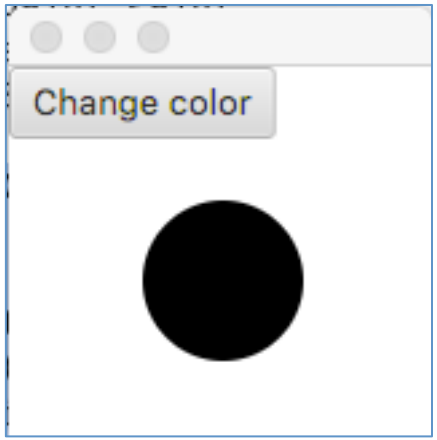
Come organizzare gli ascoltatori/osservatori

1- Self Listener

```
public class ColoredCircle extends Application
        implements EventHandler<ActionEvent> {
  Circle c=null;
  @Override
  public void start(
          Stage primaryStage) {
    c=new Circle(80,80,30);
    c.setFill(
      Paint.valueOf("BLACK"));
    Button btn = new Button();
    btn.setText("Change color");
    btn.setOnAction(this);
    Group root = new Group(btn);
    root.getChildren().add(c);
    Scene scene = new Scene(root, 300, 250);
    primaryStage.setScene(scene);
    primaryStage.show();
  }
  @Override
  public void handle(ActionEvent event) {
     checkAndChangeColor();
  }
```

```
//continua la classe ColoredCircle
void checkAndChangeColor(){
  if (c.getFill().equals
                (Paint.valueOf("BLACK")))
      c.setFill(Paint.valueOf("RED"));
  else c.setFill(Paint.valueOf("BLACK"));
}
public static void main(String[] args) {
  launch(args);
}

}
```



2ListenerEsterno

```
public class ColoredCircle
  extends Application {
  Circle c=null;
  @Override
  public void start(
         Stage primaryStage) {
    c=new Circle(80,80,30);
    c.setFill(
      Paint.valueOf("BLACK"));
    Button btn = new Button();
    btn.setText("Change color");
    btn.setOnAction(new Listener(this));
    Group root = new Group(btn);
    root.getChildren().add(c);
    Scene scene =
       new Scene(root, 300, 250);
    primaryStage.setScene(scene);
    primaryStage.show();
  }
```

```
//continua la classe ColoredCircle
void checkAndChangeColor(){
  if (c.getFill().equals
                (Paint.valueOf("BLACK")))
      c.setFill(Paint.valueOf("RED"));
  else c.setFill(Paint.valueOf("BLACK"));
}
public static void main(String[] args) {
  launch(args);
}
}
```

```
class Listener implements
    EventHandler<ActionEvent>{
 ColoredCircle c=null;
 Listener(ColoredCircle a) {c=a;}
 @Override
  public void handle(ActionEvent event) {
     c.checkAndChangeColor();
}}
```

3ListenerInterno

```
public class ColoredCircle
  extends Application {
  Circle c=null;
  @Override
  public void start(
          Stage primaryStage) {
    c=new Circle(80,80,30);
    c.setFill(
      Paint.valueOf("BLACK"));
    Button btn = new Button();
    btn.setText("Change color");
    btn.setOnAction(new Listener());
    Group root = new Group(btn);
    root.getChildren().add(c);
    Scene scene =
       new Scene(root, 300, 250);
    primaryStage.setScene(scene);
    primaryStage.show();
  }
```

```
//continua la classe ColoredCircle
void checkAndChangeColor(){
  if (c.getFill().equals
                (Paint.valueOf("BLACK")))
      c.setFill(Paint.valueOf("RED"));
  else c.setFill(Paint.valueOf("BLACK"));
}
public static void main(String[] args) {
  launch(args);
}
class Listener implements
    EventHandler<ActionEvent>{
@Override
  public void handle(ActionEvent event) {
     checkAndChangeColor();
}}
}
```

4 Listener Interno Anonimo

```
public class ColoredCircle
  extends Application {
  Circle c=null;
  @Override
  public void start(
          Stage primaryStage) {
    c=new Circle(80,80,30);
    c.setFill(
      Paint.valueOf("BLACK"));
    Button btn = new Button();
    btn.setText("Change color");
    btn.setOnAction(new EventHandler<ActionEvent>(){
        @Override
        public void handle(ActionEvent event) {
            checkAndChangeColor();
    }};
    Group root = new Group(btn);  root.getChildren().add(c);
    Scene scene = new Scene(root, 300, 250);
    primaryStage.setScene(scene);  primaryStage.show();
  }
```

```
//continua la classe ColoredCircle
void checkAndChangeColor(){
  if (c.getFill().equals
                (Paint.valueOf("BLACK")))
      c.setFill(Paint.valueOf("RED"));
  else c.setFill(Paint.valueOf("BLACK"));
}
public static void main(String[] args) {
  launch(args);
}
}
```

