Note sull’esame

Esame – Iscrizione

- **Iscrizione obbligatoria (fino a 3 giorni prima)**

  No iscrizione, no exam

- **De-iscrizione (obbligatoria! fino a 24 ore prima, eventualmente via mail. Last minute – only for serious reasons-, via mail.)**

  Chi, iscritto, è assente senza una giustificazione credibile salta l'appello successivo

- Assumete le vostre responsabilità!

  No excuses. No exceptions.

Materiale ammesso all'esame

1a prova:

- Orologio
- Penne
- Documento
- N.Matricola !

2a prova:

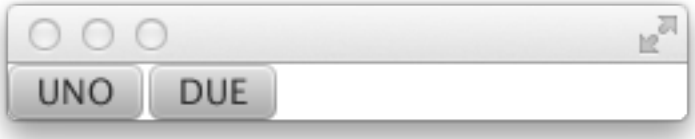
- Tutto il materiale cartaceo che volete – solo per uso personale.
- API in locale, in Netbeans.
- Copia delle slides
- Copia dei tutorials JavaFX

Registrazione (Informatici)

- Dopo aver superato ENTRAMBI I moduli dovete iscrivervi su Esse3 all'appello di REGISTRAZIONE.
- Un modulo superato ha validità 1 anno entro il quale occorre passare l'altro modulo
- NON possiamo fare registrazioni parziali.

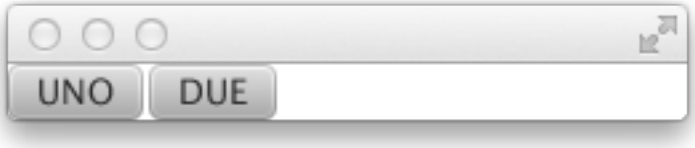
Event Chain

Event chain v.1- 1



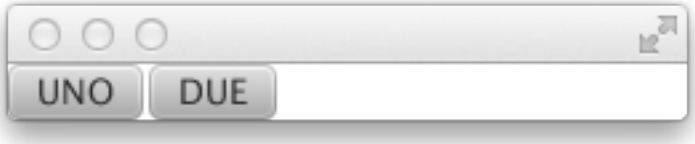

```
public class EventFilterDemo extends Application {
   public void start(final Stage stage) {
      EventHandler handler=new EventHandler<ActionEvent>() {
         @Override
         public void handle(ActionEvent t) {
            EventTarget target=t.getTarget();
            Object source=t.getSource();
            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);
   }
};
```

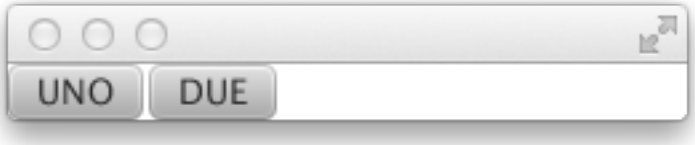
Event chain v.1- 3




```
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



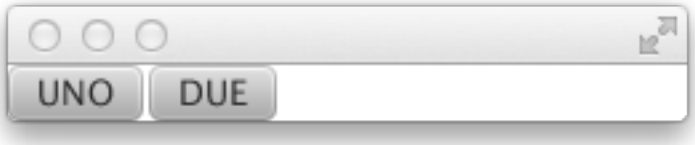

```
FILTER:STAGE javafx.stage.Stage@7c1031ba ==> Button[id=BUTTON-1, styleClass=button]
FILTER:SCENE javafx.scene.Scene@b30e9f8 ==> Button[id=BUTTON-1, styleClass=button]
FILTER:STACKPANE TilePane[id=STACKPANE, styleClass=root] ==> Button[id=BUTTON-1, styleClass=button]
FILTER:BUTTON-1 Button[id=BUTTON-1,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

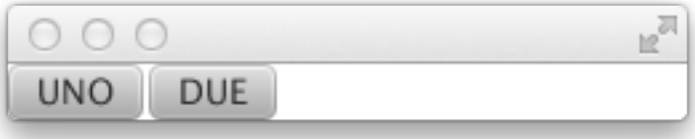
t.consume() evita che l'evento sia passato

