Accesso a variabili da classi interne

```
public class ColoredCircle
  extends Application {
  Color color=null;
  @Override
  public void start(
     …
     color=Color.BLACK;
      …
      Button btn = new Button();
      btn.addEventHandler(ActionEvent, new Listener()});
    …
  }
  class Listener implements EventHandler<ActionEvent>{
       color=Color.BLACK;
  }
}
```

NO!

Accesso a variabili da classi interne

```
public class ColoredCircle
  extends Application {
  Color color=null;
  @Override
  public void start(
     …
     color=Color.BLACK;
      …
      Button btn = new Button();
      btn.addEventHandler(ActionEvent, new Listener()});
    …
  }
  void setColor(Color c){this.color=c; }
  class Listener implements EventHandler<ActionEvent>{
       setColor(Color.BLACK);
  }
}
```

SI!

Posizionamento automatico: Layouts di base

https://docs.oracle.com/javase/8/javafx/layout-tutorial/index.html

Posizionamento di un Node

Non c’é setX (anche se alcune sottoclassi lo hanno), ma ci sono

- setLayoutX
- setTranslateX

- A che servono?

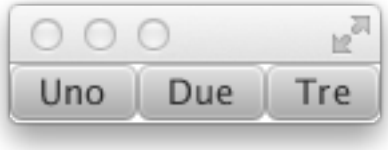
Absolute positioning: Pane

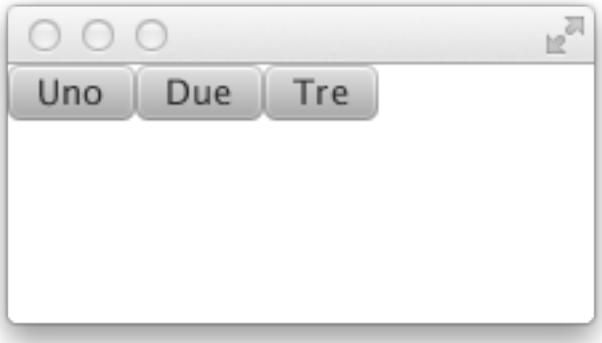
```
public void start(Stage primaryStage) {
    primaryStage.setTitle("Hello World!");
    Button btn = new Button();
    btn.setText("'Hello World'");
    Pane root = new Pane();
    btn.setLayoutX(250);
    btn.setLayoutY(220);
    root.getChildren().add(btn);
    primaryStage.setScene(new Scene(root, 300, 250));
    primaryStage.show();
}
```

Evitare di usarlo!

Layout: HBox

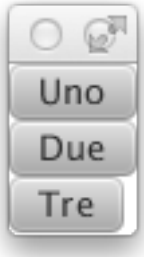
```
public class Layout1 extends Application {
    public void start(Stage stage) {
        Pane layout=new HBox();
        layout.getChildren().add(new Button("Uno"));
        layout.getChildren().add(new Button("Due"));
        layout.getChildren().add(new Button("Tre"));
        Group root = new Group(layout);
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.show();
    }}…}
```