Black Magic: Lambda expression

```
btn.setOnAction(new EventHandler<ActionEvent>(){
    @Override
    public void handle(ActionEvent event) {
        checkAndChangeColor();
}};

btn.setOnAction(event -> { checkAndChangeColor(); });
```

1) **`setOnAction`** si aspetta un **`EventHandler<ActionEvent>`**
2) **`EventHandler<ActionEvent>`** ha un solo metodo con un parametro:
    1) **`handle(ActionEvent)`**

quindi **`event`** è di tipo **`ActionEvent`**

Se avessi un (solo) metodo void scriverei ()->
Se avessi un (solo) metodo con due parametri scriverei (x, y) ->
I tipi dei parametri sarebbero dedotti dai tipi del metodo in questione.

5 Lambda expression

```
public class ColoredCircle
  extends Application {
  Circle c=null;
  @Override
  public void start(
          Stage primaryStage) {
    c=new Circle(80,80,30);
    c.setFill(
      Paint.valueOf("BLACK"));
    Button btn = new Button();
    btn.setText("Change color");
    btn.setOnAction(event -> { checkAndChangeColor(); });
    Group root = new Group(btn);  root.getChildren().add(c);
    Scene scene = new Scene(root, 300, 250);
    primaryStage.setScene(scene);
    primaryStage.show();
  }
```

```
//continua la classe ColoredCircle
void checkAndChangeColor(){
  if (c.getFill().equals
                (Paint.valueOf("BLACK")))
      c.setFill(Paint.valueOf("RED"));
  else c.setFill(Paint.valueOf("BLACK"));
}
public static void main(String[] args) {
  launch(args);
}
}
```

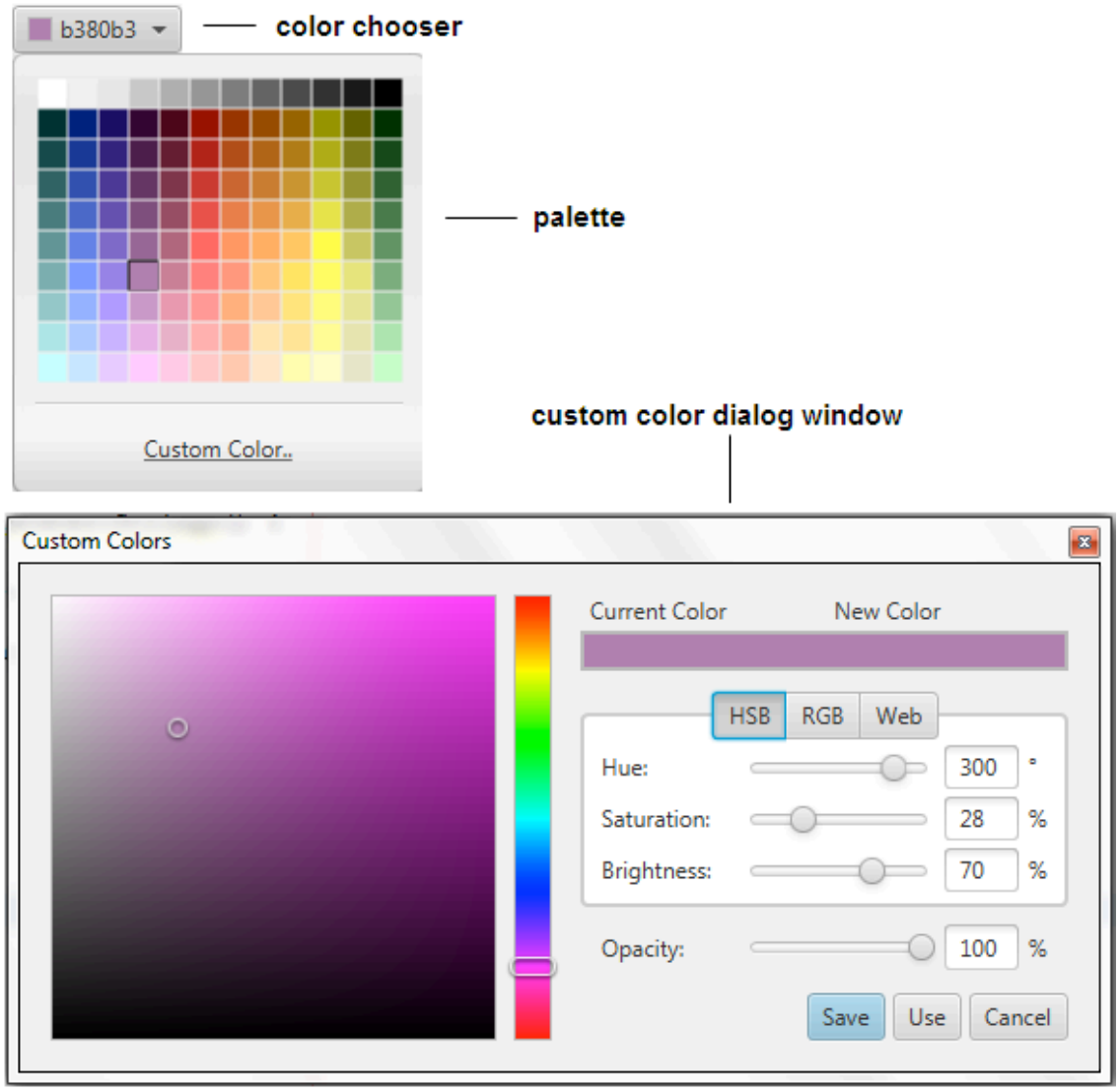
Tre piccole cose:

-@override

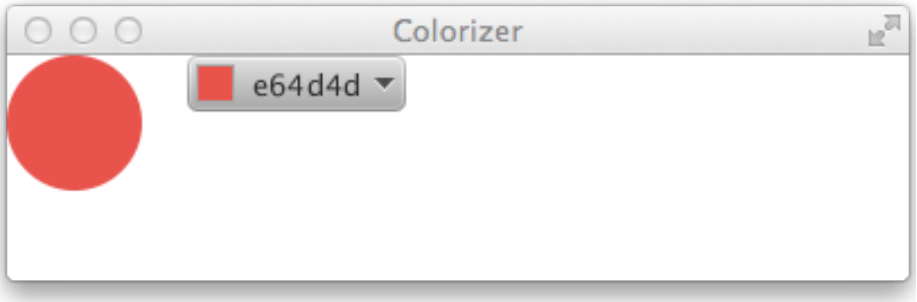
- ColorPicker

- Convenience Methods

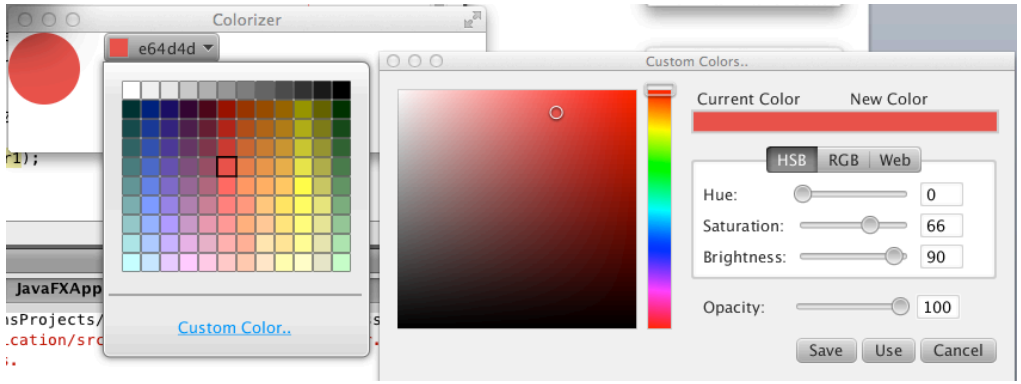
ColorPicker



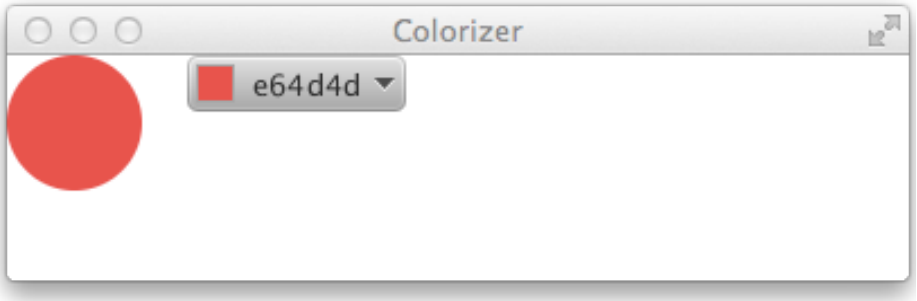
ColorPicker



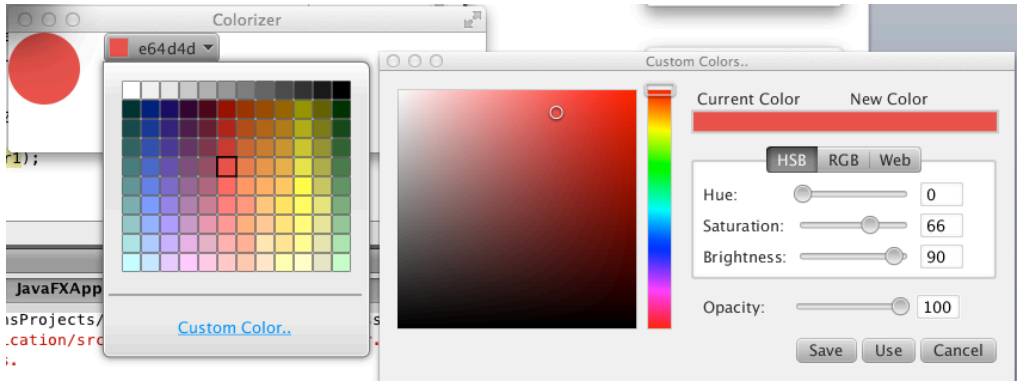

```
public class Colorizer extends Application {
  public void start(final Stage stage) {
    final Circle circ = new Circle(40, 40, 30);
    final ColorPicker colorPicker1 = new ColorPicker(Color.BLACK);

    colorPicker1.addEventHandler(ActionEvent.ACTION, new EventHandler() {
            @Override
            public void handle(Event t) {
                System.out.println(t.getEventType());
                circ.setFill(colorPicker1.getValue());
    }});
    Scene scene = new Scene(new HBox(20), 400, 100);
    HBox box = (HBox) scene.getRoot();
    box.getChildren().addAll(circ,colorPicker1);
