

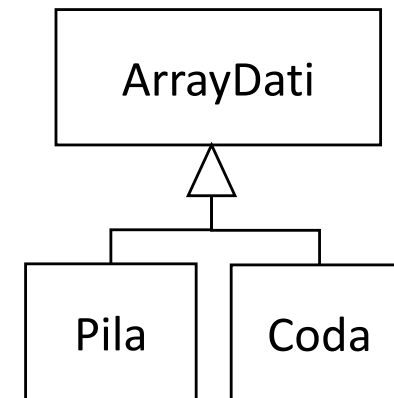
Classi astratte

Classi e metodi astratti

- Classi e metodi astratti sono definiti tali mediante la parola chiave **abstract**
- Un **metodo astratto** è un metodo per il quale non è specificata alcuna implementazione
- Una **classe** deve obbligatoriamente essere definita **abstract** se contiene *almeno* un metodo astratto (o se ne eredita uno senza implementarlo)
- 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

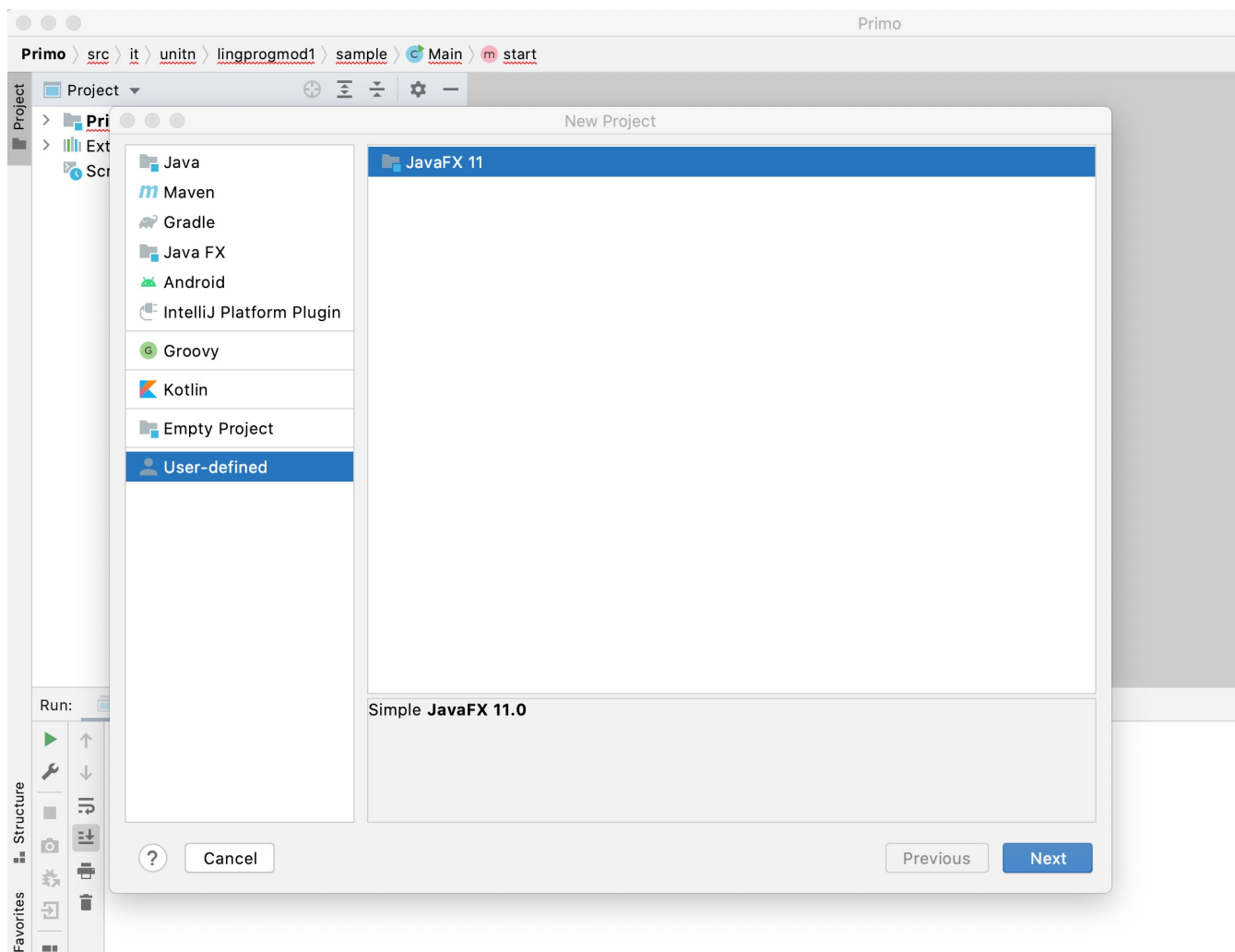
```
abstract public class ArrayDati {  