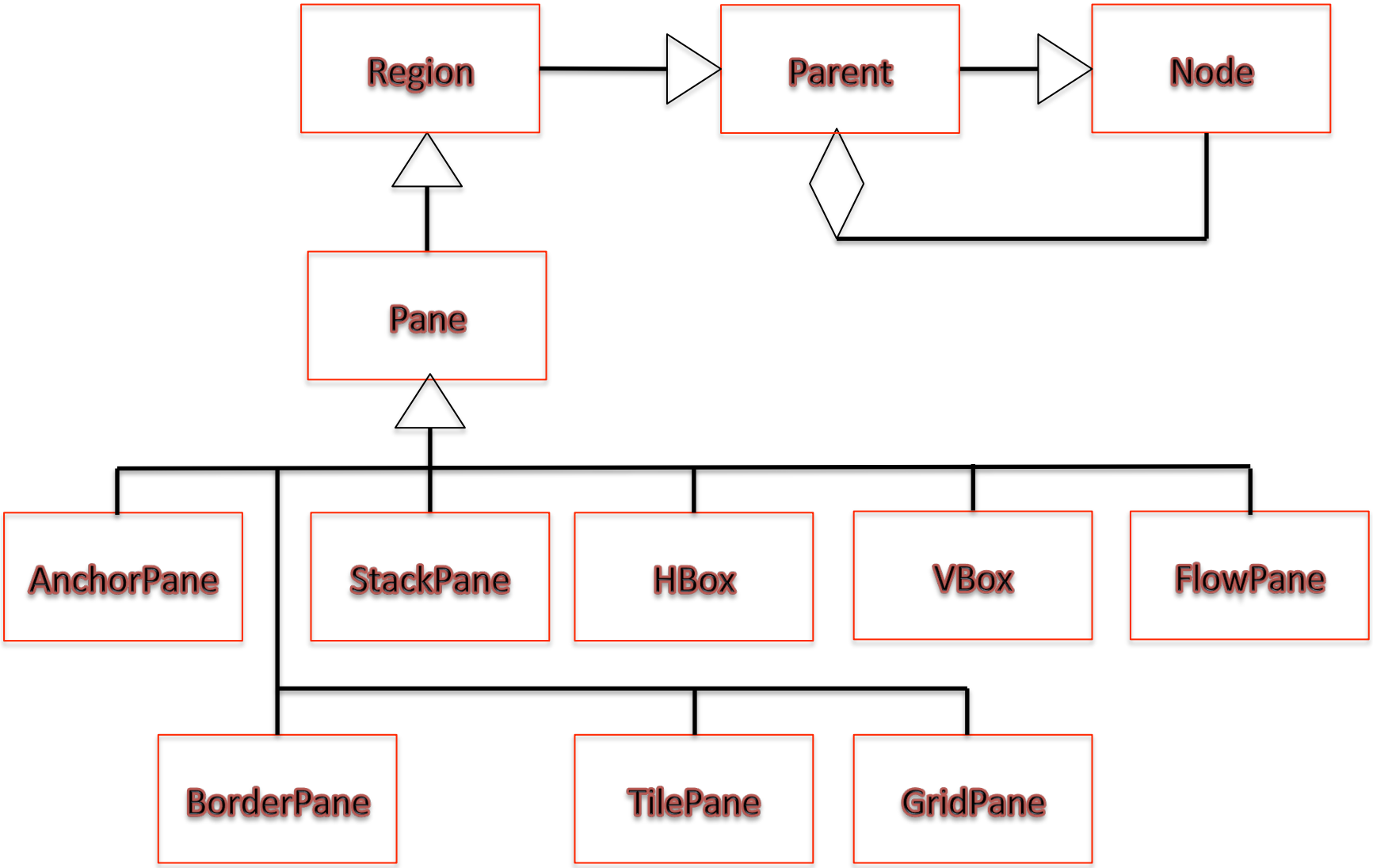


Layout: VBox

```
public class Layout1 extends Application {
    public void start(Stage stage) {
        Pane layout=new VBox();
        layout.getChildren().add(new Button("Uno"));
        layout.getChildren().add(new Button("Due"));
        layout.getChildren().add(new Button("Tre"));
        Group root = new Group(layout);
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.show();
    }}…}
```



MediaView - Media



Container classes that automate common layout models

- The HBox class arranges its content nodes horizontally in a single row.
- The VBox class arranges its content nodes vertically in a single column.
- The StackPane class places its content nodes in a back-to-front single stack.
- The TilePane class places its content nodes in uniformly sized layout cells or tiles
- The FlowPane class arranges its content nodes in either a horizontal or vertical “flow,” wrapping at the specified width (for horizontal) or height (for vertical) boundaries.
- The BorderPane class lays out its content nodes in the top, bottom, right, left, or center region.
- The AnchorPane class enables developers to create anchor nodes to the top, bottom, left side, or center of the layout.
- The GridPane class enables the developer to create a flexible grid of rows and columns in which to lay out content nodes.

