Costruttori

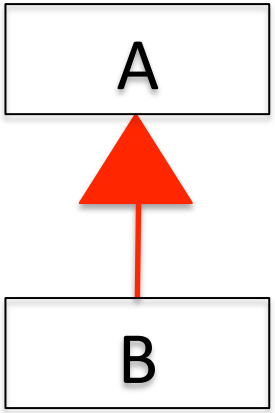
Definizione dei costruttori

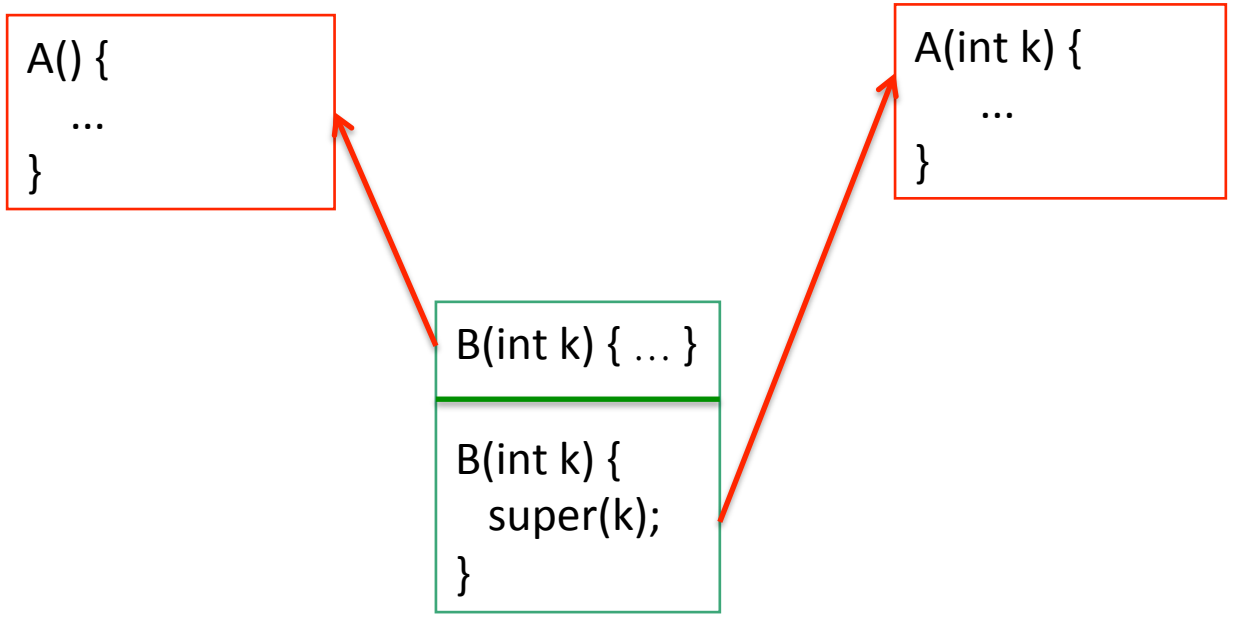
- Se per una classe A non scrivo nessun costruttore, il sistema automaticamente crea il costruttore A();

- Se invece definisco almeno un costruttore non void, ad es. A(int s), il sistema non crea il costruttore A();

Definizione dei costruttori

- Se B è figlia di A, il costruttore di B come prima cosa invoca A(), a meno che la prima istruzione non sia una super.





Invocazione dei costruttori

```
public class A {
  public A() {
    System.out.println("Creo A");
  }
}
public class B extends A {
  public B() {
    System.out.println("Creo B");
  }
  public B(int k) {
    System.out.println("Creo B_int”);
  }
}
```

```
public static void main(String [] a) {
   B b=new B(1);
 }
```

Caso 1.
Qual è l’output ?

Invocazione dei costruttori

```
public class A {
  public A(int k) {
    System.out.println("Creo A");
  }
}
public class B extends A {
  public B() {
    System.out.println("Creo B");
  }
  public B(int k) {
    System.out.println("Creo B_int”);
  }
}
```

```
public static void main(String [] a) {
    B b=new B(1);
  }
```

Caso 2.
Qual è l’output ?

Invocazione dei costruttori

```
public class A {
  public A(int k) {
    System.out.println("Creo A");
  }
}
public class B extends A {
  public B() {
    System.out.println("Creo B");
  }
  public B(int k) {
    super(k);
    System.out.println("Creo B_int”);
  }
}
```

```
public static void main(String [] a) {
    B b=new B(1);
  }
```

Caso 3.
Qual è l’output ?

La chiamata a super DEVE essere la prima istruzione!

Chiamata ad altro costruttore della stessa classe

```
public class P {
   float[] x;
   public P(int k) {
      x=new float[k];
   }
   public P() {
      this(100);
   }
```

La chiamata a this DEVE essere la prima istruzione!

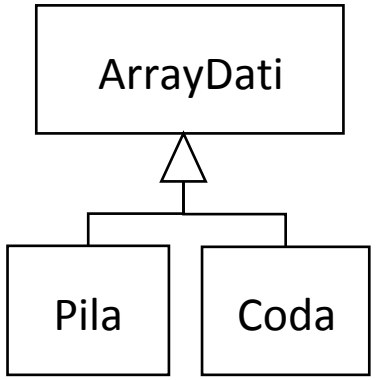
Classi astratte

Classi e metodi astratti

- Un **metodo astratto** è un metodo per il quale non è specificata alcuna implementazione
- Una **classe astratta** è tale se contiene almeno un metodo astratto
- Classi e metodi astratti sono definiti tali mediante la parola chiave **`abstract`**
- Non è possibile creare istanze di una classe astratta: bisogna definire una loro sottoclasse, che ne implementa i metodi astratti
- Le classi astratte sono molto utili per introdurre astrazioni di alto livello

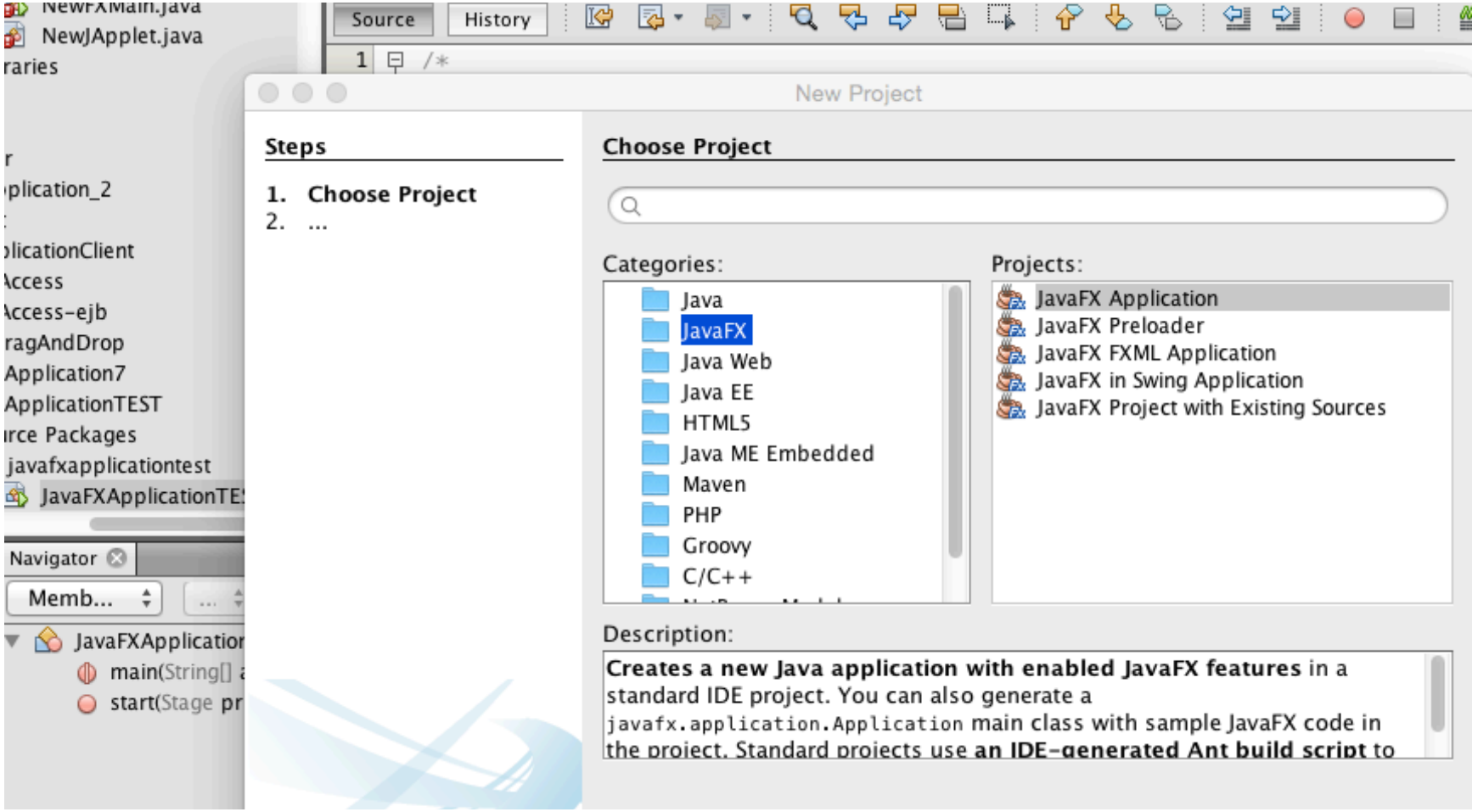
Pila, Coda, e classi astratte

```
abstract public class ArrayDati {
  int size;
  int defaultGrowthSize;
  int marker;
  int contenuto[];
  abstract public int estrai();
  … // implementazione altri metodi
}
public class Pila extends ArrayDati {
  public int estrai() { … }
}
public class Coda extends ArrayDati {
  public int estrai() { … }
}
```



