

Fondamenti di Java

Static

Modificatori: static

Variabili e metodi associati ad una Classe anziché ad un Oggetto sono definiti “static”.

Le variabili statiche servono come singola variabile **condivisa tra le varie istanze**

I metodi possono essere richiamati **senza creare una istanza.**

Variabili "static": esempio 1

```
public class S {
    static int instanceCount = 0; //variabile "di classe"
    S() {instanceCount++;}
}

public class A {
    public static void main(String a[]) {
        new A();
    }
    A() {
        for (int i = 0; i < 10; ++i) {
            S instance=new S();
        }
        System.out.println("# of instances:
            "+S.instanceCount);
    }
}
```

Output:

of instances: 10

Variabili "static": esempio 2

```
class S {
    static int instanceCount = 0; //variabile "di classe"
    S() {instanceCount++;}
    public void finalize() {instanceCount--;}
}

public class A {
    public static void main(String a[]) {
        new A();
    }
    A() {
        for (int i = 0; i < 10; ++i) {
            S instance=new S();
        }
        System.out.println("# of instances:"+S.instanceCount);
        System.gc();
        System.out.println("# of instances: "+S.instanceCount);
    }
}
```

Output:

```
# of instances: 10
# of instances: 0
```

Metodi "static": esempio 1

```
class S {
    static int instanceCount = 0; //variabile "di classe"
    S() {instanceCount++;}
    static void azzerContatore() {instanceCount=0;}
}

public class A {
    public static void main(String a[]) {
        new A();

    }
    A() {
        for (int i = 0; i < 10; ++i) {
            if (i%4==0) S.azzerContatore();
            S instance=new S();
        }
        System.out.println("instanceCount:
"+S.instanceCount);
    }
}
```

**Può agire solo su
variabili statiche!**

Output:
instanceCount: 2

**Ruolo:
Metodi che agiscono su
variabili statiche**

metodi "static": esempio 2

**Notare la
maiuscola!
(per convenzione)**

```
Math.sqrt(double x);  
System.gc();  
System.arraycopy(...);  
System.exit();  
Integer.parseInt(String s);  
Float.parseFloat(String s);
```

**Ruolo:
analogo alle
librerie del C**

Che cos'è :
`System.out.println()` ?

Attenzione!

Usare variabili static per condividere informazioni è una CATTIVA PRASSI.

- Le variabili globali violano il principio di encapsulazione dello stato nell'oggetto.
- Quando usate, dovrebbero in genere essere static final
- Il ciclo di vita delle variabili globali è "eterno": vengono istanziate a inizio esecuzione e restano vive fino alla fine.
- Presentano problemi in caso di multithreading

Perchè il main è "static"?

```
public class A {  
    String s="hello";  
    public static void main(String a[]) {  
        System.out.println(s);  
    }  
}
```

Non static variable s cannot be referenced from static context

```
public class A {  
    String s="hello";  
    public static void main(String a[]) {  
        new A();  
    }  
    A() {  
        System.out.println(s);  
    }  
}
```

