Grunnleggende JavaFX

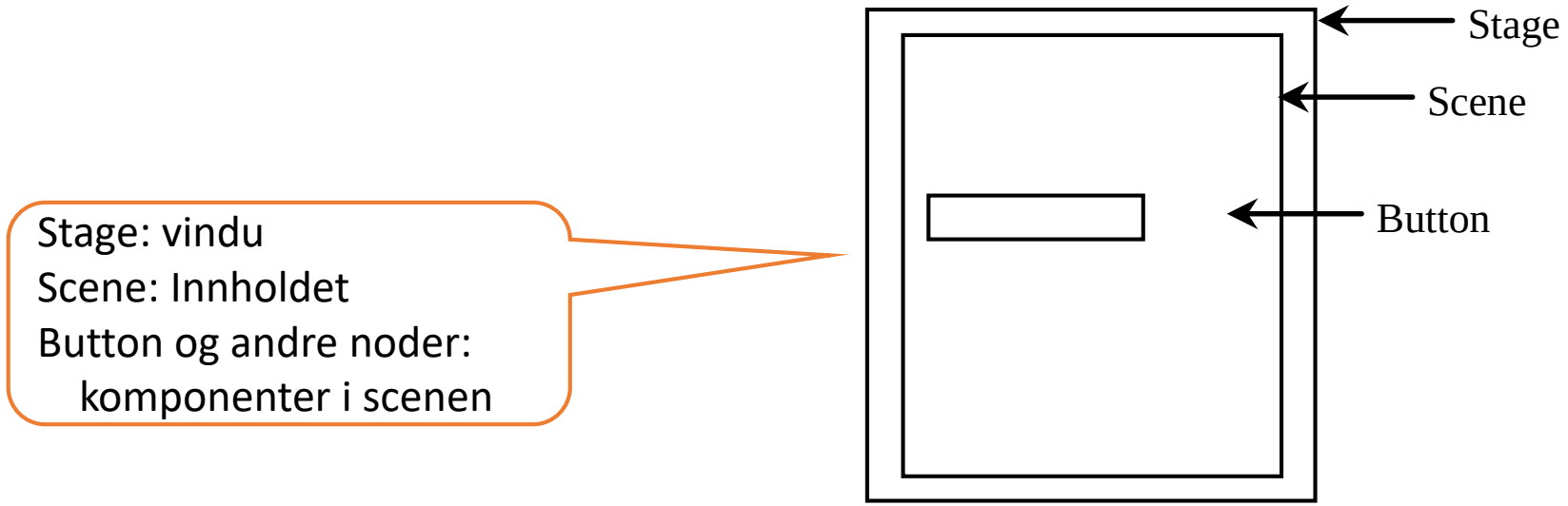
- JavaFX – Swing – AWT
- Stage – scene – node
- Properties – egenskaper
- Farger, fonter og bilder
- Pane – paneler for å strukturere layout
- Vise tekst og figurer

AWT -> Swing -> JavaFX

- Java kom på tidlig 90-tall med AWT. Den baserte mye av jobben på plattform-avhengige deler. Altså ulik kode på Windows, Linux, Mac, .. Ikke egnet for avanserte brukergrensesnitt.
- Swing tok over på slutten av 1990-tallet. «Lightweight» system, baserte seg i mindre grad på underliggende plattform. I starten problemer med å kombinere AWT og Swing, ble løst etter hvert.
- JavaFX kom ca. 2008. Mer moderne, støtter «multitouch» på berøringsskjermer. 2D og 3D støtte, animasjon, lyd og video.

Lage JavaFX applikasjon

- Lag klasse som arver fra javafx.application.Application
- Reimplementer start-metode:
  `public void start(Stage primaryStage){...}`
- Komponenter: Stage, Scene, Node



Hello world

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;

public class MyJavaFX extends Application {
  @Override
  public void start(Stage primaryStage) {
    // Create a button and place it in the scene
    Button btHW = new Button("Hello World");
    Scene scene = new Scene(btHW, 200, 250);
    primaryStage.setTitle("MyJavaFX"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
```

```
  /**
   * The main method is only needed for
   * the IDE with limited JavaFX support.
   * Not needed for running from the
   * command line.
   */
  public static void main(String[] args) {
    launch(args);
  }
} // end class MyJavaFX
```

launch-metoden oppretter et MyJavaFX-objekt, oppretter et Stage-objekt (primaryStage), og kaller start-metoden

Flere vinduer