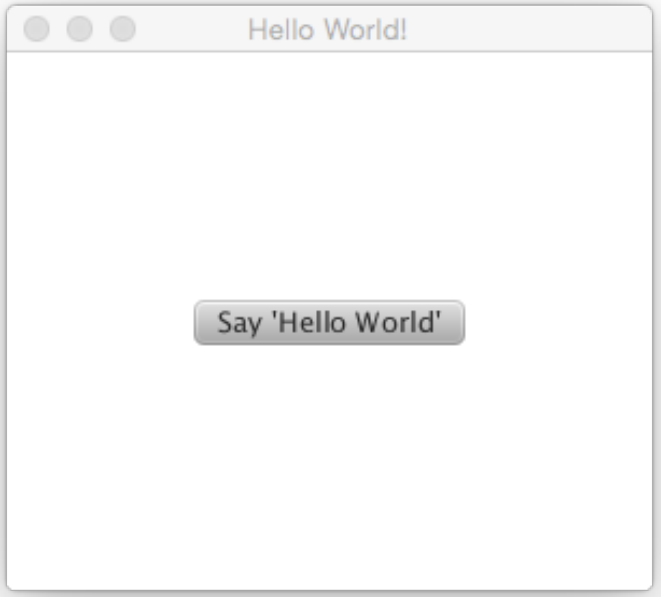
Palestra di Java con la grafica: Java FX - parte 1

Creazione di una applicazione JavaFX



```
public class JavaFXApplicationTEST extends Application {
    @Override
    public void start(Stage primaryStage) {
      Button btn = new Button();
    btn.setText("Say 'Hello World'");
    btn.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            System.out.println("Hello World!");
        }
    });
    StackPane root = new StackPane();
    root.getChildren().add(btn);
        Scene scene = new Scene(root, 300, 250);
        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

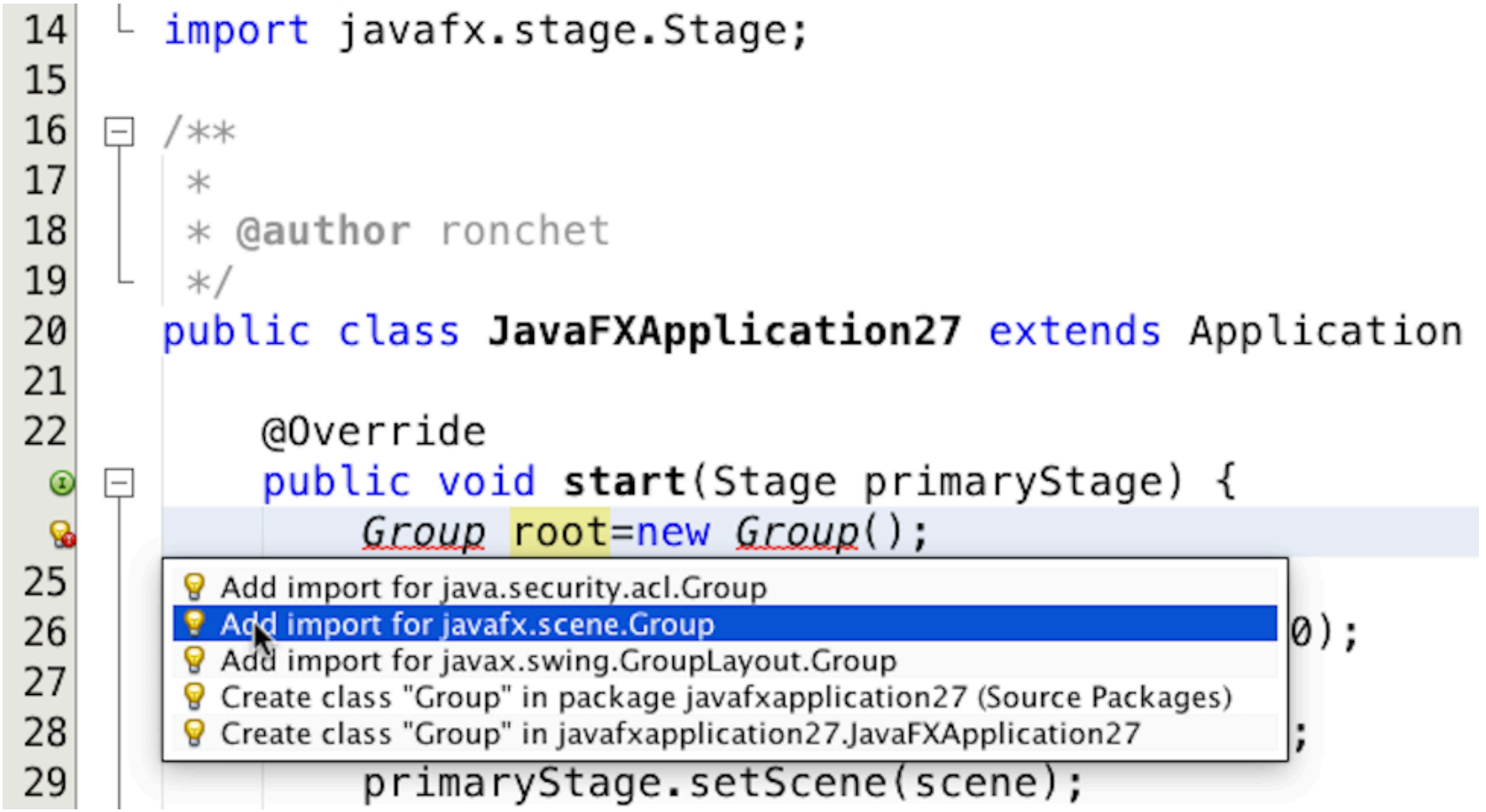
Java FX



Java FX

```
public class JavaFXApplicationTEST extends Application {
    @Override
    public void start(Stage primaryStage) {
        Group root = new Group();
        Scene scene = new Scene(root, 300, 250);
        scene.setFill(Color.AQUAMARINE);
        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

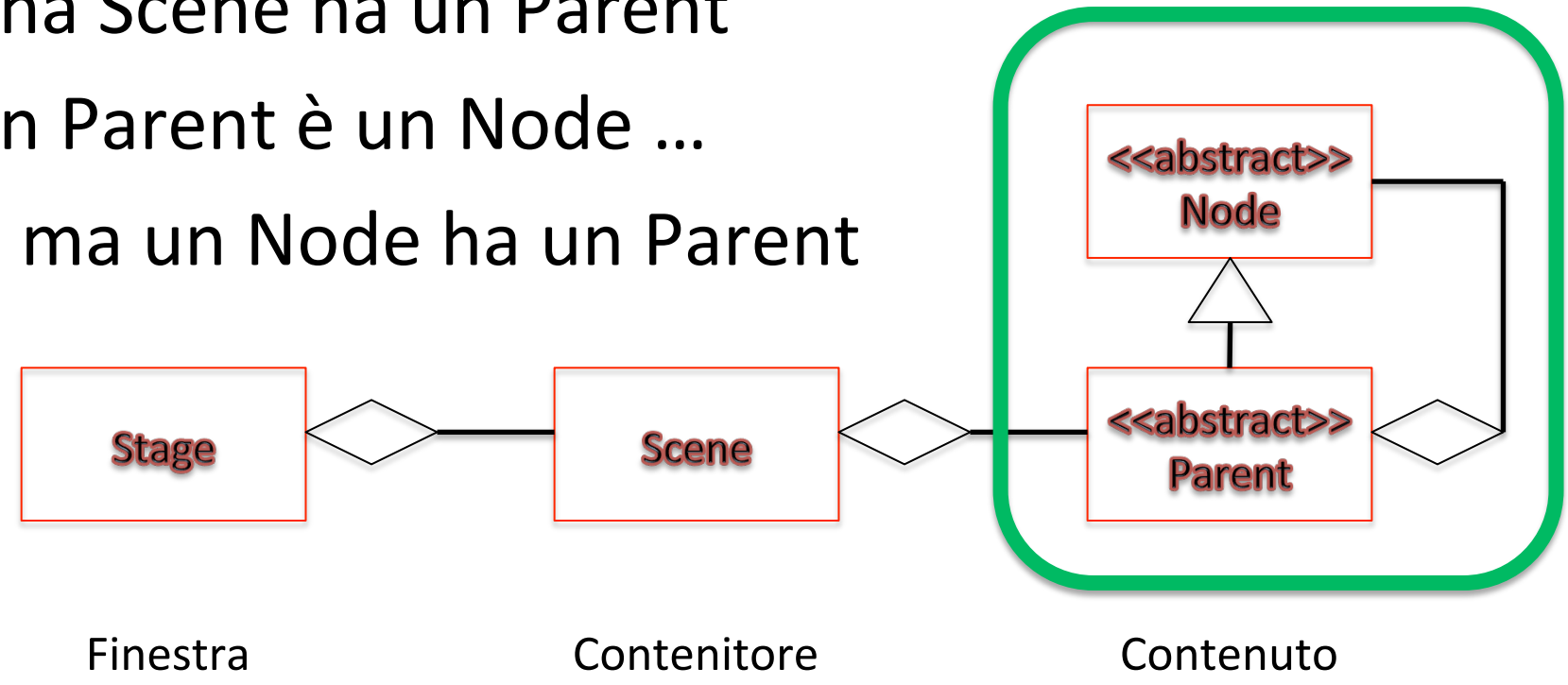
Attenzione a importare il package giusto!