    int size;  
    int defaultGrowthSize;  
    int marker;  
    int contenuto[];  
    abstract public int estrai();  
    ... // implementazione altri metodi  
}  
  
public class Pila extends ArrayDati {  
    public int estrai() { ... }  
}  
  
public class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



Palestra di Java con la grafica:

Java FX - parte 1

Creazione di una applicazione JavaFX



vedi video tutorial sulla configurazione di IntelliJ !

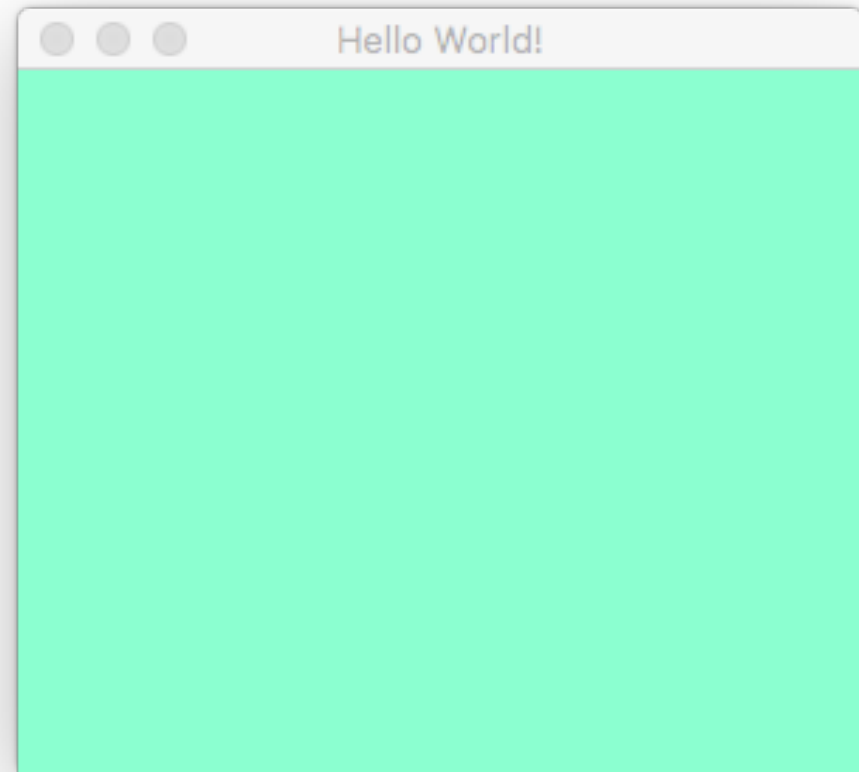
Java FX