```
public class MultipleStageDemo extends
Application {
  @Override
  public void start(Stage primaryStage) {
    Scene scene = new Scene(
      new Button("OK"), 200, 250);
    primaryStage.setTitle("MyJavaFX");
    primaryStage.setScene(scene);
    primaryStage.show();
    Stage stage = new Stage();
    stage.setTitle("Second Stage");
    stage.setScene(new Scene(
      new Button("New Stage"), 200, 250));
    stage.show();
  }
```

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

Vis første vindu

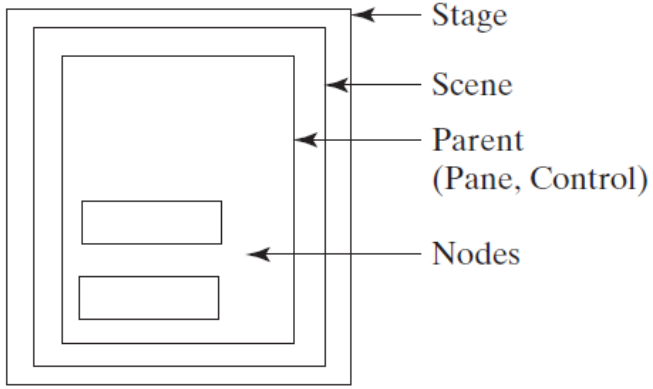
Vis andre vindu

Bestemme layout med «pane» - panel

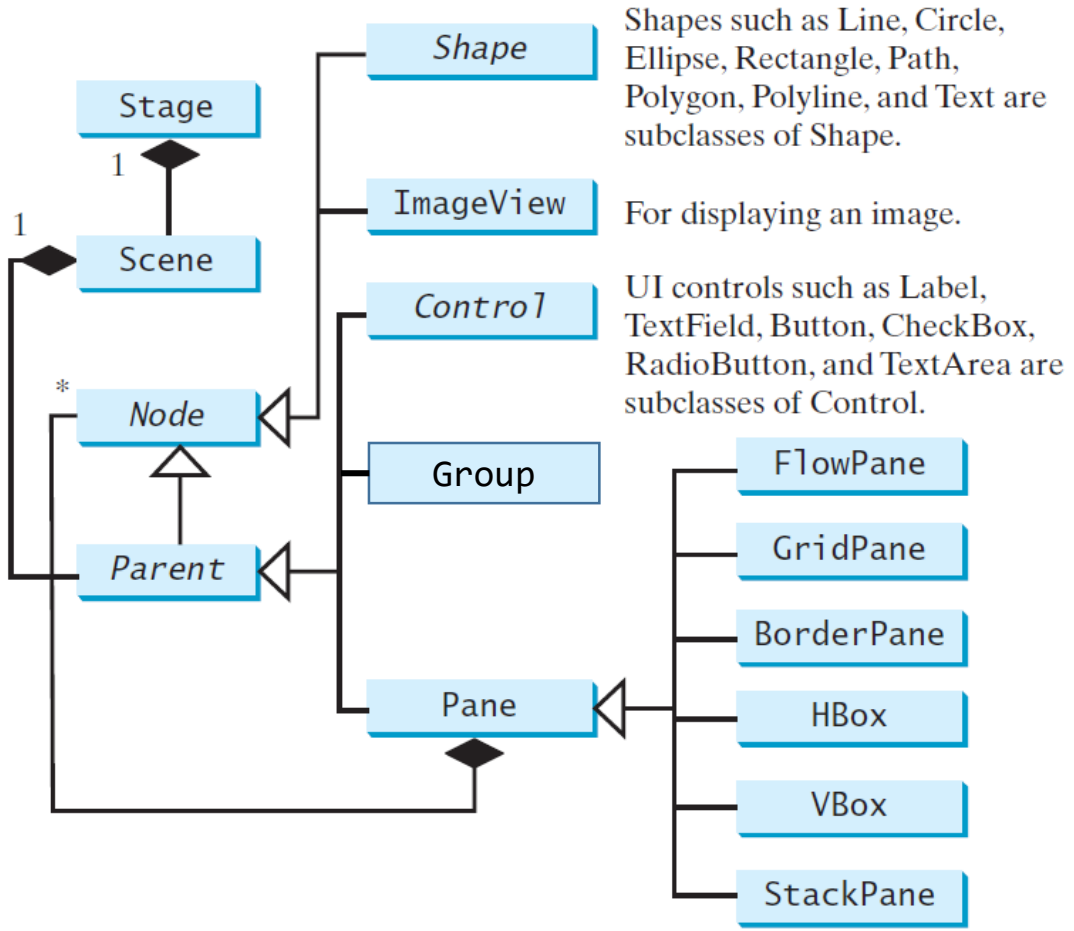
- Pane: Container (beholder) som kan holde på andre noder.
- Shape: Tekst og figurer som linje, sirkel, ellipse, rektangel, polygon, ...
- UI control: trykknapper, avkrysningsbokser, tekstfelt, ...
- Group: Beholder som inneholder noder.
- ImageView: For å vise bilder.

Alle disse fem (over) er selv noder!

Noder



(a)



(b)

Åpne piler viser subklassing.
◆ viser objektrelasjon (komposisjon)
En stage har én eller flere scener.
En scene har én eller flere Parent. En Pane har én eller flere Noder.

Tegn av figuren, men tegn *bare* klassehierarkiet, med superklassen øverst.

ButtonInPane

```
public class ButtonInPane extends Application {
  @Override
  public void start(Stage primaryStage) {
    StackPane pane = new StackPane();
    pane.getChildren().add(new Button("OK"));
    Scene scene = new Scene(pane, 200, 50);
    primaryStage.setTitle("Button in a pane");
    primaryStage.setScene(scene);
    primaryStage.show();
  }

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

Button i Pane,
Pane i Scene,
Scene i Stage!

ShowCircle

Vanlig «data-koordinatsystem» med (0,0) oppe til venstre. X mot høyre og Y nedover.


```
public class ShowCircle extends Application {
  @Override
  public void start(Stage primaryStage) {
    Circle circle = new Circle();
    circle.setCenterX(100);
    circle.setCenterY(100);
    circle.setRadius(50);
    circle.setStroke(Color.BLACK);
    circle.setFill(Color.WHITE);

    Pane pane = new Pane();
    pane.getChildren().add(circle);

    Scene scene = new Scene(pane, 200, 200);
    primaryStage.setTitle("ShowCircle");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
  // Og main-metode
}
```

Property Binding

- Introdusert i JavaFX
- Asymetrisk kobling mellom to objekter
- Om det ene objektet (source / bindable / observable object) endres, vil det andre (target / binding object, binding property) automatisk endre seg
- Koordinatene i ShowCircle tilsier at sirkelen tegnes midt i vinduet. Dersom vinduet resizes, vil ikke lenger sirkelen være i midten. Løses med property binding – kall på bind-metoden:
- `target.bind(source);`
- Mange slike «property» kan brukes både som target og source

```
Circle circle = new Circle();
circle.centerXProperty().bind(pane.widthProperty().divide(2));
circle.centerYProperty().bind(pane.heightProperty().divide(2));
```

Property Binding II

- Til mange egenskaper (som centerX) er det altså knyttet tre metoder:
- Setter-metode: void setCenterX(double)
- Getter-metoder: double getCenterX()
- Property-metode: DoubleProperty centerXProperty()

- Numeriske egenskaper (som centerX, width og height) har metoder for aritmetikk:
- add, subtract, divide, multiply.

```
public class BindingDemo {
  public static void main(String[] args) {
    DoubleProperty d1 = new SimpleDoubleProperty(1);
    DoubleProperty d2 = new SimpleDoubleProperty(2);
    d1.bind(d2); // Bind d1 with d2
    System.out.println("d1 is " + d1.getValue()
      + " and d2 is " + d2.getValue());
    d2.setValue(70.2);
    System.out.println("d1 is " + d1.getValue()
      + " and d2 is " + d2.getValue());
  }
}
```

Bidirectional – tovegs – binding

- To egenskaper kan bindes tovegs, symmetrisk.
- Endres den ene (uansett hvilken) så vil den andre endre seg.

```
public class BidirectionalBindingDemo {
  public static void main(String[] args) {
    DoubleProperty d1 = new SimpleDoubleProperty(1);
    DoubleProperty d2 = new SimpleDoubleProperty(2);
    d1.bindBidirectional(d2);  // her vil d1 bli lik d2.
    System.out.println("d1 is " + d1.getValue()
      + " and d2 is " + d2.getValue());
    d1.setValue(50.1);              // og dermed d2 også
    System.out.println("d1 is " + d1.getValue()
      + " and d2 is " + d2.getValue());
    d2.setValue(70.2);              // og dermed d1 også
    System.out.println("d1 is " + d1.getValue()
      + " and d2 is " + d2.getValue());
  }
}
```

Node – setStyle og rotate

- Noder kan «styles» med CSS-syntaks:

```
circle.setStyle("-fx-stroke: black; -fx-fill: red;");
```

- Den tilsvarer:

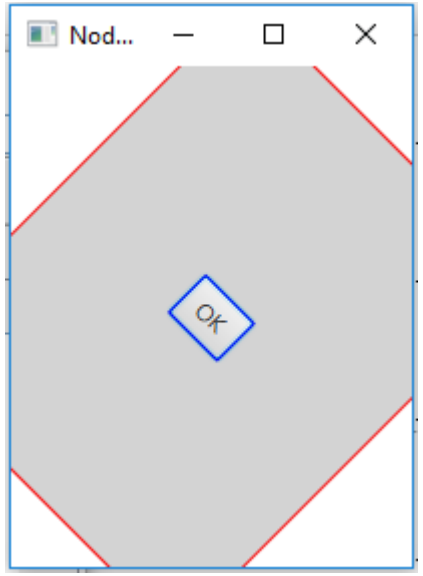
```
circle.setStroke(Color.BLACK);
circle.setFill(Color.RED);
```

- Ved feil i style-angivelsen vil den bare bli ignorert
- Noder kan roteres med rotate-egenskapen, vinkel oppgis i *grader*!

```
button.setRotate(80);
```

NodeStyleRotateDemo

```
@Override
 public void start(Stage primaryStage) {
   StackPane pane = new StackPane();
   Button btOK = new Button("OK");
   btOK.setStyle("-fx-border-color: blue;");
   pane.getChildren().add(btOK);

   pane.setRotate(45);
   pane.setStyle("-fx-border-color: red; -fx-background-color: lightgray;");

   Scene scene = new Scene(pane, 200, 250);
   primaryStage.setTitle("NodeStyleRotateDemo");
   primaryStage.setScene(scene);
   primaryStage.show();
 }
```



Color-klassa (javafx.scene.paint.Color)

Opacity – det motsatte av gjennomsiktighet. Immutable klasse.

The getter methods for property values are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.paint.Color

```
-red: double
-green: double
-blue: double
-opacity: double
```

```
+Color(r: double, g: double, b:
   double, opacity: double)
+brighter(): Color
+darker(): Color
+color(r: double, g: double, b:
   double): Color
+color(r: double, g: double, b:
   double, opacity: double): Color
+rgb(r: int, g: int, b: int):
   Color
+rgb(r: int, g: int, b: int,
   opacity: double): Color
```

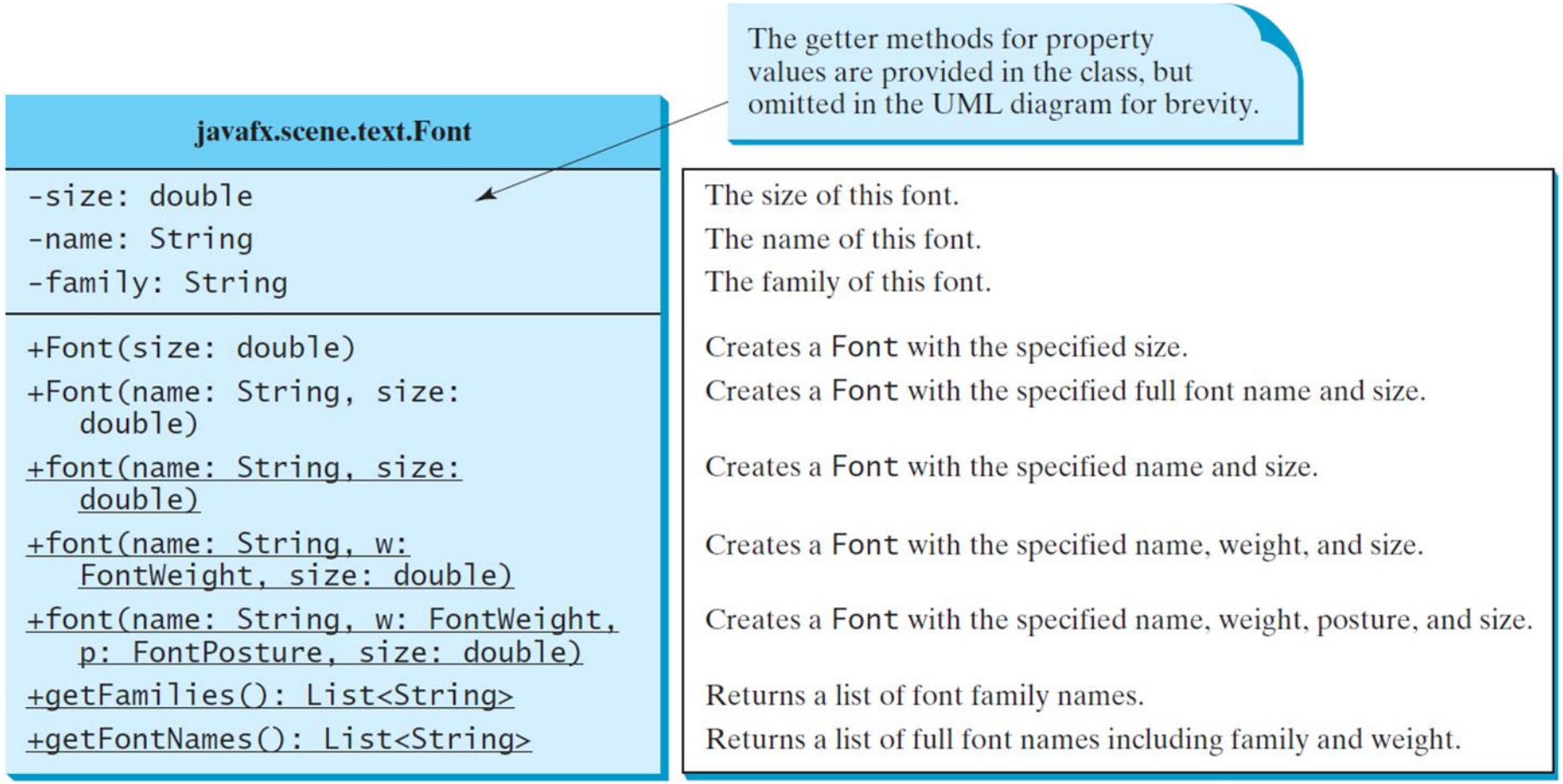
-red: double	The red value of this Color (between 0.0 and 1.0).
-green: double	The green value of this Color (between 0.0 and 1.0).
-blue: double	The blue value of this Color (between 0.0 and 1.0).
-opacity: double	The opacity of this Color (between 0.0 and 1.0).
+Color(r: double, g: double, b: double, opacity: double)	Creates a Color with the specified red, green, blue, and opacity values.
+brighter(): Color	Creates a Color that is a brighter version of this Color.
+darker(): Color	Creates a Color that is a darker version of this Color.
+color(r: double, g: double, b: double): Color	Creates an opaque Color with the specified red, green, and blue values.
+color(r: double, g: double, b: double, opacity: double): Color	Creates a Color with the specified red, green, blue, and opacity values.
+rgb(r: int, g: int, b: int): Color	Creates a Color with the specified red, green, and blue values in the range from 0 to 255.
+rgb(r: int, g: int, b: int, opacity: double): Color	Creates a Color with the specified red, green, and blue values in the range from 0 to 255 and a given opacity.