hello

I parametri del
main sono inclusi
in un vettore di
String

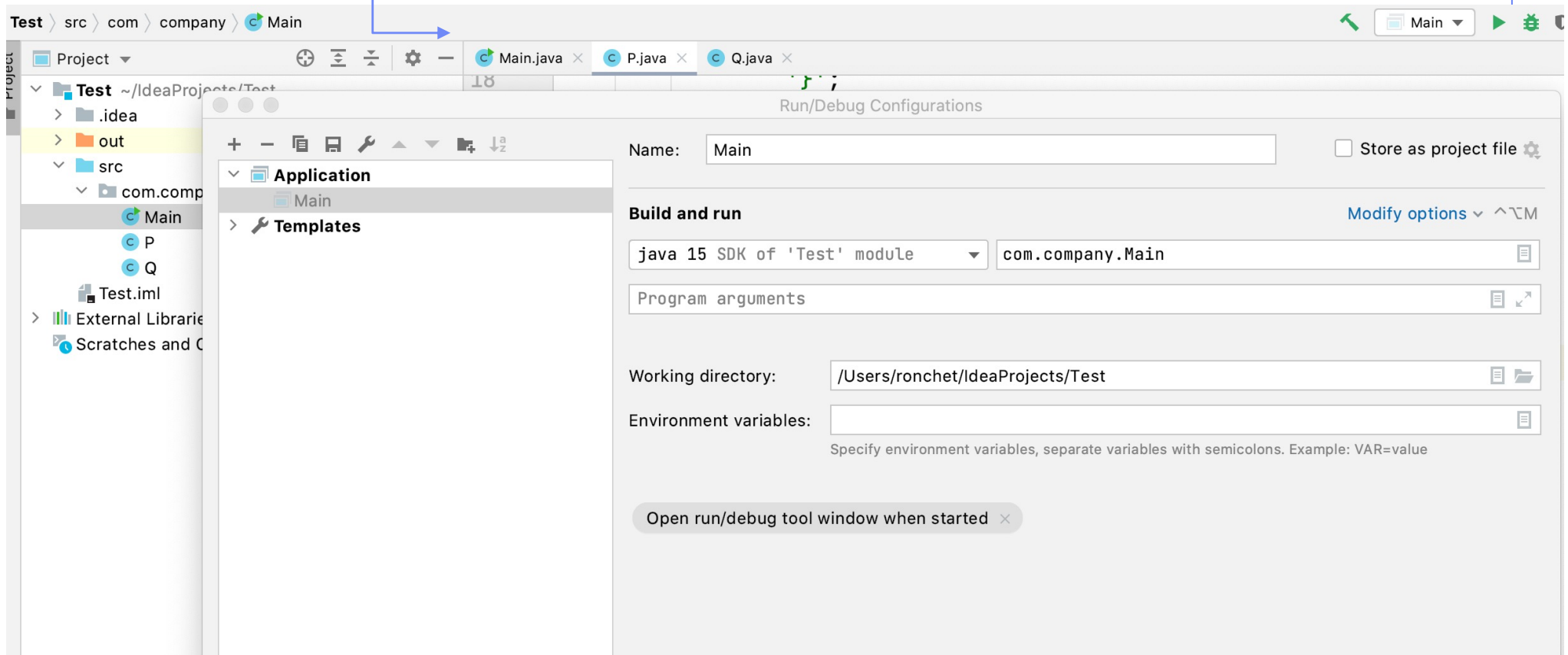
Parametri di ingresso

```
/* sum and average command lines */  
class SumAverage {  
    public static void main (String args[]) {  
        int sum = 0;  
        float avg = 0;  
        for (int i = 0; i < args.length; i++) {  
            sum += Integer.parseInt(args[i]);  
        }  
        System.out.println("Sum is: " + sum);  
        System.out.println("Average is: "  
            + (float)sum / args.length);  
    }  
}
```

Parametri di ingresso – altro esempio

```
/* sum and average command lines */  
class ElementShower{  
    public static void main(String a[]){  
        System.out.println(a.length);  
        int arg=0;  
        for (String s:a) {  
            System.out.println("argomento " +  
                               (++arg)+" : "+s);  
        }  
    }  
}
```

Parametri di ingresso in IntelliJ



Posizionamento di componenti in Java FX

Posizionamento di un Node

Non vengono forniti metodi per il posizionamento assoluto ...