```
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);  
    }  
}
```

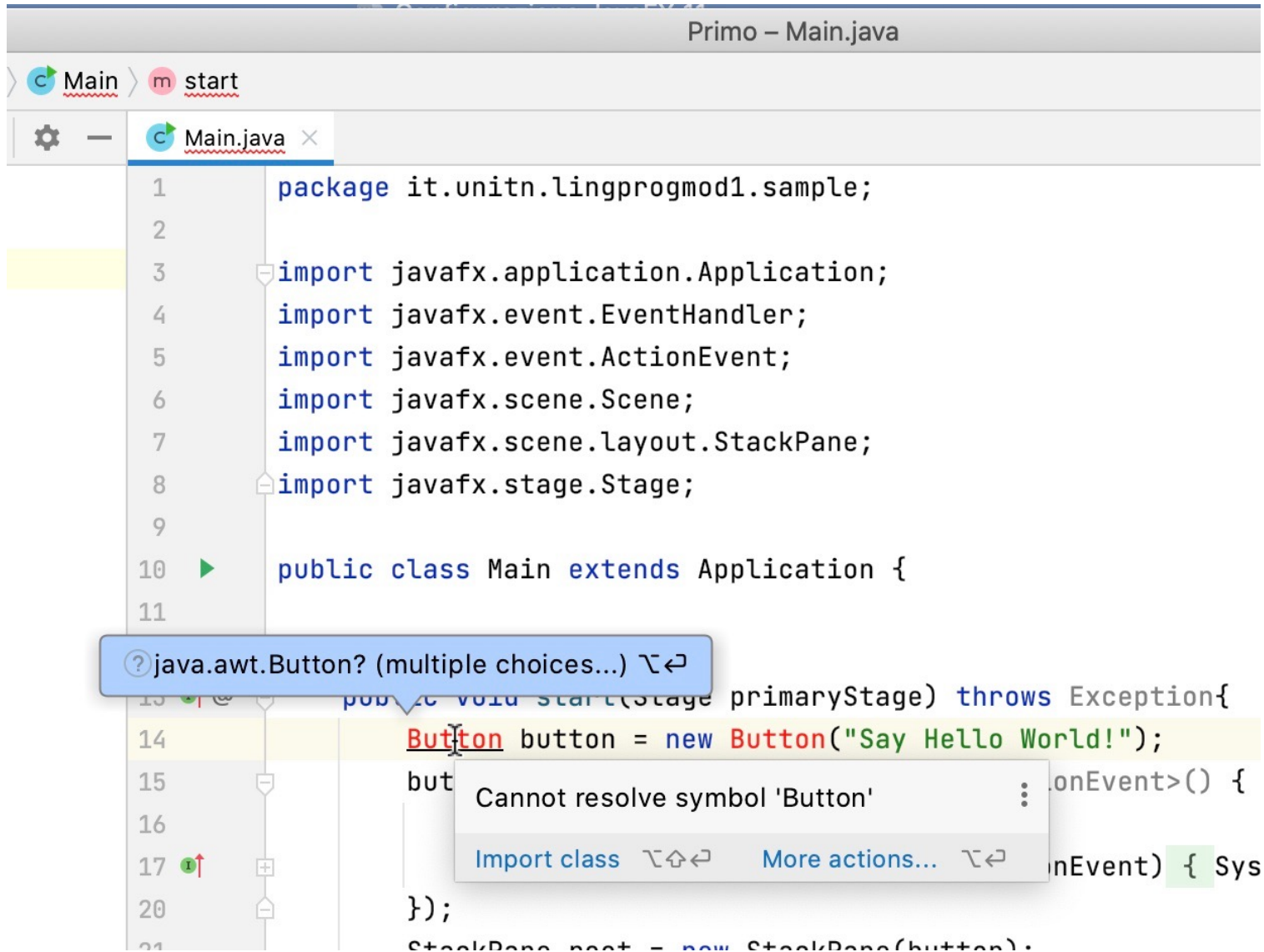


Java FX

```
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!



```
Primo – Main.java
Main start
Main.java
1 package it.unitn.lingprogmod1.sample;
2
3 import javafx.application.Application;
4 import javafx.event.EventHandler;
5 import javafx.event.ActionEvent;
6 import javafx.scene.Scene;
7 import javafx.scene.layout.StackPane;
8 import javafx.stage.Stage;
9
10 public class Main extends Application {
11
12     public void start(Stage primaryStage) throws Exception{
13
14         Button button = new Button("Say Hello World!");
15         button.setOnAction(new EventHandler() {
16             @Override public void handle(ActionEvent event) { Sys
17
18         });
19         StackPane root = new StackPane(button);
```