Konstanter:
Color.BLACK
Color.RED
Color.BLUE
Color.GREEN
Color.GRAY
Color.WHITE
og mange fler!

Font klassa (javaFfx.scene.text.Font)

- Bestemmer fontnavn (familie), vekt, holdning og størrelse (f.eks. *Calibri, normal, kursiv, 20pt*)
- Brukes når tekst skal vises i GUI
- Liste av fontnavn skaffes med: `List<String> Font.getFontNames()`
- Holdninger (postures): FontPosture.REGULAR og FontPostur.ITALIC
- Vekter: FontWeight.THIN, FontWeight.EXTRA_LIGHT, FontWeight.LIGHT, FontWeight.NORMAL, FontWeight.MEDIUM, FontWeight.SEMI_BOLD, FontWeight.BOLD, FontWeight.EXTRA_BOLD, FontWeight.BLACK
- FontPosture og FontWeight er *enum-typer*. (Oppramstyper)

Font klassa


The getter methods for property values are provided in the class, but omitted in the UML diagram for brevity.
javafx.scene.text.Font
-size: double
-name: String
-family: String
+Font(size: double)
+Font(name: String, size: double)
+font(name: String, size: double)
+font(name: String, w: FontWeight, size: double)
+font(name: String, w: FontWeight, p: FontPosture, size: double)
+getFamilies(): List<String>
+getFontNames(): List<String>
The size of this font.
The name of this font.
The family of this font.
Creates a Font with the specified size.
Creates a Font with the specified full font name and size.
Creates a Font with the specified name and size.
Creates a Font with the specified name, weight, and size.
Creates a Font with the specified name, weight, posture, and size.
Returns a list of font family names.
Returns a list of full font names including family and weight.

class FontDemo

```
// Create a label and set its properties
 Label label = new Label("JavaFX");
 label.setFont(Font.font("Times New Roman",
    FontWeight.BOLD, FontPosture.ITALIC, 20));
 pane.getChildren().add(label);
```

Label:
lite tekstfelt,
en nodetype

Resten av klassa er
standard, se læreboka

Image & ImageView

- Image (javafx.scene.image.Image) representerer et bilde internt

```
Image image = new Image(filnavn);
Image image = new Image(URL);
```

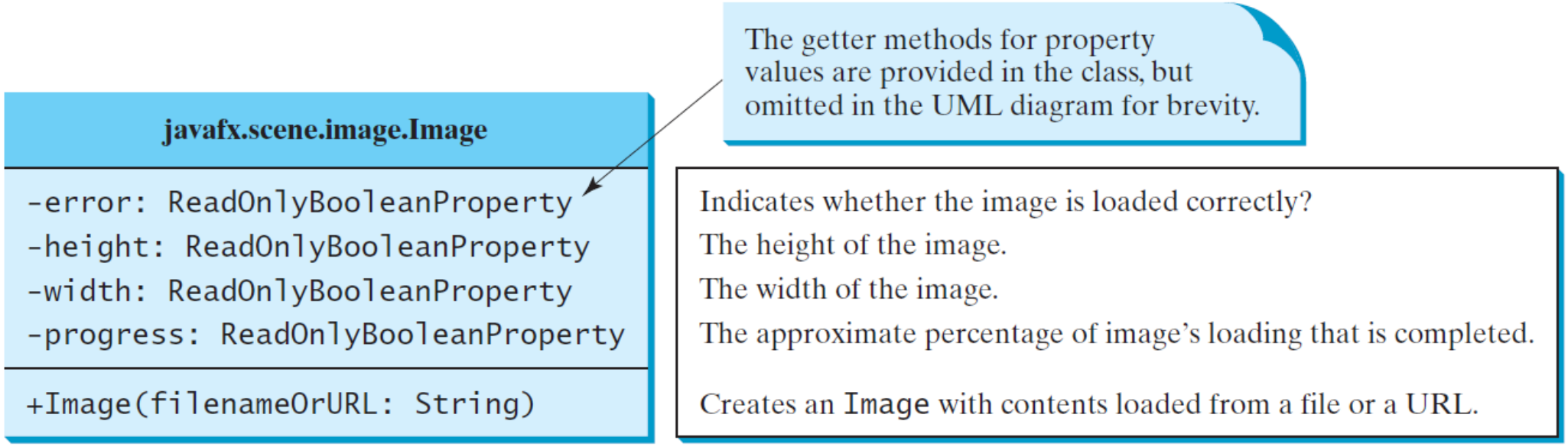
- ImageView (javafx.scene.image.ImageView) er en Node som kan vises.
- ImageView kan lages fra Image:

```
ImageView imageView = new ImageView(image);
```

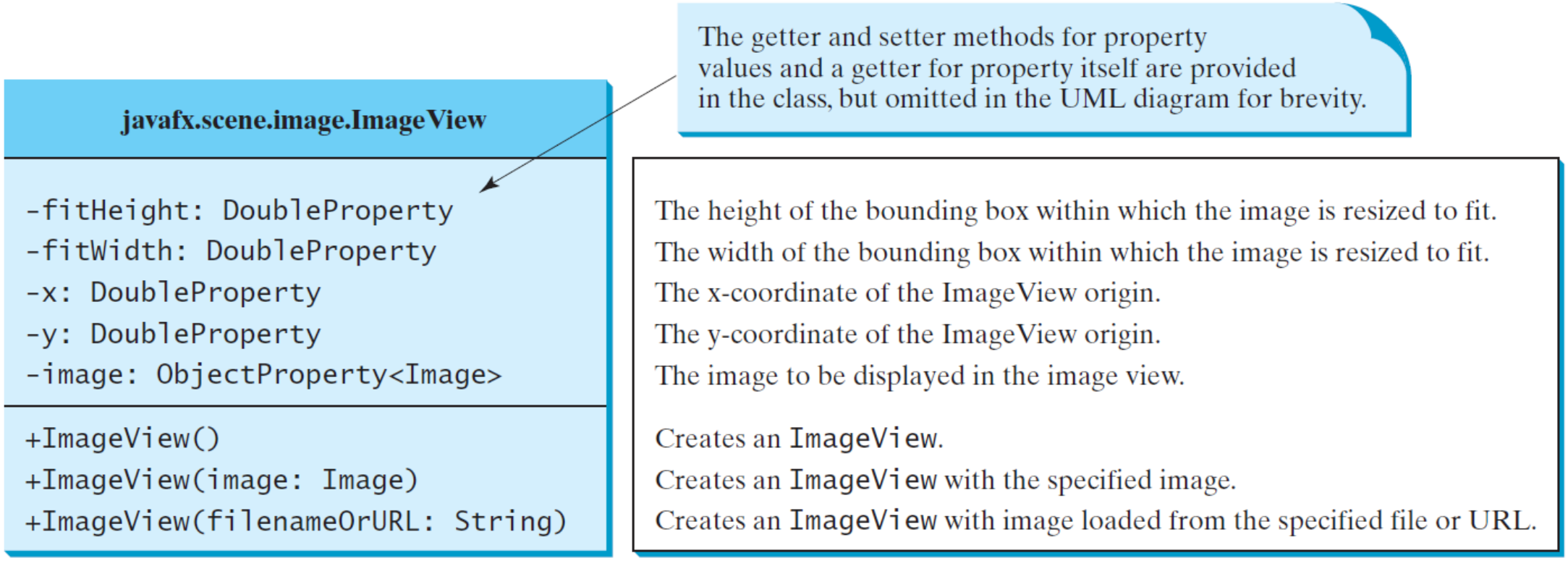
- eller direkte fra fil eller URL:

```
ImageView imageView = new ImageView(URL);
```