Event chain v.2 - 1



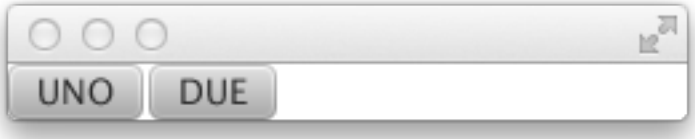

```
public void start(final Stage stage) {
        class SuperHandler implements EventHandler<ActionEvent>{
            SuperHandler() {  super(); }
            protected EventTarget target;
            protected Object source;
            protected String id;
            @Override
            public void handle(ActionEvent 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 filter=new SuperHandler () {
   public void handle(ActionEvent t) {
      super.handle(t);
      System.out.println("FILTER:"+id+" "+source+" ==> "+target);
   }
};
SuperHandler handler=new SuperHandler() {
    public void handle(ActionEvent t) {
      super.handle(t);
      System.out.println("HANDLER:"+id+" "+source+" ==> "+target);
   }
};
```

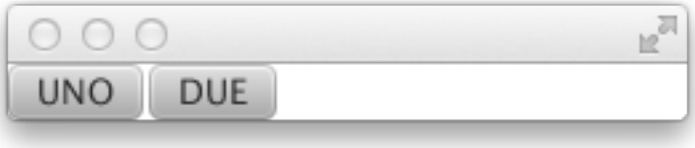
Event chain – cutter 1




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

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

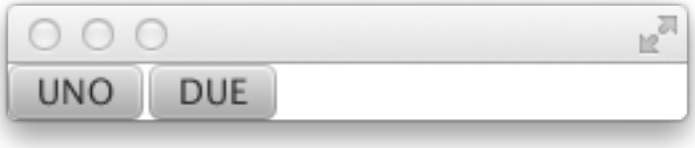
Event chain – cutter 2a




```
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]

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

Lettura suggerita…

http://docs.oracle.com/javase/8/javase-clienttechnologies.htm

Javadoc

Output

getImage

public Image getImage(URL url, String name)

- Returns an Image object that can then be painted on the screen. The url argument must specify an absolute URL. The name argument is a specifier that is relative to the url argument. This method always returns immediately, whether or not the image exists. When this applet attempts to draw the image on the screen, the data will be loaded. The graphics primitives that draw the image will incrementally paint on the screen.

  Parameters:

  url - an absolute URL giving the base location of the image

  name - the location of the image, relative to the url argument

  Returns:

  the image at the specified URL

  See Also:

  Image

Tags

Include tags in the following order:

- @author (classes and interfaces only, required)
  @version (classes and interfaces only, required.)
  @param (methods and constructors only)
  @return (methods only)
  @exception (@throws is a synonym added in Javadoc 1.2)
  @see (additional references)
  @since (since what version/ since when is it available?)
  @serial (or @serialField or @serialData)
  @deprecated (why is deprecated, since when, what to use)

http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

Un suggerimento

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 i riga
   * @param j colonna
   * @return l'elemento trovato
   */
  Node standardGetElementAt(int i, int j) {
    for (Node x : 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 scriverlo così:

Animation in JavaFX

Animation

- provides the core functionality of all animations used in the JavaFX runtime.
- Methods:
  - play()
  - playFromStart()
  - pause() (keeps the play head to the current position)
  - stop() (reset its play head to the initial position)
- The Animation progresses in the direction and speed specified by rate
- An animation can run in a loop by setting cycleCount.
- An animation with indefinite duration (a cycleCount of INDEFINITE) runs repeatedly
- If the autoReverse –flag Is set, the animation run back and forth while looping

subclasses: **Transition** and **Timeline**

Transition

Offers a simple framework to define animation, and provides all the basic functionality defined in Animation.

- Transition requires:
  - the implementation of a method interpolate(double) which is the called in each frame, while the Transition is running.
  - to set the duration of a single cycle with Animation.setCycleDuration(javafx.util.Duration).

Esempio di Transition

```
public void start(Stage primaryStage) {
  final String content = ”JavaFX Animation";
  final Text text = new Text(10, 20, "");
  final Animation animation = new Transition() {
    { setCycleDuration(Duration.millis(2000)); }
    protected void interpolate(double frac) {
       final int length = content.length();
       final int n = Math.round(length * (float) frac);
       text.setText(content.substring(0, n));
    }
  };
  StackPane root = new StackPane();
  root.getChildren().add(text);
  Scene scene = new Scene(root, 300, 250);
  primaryStage.setScene(scene);
  primaryStage.show();
  animation.play();
}
```

Timeline

- A Timeline can be used to define a free form animation of any WritableValue, e.g. all JavaFX Properties.

- A Timeline, defined by one or more KeyFrames, processes individual KeyFrame sequentially, in the order specified by KeyFrame.time. The animated properties, defined as key values in KeyFrame.values, are interpolated to/from the targeted key values at the specified time of the KeyFrame to Timeline's initial position, depends on Timeline's direction.

- Timeline processes individual KeyFrame at or after specified time interval elapsed, it does not guarantee the timing when KeyFrame is processed.

- If a KeyFrame is not provided for the time==0s instant, one will be synthesized using the target values that are current at the time Animation.play() or Animation.playFromStart() is called.

KeyFrame

Defines target values at a specified point in time for a set of variables that are interpolated along a Timeline.

The developer controls the interpolation of a set of variables for the interval between successive key frames by providing a target value and an Interpolator associated with each variable. The variables are interpolated such that they will reach their target value at the specified time.

An onFinished function is invoked on each KeyFrame if one is provided.

Esempio di animazione 1.1

