



Politecnico  
di Torino



e-Lite



# JavaFX – a Crash Course

Tecniche di Programmazione – A.A. 2022/2023

# JavaFX applications

---

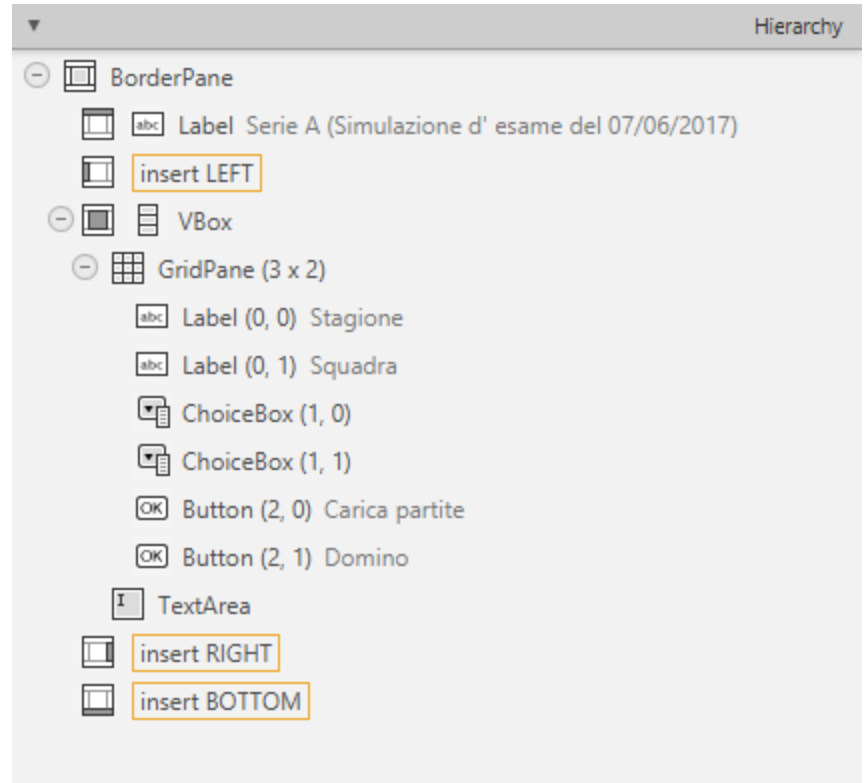


# Application structure

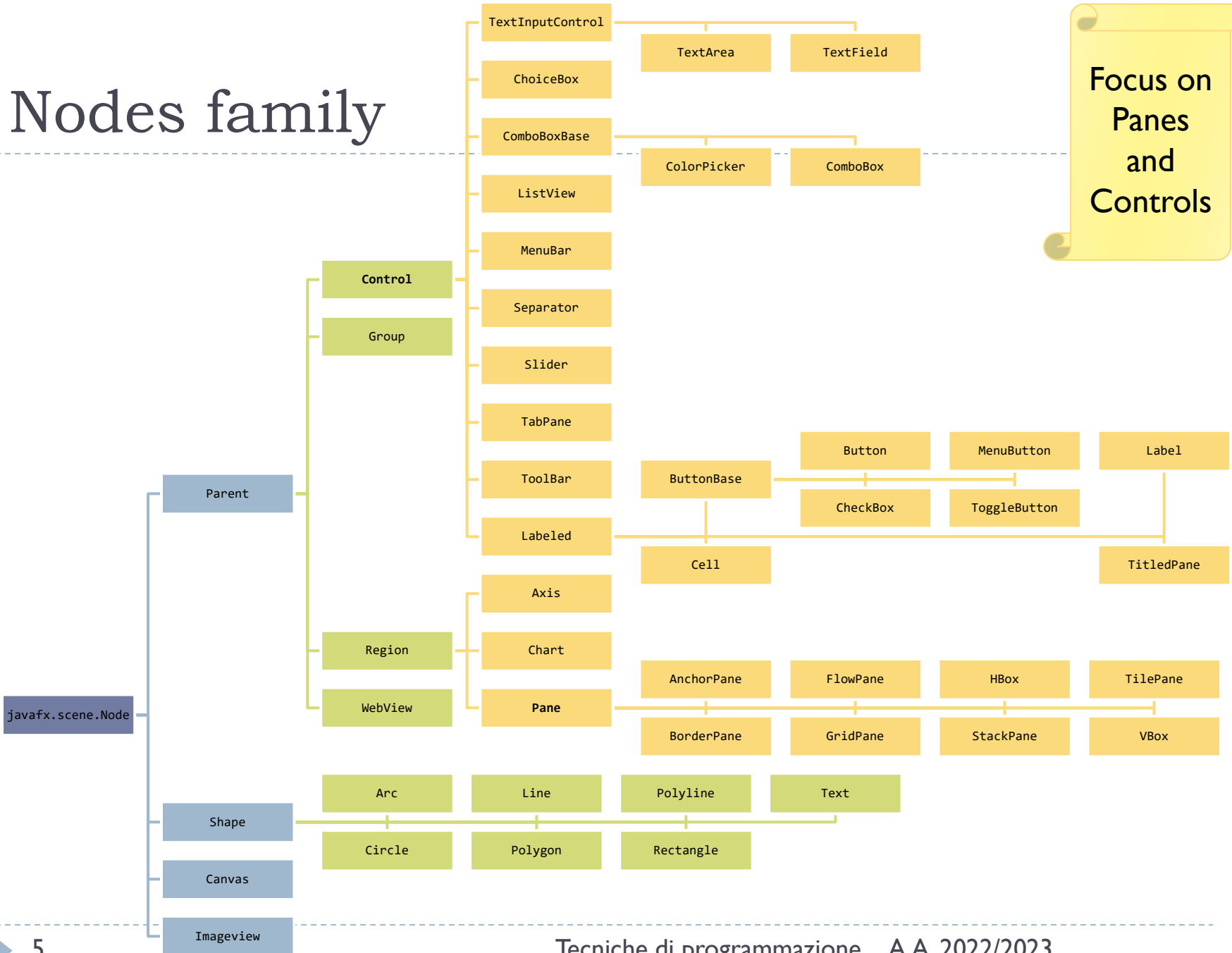


- ▶ **Stage**: where the application will be displayed (e.g., a Windows' window)
- ▶ **Scene**: one container of Nodes that compose one “page” of your application
- ▶ **Node**: an element in the Scene, with a visual appearance and an interactive behavior.
  - ▶ Nodes may be hierarchically nested

# Nested nodes



# Nodes family



# Essential Reference

## ► JavaFX JavaDoc API

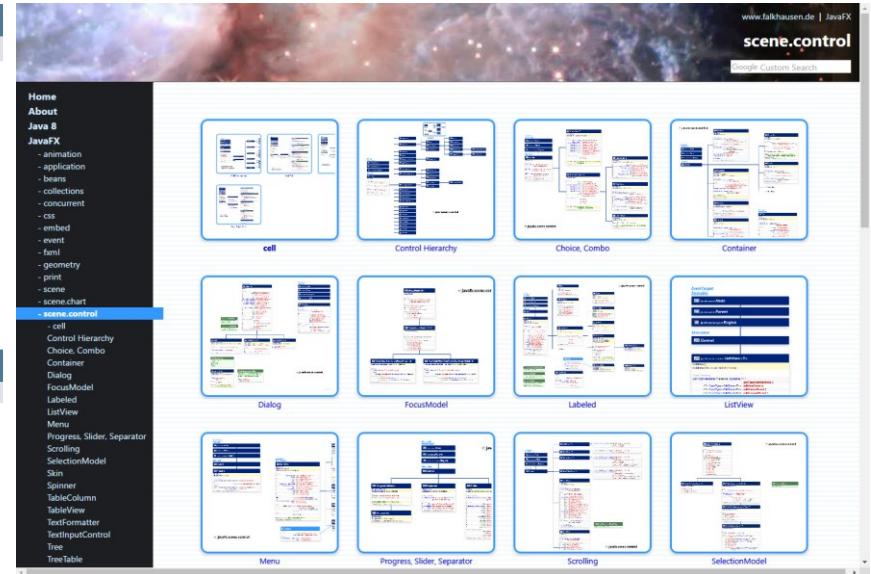
► <https://openjfx.io/javado c/11/>

## ► JavaFX Class Diagrams

► <http://falkhausen.de/Java FX-10/index.html>

OVERVIEW	MODULE	PACKAGE	CLASS	USE	TREE	DEPRECATED	INDEX	HELP
ALL CLASSES								
SEARCH: <input type="text"/>								
JavaFX 11								
Modules								
Module	Description							
javafx.base	Defines the base APIs for the JavaFX UI toolkit, including APIs for bindings, properties, collections, and events.							
javafx.controls	Defines the UI controls, charts, and skins that are available for the JavaFX UI toolkit.							
javafx.fxml	Defines the FXML APIs for the JavaFX UI toolkit.							
javafx.graphics	Defines the core scenegraph APIs for the JavaFX UI toolkit (such as layout containers, application lifecycle, shapes, transformations, canvas, input, painting, image handling, and effects), as well as APIs for animation, css, concurrency, geometry, printing, and windowing.							
javafx.media	Defines APIs for playback of media and audio content, as part of the JavaFX UI toolkit, including MediaPlayer and MediaView.							
javafx.swing	Defines APIs for the JavaFX / Swing interop support included with the JavaFX UI toolkit, including SwingNode (for embedding Swing inside a JavaFX application) and JFXPanel (for embedding JavaFX inside a Swing application).							
javafx.web	Defines APIs for the WebView functionality contained within the JavaFX UI toolkit.							

OVERVIEW	MODULE	PACKAGE	CLASS	USE	TREE	DEPRECATED	INDEX	HELP
ALL CLASSES								
Report a bug or suggest an enhancement								
Copyright © 2006, 2018, Oracle and/or its affiliates. All rights reserved.								