class Image


javafx.scene.image.Image
-error: ReadOnlyBooleanProperty
-height: ReadOnlyBooleanProperty
-width: ReadOnlyBooleanProperty
-progress: ReadOnlyBooleanProperty
+Image(filenameOrURL: String)
The getter methods for property values are provided in the class, but omitted in the UML diagram for brevity.
Indicates whether the image is loaded correctly?
The height of the image.
The width of the image.
The approximate percentage of image's loading that is completed.
Creates an Image with contents loaded from a file or a URL.

class ImageView



ShowImage demo


```
@Override // Override the start method in the Application class
 public void start(Stage primaryStage) {
  // Create a pane to hold the image views
  Pane pane = new HBox(10);              // Noder legges ut Horisontalt
  pane.setPadding(new Insets(5, 5, 5, 5));
  Image image = new Image("https://home.usn.no/lonnesta/Tor_Lonnestad.jpg");
  pane.getChildren().add(new ImageView(image));

  ImageView imageView2 = new ImageView(image);
  imageView2.setFitHeight(100);
  imageView2.setFitWidth(100);
  pane.getChildren().add(imageView2);

  ImageView imageView3 = new ImageView(image);
  imageView3.setRotate(90);
  pane.getChildren().add(imageView3);

  // Create a scene and place it in the stage
  Scene scene = new Scene(pane);
  primaryStage.setTitle("ShowImage"); // Set the stage title
  primaryStage.setScene(scene); // Place the scene in the stage
  primaryStage.show(); // Display the stage
 }
```

