

JavaFX

- JavaFX is een java library om een gebruiksvriendelijke grafische user interface op te bouwen.
- **Deel1: JavaFX Fundamenten**
We leren volledig handmatig een grafische user interface opbouwen in java
- **Deel2: JavaFX Scene Builder**
We gebruiken Scene Builder, een tool waarmee je visueel de layout van schermen kan opbouwen. Deze tool kan je integreren in Eclipse.

HO
GENT₂

JavaFX

- JavaFX wordt in het handboek besproken in hoofdstuk 12 en 13, enkel het gebruik van JavaFX Scene Builder wordt toegelicht.
- Heel wat terminologie en technieken, behandeld in deze slides zal je verspreid terug vinden in het handboek over de 2 hoofdstukken, maar deze slides wijken dus meer af van het handboek dan de andere hoofdstukken.



Deel 1: JavaFX Fundamenten



Leerinhoud JavaFX Fundamenten

- Opbouw van een JavaFX applicatie
 - UI componenten
 - Event handling
 - Layout managers
 - Menu's, toolbars
- en Alert dialogs

Doelstellingen

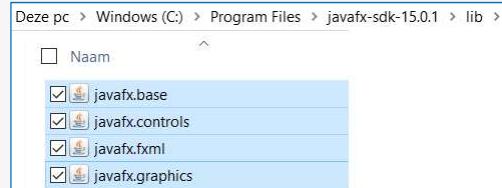
- Kan overerving en polymorfisme toepassen in Java
- Kan exception handling inbouwen in Java
- Kan strings verwerken in Java
- Kan GUI-componenten gebruiken in Java**
- Kan een gepaste file/stream correct implementeren in Java
- Kan de verantwoordelijkheden van elke laag in een drielagenstructuur toepassen.**
- Kan een gepaste collection kiezen en correct gebruiken in Java
- Kan een collection stream gebruiken in Java
- Kan een interface declareren en implementeren oa gebruikmakend van lambda expressies

1 Inleiding

- Grafische Gebruikers Interface (GUI)
 - Uitgesproken "GOE-iee"
 - Plaatst de gebruikers in een vertrouwde omgeving
 - Wordt opgebouwd met UI componenten (ook UI controls genoemd) zoals schuifbalken, icoon, etc.
 - Via de muis en het toetsenbord heeft de gebruiker interactie met de GUI componenten, etc.
- Voorgangers: Swing en AWT (Abstract Windowing Toolkit)
- JavaFX:
 - is de opvolger van Swing. Op termijn zal Swing vervangen worden door JavaFX als standaard GUI library.
 - eerste release: 4-12-2008

1.1 Een eerste voorbeeld: FXVoorbeeld1

- Maak nieuw project in Eclipse:
 - File – New - Project – JavaFX Project - Next
 - Project name: FXVoorbeeld1 – Next
 - Java Settings:
 - klik op de tab Libraries - Modulepath – Add External JARs
 - Ga naar de map javafx-sdk-15.0.1\lib en selecteer de 4 bovenste bestanden – Openen
 - Next – Package name: main
 - Finish



**HO
GENT**

7

1.1 Een eerste voorbeeld: FXVoorbeeld1

- Pas het project aan:
 - De klasse Main is een applicatie met een lege BorderPane (zie verder – slide 39)
 - Hernoem Main naar StartUp
 - De methode start moet aangepast worden (zie verder – slide 12)
 - Voorlopig gebruiken we nog geen stylesheets: we verwijderen in main het bestand application.css
 - In de package gui maken we een nieuwe (lege) klasse: New – Class - Name: WelkomScherm
 - We plaatsen het bestand bug.png in een package images
 - We voegen een module toe: rechts klikken op project – Configure – Create module-info.java - Create

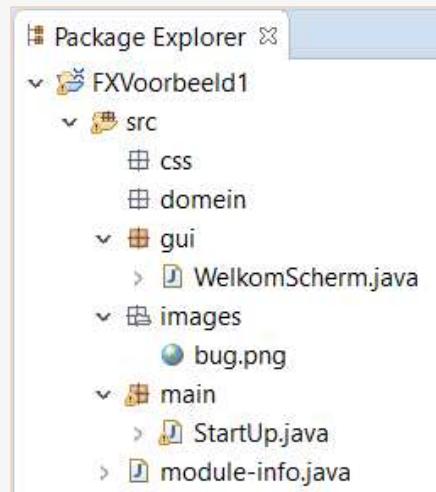
```
module-info.java
1 module FXVoorbeeld1 {
2     exports gui;
3     exports main;
4
5     requires javafx.base;
6     requires javafx.graphics;
7     requires javafx.controls;
8
9     opens gui to javafx.graphics;
10 }
```