    stage.setScene(scene);
    stage.show();
  }
  …
}
```



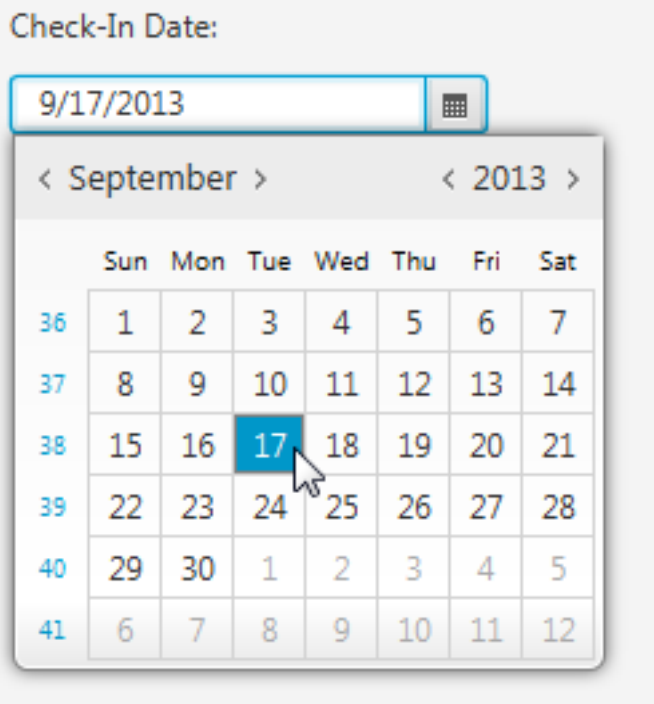
Example




```
public class Colorizer extends Application {
  public void start(final Stage stage) {
    final Circle circ = new Circle(40, 40, 30);
    final ColorPicker colorPicker1 = new ColorPicker(Color.BLACK);
    colorPicker1.setOnAction(t -> {
                System.out.println(t.getEventType());
                circ.setFill(colorPicker1.getValue());
    });
    Scene scene = new Scene(new HBox(20), 400, 100);
    HBox box = (HBox) scene.getRoot();
    box.getChildren().addAll(circ,colorPicker1);
    stage.setScene(scene);
    stage.show();
  }
  …
}
```



DatePicker



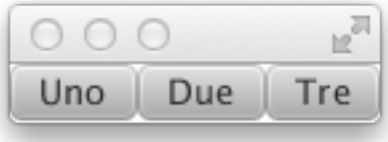
http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/date-picker.htm#CCHHJBEA