```
public class BallAnimation1 extends Application {
    int dx = 1;
    int dy = 1;
    double frameDuration=0.05;
    @Override
    public void start(final Stage primaryStage) {
        primaryStage.setTitle("Animation");
        Group root = new Group();
        Scene scene = new Scene(root, 400, 300, Color.WHITE);
        primaryStage.setScene(scene);
        addBall(scene);
        primaryStage.show();
    }
    private void addBall(final Scene scene) { ... }
}
```

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

Esempio di animazione 1.2 v-1

```
private void addBall(final Scene scene) {
    final Circle ball = new Circle(50, 50, 20);
    final Group root = (Group) scene.getRoot();
    root.getChildren().add(ball);
    Timeline tl = new Timeline();
    tl.setCycleCount(100); //Animation.INDEFINITE);
    KeyFrame moveBall = new KeyFrame(Duration.seconds(frameDuration),
            new EventHandler<ActionEvent>() {
                public void handle(ActionEvent event) {
                   ball.setTranslateX(ball.getTranslateX() + 2*dx);
                   ball.setTranslateY(ball.getTranslateY() + 0*dy);
                }
            });
    tl.getKeyFrames().add(moveBall);
    tl.play();
}
```

Esempio di animazione – v-3

```
public void handle(ActionEvent event) {
    double scale=ball.getScaleX();
    if (scale>1.5) {
        ball.setFill(Color.RED);
        ball.setOpacity(ball.getOpacity()-0.005);
    }
    if (scale>2) ball.setVisible(!ball.isVisible());
    if (scale>2.5) {
        ball.setVisible(false);
        tl.stop();
    }
    ball.setScaleX(scale+0.01);
    ball.setTranslateX(ball.getTranslateX() + 2*dx);
    ball.setTranslateY(ball.getTranslateY() + 1*dy);
}
```

espansione del solo metodo handle delle slides precedenti!

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.

- 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 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 animazione 1.2 v-2

```
private void addBall(final Scene scene) {
    final Circle ball = new Circle(50, 50, 20);
    final Group root = (Group) scene.getRoot();
    root.getChildren().add(ball)
    Timeline tl = new Timeline();
    tl.getKeyFrames().addAll(
       new KeyFrame(Duration.ZERO, // set start position
           new KeyValue(ball.translateXProperty(), 0),
           new KeyValue(ball.translateYProperty(), 0)),
       new KeyFrame(new Duration(2000), // set end position at 2s
           new KeyValue(ball.translateXProperty(), 200),
           new KeyValue(ball.translateYProperty(), 100)),
       new KeyFrame(new Duration(4000), // set end position at 4s
           new KeyValue(ball.translateXProperty(), 200),
           new KeyValue(ball.translateYProperty(), 0))
        );
        tl.play();
    }
```

Predefined Transitions

- FadeTransition
- FillTransition
- PathTransition
- PauseTransition
- RotateTransition
- ScaleTransition
- StrokeTransition
- TranslateTransition

Fade Transition 1.1