**HO
GENT**

8

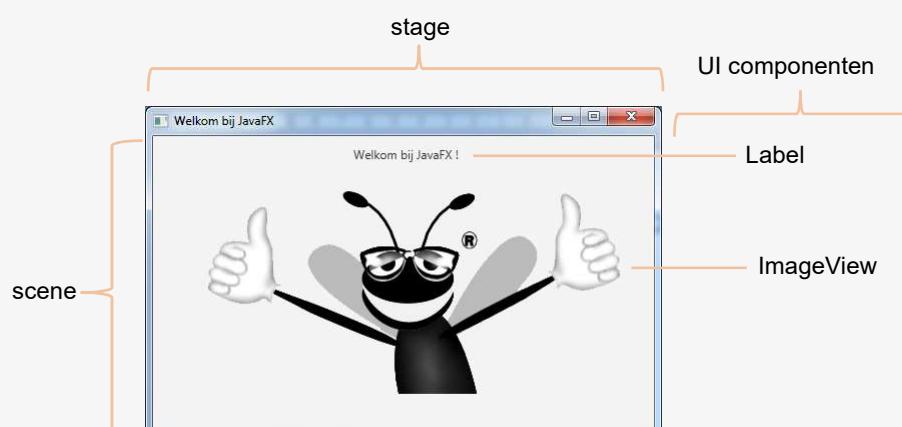
1.2 JavaFX applicatie in 3-lagen model

- FXVoorbeeld1



HO
GENT
9

1.3 Delen van het venster in JavaFX



HO
GENT
10

```

1 package gui;
2 import javafx.scene.control.Label;
3 import javafx.scene.image.Image;
4 import javafx.scene.image.ImageView;
5 import javafx.scene.layout.Pane;
6
7 public class WelkomScherm extends Pane {
8
9     public WelkomScherm() {
10         // Eerste grafische component: een label
11         Label lblWelkom = new Label("Welkom bij JavaFX !");
12
13         // Tweede grafische component, een ImageView toont een Image op het scherm
14         ImageView ivImage = new ImageView(
15             new Image(getClass().getResourceAsStream("/images/bug.png")));
16
17         // In dit voorbeeld gebruik we geen layout
18         // We geven de componenten een vaste positie mee
19         // Dit doen we met de methoden setLayoutX en setLayoutY
20         lblWelkom.setLayoutX(200);
21         lblWelkom.setLayoutY(10);
22         ivImage.setLayoutX(50);
23         ivImage.setLayoutY(50);
24
25         // Alle componenten worden verzameld in ons paneel
26         // componenten toevoegen aan een paneel
27         this.getChildren().addAll(lblWelkom, ivImage);
28     }
}

```

import uit javafx

Vraag huidige componenten van het paneel root op en voeg er 2 componenten aan toe

O
ENT 11

```

1 package main;
2
3 import javafx.application.Application;
4 import javafx.scene.Scene;
5 import javafx.stage.Stage;
6 import gui.WelkomScherm;
7
8 public class StartUp extends Application {
9
10     @Override
11     public void start(Stage primaryStage)
12     {
13         WelkomScherm root = new WelkomScherm();
14
15         Scene scene = new Scene(root, 500, 300);
16
17         primaryStage.setScene(scene);
18
19         primaryStage.setTitle("Welkom bij JavaFX");
20         primaryStage.show();
21     }
22
23
24     public static void main(String[] args)
25     {
26         launch(args);
27     }
28 }

```

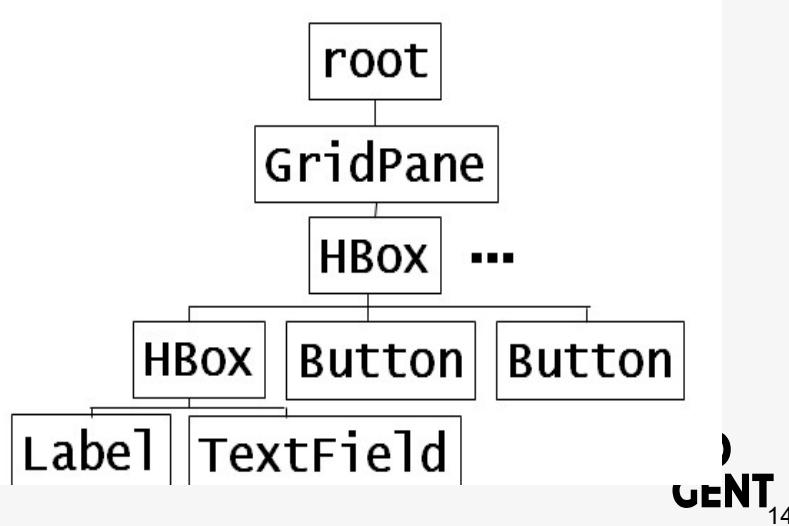
12

1.4 Opbouw van een JavaFX applicatie

- De JavaFX **Stage** is de bovenste laag van de JavaFX container
- De JavaFX **Scene** is een boomstructuur (graph) van gui-componenten
- Elke component in de scene graph is een **node**.
Een node is een instantie van een subklasse van **Node** (package javafx.scene), die gemeenschappelijke attributen en gedrag definieert voor alle nodes in een scene graph.
- De eerste node in de scene graph is de root node.

HO
GENT
₁₃

1.4 Opbouw van een JavaFX applicatie



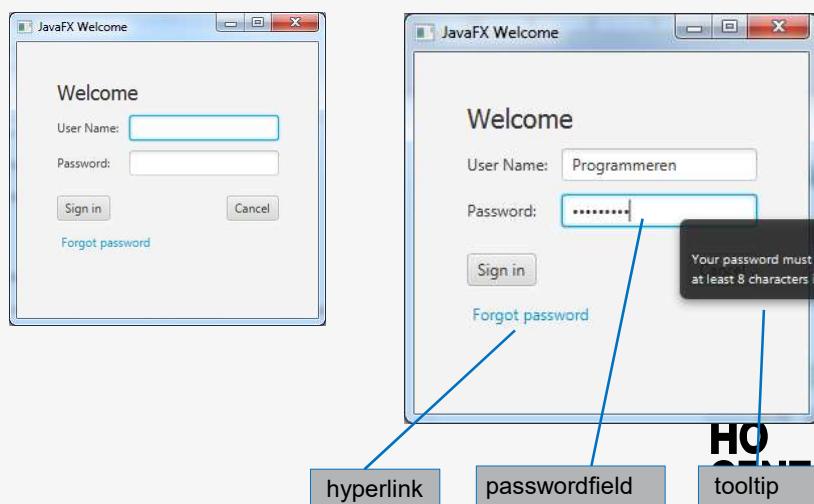
GENT
₁₄

2 Enkele componenten uit javafx.scene.control package

Component			Beschrijving
Label	Simple Label	Graphic Label	Een gebied met niet-editeerbare tekst en/of afbeelding.
TextField	Some text		Een gebied waar de gebruiker de invoer van één regel tekst via het toetsenbord intypt. Het kan ook gegevens tonen.
Button	Yellow Blue	Button	Door het klikken op deze button wordt een event uitgevoerd.
HyperLink	Hyperlink		Een HTML-label met tekst en/of afbeelding die reageert op rollovers en kliks.
PasswordField	*****		Verbergt de karakters die de gebruiker intypt.