To achieve a desired layout structure, different containers can be nested within a JavaFX application.

Layout: StackPane

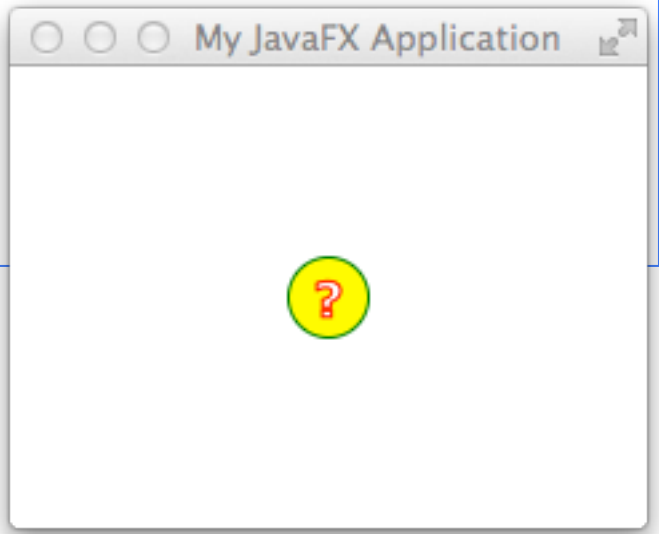
```
public class Layout1 extends Application {
    public void start(Stage stage) {
        StackPane layout=new StackPane();
        layout.getChildren().add(new Button("Uno"));
        layout.getChildren().add(new Button("Due"));
        layout.getChildren().add(new Button("Tre"));
        Group root = new Group(layout);
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.show();
    }}…}
```

Layout: StackPane

```
public class Layout1 extends Application {
    public void start(Stage stage) {
        StackPane layout=new StackPane();
        layout.getChildren().add(new Button("Uno"));
        layout.getChildren().add(new Button("Due"));
        layout.getChildren().add(new Button("Tre"));
        //Group root = new Group(layout);
        //Scene scene = new Scene(root);
        Scene scene = new Scene(layout);
        stage.setScene(scene);
        stage.show();
    }}…}
```

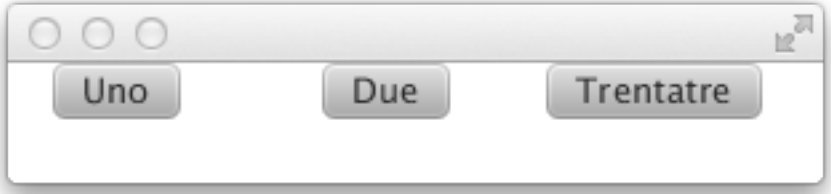
Layout: StackPane

```
public class Layout1 extends Application {
    public void start(Stage stage) {
        StackPane stack = new StackPane();
        Circle helpIcon = new Circle(15, 15, 15);
        helpIcon.setFill(Color.YELLOW);
        helpIcon.setStroke(Color.GREEN);
        Text helpText = new Text("?");
        helpText.setFont(Font.font("Verdana", FontWeight.BOLD, 18));
        helpText.setFill(Color.WHITE);
        helpText.setStroke(Color.RED);
        stack.getChildren().addAll(helpIcon, helpText);
        stack.setAlignment(Pos.CENTER);
        Scene scene = new Scene(stack);
        stage.setTitle("My JavaFX Application");
        stage.setScene(scene);
        stage.show();
    }
    }}…}
```