- es., `setX`, `setY`

- sono tuttavia presenti in alcune sottoclassi

... in compenso sono definiti i metodi
`setLayoutX`, `setTranslateX`

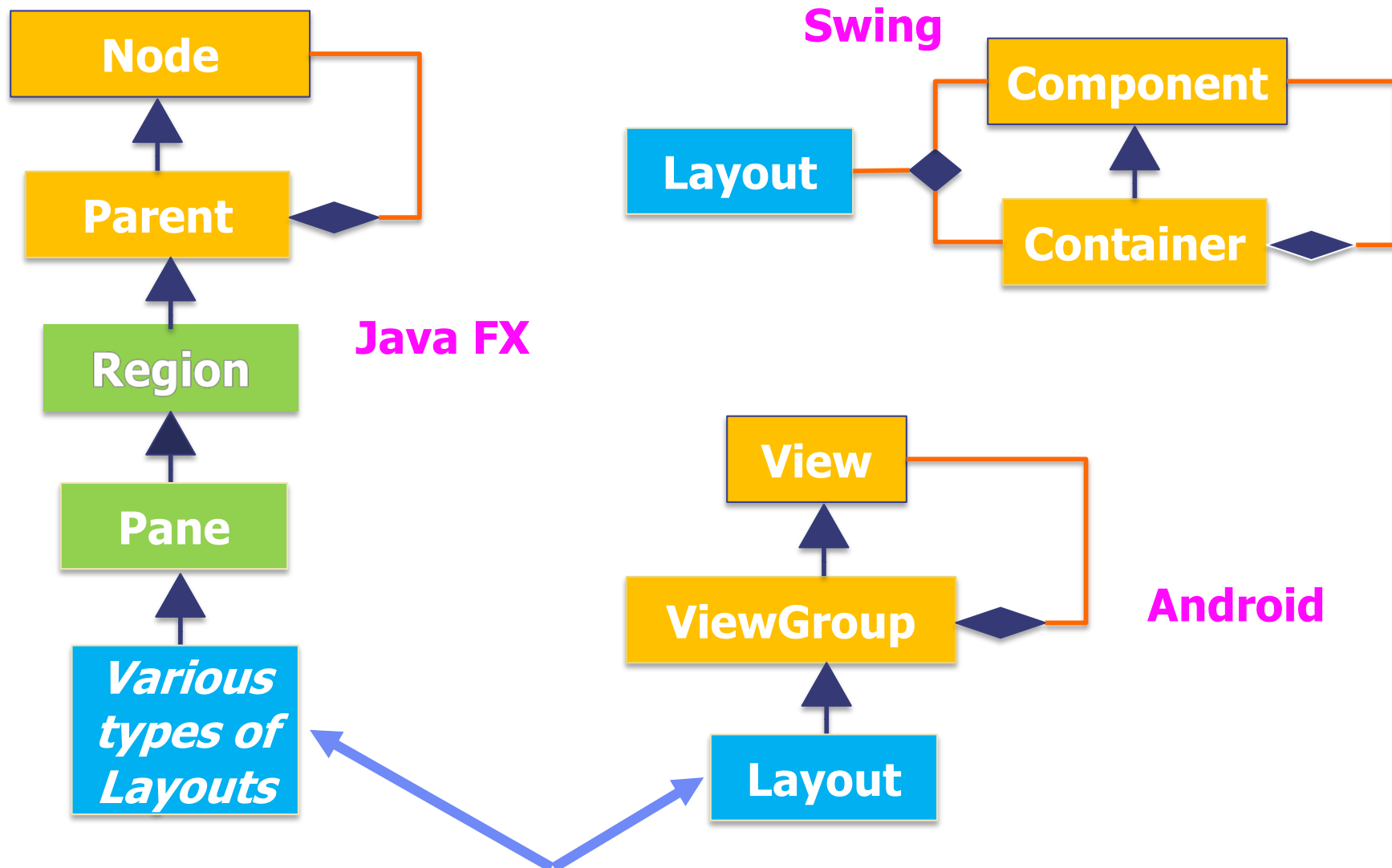
A cosa servono?

Posizionamento di un **Node**

La responsabilità di posizionare il contenuto è delegata ad una componente specifica.

In FX è attribuita ad un tipo particolare di contenitori (Parent): i Pane

FX vs. Swing and Android architectures



contenitori con una loro regola di posizionamento delle componenti

Posizionamento *automatico* mediante *layout*

L'idea è di semplificare il compito del programmatore definendo *contenitori* di oggetti che vengono posizionati secondo regole prestabilite.

Il contenitore può ignorare i desideri della componente (espressi da `setLayoutX`)

La disposizione delle componenti è *"liquida"*

Posizionamento *automatico* mediante *layout*

Fondamentale nel progetto di interfacce

JavaFX fornisce numerosi “layout managers”

Lettura raccomandata:

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

Aggiunta-rimozione di elementi

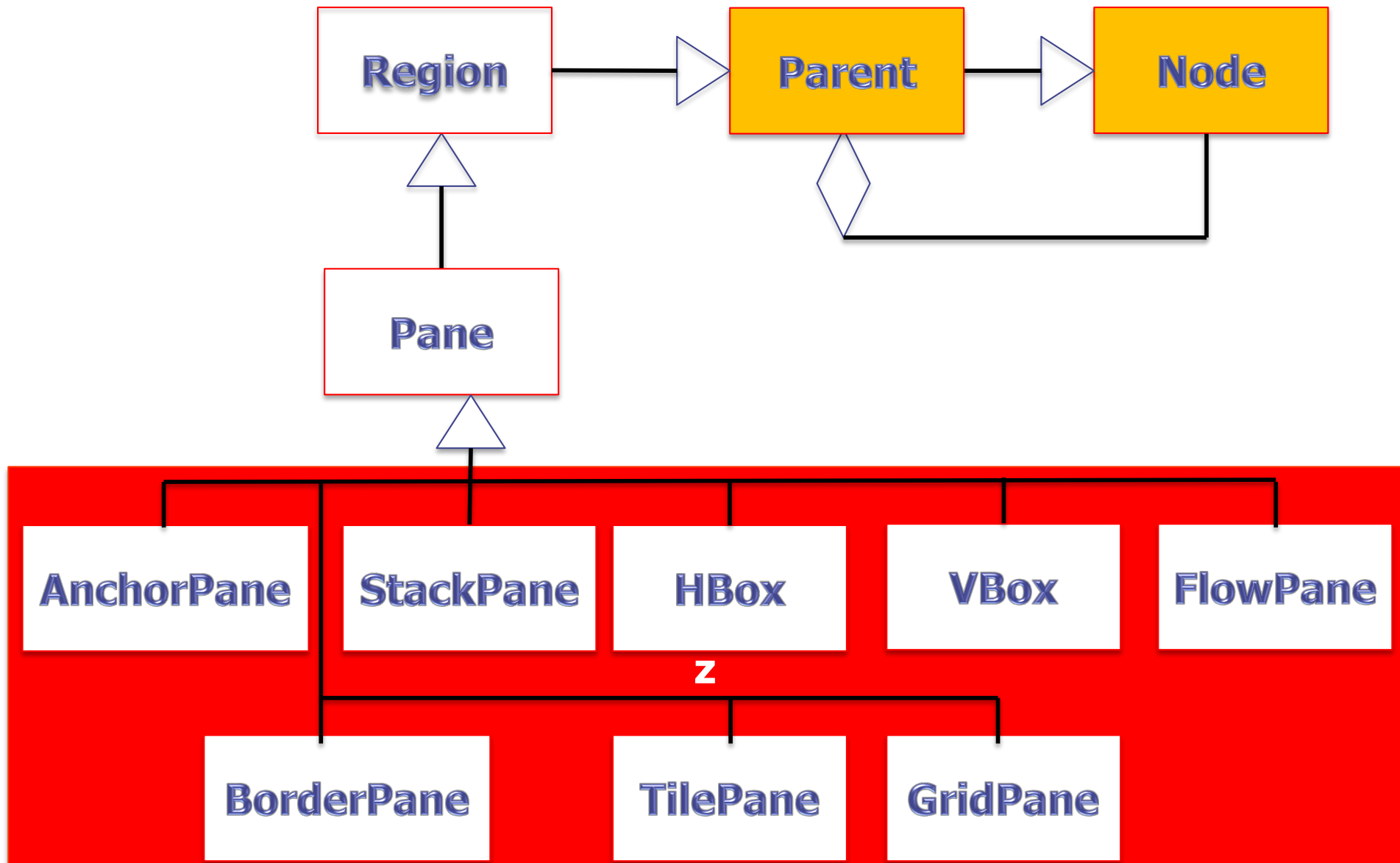
Attenzione!

Aggiunta e rimozione di elementi vengono effettuati dalla **lista dei figli (children)**, con i metodi

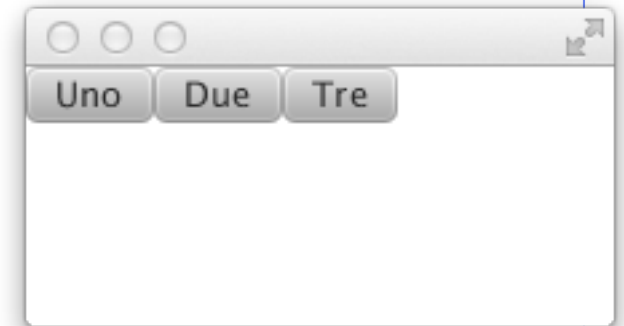
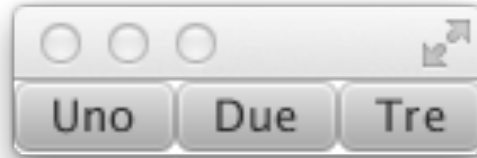
**add(Node x), addAll(Collection<Node> c),
remove(Node x), , removeAll(Collection<Node> c)**

```
Polygon triangolo= new Polygon();  
triangolo.getPoints().addAll(new Double[]{  
    0.0, 0.0, 20.0, 10.0, 10.0, 20.0 });  
Circle cerchio=new Circle(10,10,10);  
layout.getChildren().addAll(cerchio,triangolo);  
...  
layout.getChildren().remove(cerchio);
```

Layout managers predefiniti



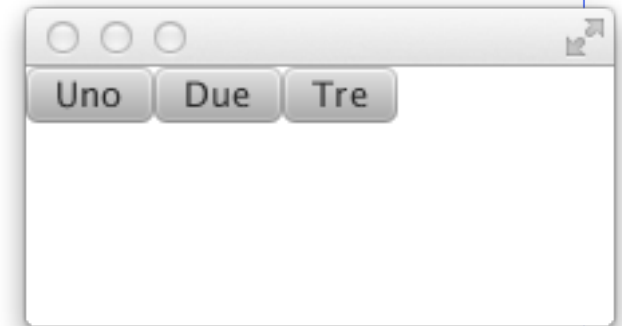
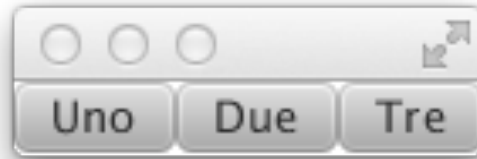
HBox



```
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();  
    }  
    ...  
}
```