# Example application structure

```
1 package it.polito.tdp.seriesa;
2
3 import it.polito.tdp.seriesa.model.Model;
4 import javafx.application.Application;
5 import javafx.stage.Stage;
6 import javafx.scene.Scene;
7 import javafx.scene.layout.BorderPane;
8 import javafx.fxml.FXMLLoader;
9
10
11 public class Main extends Application {
12     @Override
13     public void start(Stage primaryStage) {
14         try {
15             FXMLLoader loader = new FXMLLoader(getClass().getResource("SerieA.fxml"));
16             BorderPane root = (BorderPane)loader.load();
17             Scene scene = new Scene(root);
18
19             SerieAController controller = loader.getController();
20             Model model = new Model();
21             controller.setModel(model);
22
23             scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
24             primaryStage.setScene(scene);
25             primaryStage.show();
26
27         } catch (Exception e) {
28             e.printStackTrace();
29         }
30     }
31
32     public static void main(String[] args) {
33         Launch(args);
34     }
35 }
36
```

The diagram consists of five callout boxes with blue borders and white backgrounds, each containing text and connected to specific parts of the code by blue lines. The first box, labeled 'Extend javafx.application.Application', points to line 11 where 'Main extends Application'. The second box, labeled 'Load scene nodes from XML file', points to line 15 where 'FXMLLoader loader' is instantiated. The third box, labeled 'Define algorithms and data', points to lines 19-21 where the controller and model are set up. The fourth box, labeled 'Populate and show window', points to lines 23-25 where the scene is styled and the window is shown. The fifth box, labeled 'main()', points to line 32 where the 'main' method is defined.

Extend  
javafx.application.Application

Load scene nodes  
from XML file

Define algorithms  
and data

Populate and show  
window

main()

# General rules

---

- ▶ A JavaFX application extends `javafx.application.Application`
- ▶ The `main()` method should call `Application.launch()`
- ▶ The `start()` method is the main entry point for all JavaFX applications
  - ▶ Called with a Stage connected to the Operating System's window
- ▶ The content of the scene is represented as a hierarchical scene graph of Nodes
  - ▶ Stage is the top-level JavaFX container
  - ▶ Scene is the container for all content