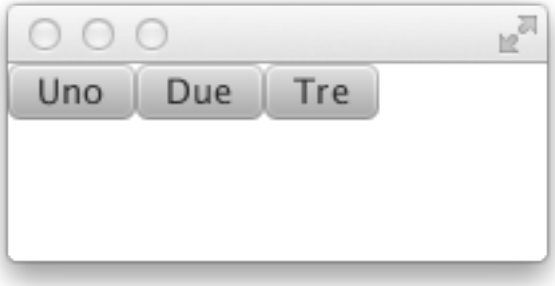


Layout: TilePane


```
public class Layout1 extends Application {
    public void start(Stage stage) {
        //Pane layout=new HBox();
        //Pane layout=new VBox();
        //StackPane layout=new StackPane();
        TilePane layout=new TilePane();
        layout.setVgap(10);
        layout.setHgap(20);
        layout.setPrefColumns(2);
        layout.getChildren().add(new Button("Uno"));
        layout.getChildren().add(new Button("Due"));
        layout.getChildren().add(new Button("Trentatre"));
         Scene scene = new Scene(layout);
        stage.setScene(scene);
        stage.show();
    }}…}
```



FlowPane




```
public class Layout1 extends Application {
    public void start(Stage stage) {
        final FlowPane layout=new FlowPane();
        layout.setPrefWrapLength(100);
        layout.getChildren().add(new Button("Uno"));
        layout.getChildren().add(new Button("Due"));
        layout.getChildren().add(new Button("Tre"));
        Scene scene = new Scene(layout);
        stage.setScene(scene);
        stage.show();
    }…
}
```

Dimensionamento delle componenti

Preferenza:

- btn.setPrefWidth(200);
- btn.setPrefHeight(200);

Vincoli:

- btn.setMinWidth(100);
- btn.setMaxWidth(250);

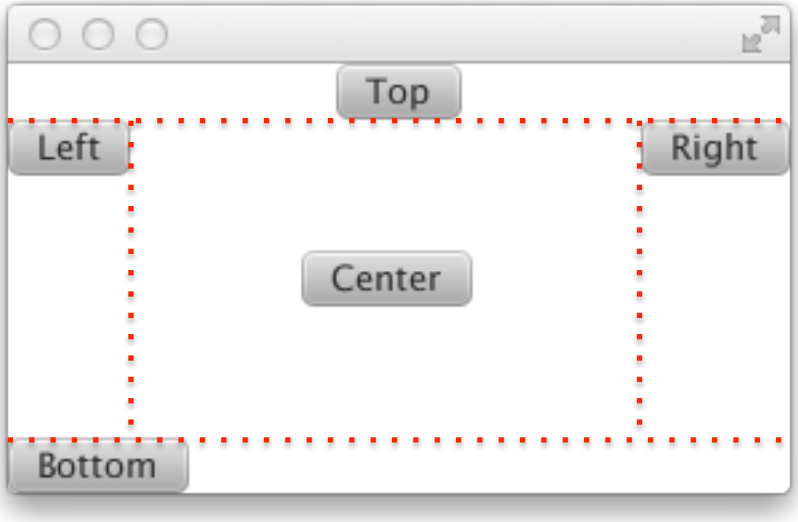
Per dettagli ed esempi di dimensionamento e allineamento, si veda:

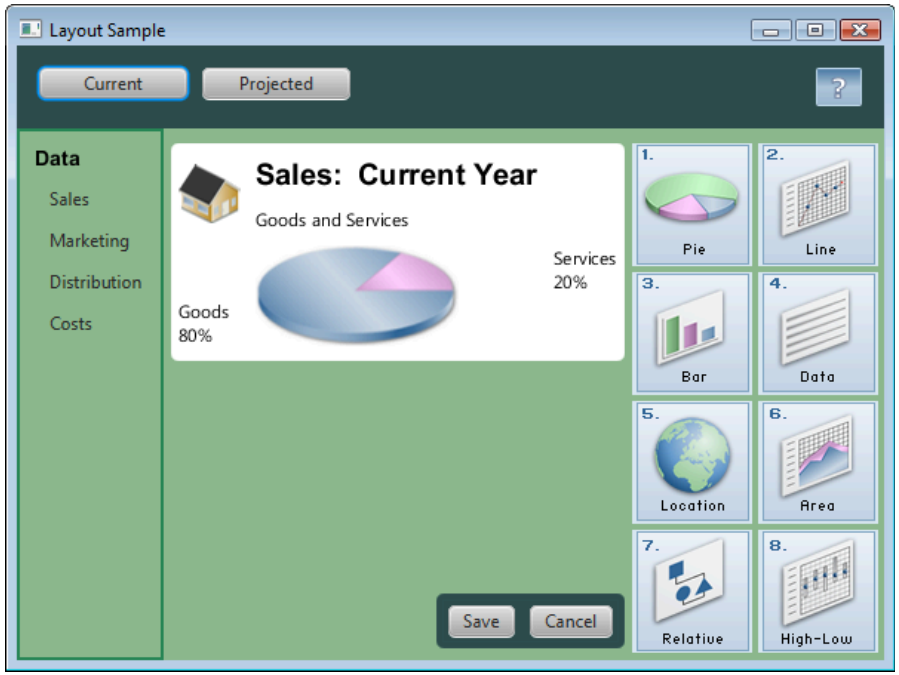
https://docs.oracle.com/javase/8/javafx/layout-tutorial/size_align.htm#JFXLY133