Allinea in orizzontale

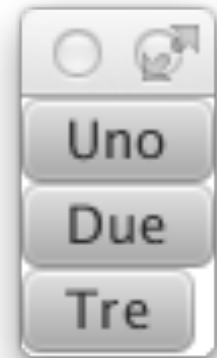
HBox



```
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(layout);  
        stage.setScene(scene);  
        stage.show();  
    }  
    ...  
}
```

Nota: è possibile creare una Scene direttamente da un Parent generico, quindi anche da un Pane o sottoclasse, ma attenzione: ognuno ha un suo allineamento predefinito

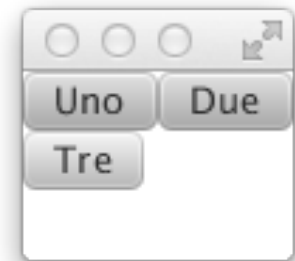
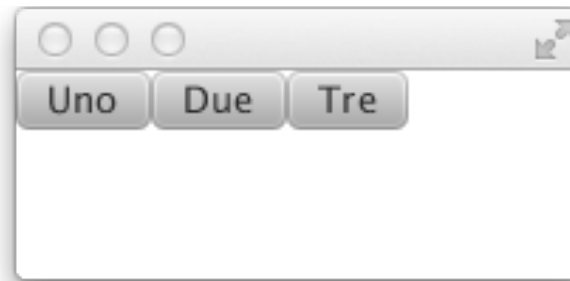
VBox



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

Allinea in verticale

FlowPane



```
public class Layout1 extends Application {  
    public void start(Stage stage) {  
        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"));  
        stage.setScene(layout);  
        stage.show();  
    } ...  
}
```

Organizza gli elementi in una sequenza continua, che va «a capo» a una distanza configurabile

StackPane

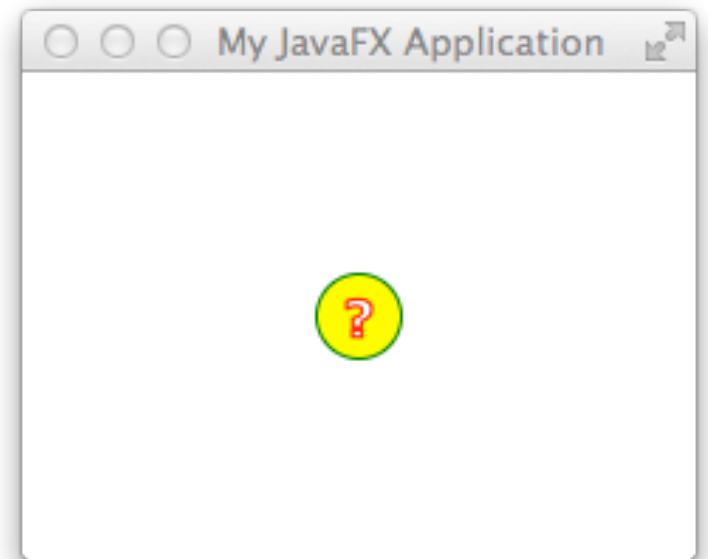
```
public class Layout1 extends Application {  
    public void start(Stage stage) {  
        StackPane layout = new StackPane();  
        layout.getChildren().add(new Button("Uno"));  
        layout.getChildren().add(new Button("Due"));  
        layout.getChildren().add(new Button("Tre"));  
 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



Impila gli elementi

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();  
    }  
}
```