? java.awt.Button? (multiple choices...) ↵↵

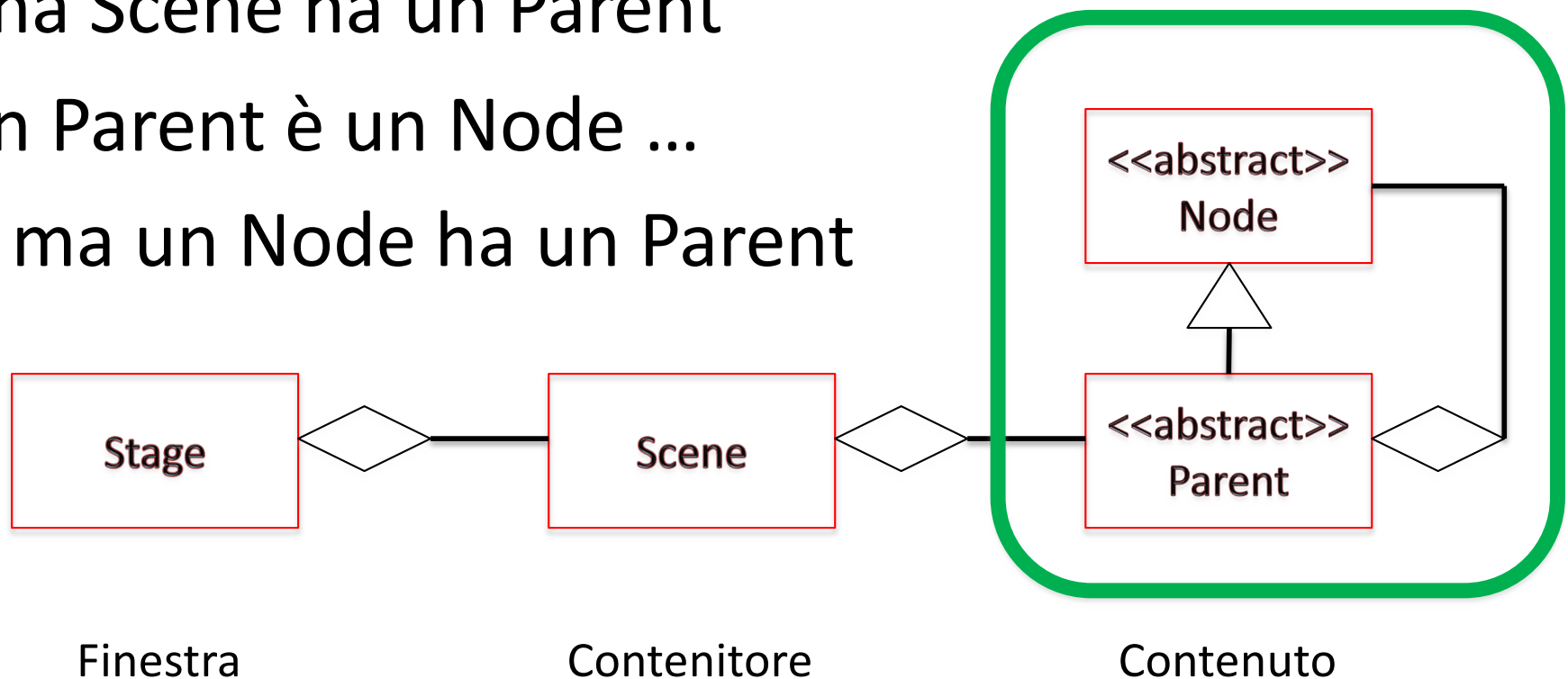
Cannot resolve symbol 'Button' ⋮