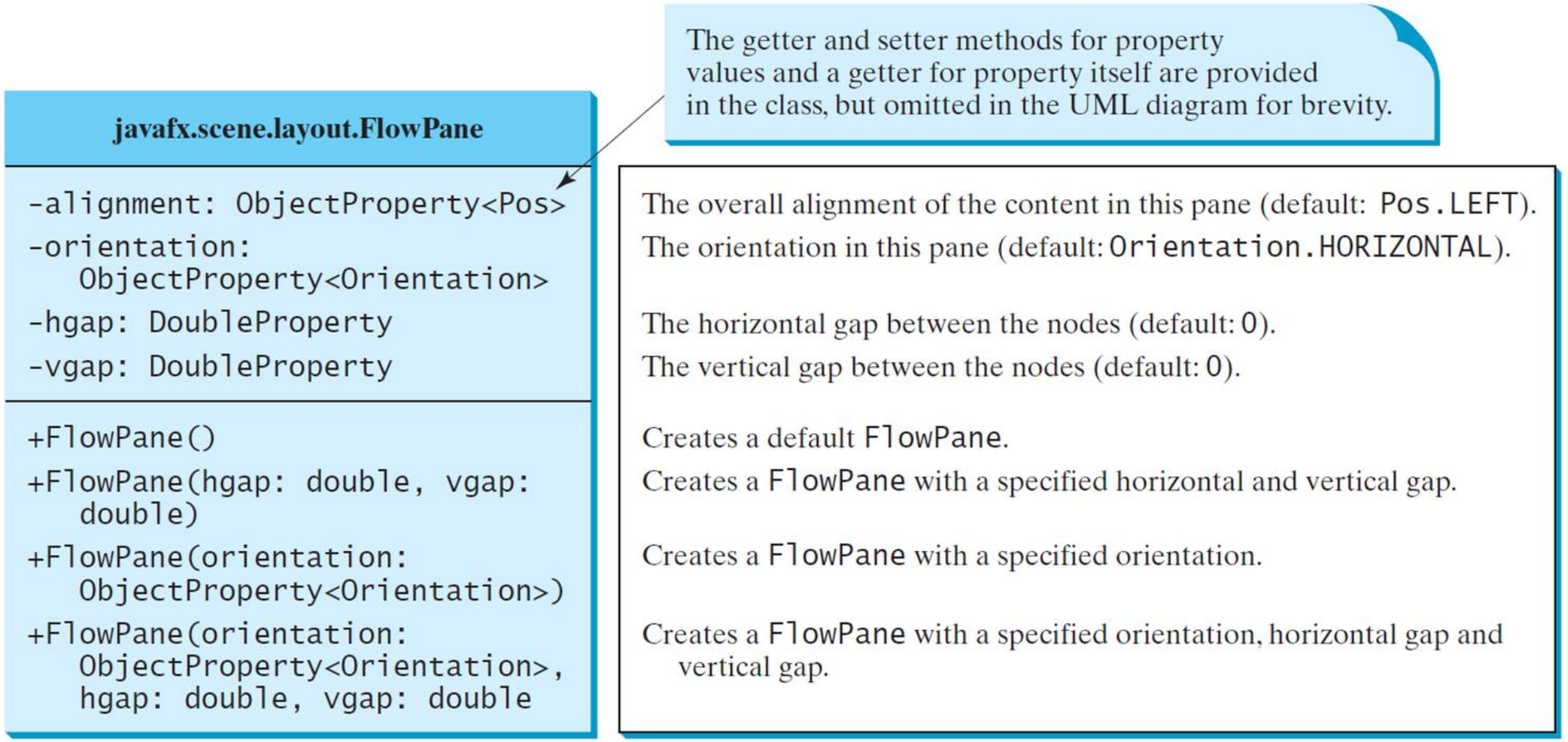
Insets: marger

Pane – beholder som inneholder noder

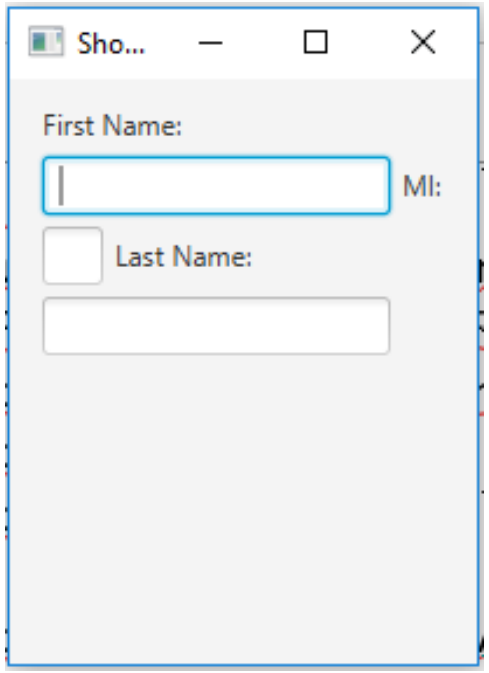
Class	*Description*
`Pane`	Base class for layout panes. It contains the `getChildren()` method for returning a list of nodes in the pane.
`StackPane`	Places the nodes on top of each other in the center of the pane.
`FlowPane`	Places the nodes row-by-row horizontally or column-by-column vertically.
`GridPane`	Places the nodes in the cells in a two-dimensional grid.
`BorderPane`	Places the nodes in the top, right, bottom, left, and center regions.
`HBox`	Places the nodes in a single row.
`VBox`	Places the nodes in a single column.

FlowPane

Nodene legges «linjevis» eller «kolonnevis», avhengig av «orientation»



FlowPane demo

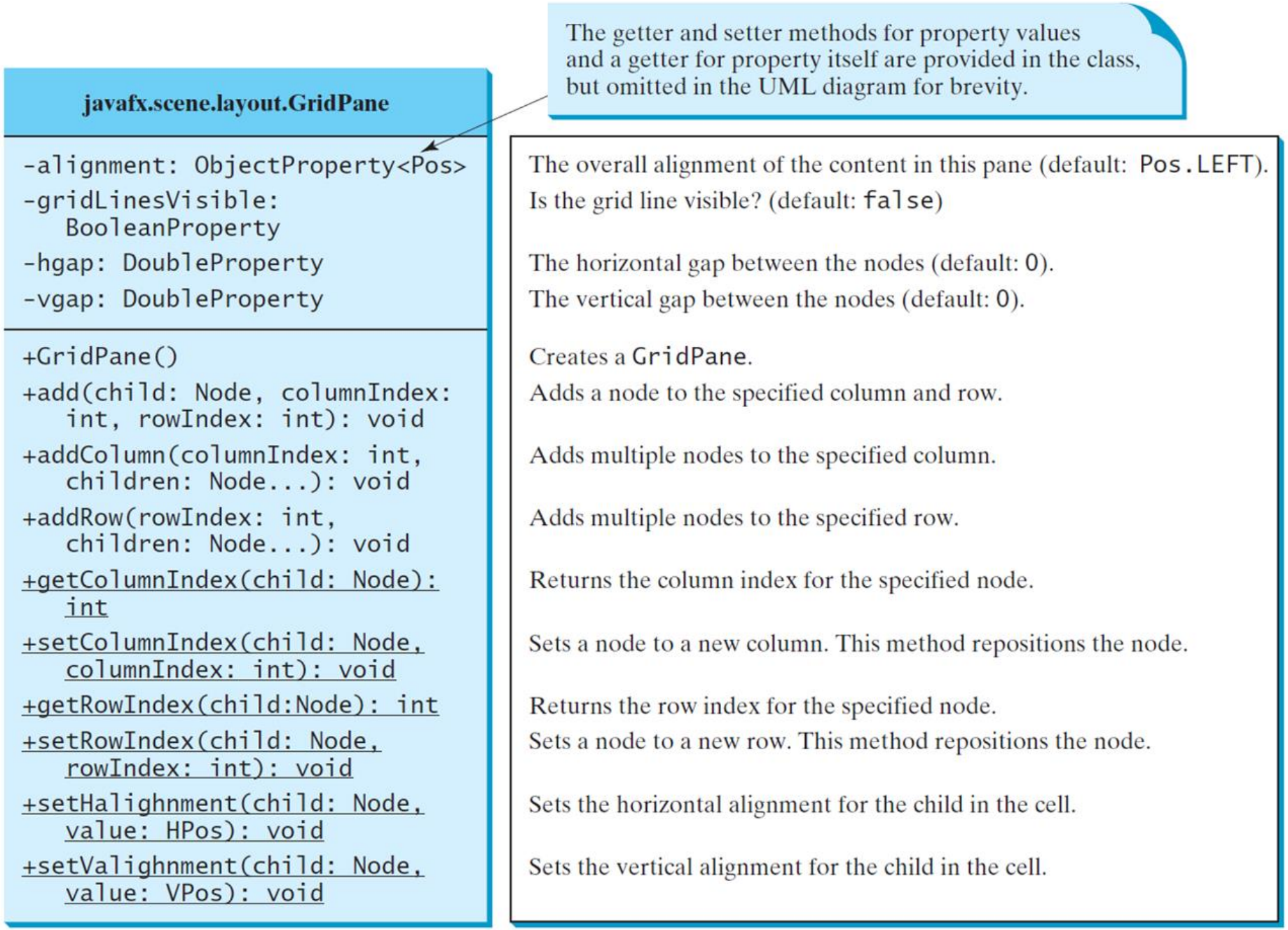


Noder plassert linjevis

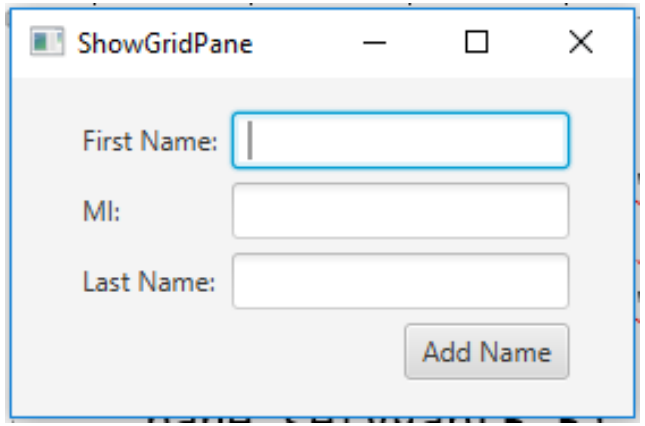
```
@Override
  public void start(Stage primaryStage) {
    FlowPane pane = new FlowPane();
    pane.setPadding(new Insets(11, 12, 13, 14));
    pane.setHgap(5);  // Avstand mellom noder
    pane.setVgap(5);

    pane.getChildren().addAll(new Label("First Name:"),
      new TextField(), new Label("MI:"));
    TextField tfMi = new TextField();
    tfMi.setPrefColumnCount(1);
    pane.getChildren().addAll(tfMi, new Label("Last Name:"),
      new TextField());

    Scene scene = new Scene(pane, 200, 250);
    primaryStage.setTitle("ShowFlowPane");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
```