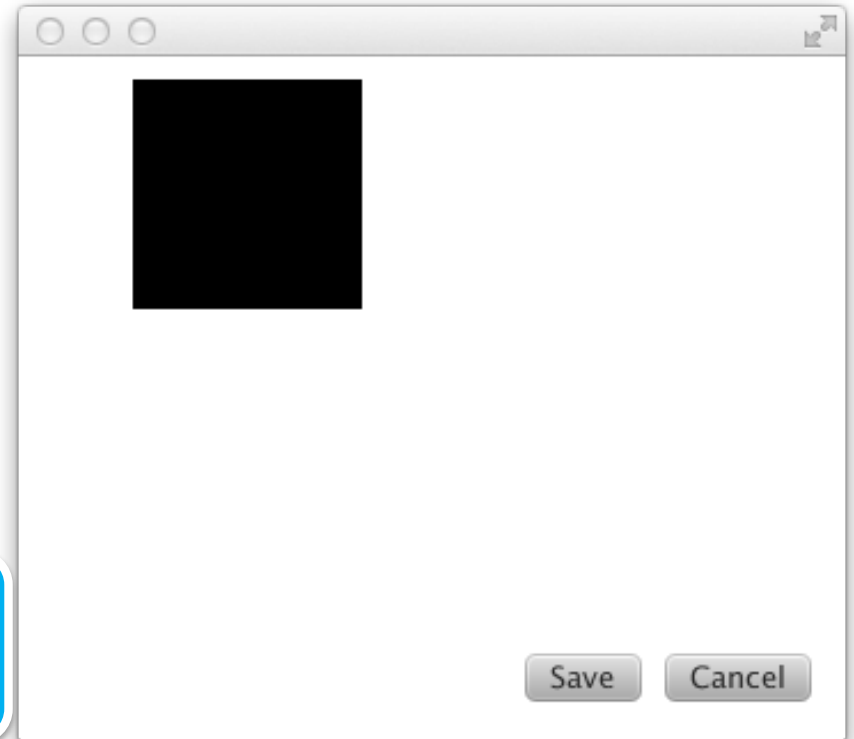
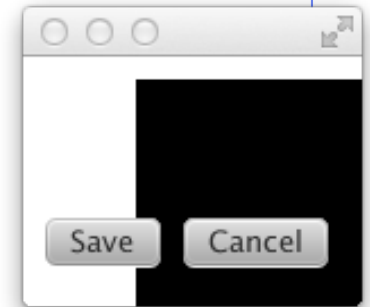


**StackPane è utile per creare
«overlay» di elementi**

AnchorPane

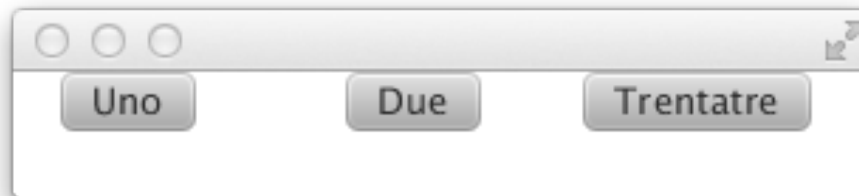
```
public void start(Stage stage) {  
    AnchorPane layout = new AnchorPane();  
    Button buttonSave = new Button("Save");  
    Button buttonCancel = new Button("Cancel");  
    HBox hb = new HBox();  
    hb.setPadding(new Insets(0, 10, 10, 10));  
    hb.setSpacing(10);  
    hb.getChildren().addAll(buttonSave,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

**Permette di «ancorare»
elementi a una zona**



TilePane

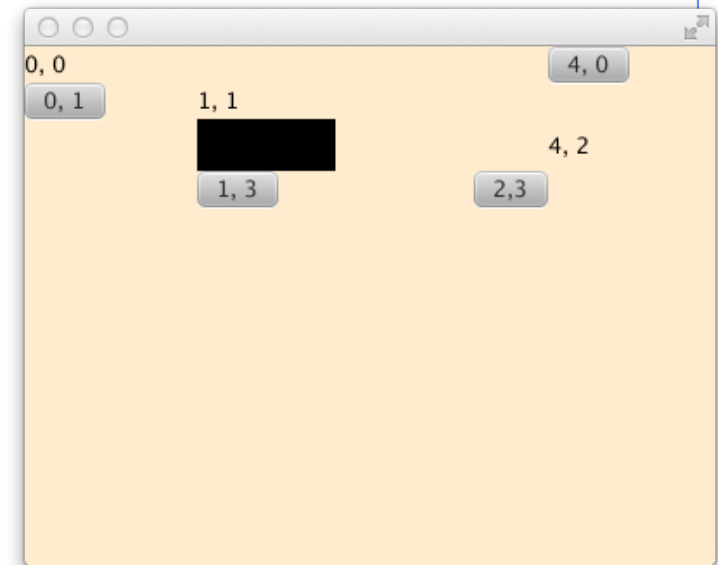
```
public class Layout1 extends Application {  
    public void start(Stage stage) {  
        TilePane layout = new TilePane();  
        layout.setVgap(10);  
        layout.setHgap(20);  
        layout.setPrefColumns(2);  
        layout.getChildren().add(new Button("Uno"));  
        layout.getChildren().add(new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



Organizza gli elementi in una griglia di celle di eguale dimensione

GridPane

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



**vedi
documentazi
one!**

**Organizza gli elementi in una griglia di cui
non è necessario dare dimensione prefissata**

Identificazione dell'elemento (i,j) in un GridPane

/**

- * implementazione generale del metodo per trovare quale elementi si trovi
 - * in posizione i,j in un GridPane.
 - * @param dp il GridPane in cui cercare
 - * @param i riga
 - * @param j colonna
 - * @return l'elemento trovato
- */