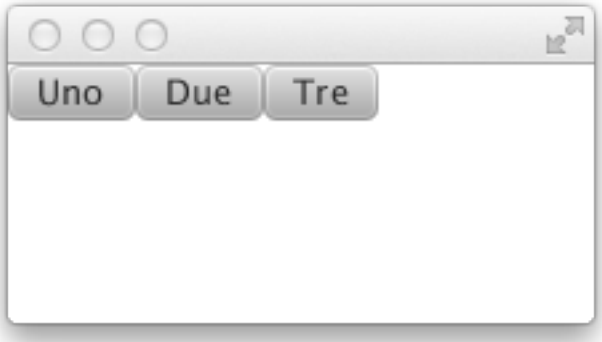
Posizionamento automatico: Layouts di base

http://docs.oracle.com/javafx/2/layout/jfxpub-layout.htm

Layout: HBox

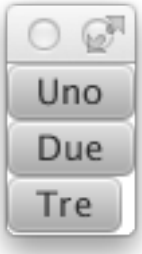
```
public class Layout1 extends Application {
    public void start(Stage stage) {
        Pane layout=new HBox();
        layout.getChildren().add(new Button("Uno"));
        layout.getChildren().add(new Button("Due"));
        layout.getChildren().add(new Button("Tre"));
        Group root = new Group(layout);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



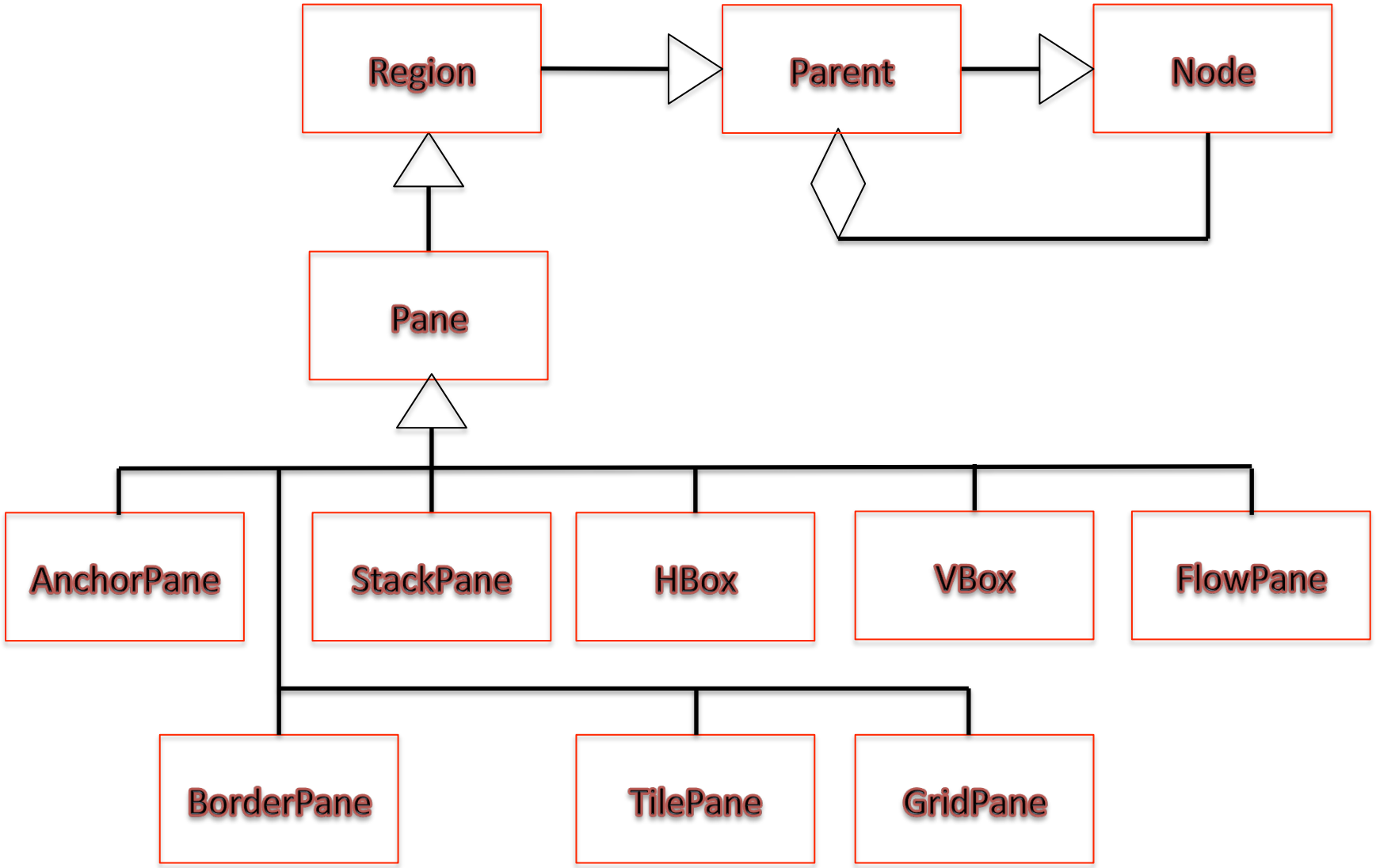


Layout: VBox