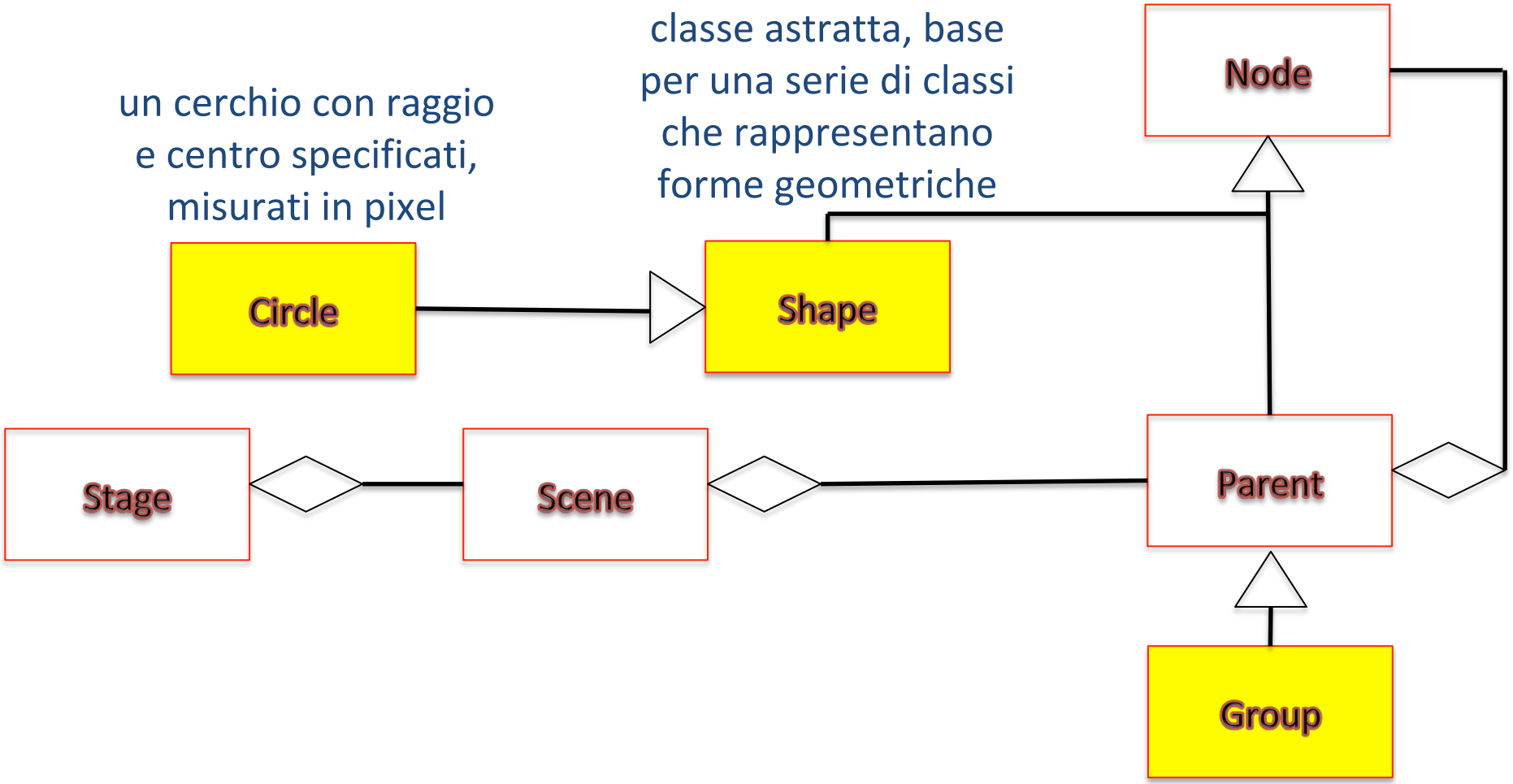
Stage/Scene/Parent/Node

Stage = “finestra”

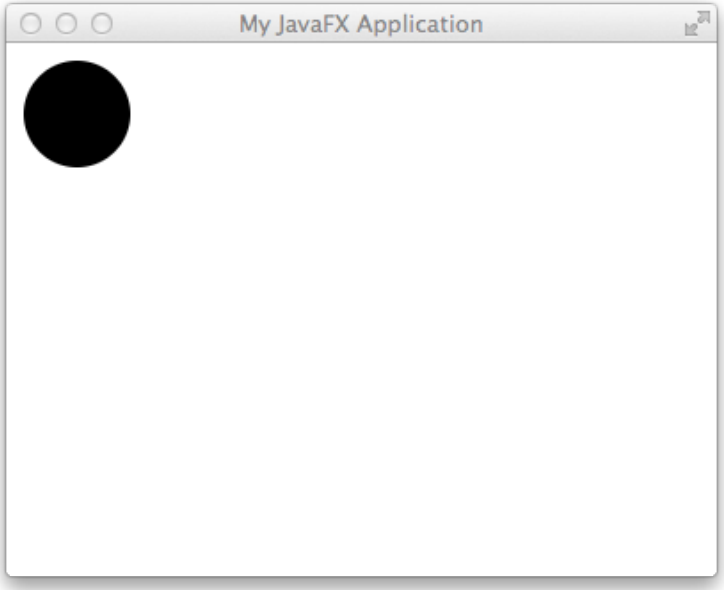
- Uno Stage contiene una Scene
- Una Scene ha un Parent
- Un Parent è un Node …
- … ma un Node ha un Parent



Group – Shape - Circle

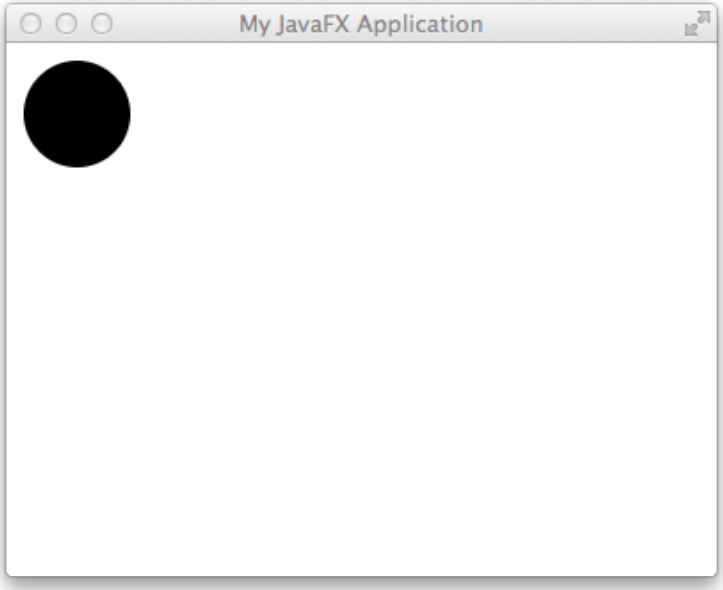


Applicazione minima




```
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;
public class MinimalApp extends Application {
      public void start(Stage stage) {
          Circle circ = new Circle(40, 40, 30);
          Group root = new Group(circ);
          Scene scene = new Scene(root, 400, 300);
          stage.setTitle("My JavaFX Application");
          stage.setScene(scene);
          stage.show();
      }
      public static void main(String[] args) {
        Application.launch(args);
      }
}
```

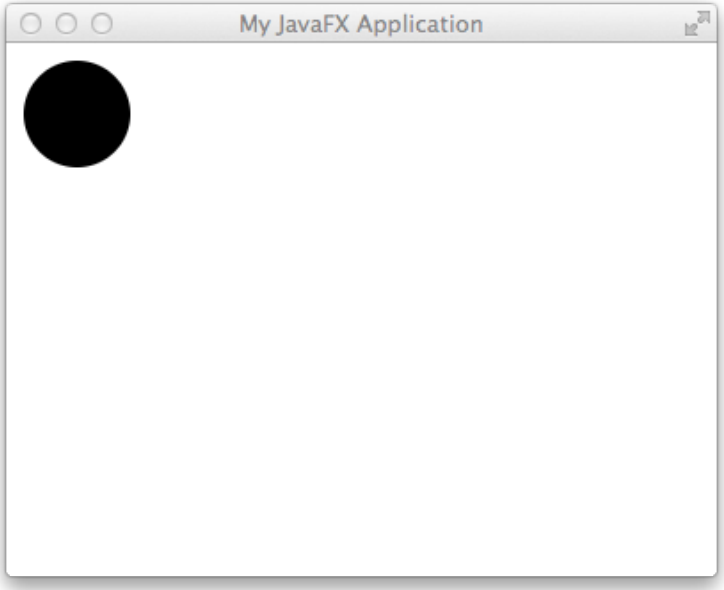
Applicazione minima




```
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;
public class MinimalApp extends Application {
      public void start(Stage stage) {
          Node circ = new Circle(40, 40, 30);
          Parent root = new Group(circ);
          Scene scene = new Scene(root, 400, 300);
          stage.setTitle("My JavaFX Application");
          stage.setScene(scene);
          stage.show();
      }
      public static void main(String[] args) {
        Application.launch(args);
      }
}
```

Superclasse a Sx dell'uguale!