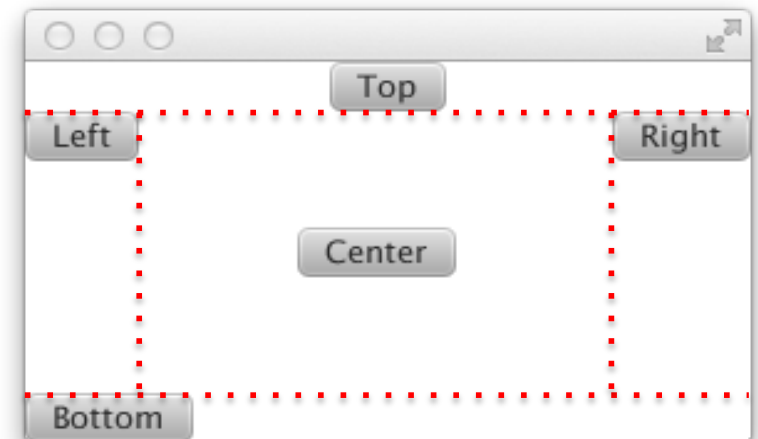
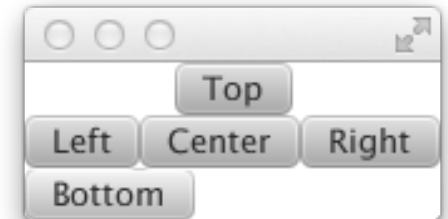
```
Node getElementAt(GridPane gp, int i, int j) {  
    for (Node x :gp. getChildren()) {  
        if ((GridPane.getRowIndex(x) == i) &&  
            (GridPane.getColumnIndex(x) == j)) {  
            return x;  
        }  
    }  
    return null;  
}
```

GridPane ha molti metodi interessanti e utili, ma ne manca uno che restituisca l'oggetto che si trova in posizione i,j nella matrice. Possiamo supplire così:

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();  
    }  
    ...  
}
```

**Organizza gli
elementi in
«zone»**



BorderPane

Combinazioni di layout



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.

Posizionamento assoluto: Pane

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

Diciamo alla
componente dove
vogliamo che lei si
posizioni

In generale, *da evitare!*

Problemi con il posizionamento...

È possibile modificare il posizionamento automatico degli elementi

- `setLayoutX` e `setTranslateX` (e analoghi per la Y)

NOTA: in genere `layoutX` è stabilito dal contenitore, cambiarlo a mano non serve a nulla!

Dalle API...

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.

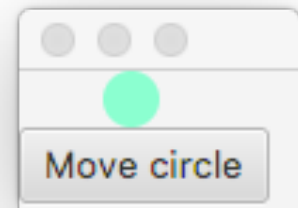
If the node is managed and has a **Region** (or subclass) as its parent, then the layout region will set layoutX according **to its own layout policy**.

If the node is unmanaged or parented by a **Group**, then the application may set layoutX directly to position it.

Esempio di posizio- namento

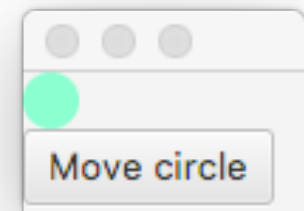
```
class Controller implements EventHandler<ActionEvent>{
    int inc = 0; Circle c; Controller(Circle c)
        @Override
        public void handle(ActionEvent event) {
            inc += 10;
            c.setLayoutX(inc % 100);
            //c.setCenterX(inc % 100);
            //c.setTranslateX(inc % 100);
            System.out.println(c.getLayoutX()
                + " " + c.getTranslateX()+" "+c.getCenterX());}}}
```

```
public class MoveBall extends Application{
    @Override
    public void start(Stage primaryStage) {
        Circle c = new Circle(200, 200, 10);
        c.setFill(Color.AQUAMARINE);
        Button btn = new Button();
        btn.setText("Move circle");
        btn.setOnAction(new Controller(c));
        VBox root = new VBox();
        root.getChildren().addAll(c, btn);
        Scene scene = new Scene(root, 100, 50);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```



Settando la LayoutX...

```
class Controller implements EventHandler<ActionEvent>{
    int inc = 0; Circle c; Controller(Circle c)
        @Override
    public void handle(ActionEvent event) {
        inc += 10;
        c.setLayoutX(inc % 100);
        //c.setCenterX(inc % 100);
        //c.setTranslateX(inc % 100);
        System.out.println(c.getLayoutX()
            + " " + c.getTranslateX()+" "+c.getCenterX());}}}
```