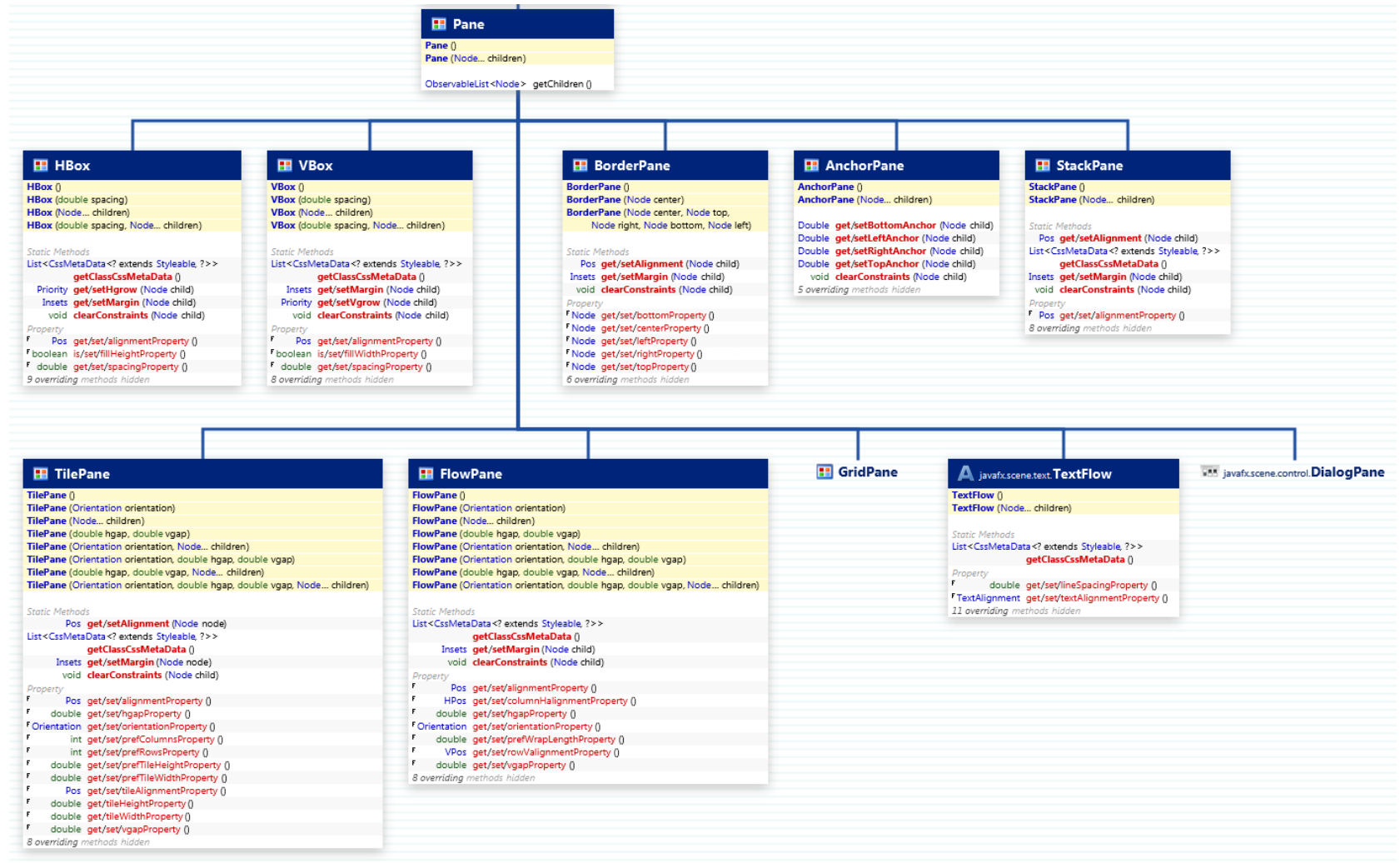


# Nodes

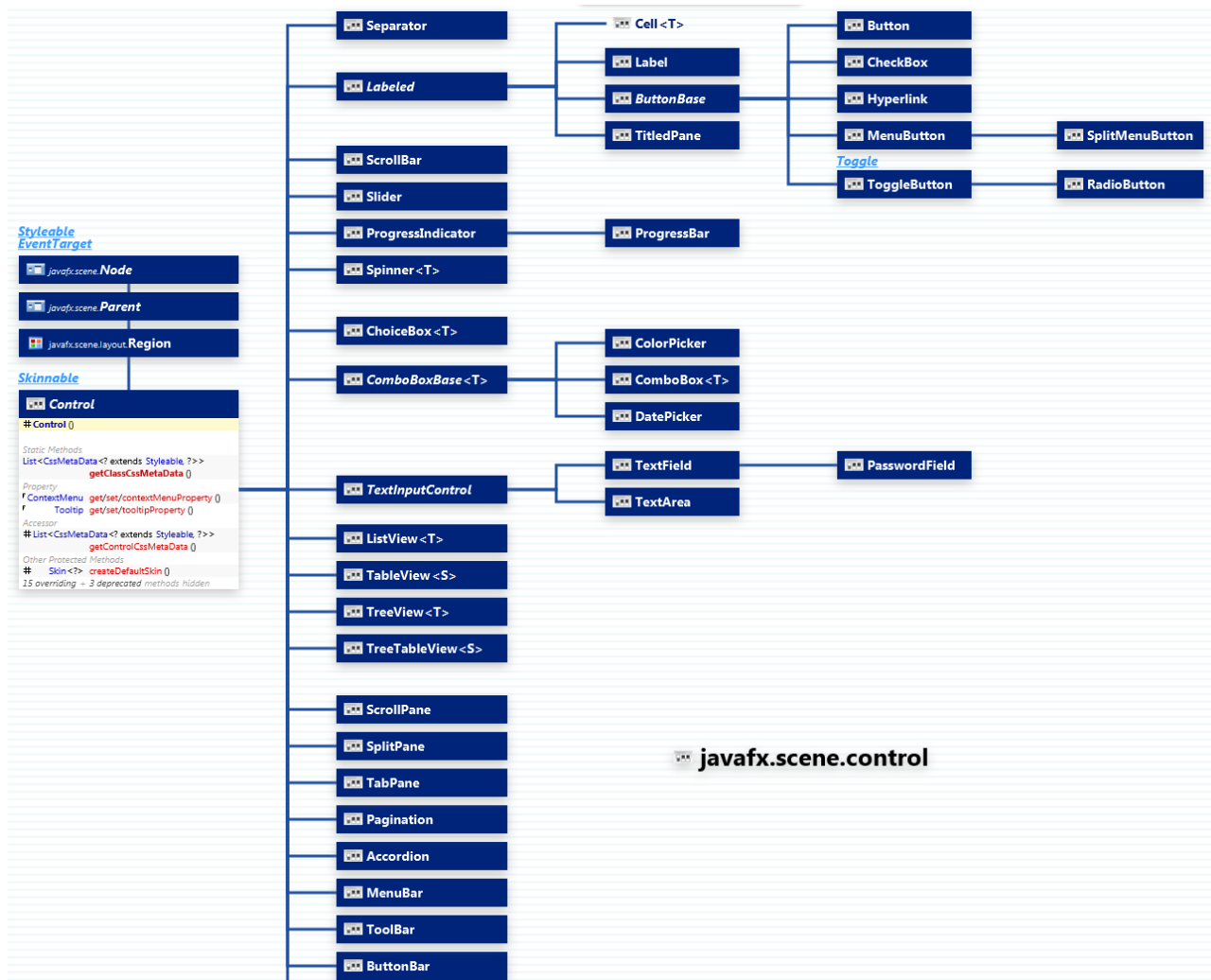
---

- ▶ The Scene is populated with a tree of Nodes
  - ▶ Layout components (Panels)
  - ▶ UI Controls
  - ▶ Charts
  - ▶ Shapes
- ▶ Nodes have Properties
  - ▶ Visual (size, position, z-order, color, ...)
  - ▶ Contents (text, value, data sets, ...)
  - ▶ Programming (event handlers, controller)
- ▶ Nodes generate Events
  - ▶ UI events
- ▶ Nodes can be styled with CSS