Applicazione minima

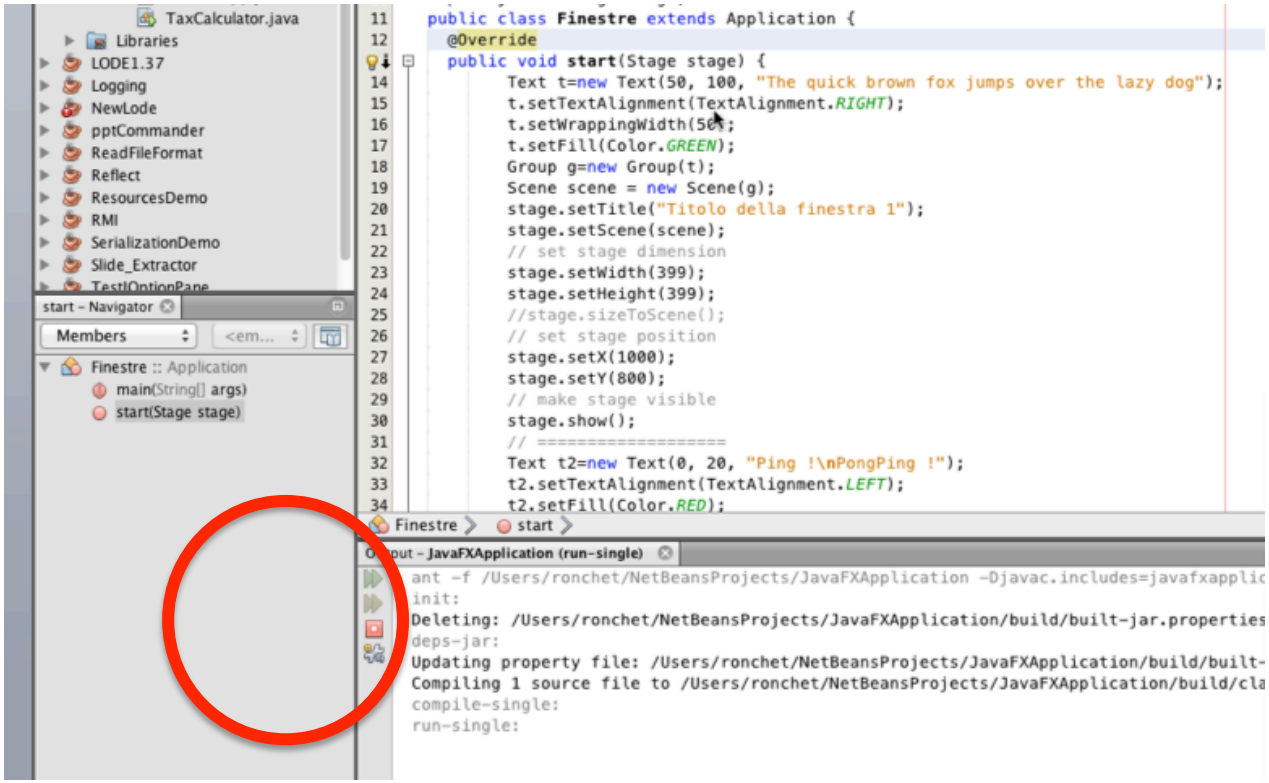



```
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;
public class MinimalApp extends Application {
      public void start(Stage stage) {
          Circle circ = new Circle(40, 40, 30);
          Group root = new Group();
          root.getChildren().addAll(circ);
          Scene scene = new Scene(root, 400, 300);
          stage.setTitle("My JavaFX Application");
          stage.setScene(scene);
          stage.show();
      }
      public static void main(String[] args) {
        Application.launch(args);
      }
```

Modo alternativo di aggiungere componenti

Quando termina l'esecuzione?

- Il *processo* associato a un'applicazione JavaFX rimane attivo anche dopo la fine di **`start()`**
  - l'applicazione va esplicitamente terminate
  - un *processo* è un *programma* in esecuzione, con il suo stato



User Input

User input – senza grafica

```
import java.util.Scanner;
...
Scanner scanner = new Scanner(System.in);
System.out.println(”dimmi qualcosa");
String inputString = scanner.nextLine();
    ...
System.out.println(inputString);
```

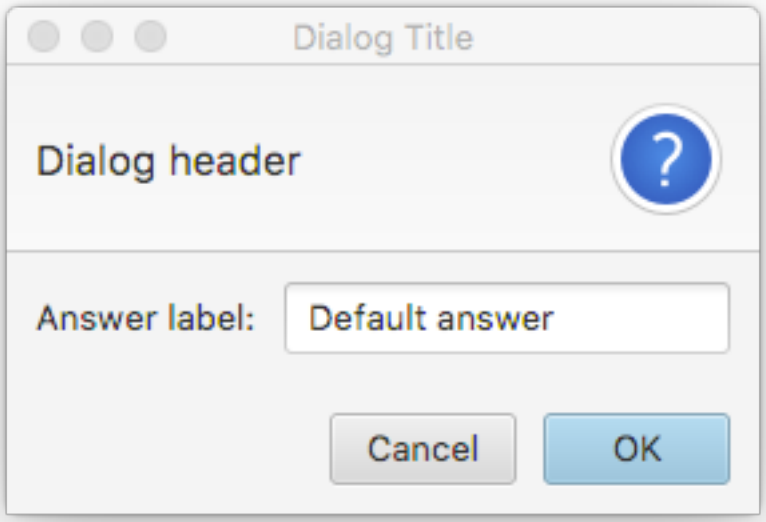
User input – con grafica

```
Alert alert = new Alert(AlertType.INFORMATION);
alert.setTitle("Alert Title");
alert.setHeaderText("Alert header:");
alert.setContentText("Alert Text");
alert.showAndWait();
```

User input – con grafica

```
TextInputDialog dialog = new TextInputDialog("Default
    answer");
dialog.setTitle("Dialog Title");
dialog.setHeaderText("Dialog header");
dialog.setContentText("Answer label:");
String s= dialog.showAndWait().get();
```

Non è proprio il modo giusto per farlo, ma per ora…



Conversione di stringhe in numeri

Conversione di stringhe in numeri

```
String s="10";
int i=Integer.parseInt(s);

String pi="3.1415026535";
float π=Float.parseFloat(pi);
```

Che succede se faccio

```
String s=”pippo";
int i=Integer.parseInt(s);
```

?

Errore!

```
Exception in Application start method
java.lang.reflect.InvocationTargetException
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at com.sun.javafx.application.LauncherImpl.launchApplicationWithArgs(LauncherImpl.java:389)
        at com.sun.javafx.application.LauncherImpl.launchApplication(LauncherImpl.java:328)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at sun.launcher.LauncherHelper$FXHelper.main(LauncherHelper.java:767)
Caused by: java.lang.RuntimeException: Exception in Application start method
        at com.sun.javafx.application.LauncherImpl.launchApplication1(LauncherImpl.java:917)
        at com.sun.javafx.application.LauncherImpl.lambda$launchApplication$155(LauncherImpl.java:182)
        at java.lang.Thread.run(Thread.java:745)
Caused by: java.lang.NumberFormatException: For input string: "z"
        at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:2043)
        at sun.misc.FloatingDecimal.parseFloat(FloatingDecimal.java:122)
        at java.lang.Float.parseFloat(Float.java:451)
        at javafxapplication27.JavaFXApplication27.start(JavaFXApplication27.java:36)
        at com.sun.javafx.application.LauncherImpl.lambda$launchApplication1$162(LauncherImpl.java:863)
        at com.sun.javafx.application.PlatformImpl.lambda$runAndWait$175(PlatformImpl.java:326)
        at com.sun.javafx.application.PlatformImpl.lambda$null$173(PlatformImpl.java:295)
        at java.security.AccessController.doPrivileged(Native Method)
        at com.sun.javafx.application.PlatformImpl.lambda$runLater$174(PlatformImpl.java:294)
        at com.sun.glass.ui.InvokeLaterDispatcher$Future.run(InvokeLaterDispatcher.java:95)
Exception running application javafxapplication27.JavaFXApplication27
```

Gestione degli errori

Proteggiamo questo codice

```
Scanner scanner = new Scanner(System.in);
String inputString;
int z;
System.out.println("dammi un numero");
inputString= scanner.nextLine();
z=Integer.parseInt(inputString);
System.out.println("input valido!");
```

Blocco try - catch

```
Scanner scanner = new Scanner(System.in);
String inputString;
int z;
 System.out.println("dammi un numero");
inputString= scanner.nextLine();
try {
      int z=Integer.parseInt(inputString);
     System.out.println("input valido!");
} catch (NumberFormatException ex) {
     System.out.println("input non valido!");
}
```

Possiamo fare di meglio

```
Scanner scanner = new Scanner(System.in);
String inputString;
int z;
boolean failure=true;
do {
  try {
       System.out.println("dammi un numero");
       inputString= scanner.nextLine();
       int z=Integer.parseInt(inputString);
       failure=false;
  } catch (NumberFormatException ex) {
       failure=true;
  }
} while (failure);
```

Clausola finally

Nota: c'è una ulteriore clausola!

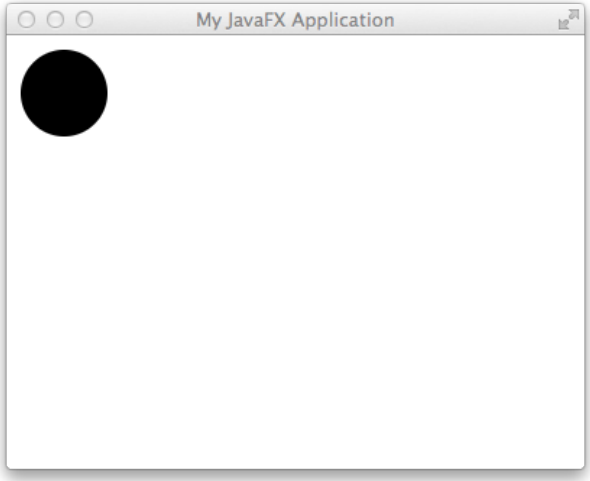
```
try {
      codice che potrebbe generare errore
} catch (NumberFormatException ex) {
    codice da eseguire se si verifica un errore
} finally {
   codice da eseguire sia che ci sia stato un errore,
   sia che non ci sia stato.
}
```

finally

The finally block always executes when the try block exits. This ensures that the finally block is executed even if an unexpected exception occurs. But finally is useful for more than just exception handling — it allows the programmer to avoid having cleanup code accidentally bypassed by a return, continue, or break. Putting cleanup code in a finally block is always a good practice, even when no exceptions are anticipated.

Esercizio

Esercizio per casa



Modificare il codice che disegna un disco, in modo che dopo aver disegnato la finestra con il cerchio (dopo la stage.show()), all'utente venga chiesto tramite un Dialog il valore del raggio, e si modifichi il cerchio usando il valore dato.

Si gestiscano errori, e si ponga un limite minimo e massimo ai valori accettati per il raggio.