Posizionamento automatico: Layouts avanzati

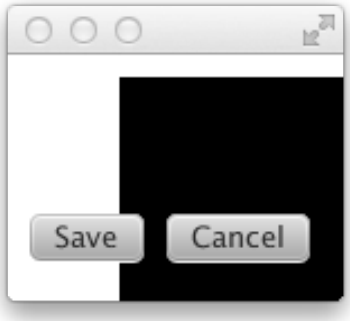
BorderPane

```
public class Layout1 extends Application {
    public void start(Stage stage) {
        BorderPane layout=new BorderPane();
        Button top=new Button("Top");
        BorderPane.setAlignment(top, Pos.TOP_CENTER);
        layout.setTop(top);
        layout.setBottom(new Button("Bottom"));
        layout.setLeft(new Button("Left"));
        layout.setRight(new Button("Right"));
        layout.setCenter(new Button("Center"));
        Scene scene = new Scene(layout);
        stage.setScene(scene);
        stage.show();
    }
…
}
```

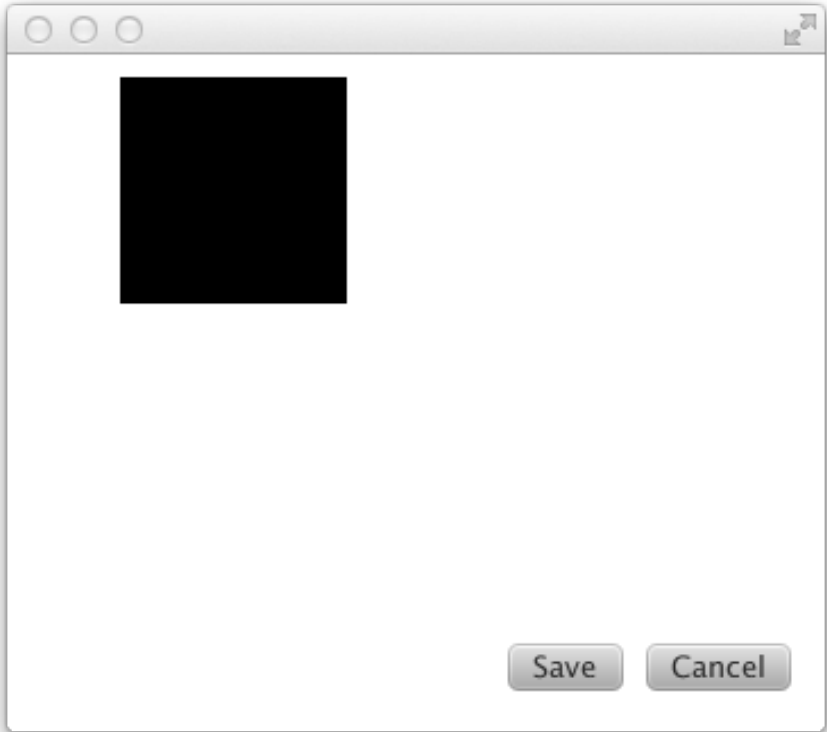




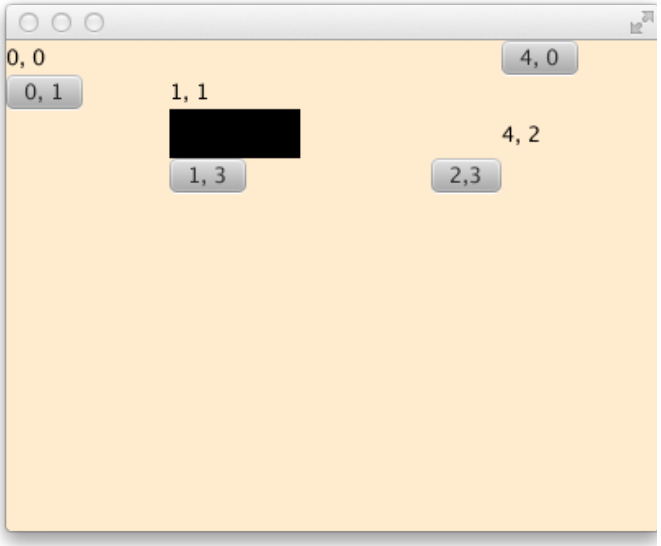
AnchorPane




```
public void start(Stage stage) {
    AnchorPane anchorpane = new AnchorPane();
    Button buttonSave = new Button("Save");
    Button buttonCancel = new Button("Cancel");
    HBox hb = new HBox();
    hb.setPadding(new Insets(0, 10, 10, 10));
    hb.setSpacing(10);
    hb.getChildren().addAll(buttonSave, buttonCancel);
    Rectangle r=new Rectangle(100,100);
    anchorpane.getChildren().addAll(r,hb);
    AnchorPane.setBottomAnchor(hb, 8.0);
    AnchorPane.setRightAnchor(hb, 5.0);
    AnchorPane.setTopAnchor(r, 10.0);
    AnchorPane.setLeftAnchor(r, 50.0);
    Scene scene = new Scene(anchorpane);
    stage.setScene(scene);