15

2.1 Een login pagina: FXVoorbeeld2



HO

GO

,6

```

package gui;

import ...;

public class RegistreerScherm extends GridPane
{
    // Dit attribuut hebben we in meerdere methoden nodig
    private Label lblMessage;

    public RegistreerScherm()
    {
        // Alineert grid in het midden
        this.setAlignment(Pos.CENTER);
        // Vrije ruimte tussen kolommen
        this.setHgap(10);
        // Vrije ruimte tussen rijen
        this.setVgap(10);

        // Vrije ruimte rond de randen van de grid (boven, rechts, onder, links)
        this.setPadding(new Insets(25, 25, 25, 25));

        Label lblTitle = new Label("Welcome");
        lblTitle.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));

        // Bij GridPane kan in elke cel een component geplaatst worden
        // Een component kan over meerdere rijen en/of kolommen geplaatst worden
        // De label wordt hier over 2 kolommen en 1 rij geplaatst
        this.add(lblTitle, 0, 0, 2, 1);

        Label lblUserName = new Label("User Name:");
        this.add(lblUserName, 0, 1);

        TextField txfUser = new TextField();
        this.add(txfUser, 1, 1);
    }
}

```

```

Label lblPassword = new Label("Password:");
this.add(lblPassword, 0, 2);

PasswordField pwfPassword = new PasswordField();
this.add(pwfPassword, 1, 2);

Tooltip tooltip = new Tooltip();
tooltip.setText(
    "\nYour password must be\n"
    + "at least 8 characters in length\n"
);
pwfPassword.setTooltip(tooltip);

Button btnSignIn = new Button("Sign in");
// We aligneren btnSignIn links
setHalignment(btnSignIn, HPos.LEFT);
this.add(btnSignIn, 0, 4);

Button btnCancel = new Button("Cancel");
// We aligneren btnCancel rechts
setHalignment(btnCancel, HPos.RIGHT);
this.add(btnCancel, 1, 4);

Hyperlink linkForgot = new Hyperlink("Forgot password");
this.add(linkForgot, 0, 5, 2, 1);

lblMessage = new Label();
this.add(lblMessage, 1, 6);

```

```

// We koppelen een event handler aan de knop Sign In
// We gebruiken hier voor method reference
    btnSignIn.setOnAction(this::buttonPushed);

// We koppelen een event handler aan de knop Cancel
// We gebruiken hier voor een lambda expressie
    btnCancel.setOnAction(evt -> lblMessage.setText("Cancel button pressed")
    );

// We koppelen een event handler aan de hyperlink
// We gebruiken hier voor een anonyme innerklasse
    linkForgot.setOnAction(new EventHandler<ActionEvent>()
    {
        @Override
        public void handle(ActionEvent evt)
        {
            lblMessage.setText("Hyperlink clicked");
        }
    });
}

// Event-afhandeling: wat er moet gebeuren als we op de knop Sign in klikken
private void buttonPushed(ActionEvent event)
{
    lblMessage.setText("Sign in button pressed");
}
}

```

```

package main;

import gui.RegistreerScherm;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class StartUp extends Application
{
    @Override
    public void start(Stage primaryStage)
    {
        RegistreerScherm grid = new RegistreerScherm();
        // grid is de root node, breedte is 300, hoogte is 275

        Scene scene = new Scene(grid, 300, 275);
        primaryStage.setScene(scene);

        primaryStage.setTitle("JavaFX Welcome");
        primaryStage.show();
    }

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

```

3 Event afhandeling

- GUI's zijn *event driven*:
 - genereert events op het ogenblik dat de gebruiker interactie heeft met een UI component
 - op een knop klikken, tekst intypen in een tekstvak, de muis bewegen, etc.
- De code die uitgevoerd wordt als antwoord op het optreden van een **event** heet een **event handler**

**HO
GENT**
21

3.1 Delen bij event handling

Er zijn 3 delen bij event handling:

- **Event source**: de UI component waarop de gebruiker inwerkt
- **Event object**: bevat de informatie over het event dat opgetreden is
- **Event listener**: ontvangt een event object bij het optreden van een event en geeft dan een antwoord via een event handler

**HO
GENT**
22

- Als programmeur moet je twee taken uitvoeren:
 - Registreer de event listener voor de event source
 - Implementeer een event-handling methode (event handler)
- Event handling kan op 3 manieren:
 - Met een **method reference**
 - Met een **lambda expressie**
 - Met een **anonieme innerklasse**

**HO
GENT**
23

3.2 Voorbeelden van event afhandeling

- Method reference:

```

79 | // We koppelen een event handler aan de knop Sign In
80 | // We gebruiker hiervoor method reference
81 |     btnSignIn.setOnAction(this::buttonPushed);

101 |     // Event-afhandeling: wat er moet gebeuren als we op de knop Sign in klikken
102 |     private void buttonPushed(ActionEvent event)
103 |     {
104 |         lblMessage.setText("Sign in button pressed");
105 |     }
  
```

- Lambda expressie:

```

83 | // We koppelen een event handler aan de knop Cancel
84 | // We gebruiken hiervoor een lambda expressie
85 |     btnCancel.setOnAction(evt -> lblMessage.setText("Cancel button pressed"))
86 |
87 |
  