```
public class FadeTransitionDemo extends Application {
    @Override
    public void start(final Stage primaryStage) {
        primaryStage.setTitle("Animation");
        Group root = new Group();
        Scene scene = new Scene(root, 400, 300, Color.WHITE);
        primaryStage.setScene(scene);
        addFadingRect(scene);
        primaryStage.show();
    }
    private void addFadingRect (final Scene scene) { ... }
}
```

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

Fade Transition 1.2

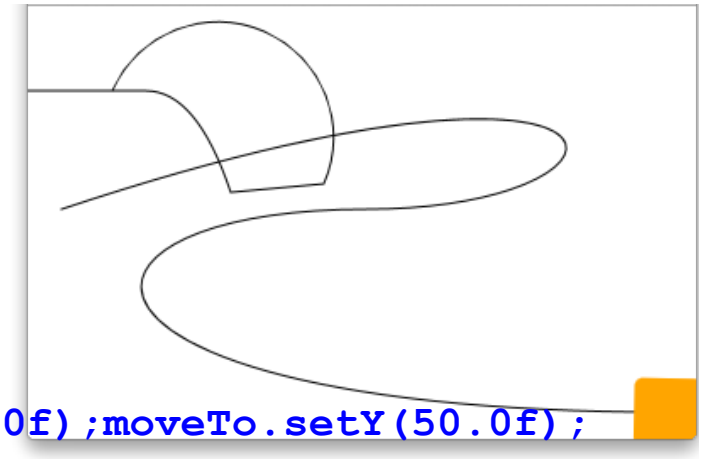
```
private void addFadingRect(final Scene scene) {
      final Rectangle rect1 = new Rectangle(10, 10, 100, 100);
      rect1.setArcHeight(20);
      rect1.setArcWidth(20);
      rect1.setFill(Color.RED);
     final Group root = (Group) scene.getRoot();
     root.getChildren().add(rect1);
     FadeTransition ft = new FadeTransition(Duration.millis(1000),
                rect1);
     ft.setFromValue(1.0);
     ft.setToValue(0.0);
     ft.setCycleCount(Timeline.INDEFINITE);
     ft.setAutoReverse(true);
     ft.play();
}
```

PathTransition 1.1

```
public class PathExample extends Application {
    @Override
    public void start(Stage primaryStage) {
        Path path = drawPath();
        final Rectangle rectPath = new Rectangle(0, 0, 40, 40);
        rectPath.setArcHeight(10);
        rectPath.setArcWidth(10);
        rectPath.setFill(Color.ORANGE);
        Group root = new Group();
        root.getChildren().addAll(path,rectPath);
        Scene scene = new Scene(root, 400, 400);
        moveObjectOnPath(path, rectPath);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    drawPath() {...}
    moveObjectOnPath (...) {...}
```

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

PathTransition 1.2



```
public Path drawPath() {
    Path path = new Path();
    MoveTo moveTo = new MoveTo(); moveTo.setX(0.0f);moveTo.setY(50.0f);
    HLineTo hLineTo = new HLineTo(); hLineTo.setX(70.0f);
    QuadCurveTo quadCurveTo = new QuadCurveTo();
    quadCurveTo.setX(120.0f); quadCurveTo.setY(110.0f);
    quadCurveTo.setControlX(100.0f); quadCurveTo.setControlY(50.0f);
    LineTo lineTo = new LineTo();
    lineTo.setX(175.0f); lineTo.setY(105.0f);
    ArcTo arcTo = new ArcTo(); arcTo.setX(50.0f); arcTo.setY(50.0f);
    arcTo.setRadiusX(50.0f); arcTo.setRadiusY(50.0f);
    path.getElements().add(moveTo); path.getElements().add(hLineTo);
    path.getElements().add(quadCurveTo); path.getElements().add(lineTo);
    path.getElements().add(arcTo);
    path.getElements().add(new MoveTo(20, 120));
    path.getElements().add(new CubicCurveTo(380, 0, 380, 120, 200, 120));
    path.getElements().add(new CubicCurveTo(0, 120, 0, 240, 380, 240));
    return path;
}
```

PathTransition 1.3

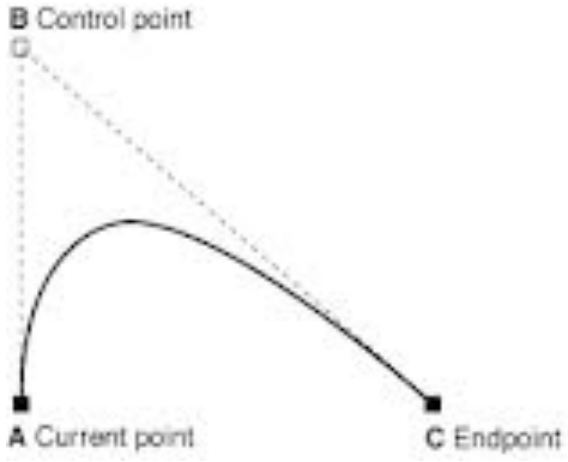
```
void moveObjectOnPath(Path path, Node node) {
        PathTransition pathTransition = new PathTransition();
        pathTransition.setDuration(Duration.millis(4000));
        pathTransition.setPath(path);
        pathTransition.setNode(node);
        pathTransition.setOrientation(
            PathTransition.OrientationType.ORTHOGONAL_TO_TANGENT);
        pathTransition.setCycleCount(Timeline.INDEFINITE);
        pathTransition.setAutoReverse(true);
        pathTransition.play();
    }
```

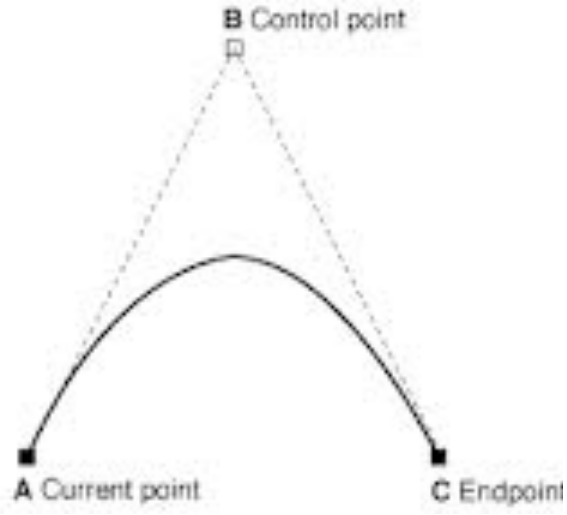
ORTHOGONAL_TO_TANGENT
The targeted node's rotation matrix is set to keep node perpendicular to the path's tangent along the geometric path.

QuadCurveTo