    stage.show();
}
```

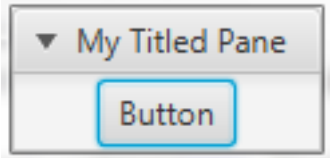


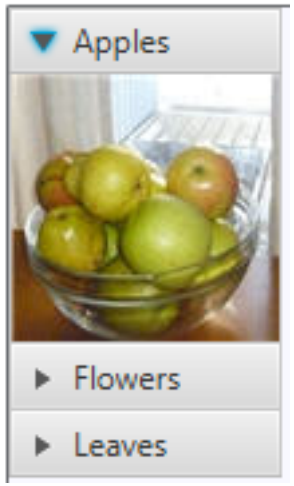
```java
public void start(Stage primaryStage) {
        double width = 400;
        double height = 300;
        GridPane gridPane = new GridPane();
        Scene scene = new Scene(gridPane,
           width, height, Color.BLANCHEDALMOND);
        gridPane.add(new Text("0, 0"), 0, 0);
        gridPane.add(new Button("0, 1"), 0, 1);
        gridPane.add(new Text("1, 1"), 1, 1);
        Rectangle r=new Rectangle(80,30);
        gridPane.add(r, 1, 2);
        gridPane.add(new Button("1, 3"), 1, 3);
        gridPane.add(new Button("2,3"), 2, 3);
        gridPane.add(new Button("4, 0"), 4, 0);
        gridPane.add(new Text("4, 2"), 4, 2);
        ColumnConstraints column1 = new ColumnConstraints(100);
        ColumnConstraints column2 = new ColumnConstraints();
        column2.setPercentWidth(40);
        column2.setHgrow(Priority.ALWAYS);
        gridPane.getColumnConstraints().addAll(column1, column2);
        primaryStage.setScene(scene);
        primaryStage.show();
}
```



GridPane

TitledPane





Accordion

http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/accordion-titledpane.htm#CACGBAHI

Come cambiare le coordinate di un oggetto posizionato da un Pane?

public final double getTranslateX()

Gets the value of the property translateX.