```

GENT
24

- Anonieme innerklasse:

```

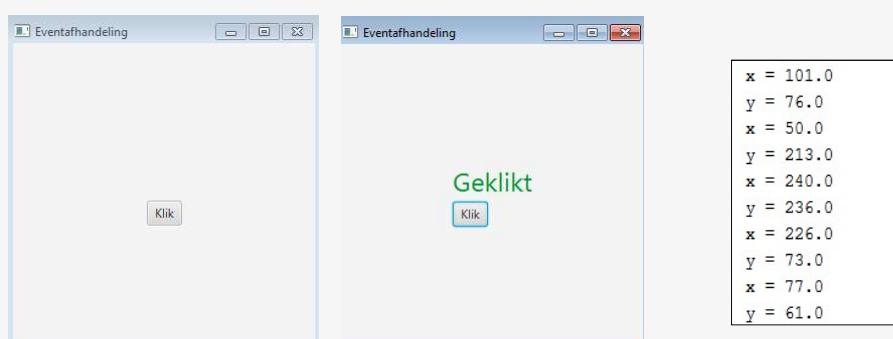
88 // We koppelen een event handler aan de hyperlink
89 // We gebruiken hiervoor een anonieme innerklasse
90 linkForgot.setOnAction(new EventHandler<ActionEvent>()
91 {
92     @Override
93     public void handle(ActionEvent evt)
94     {
95         lblMessage.setText("Hyperlink clicked");
96     }
97 });

```

De event handler ontvangt een ActionEvent, die aangeeft dat op de hyperlink geklikt werd

**HO
GENT**
25

3.3 Van publieke klasse tot anonieme innerklasse



Zie voorbeeld: DemoEventAfhandeling

**HO
GENT**
26

3.3 A. Eventafhandeling in de publieke klasse

```
public class StartUp extends Application
{
    @Override
    public void start(Stage primaryStage) throws Exception
    {
        //1. klasse scherm én klasse die eventhandler-interface implementeert
        DemoEventScherm root = new DemoEventScherm();

        Scene scene = new Scene(root,300,300);
        scene.getStylesheets().add("/css/style.css");
        primaryStage.setScene(scene);
        primaryStage.setTitle("Eventafhandeling");
        primaryStage.show();
    }
}
```



27

```
public class DemoEventScherm extends GridPane
{
    public DemoEventScherm()
    {
        Label lblBoodschap = new Label();
        Button btnKlik = new Button("Klik");

        this.add(lblBoodschap, 0, 0);
        this.add(btnKlik, 0, 1);

        this.setAlignment(Pos.CENTER);

        ActionEventAfhandeling eafh = new ActionEventAfhandeling(lblBoodschap);
        btnKlik.setOnAction(eafh);

        MouseEventAfhandeling mafh = new MouseEventAfhandeling();
        this.setOnMouseClicked(mafh);
    }
}
```

28

```
public class ActionEventAfhandeling implements EventHandler<ActionEvent>
{
    private final Label lblBoodschap;

    public ActionEventAfhandeling(Label lblBoodschap)
    {
        this.lblBoodschap = lblBoodschap;
    }

    @Override
    public void handle(ActionEvent event)
    {
        lblBoodschap.setText("Geklikt");
    }
}
```

```
public class MouseEventAfhandeling implements EventHandler<MouseEvent>
{
    @Override
    public void handle(MouseEvent event)
    {
        System.out.println("x = " + event.getSceneX());
        System.out.println("y = " + event.getSceneY());
    }
}
```

HO GENT

29

3.3 B. Eventafhandeling in de innerklasse

- Een klasse kan binnenin een andere klasse gedefinieerd worden. Zulke innerklasses worden dikwijls gebruikt voor event afhandeling.

De inner klasse heeft toegang tot alle attributen/methoden van de 'outer' klasse.

- Voorbeeld: zie volgende dia's.

HO
GENT
30

```

public class DemoEventSchermMetBenoemdeInnerKlasse extends GridPane
{
    private final Label lblBoodschap;

    public DemoEventSchermMetBenoemdeInnerKlasse()
    {
        lblBoodschap = new Label();
        Button btnKlik = new Button("Klik");
        this.add(lblBoodschap, 0, 0);
        this.add(btnKlik, 0, 1);

        this.setAlignment(Pos.CENTER);

        btnKlik.setOnAction(new InnerklasseActionEventAfhandeling());
        this.setOnMouseClicked(new InnerKlasseMouseEventAfhandeling());
    }
}

```

**HO
GENT**
31

```

private class InnerklasseActionEventAfhandeling
    implements EventHandler<ActionEvent>
{
    @Override
    public void handle(ActionEvent event)
    {
        lblBoodschap.setText("Geklikt");
    }
}

private class InnerKlasseMouseEventAfhandeling
    implements EventHandler<MouseEvent>
{
    @Override
    public void handle(MouseEvent event)
    {
        System.out.println("x = " + event.getSceneX());
        System.out.println("y = " + event.getSceneY());
    }
}

```

GENT
32

3.3 C. Eventafhandeling in de anonieme innerklasse

- I.p.v. een innerklasse te definiëren, definiëren we de implementatie van de interface onmiddellijk bij instantiatie van het object.
- We kunnen de objectreferentie dan ook ineens ter plaatse leveren, nl. als actueel argument van de methode **setOnAction**.
- We doen dit voor elke GUI-component die events levert.