```
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

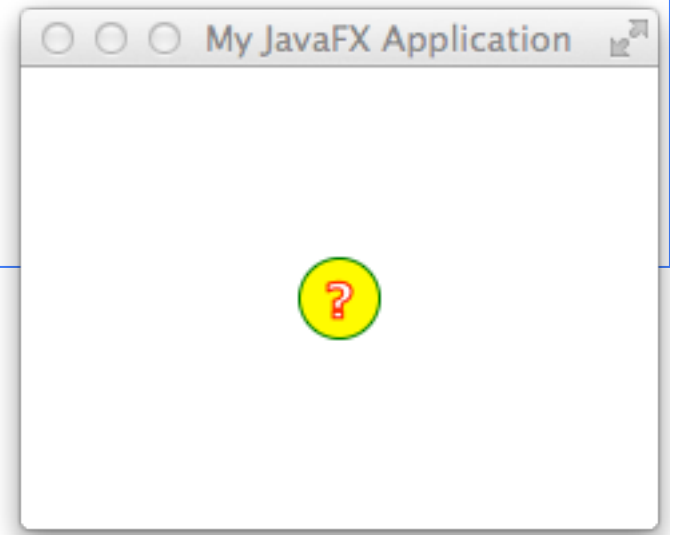
```
public class Layout1 extends Application {
    public void start(Stage stage) {
        StackPane layout=new StackPane();
        layout.getChildren().add(new Button("Uno"));
        layout.getChildren().add(new Button("Due"));
        layout.getChildren().add(new Button("Tre"));
        Group root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

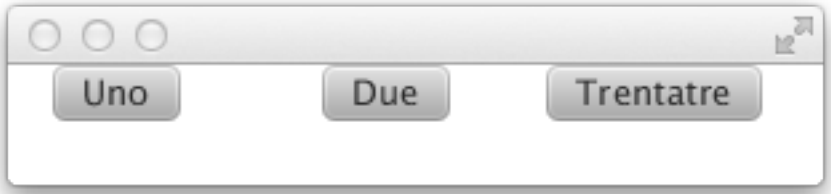


Layout: TilePane


```
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"));
        Group root = new Group(layout);
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.show();
    }}…}
```

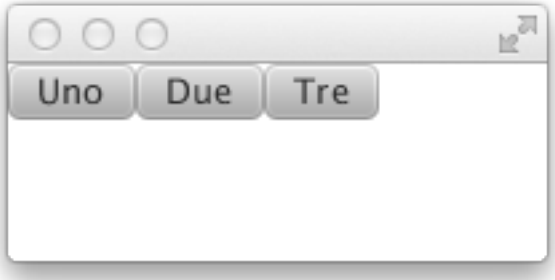
Layout: TilePane


```
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"));
        //Group root = new Group(layout);
        //Scene scene = new Scene(root);
        Scene scene = new Scene(layout);
        stage.setScene(scene);
        stage.show();
    }}…}
```



FlowPane

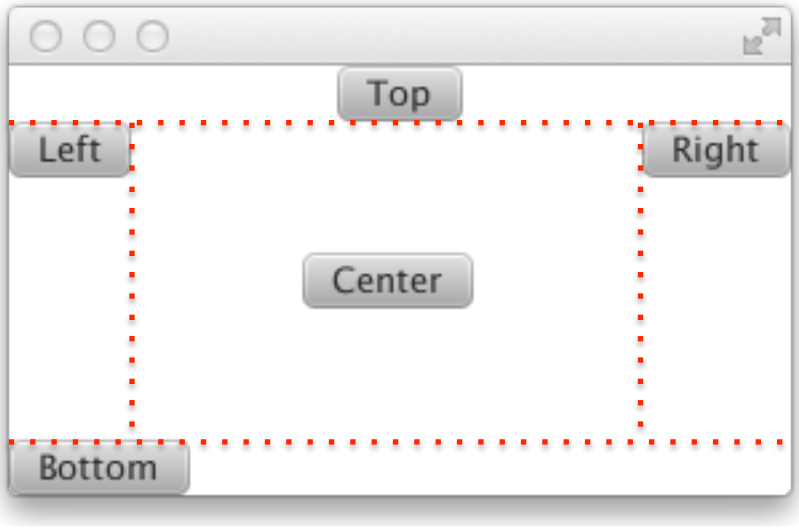
```
public class Layout1 extends Application {
    public void start(Stage stage) {
        final FlowPane layout=new FlowPane();
        layout.setPrefWrapLength(100);
        layout.getChildren().add(new Button("Uno"));
        layout.getChildren().add(new Button("Due"));
        layout.getChildren().add(new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