# Panes

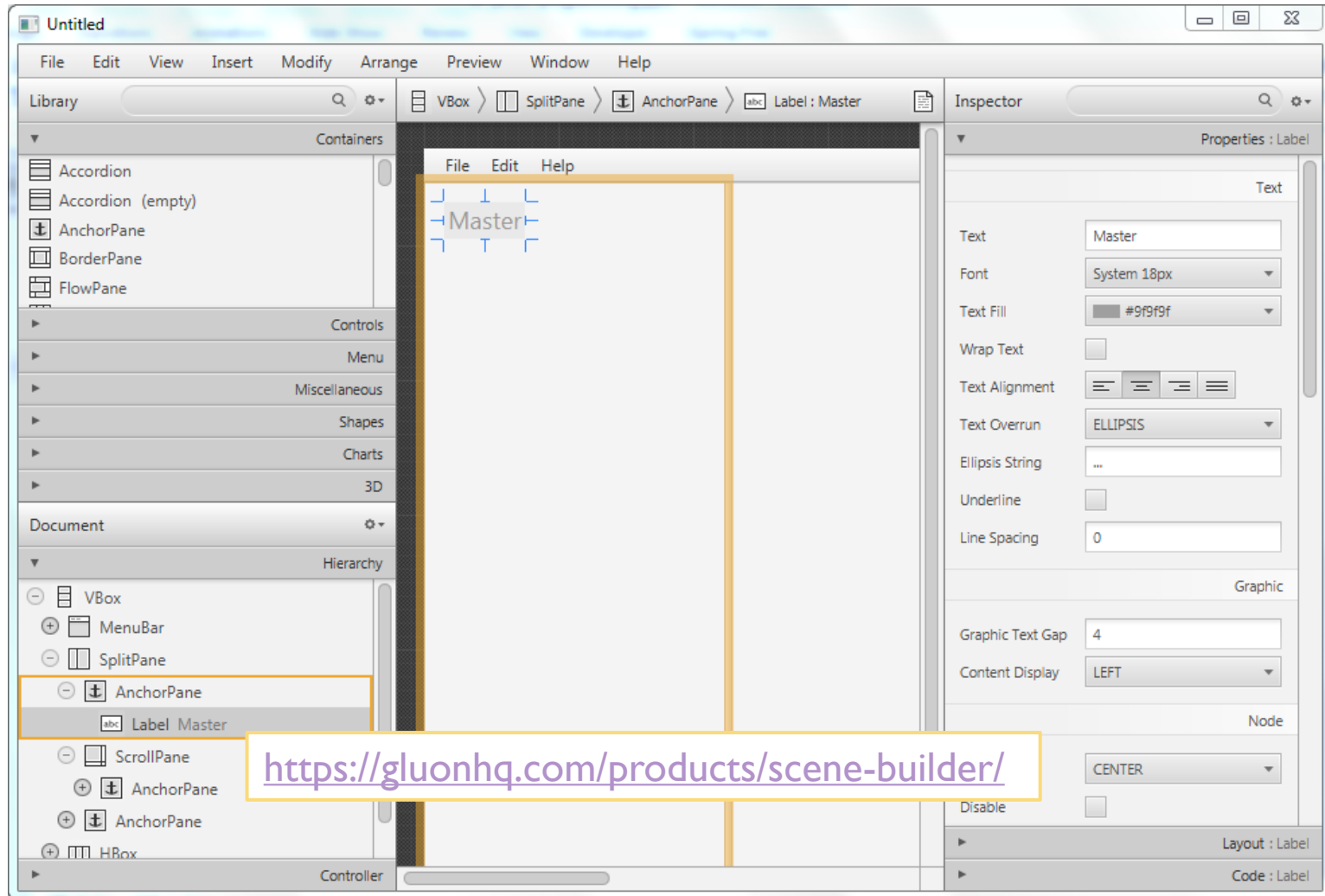


# Controls





# JavaFX Scene Builder 19



# Building a scene from FXML

---

```
public void start(Stage stage) throws Exception {  
    Parent root = FXMLLoader.load(  
        getClass().getResource("/fxml/Scene.fxml"));  
  
    stage.setTitle("Circle Demo");  
    stage.setScene(new Scene(root));  
    stage.show();  
}
```

# Key concepts in JavaFX

---

- ▶ **Property:** attributes of the Nodes, may specify content, size, color, ... Can be read and written by the application
- ▶ **Event:** every user action on one element of the GUI generates a different *event*. Events can be captured and *handled* by our code
- ▶ **Controller:** the Java class that contains
  - ▶ References to interesting Nodes
  - ▶ Event Handlers

# Properties

- ▶ Extension of the Java Beans convention
  - ▶ May be used also outside JavaFX
- ▶ Encapsulate properties of an object
  - ▶ Different types (string, number, object, collection, ...)
  - ▶ Set/Get
  - ▶ Observe changes
  - ▶ Support lazy evaluation
- ▶ Each Node has a large set of Properties
  - ▶ Can be manipulated
  - ▶ The scene updates

Properties	
Type	Property and Description
BooleanProperty	cancelButton A Cancel Button is the button that receives a keyboard VK_ESC press, if no other node in the scene co
BooleanProperty	defaultButton A default Button is the button that receives a keyboard VK_ENTER press, if no other node in the scene
Properties inherited from class javafx.scene.control.ButtonBase	
armed, onAction	
Properties inherited from class javafx.scene.control.Labeled	
alignment, contentDisplay, ellipsisString, font, graphic, graphicTextGap, labelPadding, mnemonicParsing, text, textFill, textOverrun, text, underline, wrapText	
Properties inherited from class javafx.scene.control.Control	
contextMenu, height, maxHeight, maxWidth, minHeight, minWidth, prefHeight, prefWidth, skinClassName, skin, to	
Properties inherited from class javafx.scene.Parent	
needsLayout	
Properties inherited from class javafx.scene.Node	
blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effec eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, localToParentTransform, localToSceneTransform, managed, mouseTransparent, onContextMenuRequested, onDragDete onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, o onKeyTyped, onMouseClicked, onMouseDragEntered, onMouseDragExited, onMouseDragged, onMouseDragOver, onMouseDr onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, onRotate, onRotationFinished, onRotationStarted, onScrollFinished, onScroll, onScrollStarted, onSwipeDown, onSwipeLeft, onSwipeRight, onSwi onTouchMoved, onTouchPressed, onTouchReleased, onTouchStationary, onZoomFinished, onZoom, onZoomStarted, opac pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, t visible	