**HO
GENT**
33

```
public class DemoEventSchermMetAnoniemeInnerKlassen extends GridPane
{
    public DemoEventSchermMetAnoniemeInnerKlassen()
    {
        Label lblBoodschap = new Label();
        Button btnKlik = new Button("Klik");

        this.add(lblBoodschap, 0, 0);
        this.add(btnKlik, 0, 1);
        this.setAlignment(Pos.CENTER);

        btnKlik.setOnAction(new EventHandler<ActionEvent>()
        {
            @Override
            public void handle(ActionEvent event)
            {
                lblBoodschap.setText("Joepie!");
            }
        });

        this.setOnMouseClicked(new EventHandler<MouseEvent>()
        {
            @Override
            public void handle(MouseEvent event)
            {
                System.out.println("x = " + event.getSceneX());
                System.out.println("y = " + event.getSceneY());
            }
        });
    }
}
```

**HO
GENT**
34

3.3 D. Eventafhandeling in de anonieme innerklasse: WindowEvent (versie 1)

```
public class StartUp extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {

        //3. Klasse scherm met anonieme innerklassen
        DemoEventSchermMetAnoniemeInnerKlassen root = new DemoEventSchermMetAnoniemeInnerKlassen();

        Scene scene = new Scene(root,300,300);
        scene.getStylesheets().add("/css/style.css");
        primaryStage.setScene(scene);
        primaryStage.setTitle("Eventafhandeling");
        primaryStage.show();
        primaryStage.setOnCloseRequest(new EventHandler<WindowEvent>() {
            @Override
            public void handle(WindowEvent event) {
                System.out.println("We sluiten het venster en dus... ook de applicatie");
                Platform.exit();
            }
        });
    }
}
```

x = 67.0
y = 65.0
x = 256.0
y = 74.0
We sluiten het venster en dus... ook de applicatie

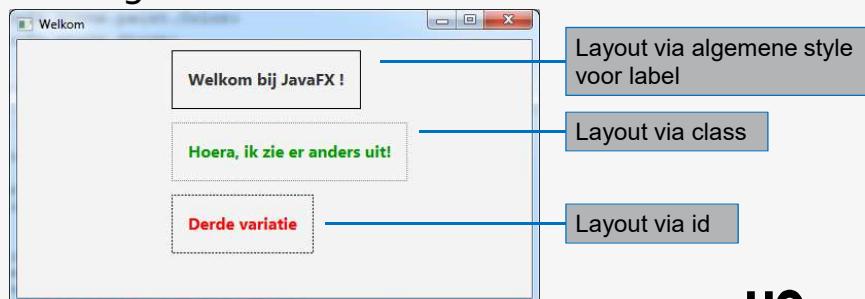
3.3 D. Eventafhandeling in de anonieme innerklasse: WindowEvent (versie 2)

```
primaryStage.setOnCloseRequest(new EventHandler<WindowEvent>()
{
    @Override
    public void handle(WindowEvent event)
    {
        Alert alert = new Alert(AlertType.CONFIRMATION);
        alert.setTitle("Bevestig");
        alert.setContentText("Wil je de applicatie afsluiten?");
        Optional result = alert.showAndWait();
        if (result.get() == ButtonType.OK)
        {
            System.out.println("We sluiten het venster en dus... ook de applicatie");
        } else
        {
            event.consume();
        }
    }
});
```

Het event wordt geannuleerd

4 CSS: FXVoorbeeld3

- Het is mogelijk om JavaFX Cascading Style Sheets (CSS) te gebruiken om de opmaak te verzorgen.



5 Layout panelen

- Nodes met afstammelingen zijn meestal layout containers die hun afstammelingen ordenen in de scene.
- Nodes die geordend zijn in een layout container zijn een combinatie van UI controls en mogelijk andere layout containers.
- Layout panelen kunnen gecombineerd worden – zie werkcollege - rode draad deel 2.

5.1 BorderPane

In een BorderPane heb je 5 gebieden om nodes in te plaatsen: top, bottom, right, left en center.



**HO
GENT**
39

5.2 HBox en VBox

In een HBox worden de nodes horizontaal in één rij geplaatst.



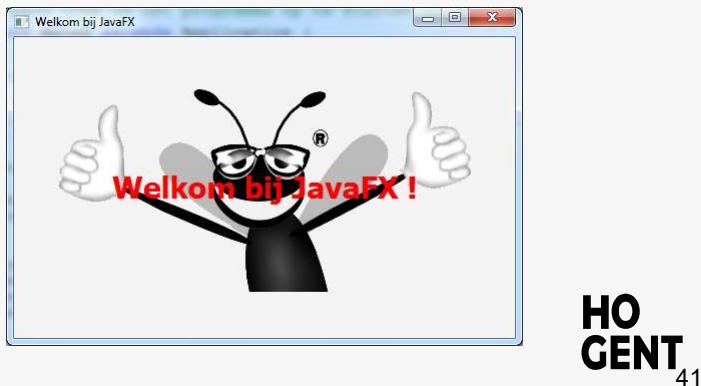
In een VBox worden de nodes verticaal in één kolom geplaatst.



**HO
GENT**
40

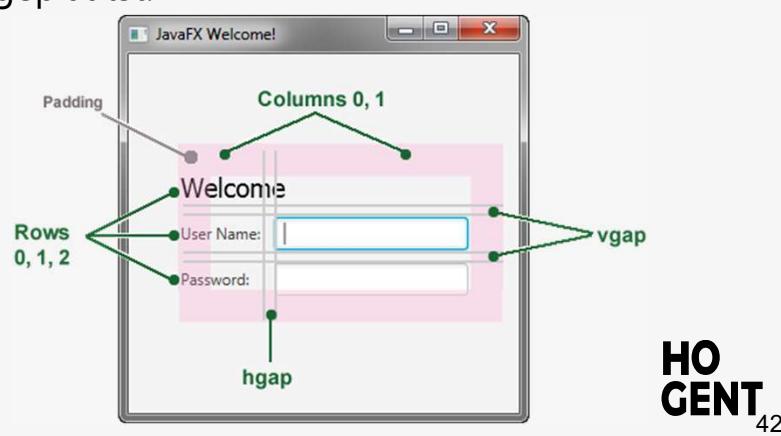
5.3 StackPane

In een StackPane worden de nodes boven elkaar geplaatst. De meest recente node komt bovenaan zoals bij een stack.



5.4 GridPane

In een GridPane worden de nodes in een flexibele tabel geplaatst.



