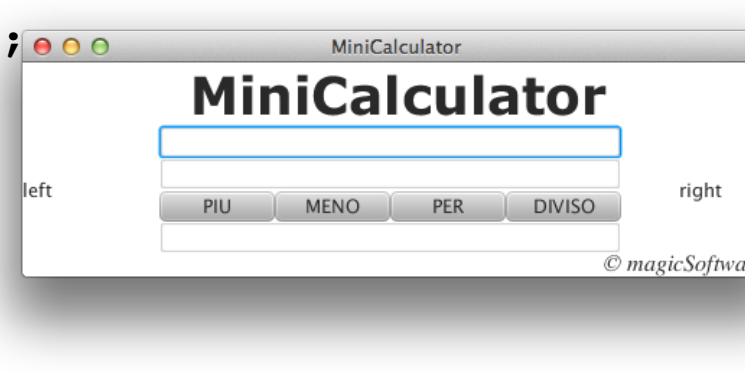
Soluzione - 3

```
// ======== Bottom
Label lb = new Label("© magicSoftware ");
lb.setFont(Font.font("Times", FontPosture.ITALIC, 16));
borderP.setBottom(lb);
BorderPane.setAlignment(lb, Pos.BOTTOM_RIGHT);

// ======== Center
// --- BOX
TilePane box = new TilePane();
box.setPrefColumns(1);
box.setMinWidth(320.0);
box.setMaxWidth(320.0);
// --- Campi
```



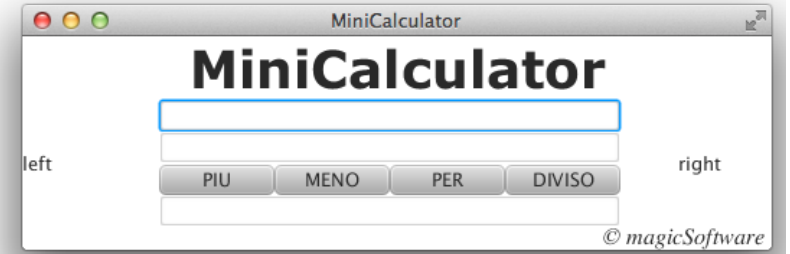
Soluzione - 4

```
// --- Campi

        // --- Bottoni
        TilePane hb = new TilePane();
        hb.setPrefWidth(320.0);
        hb.setAlignment(Pos.CENTER);
        Button sum = new Button("PIU");
        sum.setId("+");
         sum.setPrefWidth(80);
        Button divide = new Button("DIVISO");
        divide.setPrefWidth(80);
        divide.setId("/");

        Button multiply = new Button("PER");
        multiply.setPrefWidth(80);
        multiply.setId("*");

        Button subtract = new Button("MENO");
        subtract.setPrefWidth(80);
        subtract.setId("-");
```



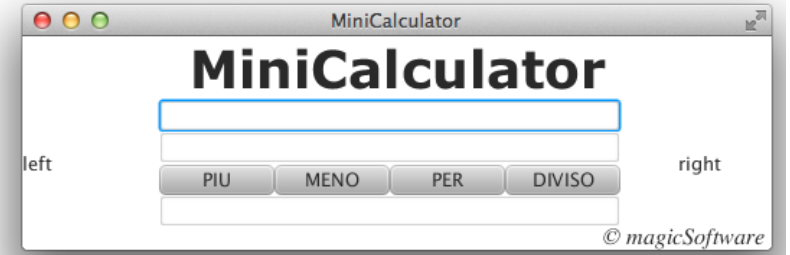
Soluzione - 5

```
        hb.getChildren().addAll(sum, subtract, multiply, divide);
        box.getChildren().addAll(input1, input2, hb, output);

        // ======== Behaviour
        MyListener list = new MyListener();
        multiply.addEventFilter(ActionEvent.ACTION, list);
        sum.addEventFilter(ActionEvent.ACTION, list);
        divide.addEventFilter(ActionEvent.ACTION, list);
        subtract.addEventFilter(ActionEvent.ACTION, list);

        borderP.setCenter(box);
        Scene scene = new Scene(borderP);
        primaryStage.setScene(scene);
        primaryStage.sizeToScene();
        primaryStage.show();
    }
```



Soluzione - 6

```
private class MyListener implements EventHandler {
  public void handle(Event t) {
        String i1 = input1.getText();
        String i2 = input2.getText();
        double o1 = 0;
        double o2 = 0;
        try {
            o1 = Double.parseDouble(i1);
            o2 = Double.parseDouble(i2);
        } catch (NumberFormatException e) {
            Label msg = new Label("Errore - Not A Number!");
            StackPane g = new StackPane();
            g.getChildren().add(msg);
            Scene stageScene = new Scene(g, 200, 200);
            Stage errorStage = new Stage();
            errorStage.setScene(stageScene);
            errorStage.show();
            return;
        }
```



Soluzione - 7

```
            String s = ((Button) t.getTarget()).getId();
            switch (s) {
                case "+":
                    output.setText("" + (o1 + o2));
                    break;
                case "*":
                    output.setText("" + (o1 * o2));
                    break;
                case "-":
                    output.setText("" + (o1 - o2));
                    break;
                case "/":
                    output.setText("" + (o1 / o2));
                    break;


            }

        }
    }
}
```