Information hiding: visibilità

Information hiding in Java

- La visibilità degli attributi e metodi di una classe `C` può essere:
  - **`public`**
    - visibili a tutti
    - vengono ereditati
  - **`protected`**
    - visibili solo alle sottoclassi
    - vengono ereditati
  - **«package» (nessun modificatore specificato)**
    - visibili solo alle classi dichiarate nel package di `C`
  - **`private`**
    - visibili solo all'interno della stessa classe
    - non visibili nelle sottoclassi

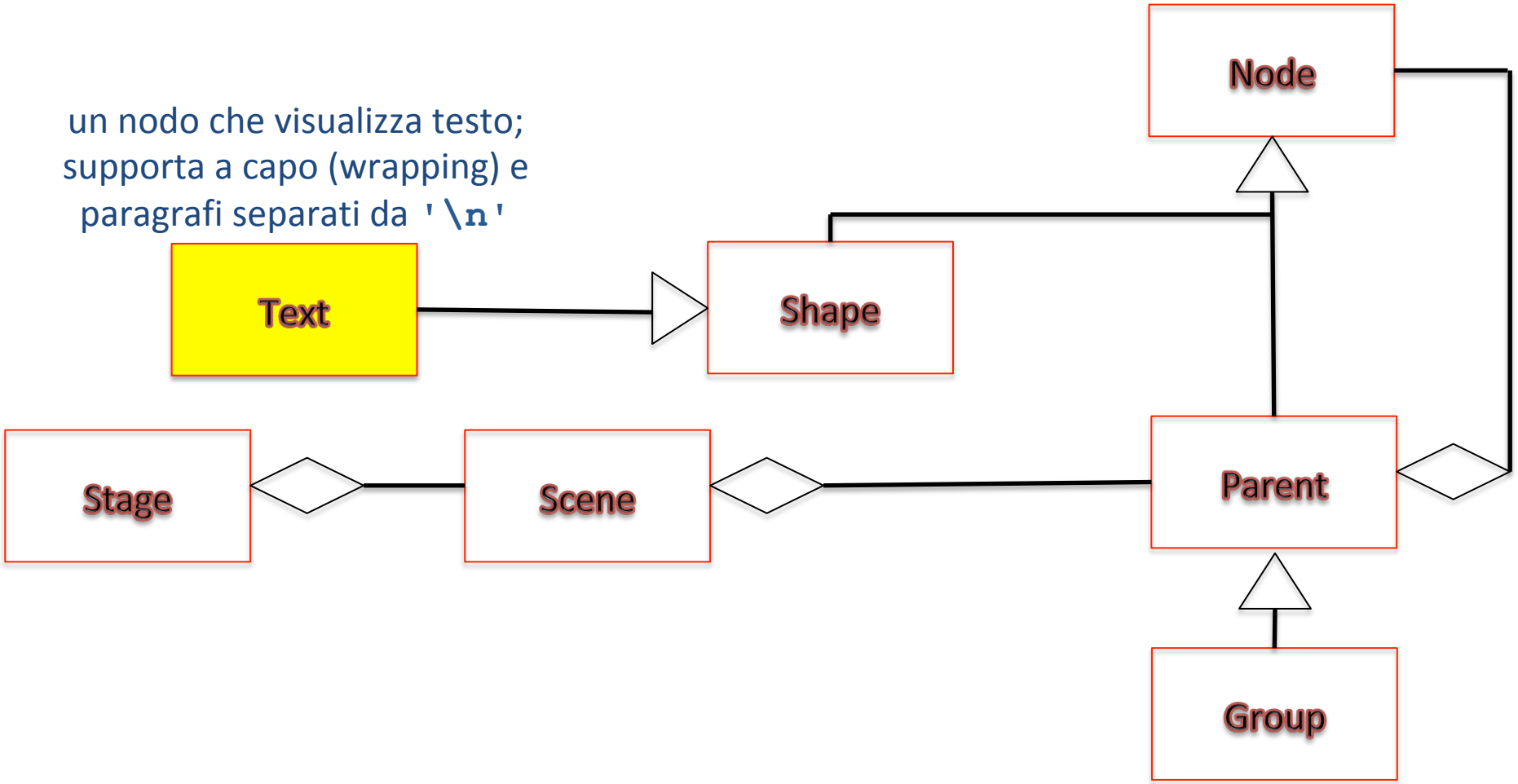
Modificatori di visibilità

	visibilità			
modificatore	**classe**	**package**	**sottoclasse**	**mondo**
`private`	Y	N	N	N
“package”	Y	Y	N	N
`protected`	Y	Y	Y	N
`public`	Y	Y	Y	Y

Palestra di Java con la grafica: Java FX - parte 2

Shape & Text

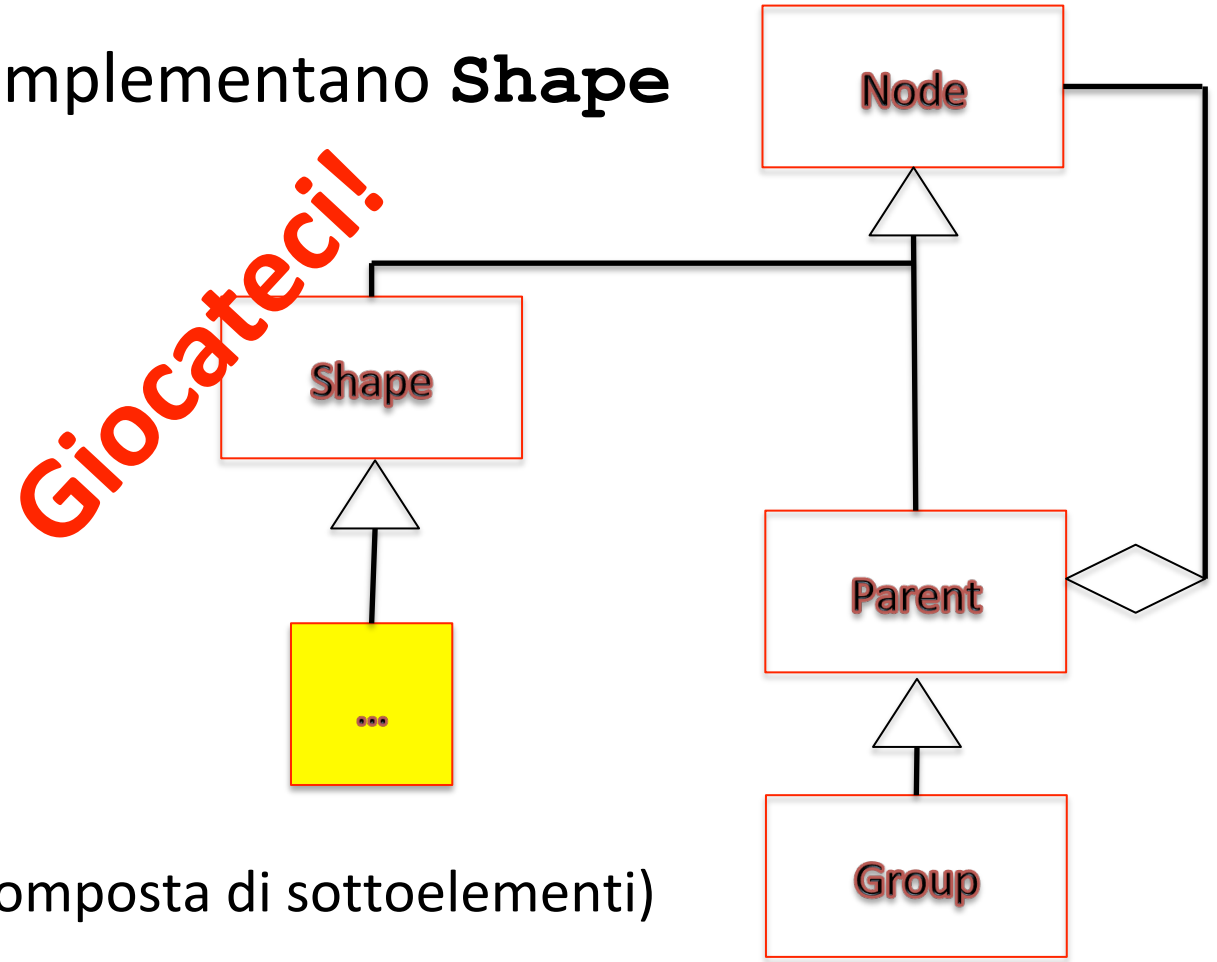
un nodo che visualizza testo;
supporta a capo (wrapping) e
paragrafi separati da `'\n'`