5.5 FlowPane en TilePane

In een horizontale FlowPane worden de nodes in een rij geplaatst. Indien de beschikbare breedte overschreden wordt dan komt de node in een nieuwe rij.

In een verticale FlowPane worden de nodes in een kolom geplaatst. Indien de beschikbare hoogte overschreden wordt dan komt de node in een nieuwe kolom.

In een TilePane worden de nodes geplaatst in cellen die allemaal even groot zijn. De grootte van elke cel wordt bepaald door de grootste nodige. Of zelf instellen via prefTileWidth en prefTileHeight!



5.6 AnchorPane

In een AnchorPane kunnen er ankerpunten geplaatst worden bij de top, left, right, center en bottom van de layout.



5.7 Grootte van de nodes

Stel de minimum, preferred en/of maximum size in van de componenten



5.8 Uitlijning van de nodes

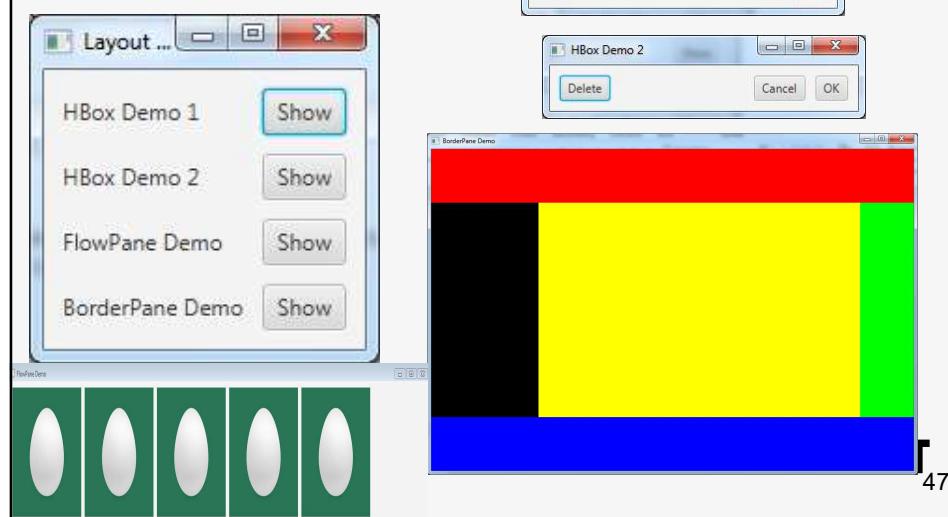
Gebruik de methode `setAlignment(Pos value)`

- Pos - verticaal en horizontaal uitlijnen
De waarde links van de underscore geeft de verticale uitlijning weer, de waarde rechts van de underscore geeft de horizontale uitlijning weer.
Bijvoorbeeld: `Pos.BOTTOM_LEFT` lijnt verticaal uit aan de onderkant en horizontaal aan de linkerkant
- Voorbeeld2: `this.setAlignment(Pos.CENTER);`

HO
GENT₄₆

5.9 Demo van layout panelen

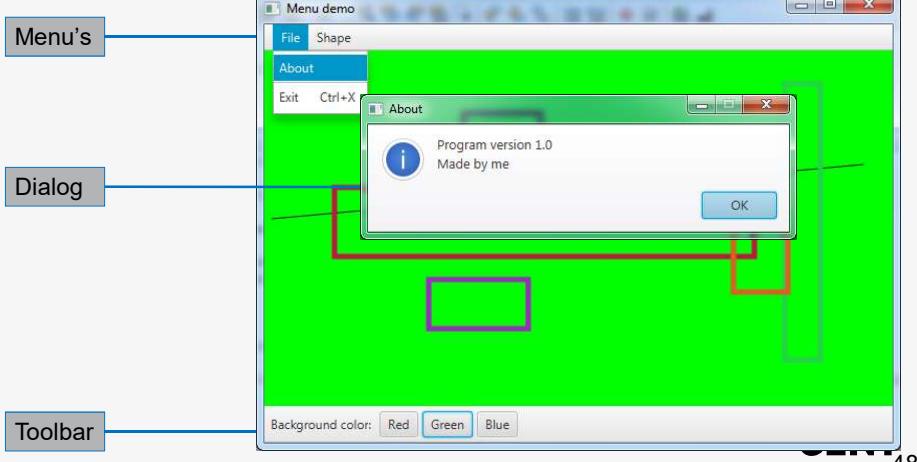
FXVoorbeeld4



47

6 Menu's, Toolbar en Dialog

FXVoorbeeld5



48

6.1 Menu

- Door te werken met menu's kan de gebruiker acties uitvoeren zonder extra componenten op de GUI
- Een menu wordt beheerd door een object van **MenuBar**
- Een menu is opgebouwd uit menu componenten zoals **Menu** (submenu), **MenuItem**, **CheckMenuItem**, **RadioMenuItem**, ...
- Een menu is toegankelijk via muis of via sneltoetsen (accelerator)
- Met de methode **getMenus().add()** kan je menu's toevoegen aan een **MenuBar**
- Met de methode **getItems().add()** kan je componenten toevoegen aan een **Menu**

**HO
GENT**
49

Voorbeeld van een Menu

```
MenuBar menuBar = newMenuBar();
Menu fileMenu = new Menu("File");
MenuItem aboutMenuItem = new MenuItem("About");
MenuItem exitMenuItem = new MenuItem("Exit");
exitMenuItem.setAccelerator(KeyCombination.keyCombination("Ctrl+x"));
fileMenu.getItems().addAll(aboutMenuItem, new SeparatorMenuItem(), exitMenuItem);
menuBar.getMenus().add(fileMenu);
```

Event handling bij menu:

```
aboutMenuItem.setOnAction(this::aboutClicked);
exitMenuItem.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent e) {
        Platform.exit();
    }
});
```