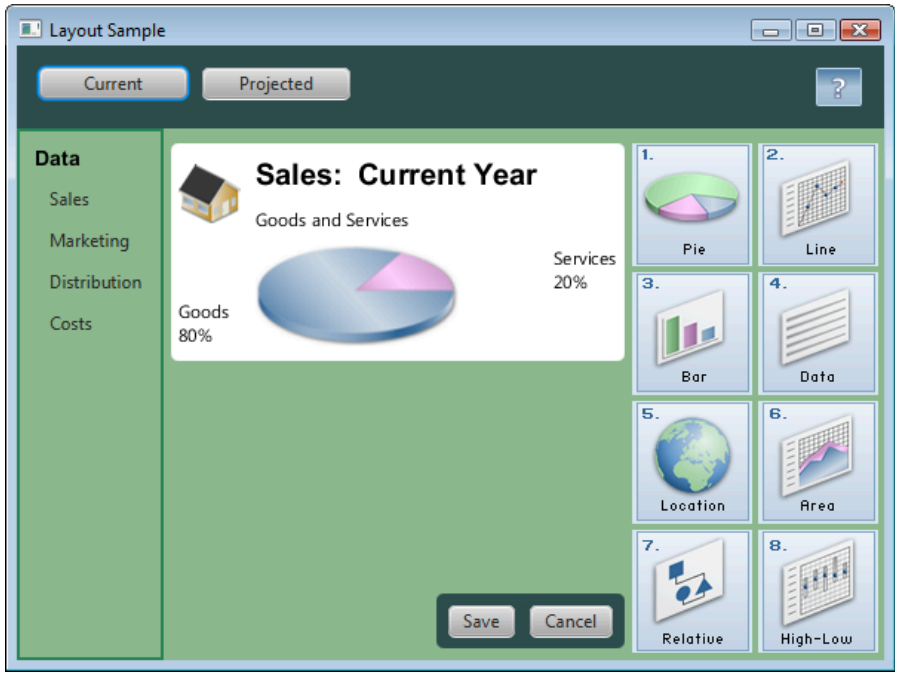


Posizionamento automatico: Layouts avanzati

BorderPane

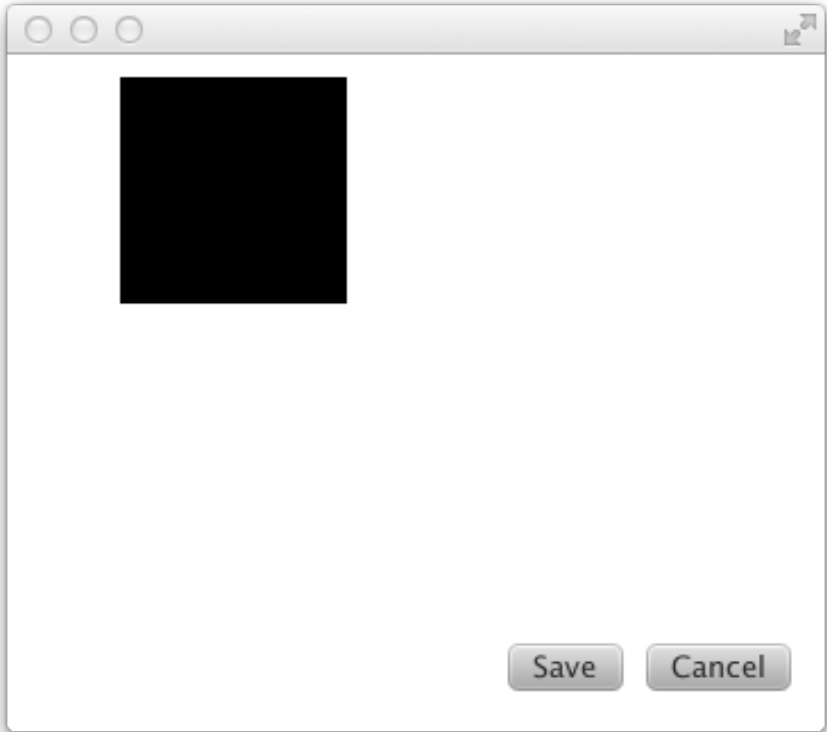
```
public class Layout1 extends Application {
    public void start(Stage stage) {
        BorderPane layout=new BorderPane();
        Button top=new Button("Top");
        BorderPane.setAlignment(top, Pos.TOP_CENTER);
        layout.setTop(top);
        layout.setBottom(new Button("Bottom"));
        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();
    }
…
}
```