```
MoveTo moveTo = new MoveTo();
moveTo.setX(0.0f);
moveTo.setY(0.0f);

QuadCurveTo quadTo = new QuadCurveTo();
quadTo.setControlX(0.0f);
quadTo.setControlY(0.0f);
quadTo.setX(100.0f);
quadTo.setY(50.0f);
```

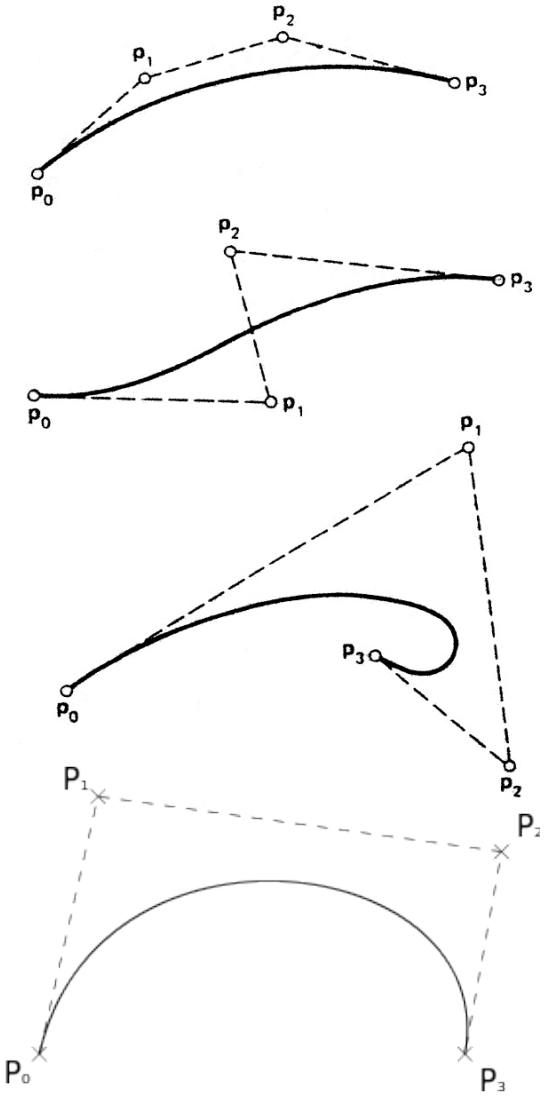




Curve di Bezier Quadratiche

$$\mathbf{B}(t) = (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2 \ , \ t \in [0,1].$$

CubicCurveTo




```
MoveTo moveTo = new MoveTo();
moveTo.setX(0.0f);
moveTo.setY(0.0f);

CubicCurveTo cubicTo = new CubicCurveTo();
cubicTo.setControlX1(0.0f);
cubicTo.setControlY1(0.0f);
cubicTo.setControlX2(100.0f);
cubicTo.setControlY2(100.0f);
cubicTo.setX(100.0f);
cubicTo.setY(50.0f);
```

Curve di Bezier Cubiche

$$\mathbf{B}(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2 t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3 \ , \ t \in [0,1].$$

Transition composition

```
SequentialTransition sq= new SequentialTransition();
sq().addAll(fadeTransition, translateTransition, ...);

ParallelTransition pt= new ParallelTransition ();
pt().addAll(fadeTransition, translateTransition, ...);
```

Altri esempi

- BouncingBallAnimation
- Animation
- HorseInMotion
- Tree