# Events

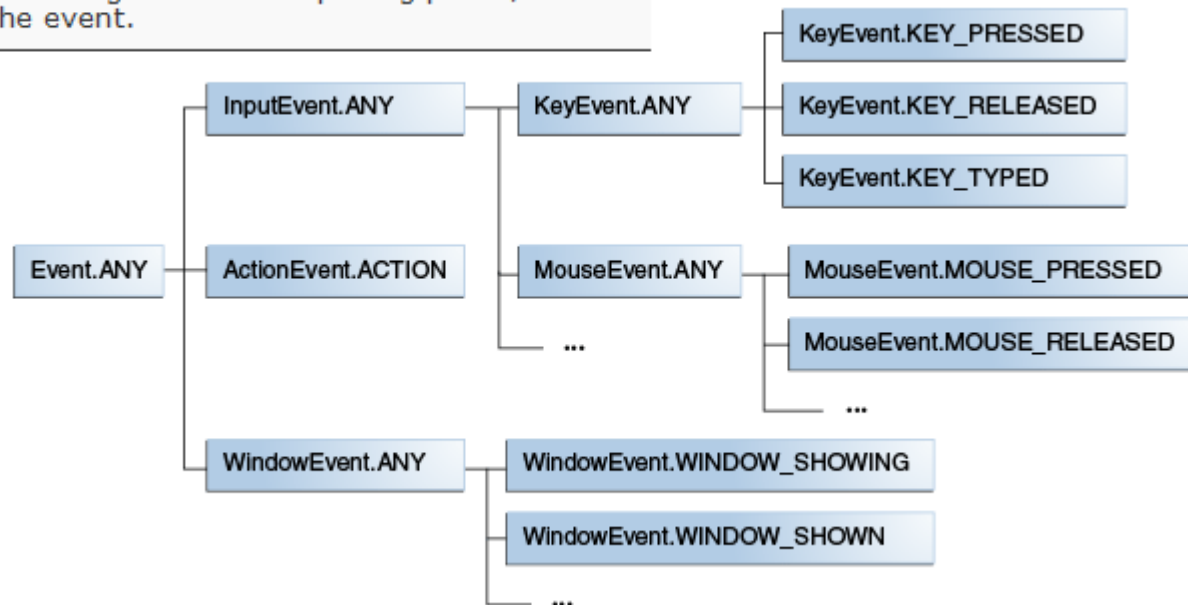
---

- ▶ FX Event (`javafx.event.Event`):
  - ▶ Event Source => a Node
  - ▶ Event Target
  - ▶ Event Type
- ▶ Usually generated after some user action
- ▶ Event Types
- ▶ You can define **event handlers** in your application