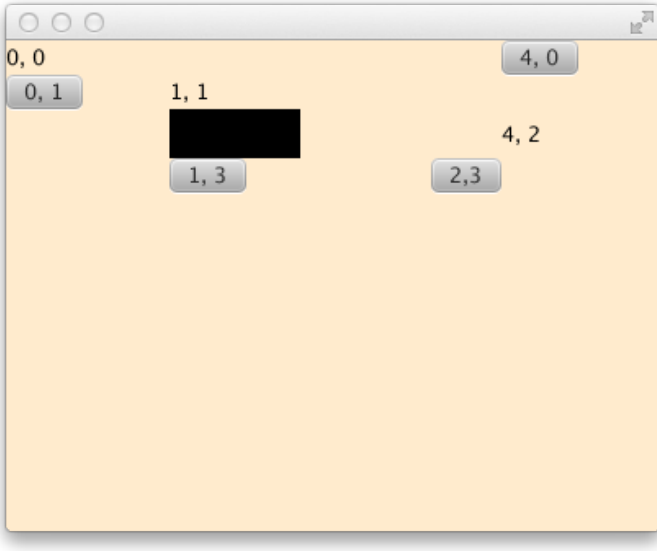


AnchorPane

```
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();
}
```

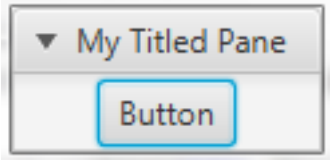


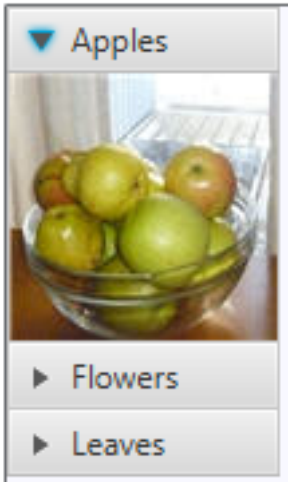
```
public void start(Stage primaryStage) {
        double width = 400;
        double height = 300;
        GridPane gridPane = new GridPane();
        Scene scene = new Scene(gridPane,
           width, height, Color.BLANCHEDALMOND);
        gridPane.add(new Text("0, 0"), 0, 0);
        gridPane.add(new 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





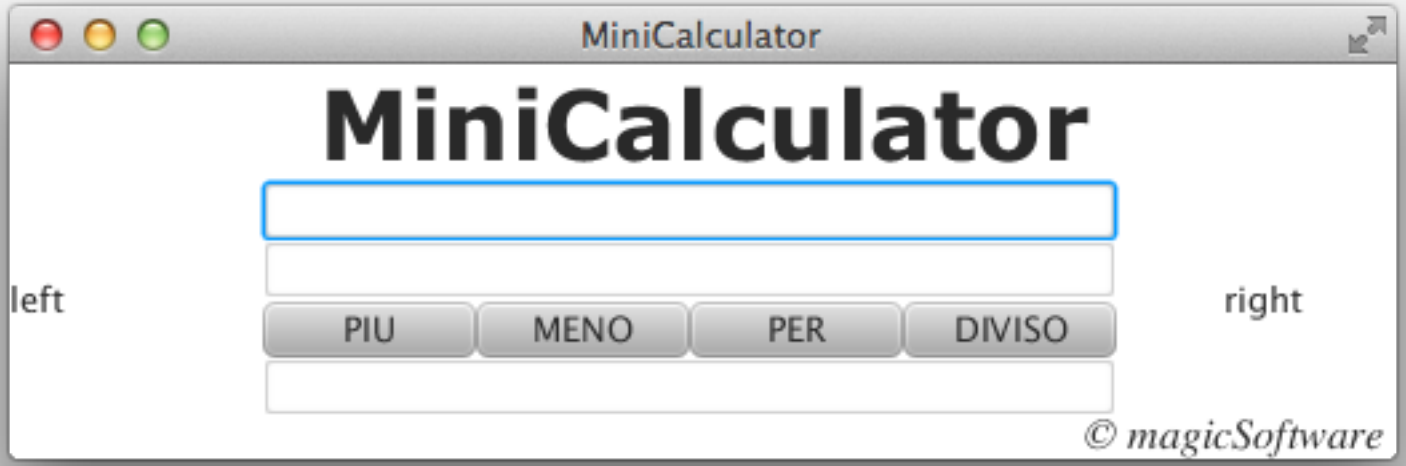
Accordion

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

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();
}
```

Esercizio


MiniCalculator
MiniCalculator
left
right
PIU
MENO
PER
DIVISO
© magicSoftware