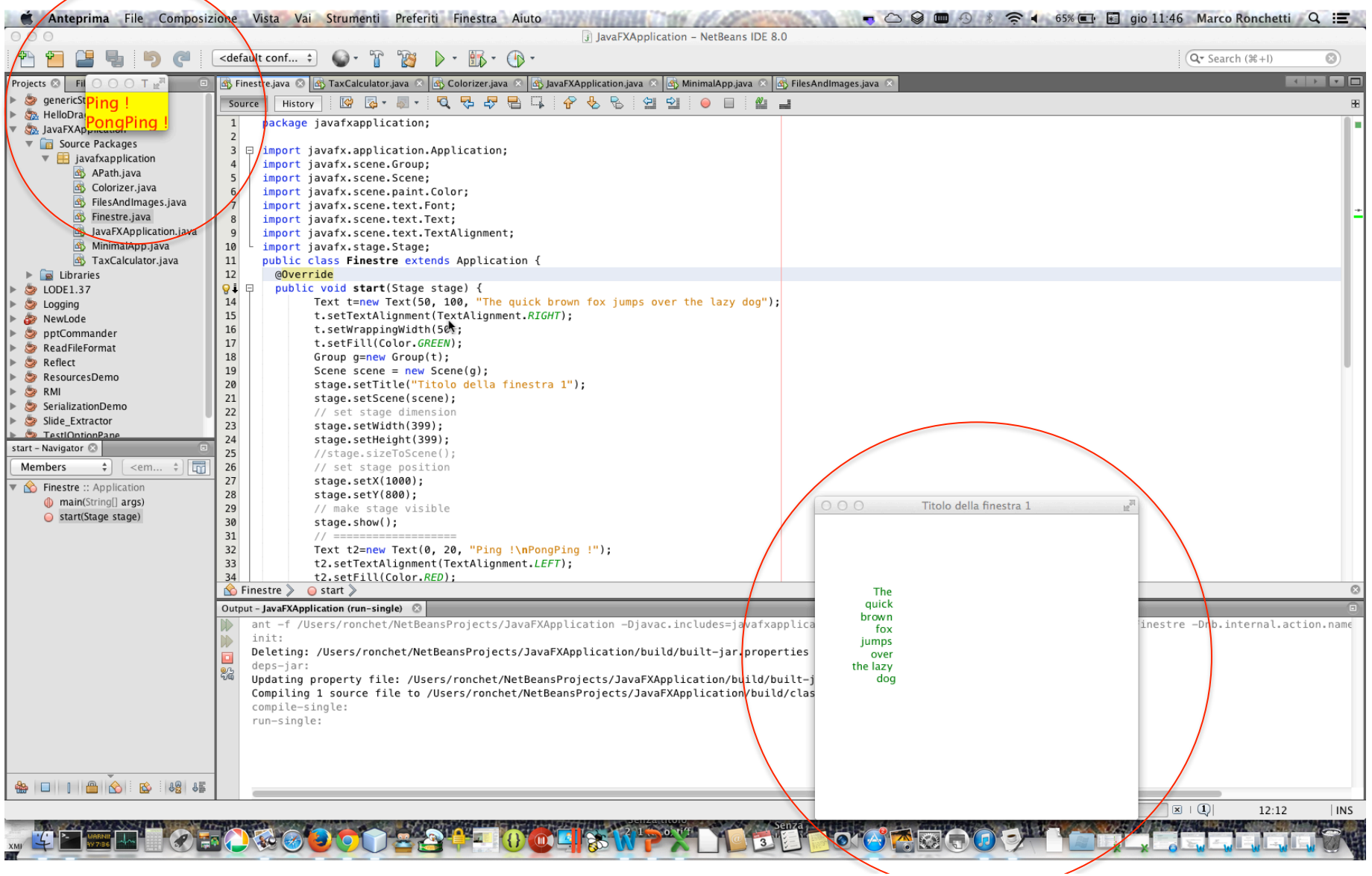
Gerarchia di Shape

Le seguenti classi implementano **`Shape`**

- **`Line`**
- **`Polyline`**
- **`Polygon`**
- **`Rectangle`**
- **`Arc`**
- **`Circle`**
- **`Ellipse`**
- **`QuadCurve`**
- **`CubicCurve`**
- **`Text`**
- **`SVGPath`** (linea composta di sottoelementi)



Esempio: Finestre multiple



Finestre multiple: prima finestra

```
public class Finestre extends Application {
public void start(Stage stage) {
    Text t=new Text(50, 100, "The quick brown fox jumps over
       the lazy dog");
    t.setTextAlignment(TextAlignment.RIGHT);
    t.setWrappingWidth(50);
    t.setFill(Paint.valueOf("GREEN"));
    Group g = new Group(t);
    Scene scene = new Scene(g);
    stage.setTitle("Titolo
                della finestra 1");
    stage.setScene(scene);
    // set stage dimension
    stage.setWidth(399);
    stage.setHeight(399);
   // set stage position
    stage.setX(1000);
    stage.setY(800);
    // make stage visible
    stage.show();
```

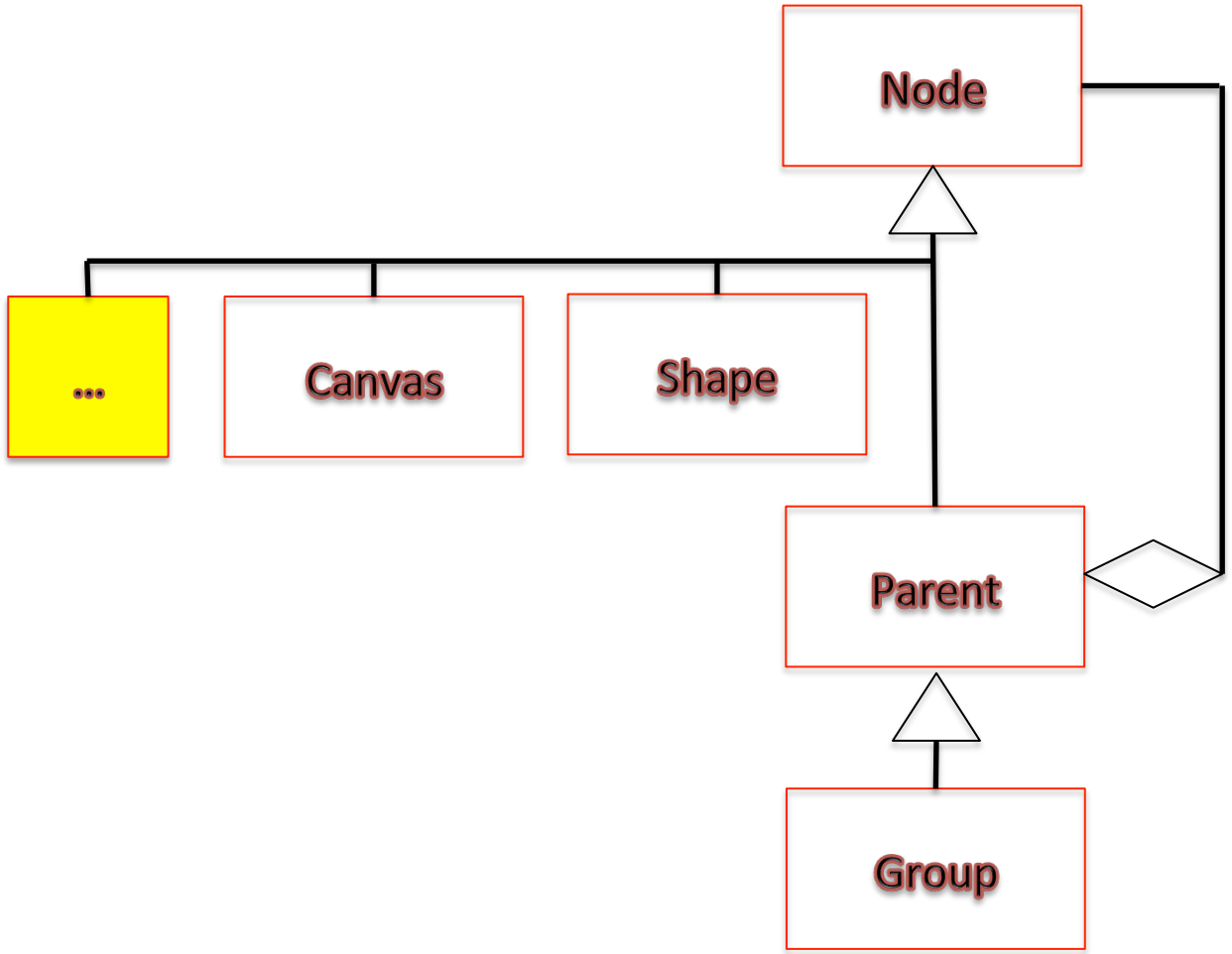
Finestre multiple: seconda finestra

```
        Text t2=new Text(0, 20, "Ping !\nPongPing !");
        t2.setTextAlignment(TextAlignment.LEFT);
        t2.setFill(Paint.valueOf("RED"));
        t2.setFont(new Font(20));
         Group g2 = new Group(t2);
         Scene scene2 = new Scene(g2);
         scene2.setFill(Paint.valueOf("YELLOW"));
         Stage stage2 = new Stage();
         stage2.setTitle("Titolo della finestra 2");
         stage2.setScene(scene2);
         stage2.sizeToScene();
         stage2.setX(100);
         stage2.setY(80);
         stage2.show();
  }
  public static void main(String[] args) {
          launch(args);
  }
}
```