Import class ↵↵↵ More actions... ↵↵

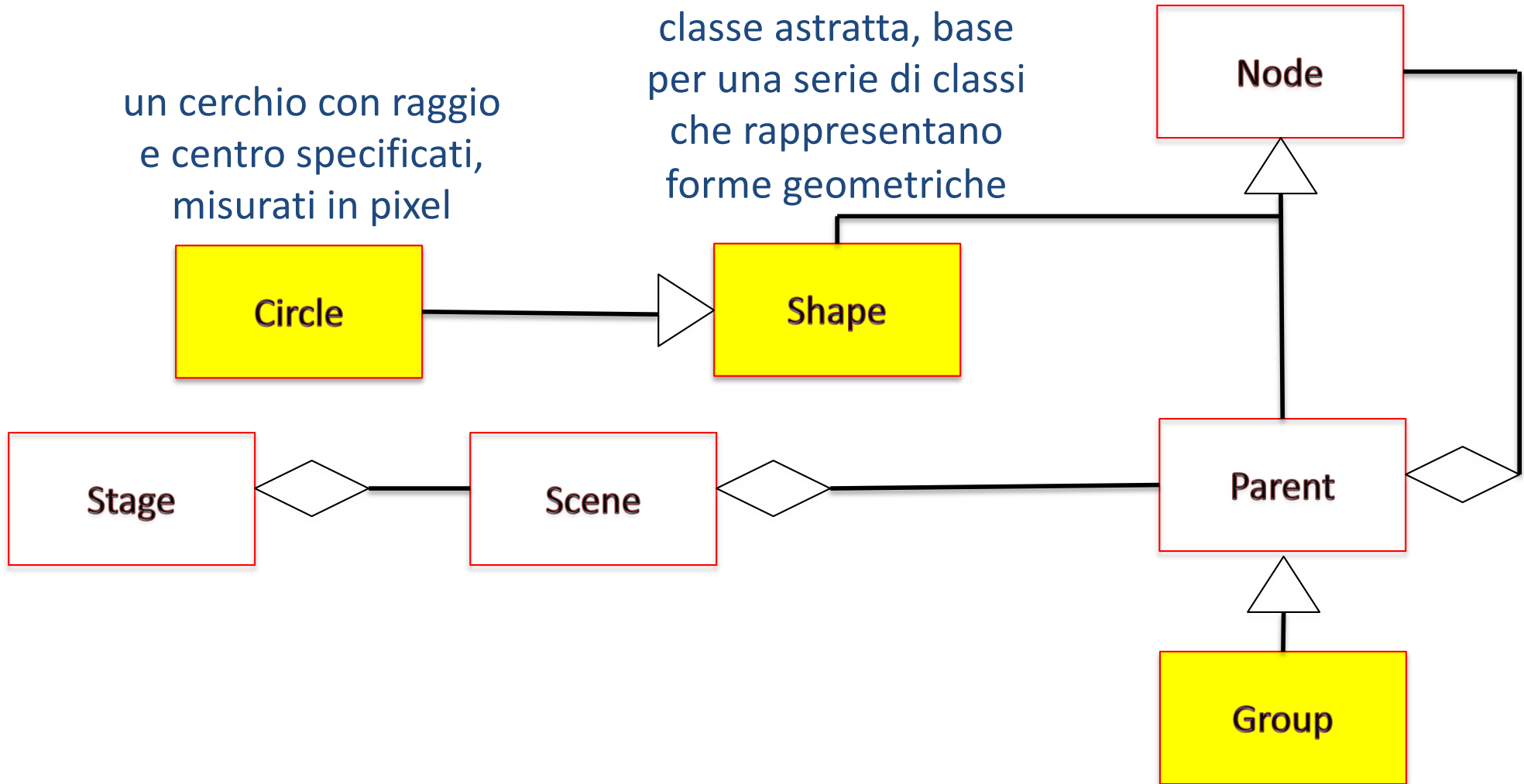
Stage/Scene/Parent/Node

Stage = “finestra”