GridPane

Plasserer nodene i en matriseform



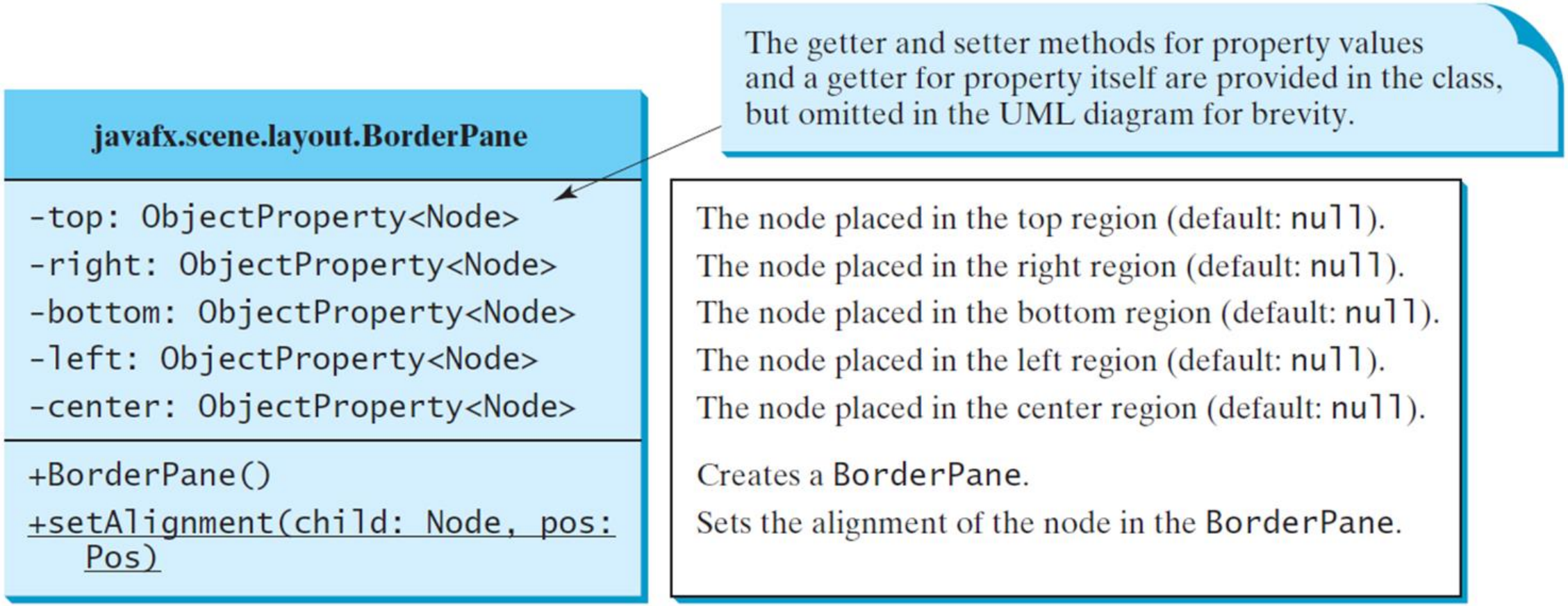
GridPane demo



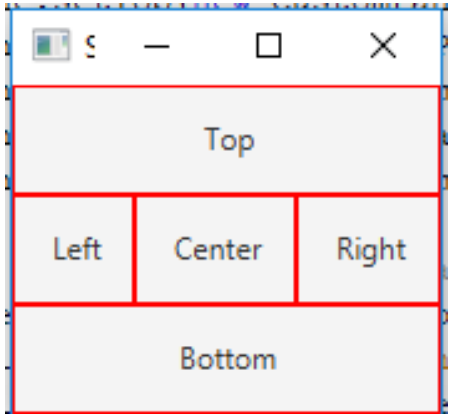

```
@Override
public void start(Stage primaryStage) {
  GridPane pane = new GridPane();
  pane.setAlignment(Pos.CENTER);
  pane.setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
  pane.setHgap(5.5);
  pane.setVgap(5.5);

  pane.add(new Label("First Name:"), 0, 0);
  pane.add(new TextField(), 1, 0);
  pane.add(new Label("MI:"), 0, 1);
  pane.add(new TextField(), 1, 1);
  pane.add(new Label("Last Name:"), 0, 2);
  pane.add(new TextField(), 1, 2);
  Button btAdd = new Button("Add Name");
  pane.add(btAdd, 1, 3);
  GridPane.setHalignment(btAdd, HPos.RIGHT);

  Scene scene = new Scene(pane);
  primaryStage.setTitle("ShowGridPane");
  primaryStage.setScene(scene);
  primaryStage.show();
}
```

BorderPane

Plasserer inntil fem noder i topp, bunn, venstre, høyre og senter.