Node hierarchy

Node

- Parent
- Shape
- Canvas



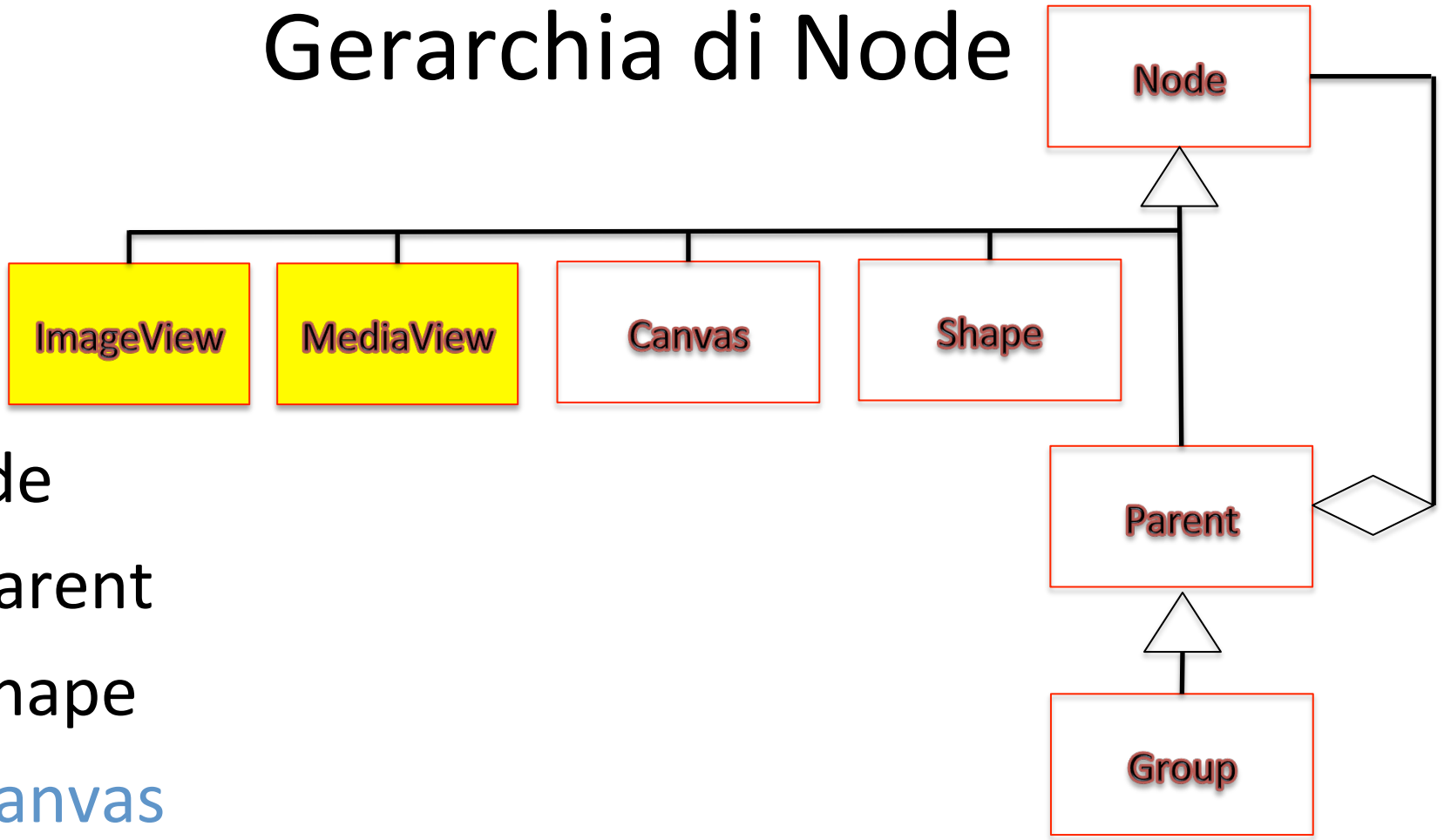
Canvas

Una "tela del pittore" con un metodo per ottenere il suo `GraphicContext` con varie primitive per disegnarci sopra:

- `fillArc()`
- `fillRect()`
- `drawImage()`
- …

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

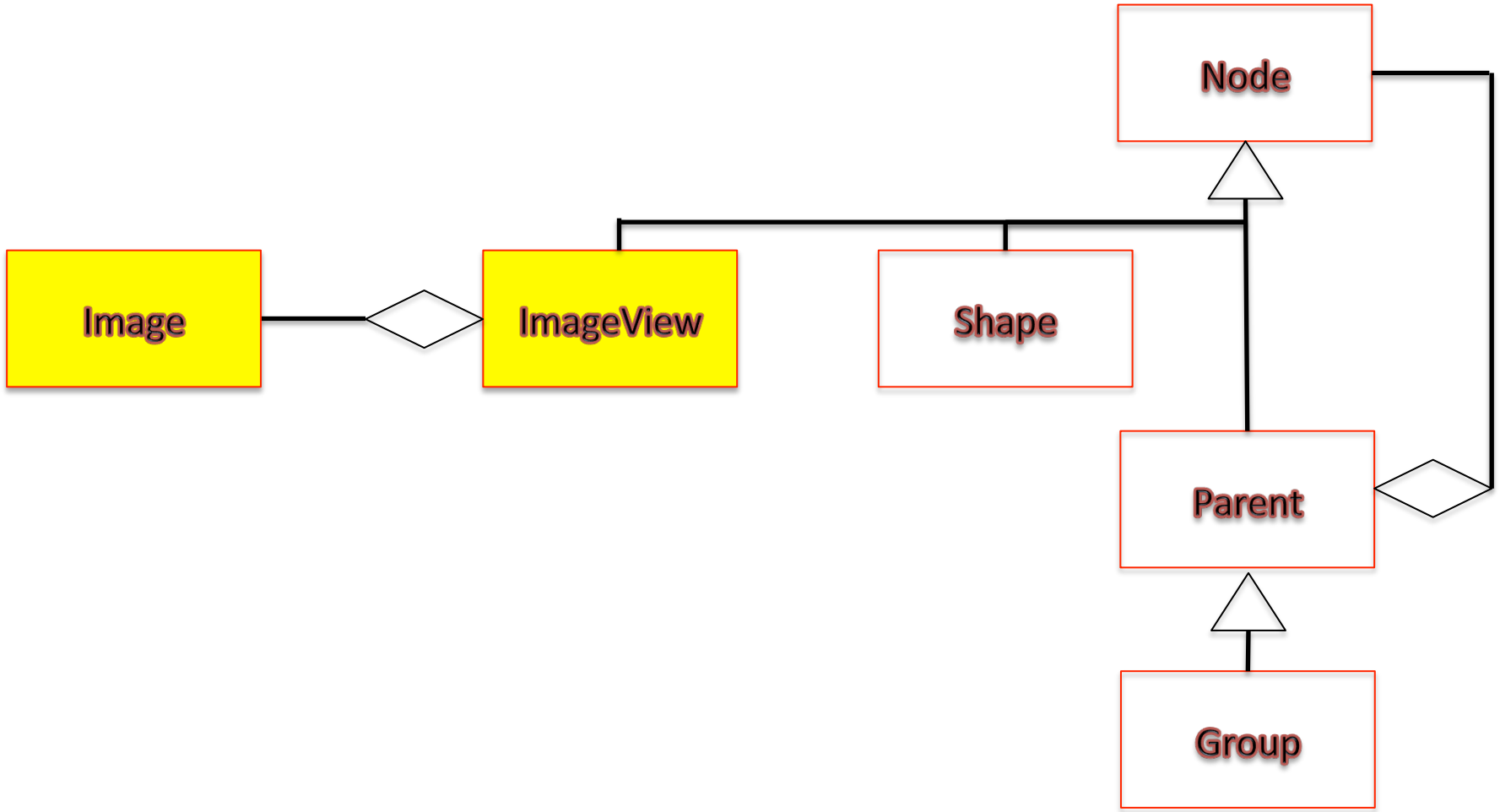
Gerarchia di Node



Node

- Parent
- Shape
- Canvas
- ImageView
- MediaView

ImageView & Image



Gerarchia di Parent (parziale)

Parent

- Control
  - superclasse di vari widget,

    tra cui FileChooser
- …

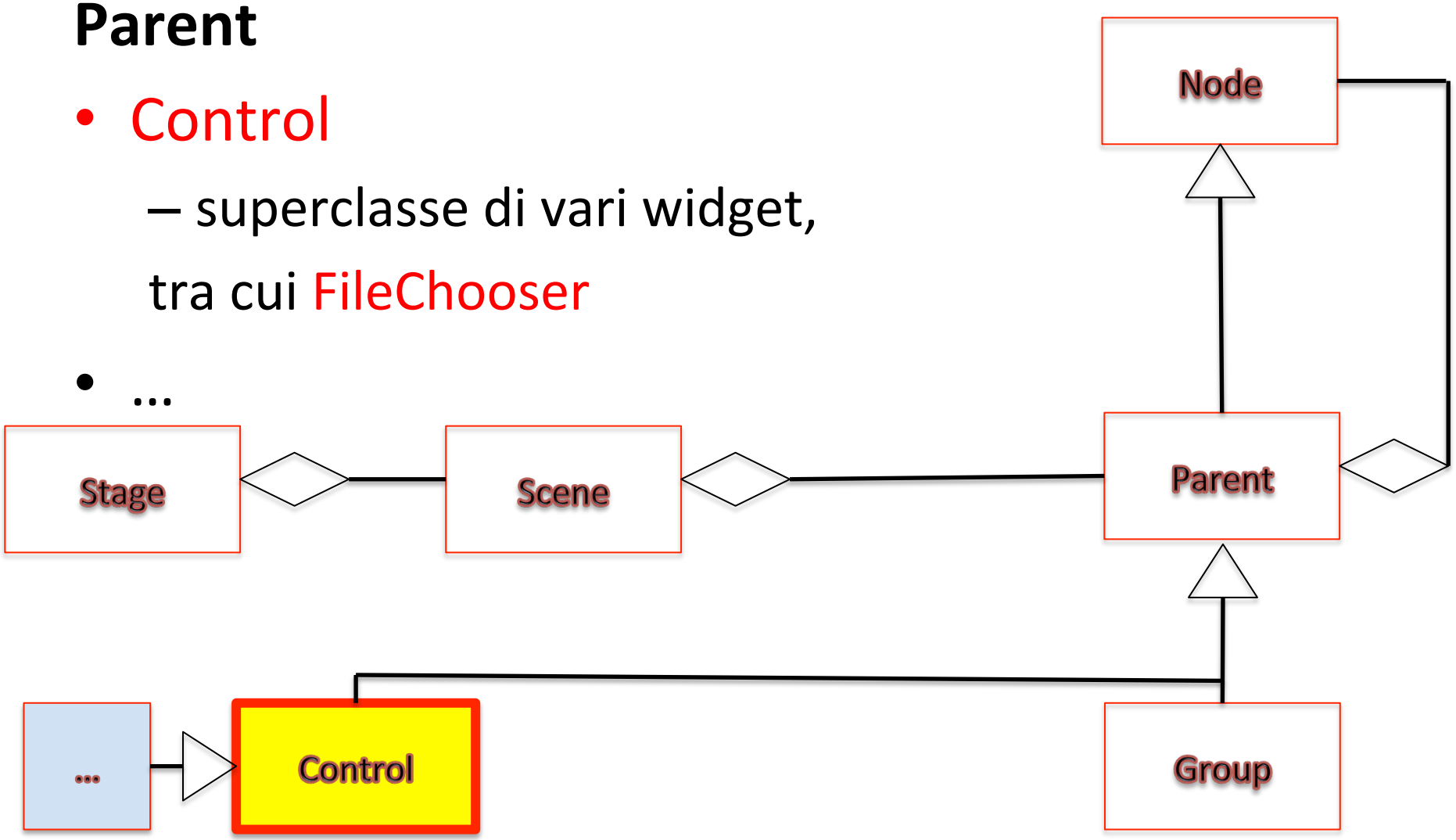
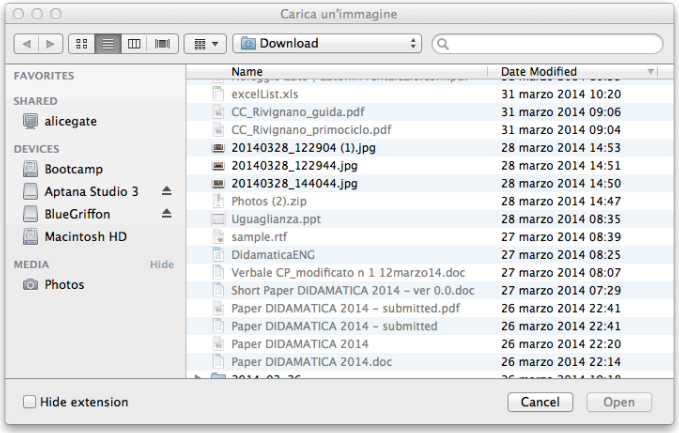