50

6.2 Toolbar

- Componenten toevoegen aan een toolbar kan via de constructor of met de methode `getItems().add()`

```
Button btnRed = new Button("Red");
Button btnGreen = new Button("Green");
Button btnBlue = new Button("Blue");
ToolBar toolBar = new ToolBar(new Label("Background color: "),
                                btnRed, btnGreen, btnBlue);
toolBar.setOrientation(Orientation.HORIZONTAL);
```

**HO
GENT** 51

51

6.3 Alert

- Voorbeeld:

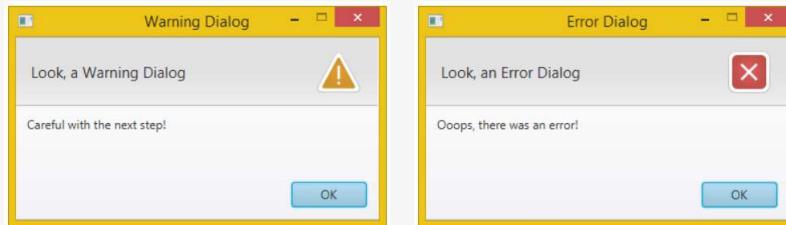


**HO
GENT** 52

52

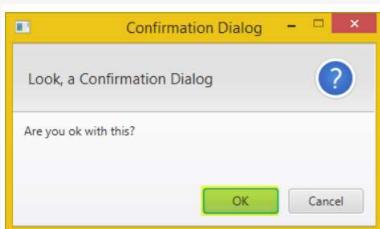
6.3 Alert

- AlertType.WARNING en AlertType.ERROR



**HO
GENT**
53

6.3 Alert



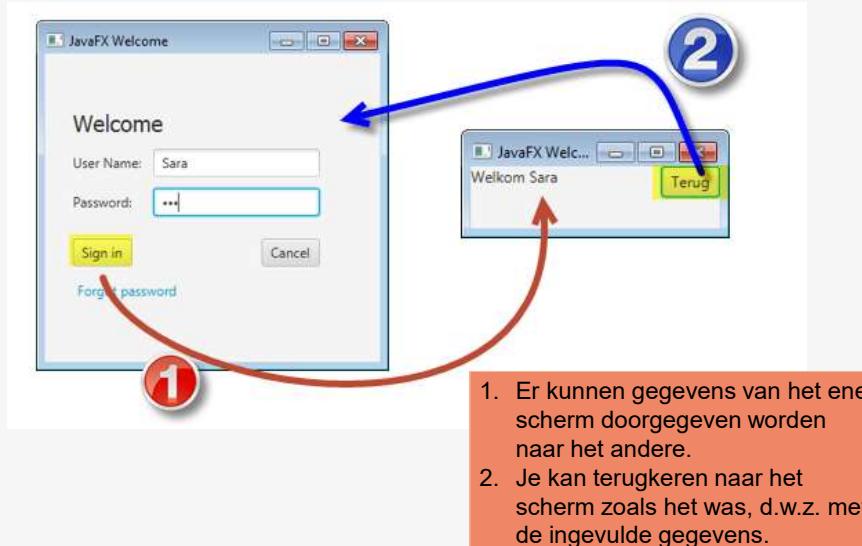
Afhankelijk van het antwoord van de gebruiker – OK of Cancel – kan je hier verschillend op reageren.

```
Alert alert = new Alert(AlertType.CONFIRMATION);
alert.setTitle("Confirmation Dialog");
alert.setHeaderText("Look, a Confirmation Dialog");
alert.setContentText("Are you ok with this?");

Optional<ButtonType> result = alert.showAndWait();
if (result.get() == ButtonType.OK){
    // ... user chose OK
} else {
    // ... user chose CANCEL or closed the dialog
}
```

**HO
GENT**
54

7.1 Wisselen van scherm (FXVoorbeeld7)



7.1 Wisselen van scherm (FXVoorbeeld7)

1

```
// Event-afhandeling: wat er moet gebeuren als we op de knop Sign in klikken
private void buttonPushed(ActionEvent event) {
    lblMessage.setText("Sign in button pressed");

    //We bouwen een nieuw scherm op waarmee we een scene maken
    //Parameter 1: naam gebruiker (ingevuld op loginscherm), willen we tonen op volgend scherm
    //Parameter 2: this, volgend scherm krijgt het huidige scherm mee om gemakkelijk te kunnen terugkeren.
    VolgendScherm vs = new VolgendScherm(txfUser.getText(), this);
    Scene scene = new Scene(vs, 200, 50);
    Stage stage = (Stage) this.getScene().getWindow();
    stage.setScene(scene);
    stage.show();
}
```

7.1 Wisselen van scherm (FXVoorbeeld7)

2

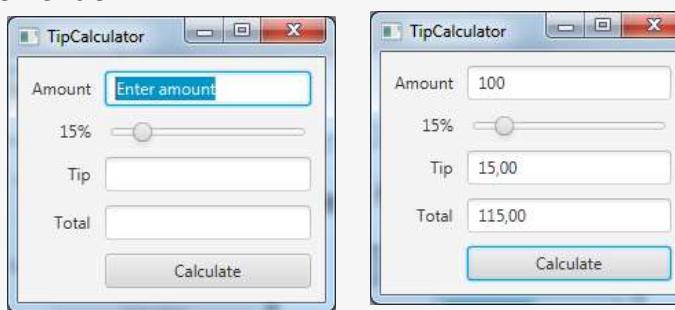
```
private void buildGui() {
    lblWelkom = new Label("Welkom " + inlognaam);
    btnTerug = new Button("Terug");
    Region spring = new Region();
    HBox.setHgrow(spring, Priority.ALWAYS);
    this.getChildren().addAll(lblWelkom,spring,btnTerug);

    btnTerug.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            Stage stage = (Stage)(getScene().getWindow());
            stage.setScene(login.getScene());
        }
    });
}
```