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

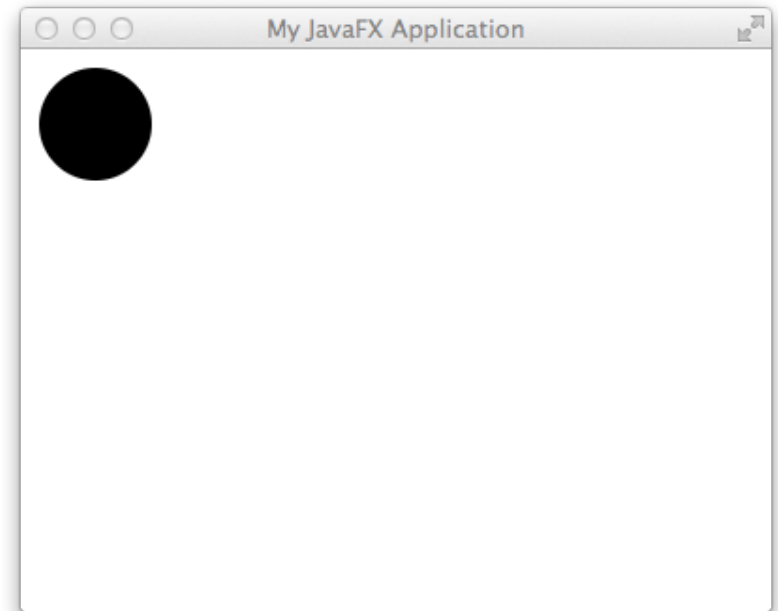


Group – Shape - Circle



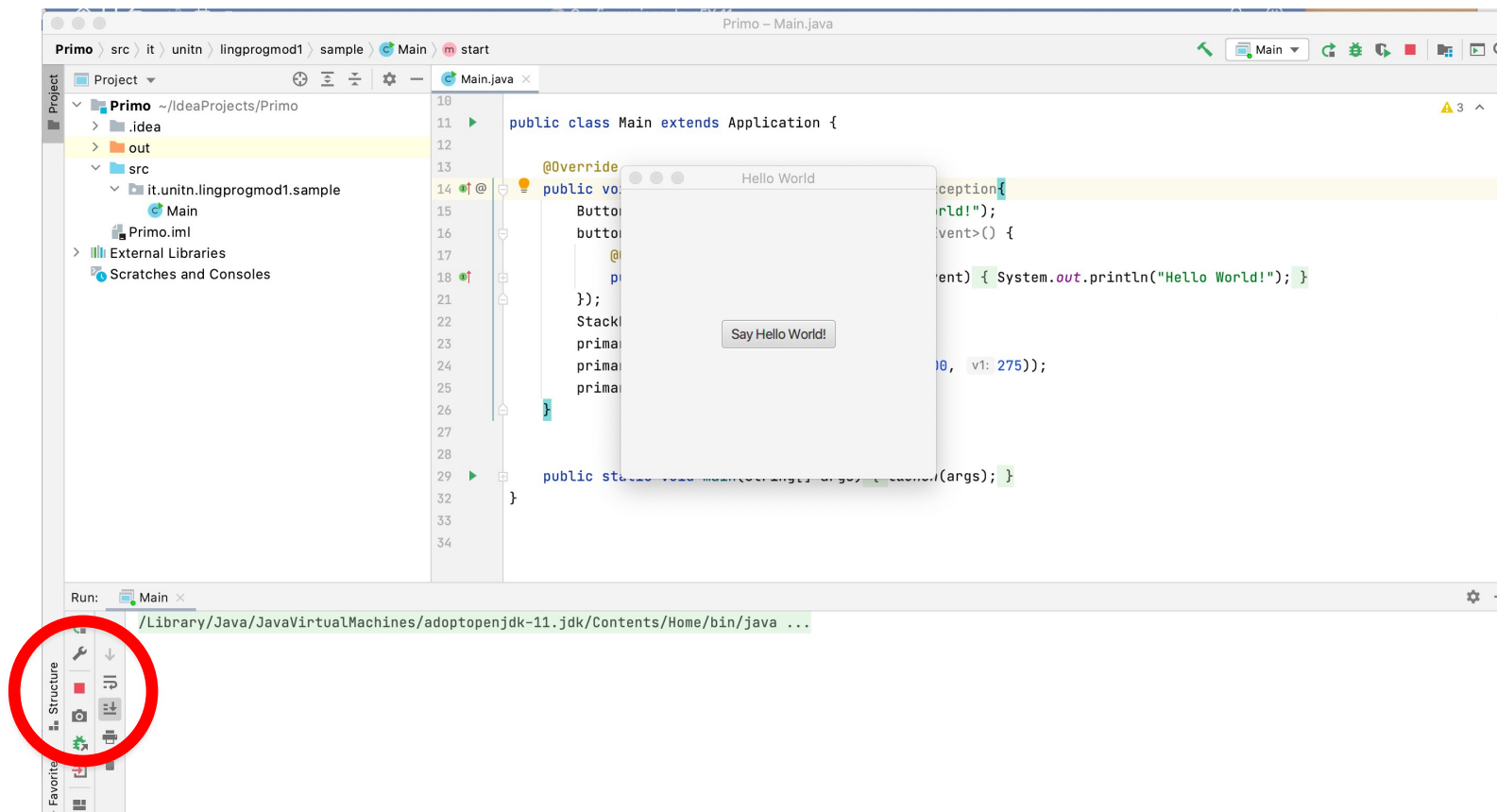
Applicazione minima

```
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);
    }
}
```