Image & File

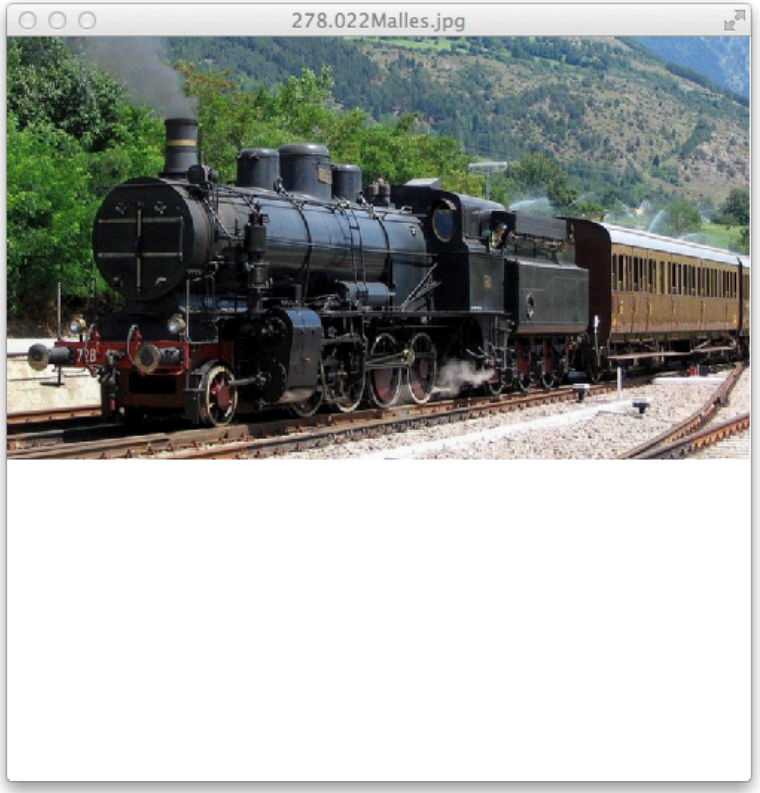



```
public class FilesAndImages extends Application {
    public void start(Stage stage) {
        FileChooser fileChooser = new FileChooser();
         fileChooser.setTitle("Carica un'immagine");
        fileChooser.getExtensionFilters().addAll(
                new FileChooser.ExtensionFilter("JPG", "*.jpg"),
                new FileChooser.ExtensionFilter("PNG", "*.png")
         );
        String url = System.getProperty("user.home");
         File f=new File(url);
        fileChooser.setInitialDirectory(f); // bugged on MacOsX
        File file = fileChooser.showOpenDialog(stage);
         if (file == null) {
             System.out.println("No file chosen");
             System.exit(1);
         }
```

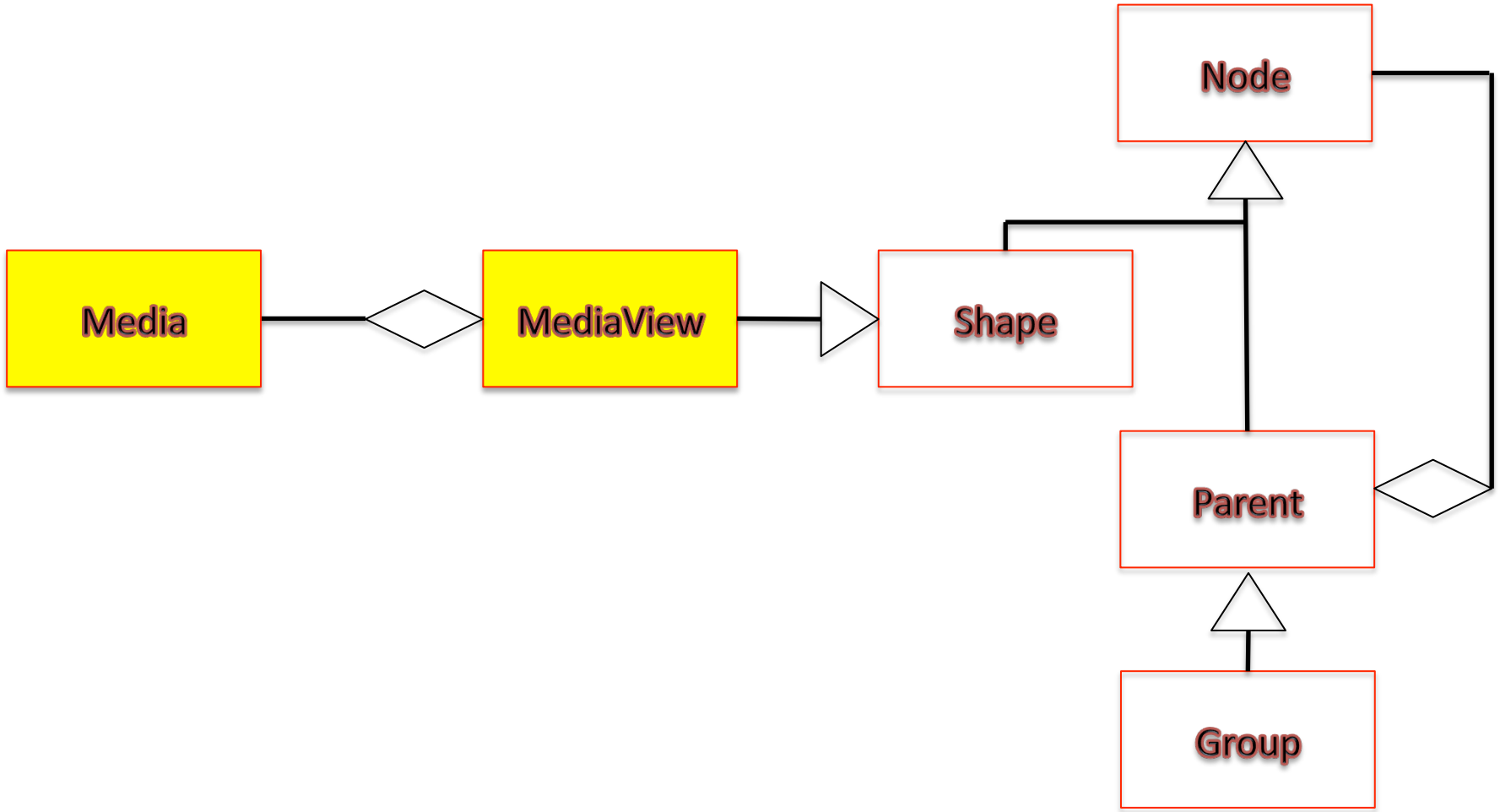
Image & File

```
        Image image = new Image("file://" +
                    file.getAbsolutePath(), 500, 500, true, true);
        ImageView iw = new ImageView(image);
        Group root = new Group(iw);
        Scene scene = new Scene(root, 500,500);
        stage.setTitle(file.getName());
        stage.setScene(scene);
        stage.sizeToScene();
        stage.show();
    }

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

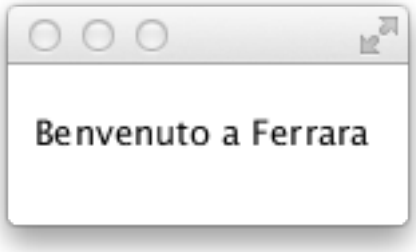


MediaView & Media



MediaView

```
public class Sounds extends Application{
    public void start(Stage stage) {
        Media media = new Media("http://www.ferraraterraeacqua.it/
it/audioguide/audioguide-di-ferrara-citta-del-rinascimento/
01_benvenuto-a-ferrara.mp3");
        MediaPlayer mediaPlayer = new MediaPlayer(media);
        mediaPlayer.setAutoPlay(true);
        // create mediaView and add media player to the viewer
        MediaView mediaView = new MediaView(mediaPlayer);
        Group root = new Group(mediaView);
        root.getChildren().add(
                new Text(10, 30,"Benvenuto a Ferrara"));
        Scene scene = new Scene(root, 150, 60);
        stage.setScene(scene);
        stage.sizeToScene();
        stage.show();
    }
    public static void main(String[] args) {
        Application.launch(args);
    }
}
```



http://docs.oracle.com/javafx/2/media/overview.htm

Parent hierarchy

Parent

- Control
  - superclasse di vari widget,

    tra cui FileChooser (lo vediamo tra poco)
- Group
- Region
- WebView

A Region is an area of the screen that can contain other nodes

WebView is a Node that manages a WebEngine and displays its content.