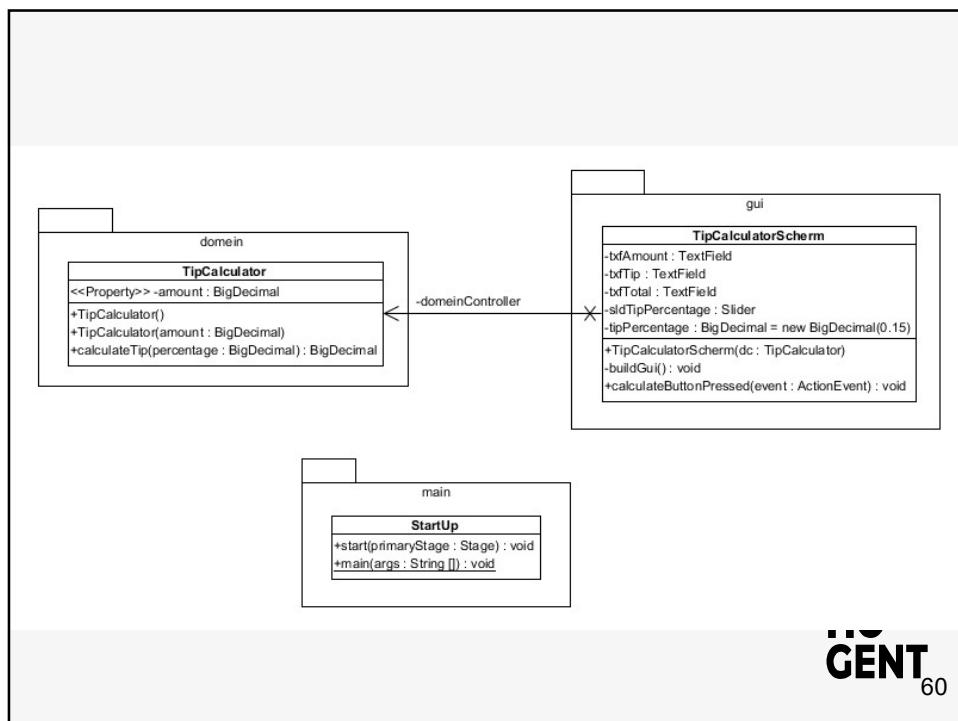
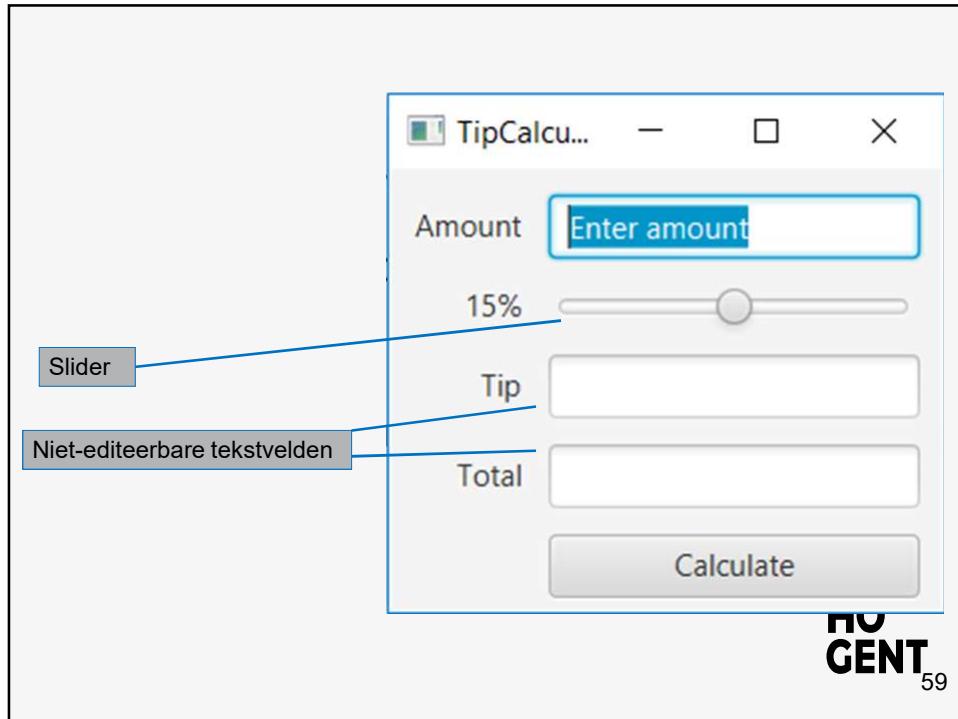
HO
GENT
57

7.2 Tipcalculator (FXVoorbeeld6)

- De Tipcalculator berekent en toont de fooi en het totaal te betalen bedrag van een restaurantrekening
- By default wordt een fooi van 15 % aangerekend
- Een ander tippercentage wordt ingesteld aan de hand van een Slider



NT
58



```
txfTip = new TextField();
txfTip.setEditable(false);
```

```
sldTipPercentage.valueProperty().addListener(
    new ChangeListener<Number>()
{
    @Override
    public void changed(ObservableValue<? extends Number> ov,
                        Number oldValue, Number newValue)
    {
        tipPercentage = BigDecimal.valueOf(newValue.doubleValue()/100.0);
        lblTipPercentage.setText(String.format("%.0f%%",
                                              tipPercentage.multiply(new BigDecimal(100.0))));
    }
});
```

HO GENT

61

Bijkomend leermateriaal

- <http://docs.oracle.com/javase/8/javase-clienttechnologies.htm>
- Workshops JavaFX, Steven Van Impe, ongepubliceerde uitgave Projecten-workshops I, 2013-2014
- <http://www.java2s.com/Tutorials/Java/JavaFX/index.htm>
- <https://code.makery.ch/library/javafx-tutorial/>
- <https://gluonhq.com/products/javafx/>

HO
GENT

62