BorderPane Demo

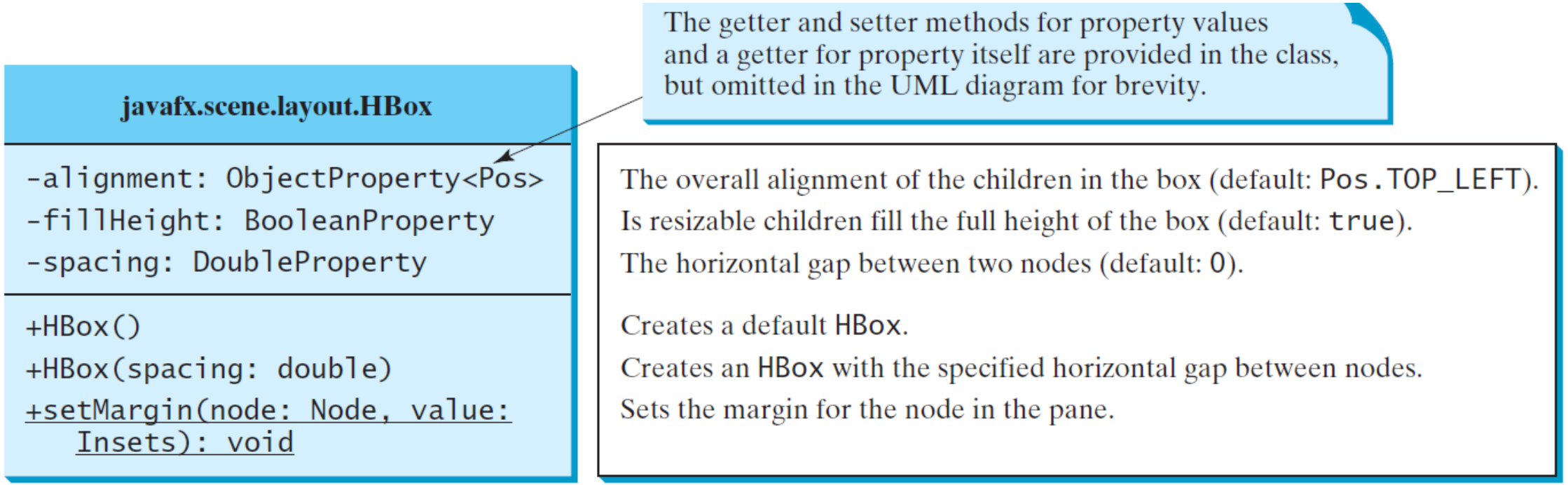



```
@Override
  public void start(Stage primaryStage) {
    BorderPane pane = new BorderPane();

   pane.setTop(new CustomPane("Top"));
    pane.setRight(new CustomPane("Right"));
    pane.setBottom(new CustomPane("Bottom"));
    pane.setLeft(new CustomPane("Left"));
    pane.setCenter(new CustomPane("Center"));

    Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowBorderPane");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
```

```
class CustomPane extends StackPane {
  public CustomPane(String title) {
    getChildren().add(new Label(title));
    setStyle("-fx-border-color: red");
    setPadding(new Insets(11.5, 12.5, 13.5,14.5));
  }
}
```

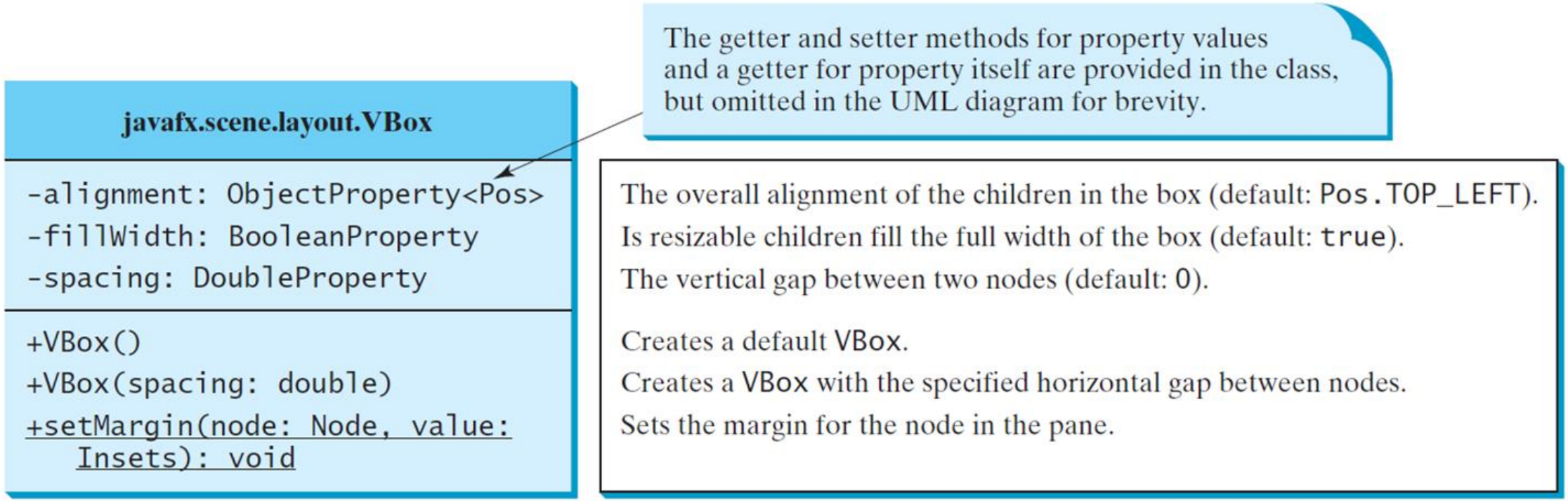
HBox

Plasserer alle noder etter hverandre horisontalt i bare én rad



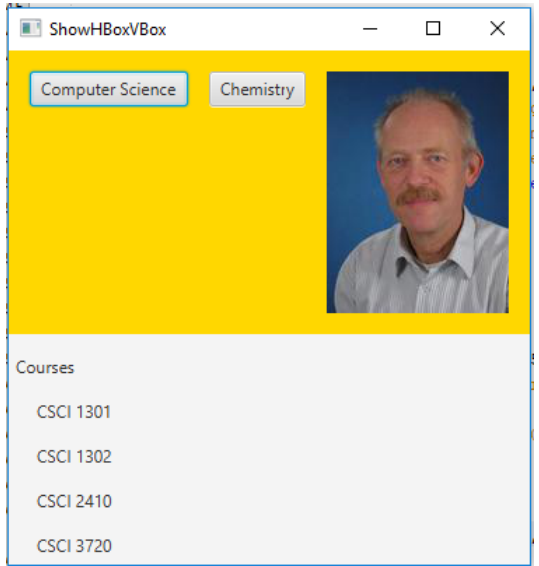
VBox

Plasserer alle noder under hverandre (vertikalt) i bare én kolonne



HBoxVboxDemo

```
@Override
public void start(Stage primaryStage) {
  BorderPane pane = new BorderPane();

  pane.setTop(getHBox());
  pane.setLeft(getVBox());

  Scene scene = new Scene(pane);
  primaryStage.setTitle("ShowHBoxVBox");
  primaryStage.setScene(scene);
  primaryStage.show();
}
```




```
private HBox getHBox() {
  HBox hBox = new HBox(15);
  hBox.setPadding(new Insets(15, 15, 15, 15));
  hBox.setStyle("-fx-background-color: gold");
  hBox.getChildren().add(new Button("Computer Science"));
  hBox.getChildren().add(new Button("Chemistry"));
  ImageView imageView =
    new ImageView(new Image("image/us.gif"));
  hBox.getChildren().add(imageView);
  return hBox;
}

private VBox getVBox() {
  VBox vBox = new VBox(15);
  vBox.setPadding(new Insets(15, 5, 5, 5));
  vBox.getChildren().add(new Label("Courses"));
  Label[] courses = {new Label("CSCI 1301"),
    new Label("CSCI 1302"), new Label("CSCI 2410"),
    new Label("CSCI 3720")};

  for (Label course: courses) {
    VBox.setMargin(course, new Insets(0, 0, 0, 15));
    vBox.getChildren().add(course);
  }
  return vBox;
}
```