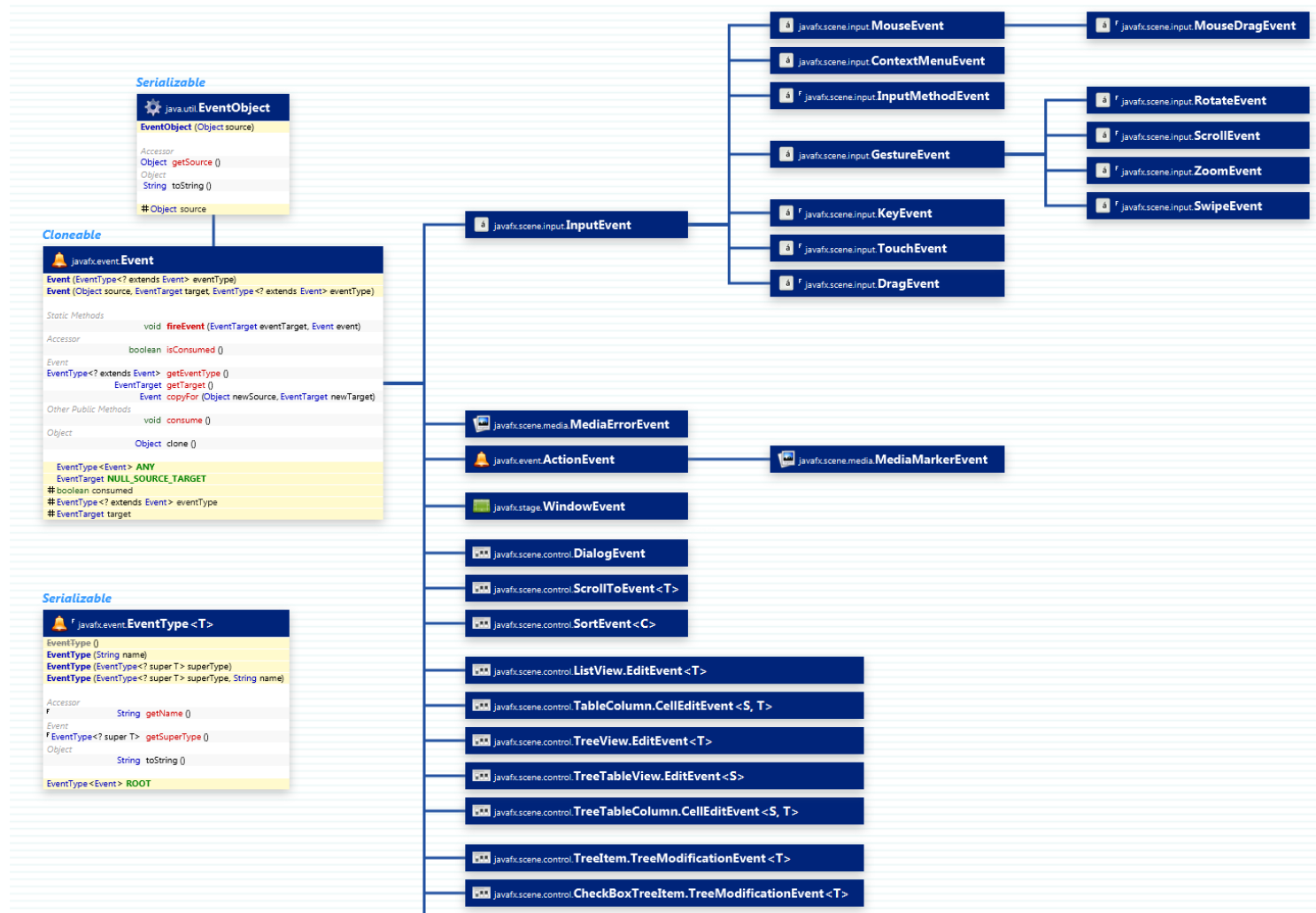


# What is an event?

Property	Description
Event type	Type of event that occurred.
Source	Origin of the event, with respect to the location of the event in the event dispatch chain. The source changes as the event is passed along the chain.
Target	Node on which the action occurred and the end node in the event dispatch chain. The target does not change, however if an event filter consumes the event during the event capturing phase, the target will not receive the event.

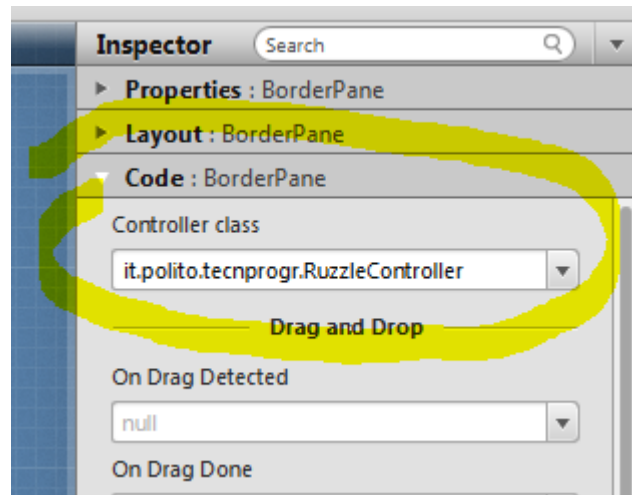


# Event types



# Defining a Controller class

- ▶ The Root element of the scene graph may specify a **fx:controller** attribute
  - ▶ `<BorderPane`  
`id="BorderPane"`  
`xmlns:fx="http://javafx.com/fxml1"`  
`fx:controller="it.polito.tdp.RuzzleController">`



# Injection of Node references

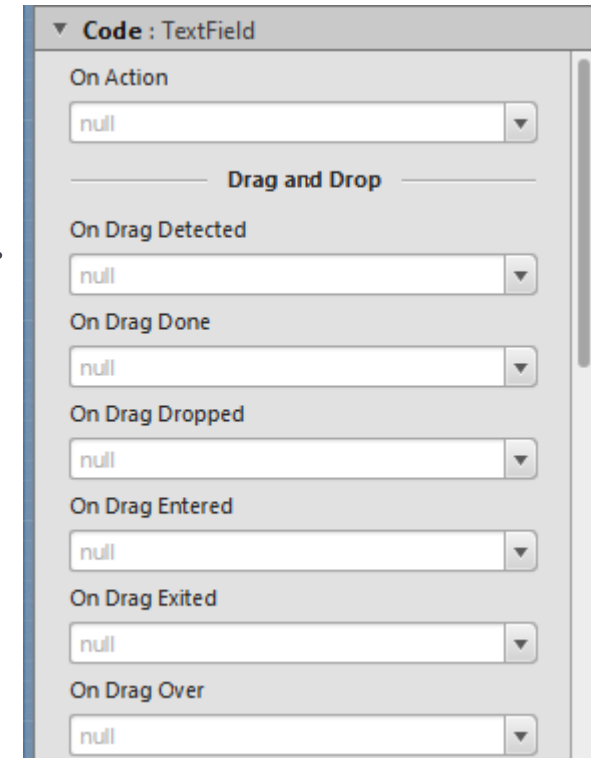
---

- ▶ The controller code may directly access various Nodes in the associated scene graph
- ▶ The attribute `@FXML` associates a Node variable with the corresponding node, with the same `fx:id` value as the variable name
- ▶ Try: View | Show Sample Controller Skeleton on the Scene Builder!

```
@FXML // fx:id="theTitle"  
private Label theTitle;
```