Quando termina l'esecuzione?

- Il *processo* associato a un'applicazione JavaFX rimane attivo anche dopo la fine di **start()**
 - l'applicazione va esplicitamente terminata
 - 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");
```

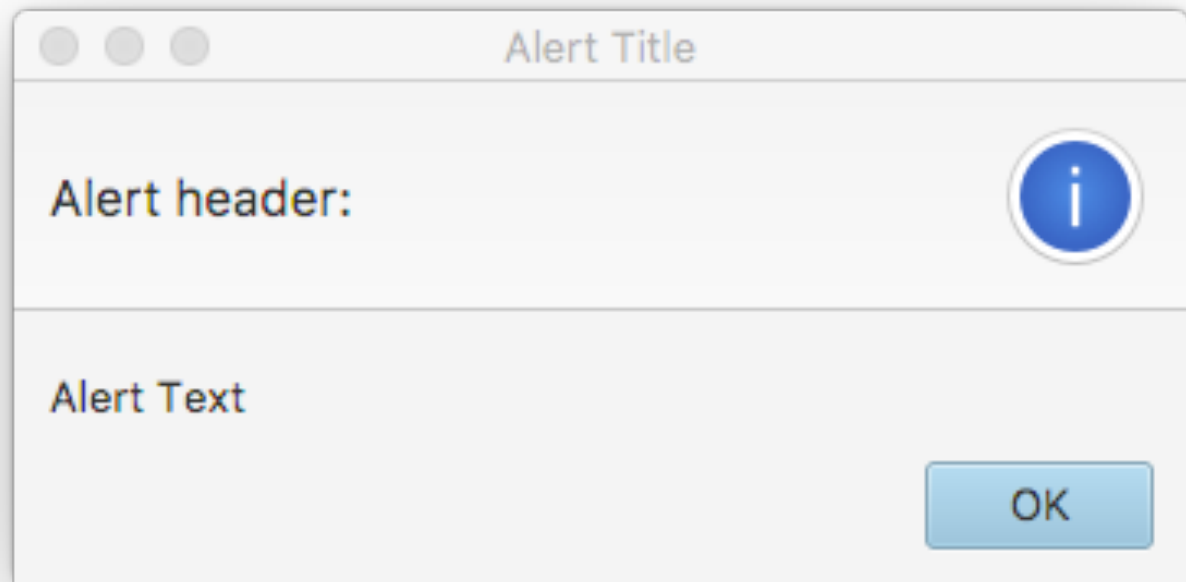
```
String inputString = scanner.nextLine();
```

```
...
```

```
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...



Esercizio

Esercizio

Giocate con il codice di JavaFX.

- Cercate di scoprire come si possa, invece del cerchio, disegnare un rettangolo o altre figure (suggerimento: esplorate le sottoclassi di Shape).