Property description:

Defines the x coordinate of the translation that is added to this Node's transform.

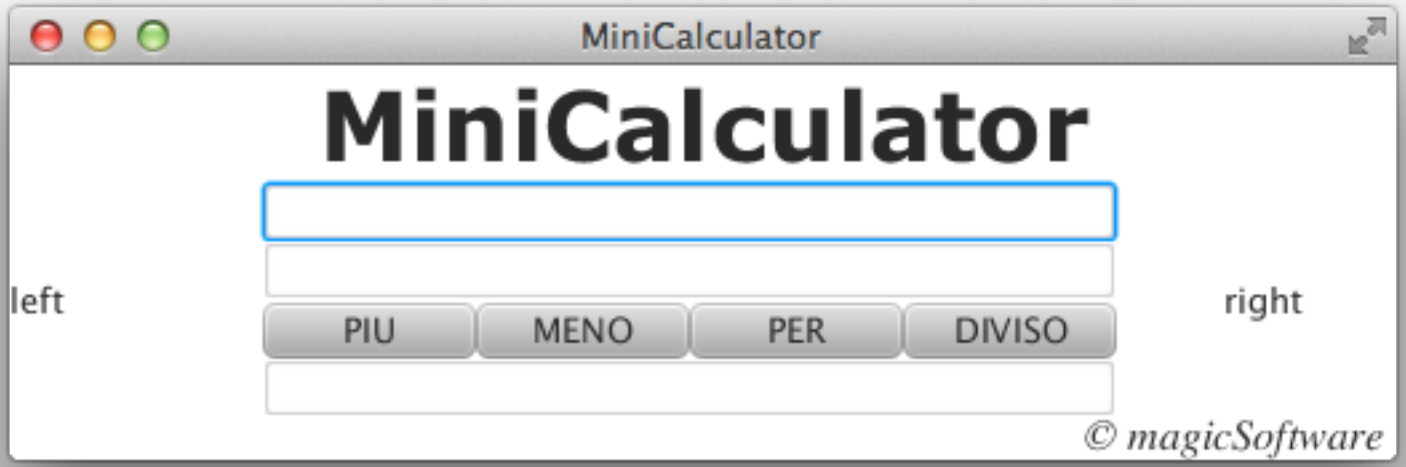
The node's final translation will be computed as layoutX + translateX, where layoutX establishes the node's stable position and translateX optionally makes dynamic adjustments to that position.

Layout e Translate properties

- getLayoutX
- setTranslateX
- getTraslateX

- stesso per Y

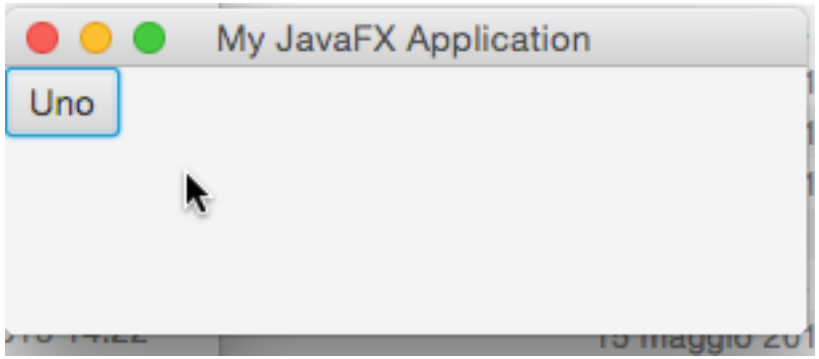
Esercizio


MiniCalculator
MiniCalculator
left
right
PIU
MENO
PER
DIVISO
© magicSoftware

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);
```

Nota 1: specializziamo l'evento gestito

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);
```