# Registration of Event Handlers

- ▶ In FXML, you may set a event handler through attributes
  - ▶ `onAction`, `onKeyTyped`, `onMouseClicked`, ... hundreds more ...
- ▶ The value should be the `#name` of a method in the controller class
  - ▶ With the right signature for the event type

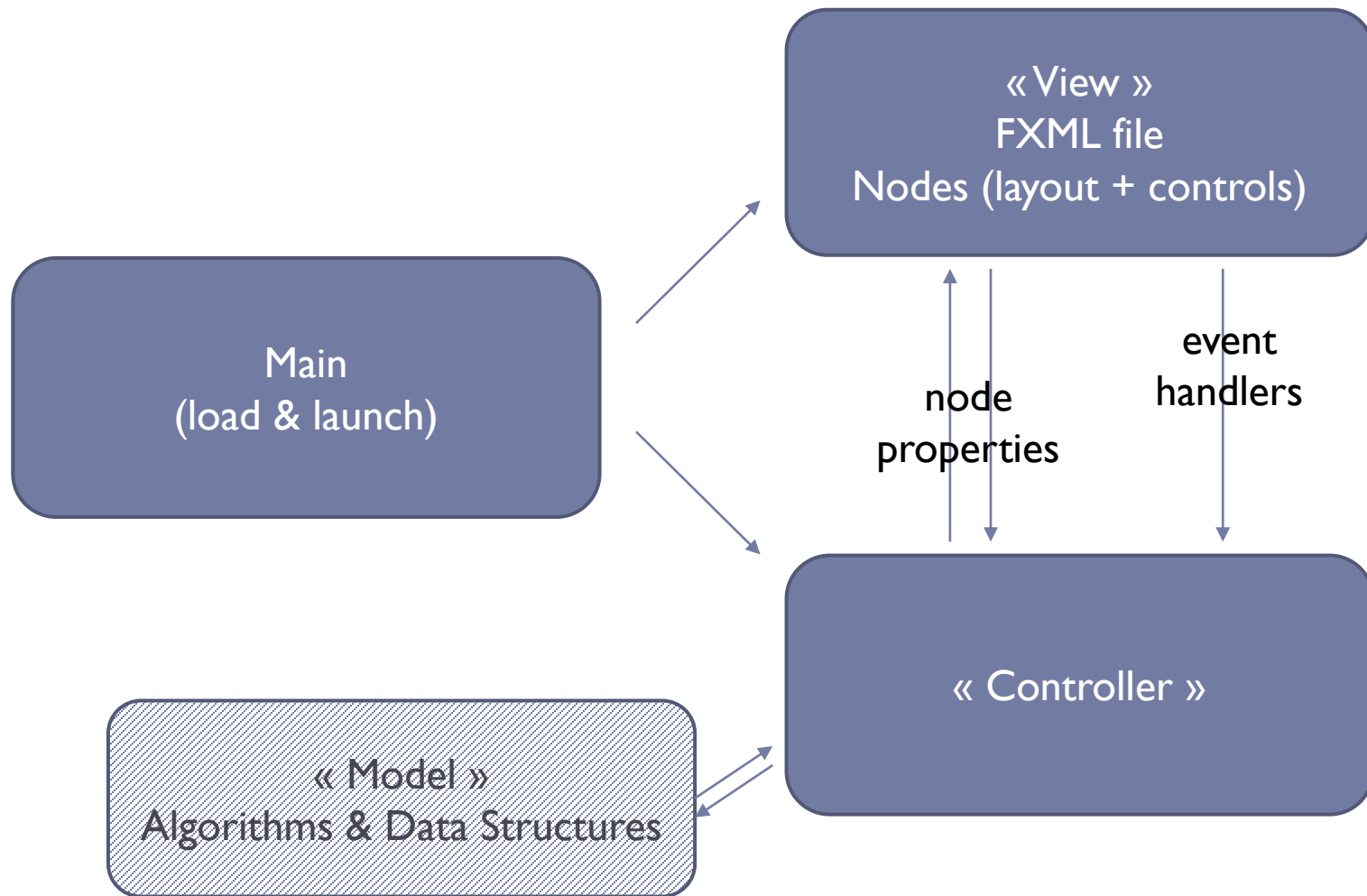


```
<Button fx:id="cercaBtn"
onAction="#doCercaParola"
text="Cerca" />
```

```
@FXML
void doCercaParola (
   (ActionEvent event) {
```






# Minimal program structure

---



# Licenza d'uso



- ▶ Queste diapositive sono distribuite con licenza Creative Commons “Attribuzione - Non commerciale - Condividi allo stesso modo (CC BY-NC-SA)”
- ▶ Sei libero:
  - ▶ di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera 
  - ▶ di modificare quest'opera 
- ▶ Alle seguenti condizioni:
  - ▶ **Attribuzione** — Devi attribuire la paternità dell'opera agli autori originali e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera. 
  - ▶ **Non commerciale** — Non puoi usare quest'opera per fini commerciali. 
  - ▶ **Condividi allo stesso modo** — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa. 
- ▶ <http://creativecommons.org/licenses/by-nc-sa/3.0/>