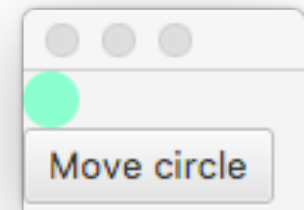


10.0	0.0	200.0
20.0	0.0	200.0
30.0	0.0	200.0
40.0	0.0	200.0
50.0	0.0	200.0

**I dati cambiano, ma non funziona:
il cerchio non si muove!**

Provando con CenterX...

```
class Controller implements EventHandler<ActionEvent>{
    int inc = 0; Circle c; Controller(Circle c)
        @Override
    public void handle(ActionEvent event) {
        inc += 10;
        //c.setLayoutX(inc % 100);
        c.setCenterX(inc % 100);
        //c.setTranslateX(inc % 100);
        System.out.println(c.getLayoutX()
+ " " + c.getTranslateX()+" "+c.getCenterX());}}}
```



```
-190.0 0.0 10.0
0.0 0.0 20.0
-10.0 0.0 30.0
-20.0 0.0 40.0
-30.0 0.0 50.0
```

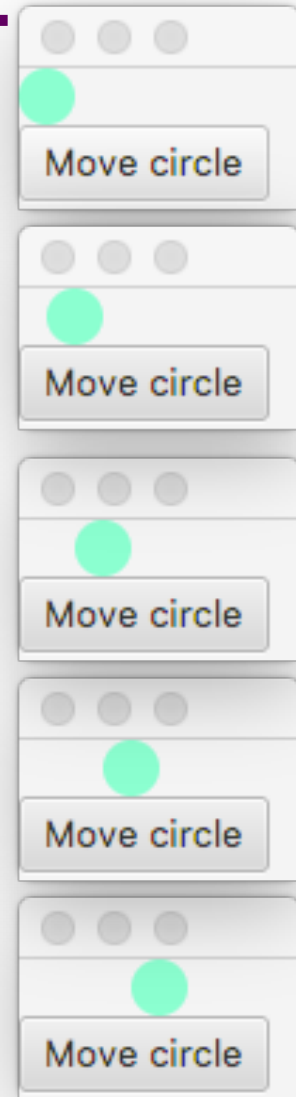
Non funziona neanche così...

e finalmente con setTranslateX...

```
class Controller implements EventHandler<ActionEvent>{  
    int inc = 0; Circle c; Controller(Circle c)  
        @Override  
    public void handle(ActionEvent event) {  
        inc += 10;  
        //c.setLayoutX(inc % 100);  
        //c.setCenterX(inc % 100);  
        c.setTranslateX(inc % 100);  
        System.out.println(c.getLayoutX()  
+ " " + c.getTranslateX()+" "+c.getCenterX());}}
```

Funziona!

```
-190.0 10.0 200.0  
-190.0 20.0 200.0  
-190.0 30.0 200.0  
-190.0 40.0 200.0
```



Dimensionamento e allineamento

È possibile dimensionare direttamente gli elementi

- Es., `btn.setPrefWidth(200);`

... oppure specificare vincoli sulle dimensioni

- Es., `btn.setMinWidth(100);`

Per esempi di dimensionamento e allineamento:

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

FXML – noi non lo trattiamo.

Anche grazie all'idea del posizionamento liquido, è possibile definire le componenti fuori dal codice, usando una programmazione dichiarativa (files XML in Android e FXML in JavaFX)

```
<BorderPane id="borderPane" xmlns:fx=http://javafx.com/fxml
  prefHeight="200" prefWidth="320">
  <stylesheets>
    <URL value="@form.css"/>
  </stylesheets>
  <top>
    <Text text="Who are you?"/>
  </top>
  <center>
    <TextField id="textfield" fx:id="textfield"/>
  </center>
</BorderPane>
```

Quasi classi: enum

Soluzione con le stringhe (def. di variabile)

```
static final String seme[] {"Cuori", "Quadri", "Fiori",  
"Picche"};
```

Soluzione con le enum (def. di tipo di dato e relativi valori ammessi!)

```
enum Seme { Cuori, Quadri, Fiori, Picche }
```

Esempio:

**i semi delle
carte**

```
public static void main(String a[]) {  
    Seme c=Seme.Cuori;  
    Seme p=Seme.Picche;  
    System.out.println(c.name()+" "+p.name());  
    System.out.println(c+" "+p); // usa toString()  
  
    System.out.println(c.ordinal()+" "+p.ordinal());  
    System.out.println(c.compareTo(p));  
  
    for (Seme x:Seme.values()) {  
        System.out.println(x);  
    }  
}
```

**Cuori Picche
Cuori Picche**

**0 3
-3**

**Cuori
Quadri
Fiori
Picche**