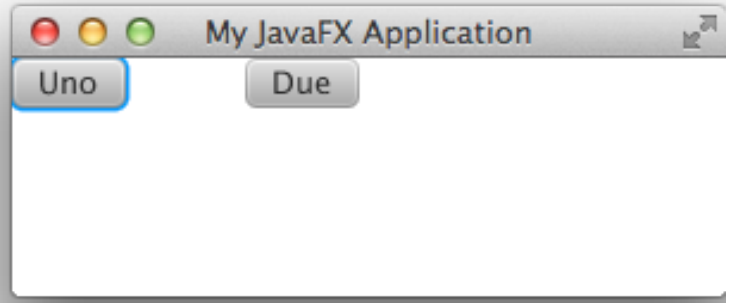
Nota 1:

Nota 2:
generiamo
un'altro evento!

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

Un app con due bottoni…

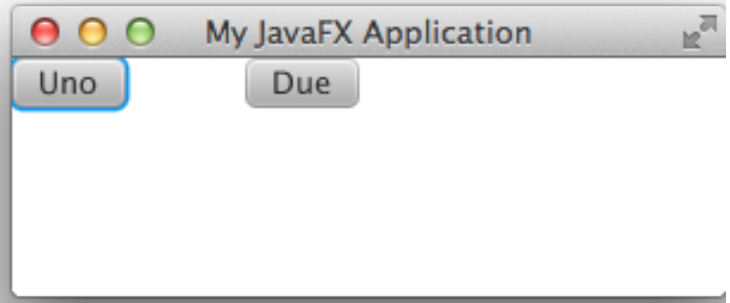
```
public class Keyboard1 extends Application {
     int counter=0;
     public void start(Stage stage) {
        TilePane box=new TilePane();
        box.setHgap(50);
        final Button b1=new Button("Uno");
        final Button b2=new Button("Due");
        box.getChildren().addAll(b1,b2);
         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

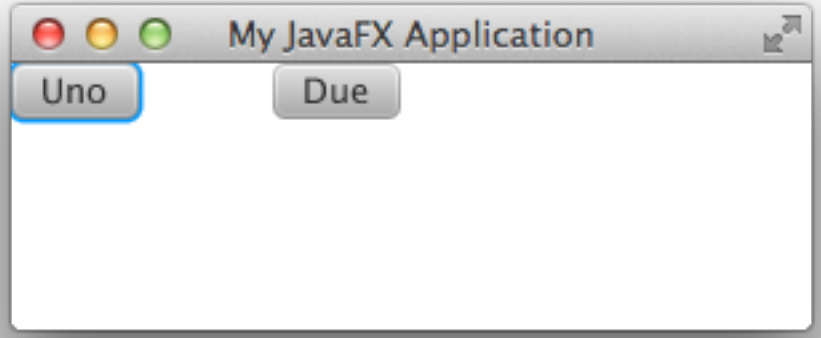
...catturiamo gli eventi di keyboard

```
// 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);
```

Lasciamo invariato il controllo della tastiera

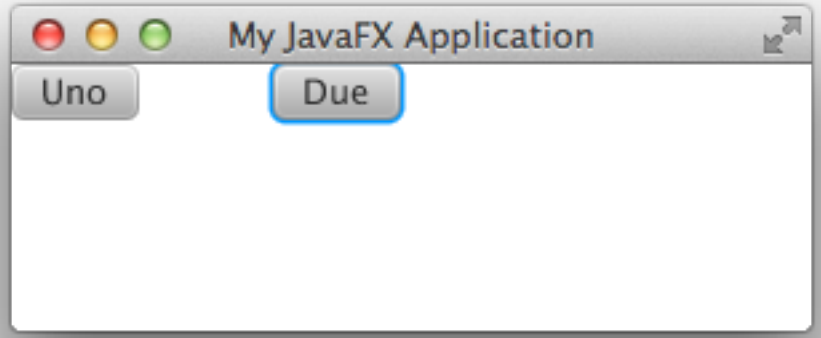
Button[id=null, styleClass=button] => Button[id=null, styleClass=button]

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;
                            }
                        }
                    };
```