HoGent

BEDRIJF
EN
ORGANISATIE

H25: JavaFX Fundamenten

Leerinhoud JavaFX Fundamenten

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

Deelcompetentie 1: een programma ontwikkelen waarin meerdere klassen met elkaar interageren (TI01)
Indicatoren:
1.1. Kan overerving en polymorfisme toepassen in Java
1.2. Kan exception handling inbouwen in Java
1.3. Kan strings verwerken in Java
1.4. **Kan GUI-componenten gebruiken in Java**

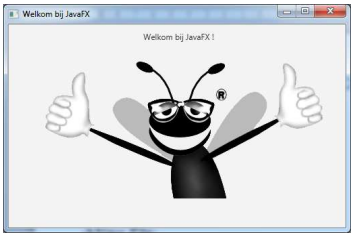
1 Inleiding

- 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.
- Swing: ontwikkeld in 1997.
- JavaFX:
  - is de opvolger van Swing. Op termijn zal Swing vervangen worden door JavaFX als standaard GUI library.
  - eerste release: 4-12-2008

HoGent 3

1.1 Een eerste voorbeeld: FXVoorbeeld1

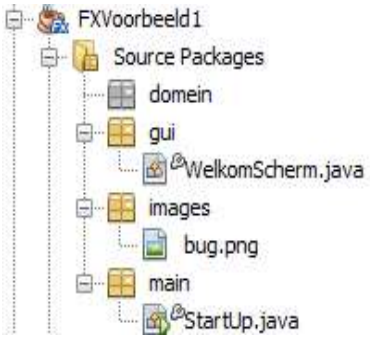
- Maak nieuw project in NetBeans:
  - →File - New Project
  - →Categories: JavaFX
  - →Projects: JavaFX Application
  - →Project name: FXVoorbeeld1
  - →Maak package main
  - →New / Other:
    - → Categories: JavaFX
    - → File Types: JavaFX Main Class
    - → Class Name: StartUp
      - →Je krijgt een ‘Hello World’ applicatie
      - →Pas de start methode aan (zie verder – slide 8)!
  - →Plaats bug.png in map (package) images
  - →Maak package gui
  - →New Class: WelkomScherm.java



HoGent 4

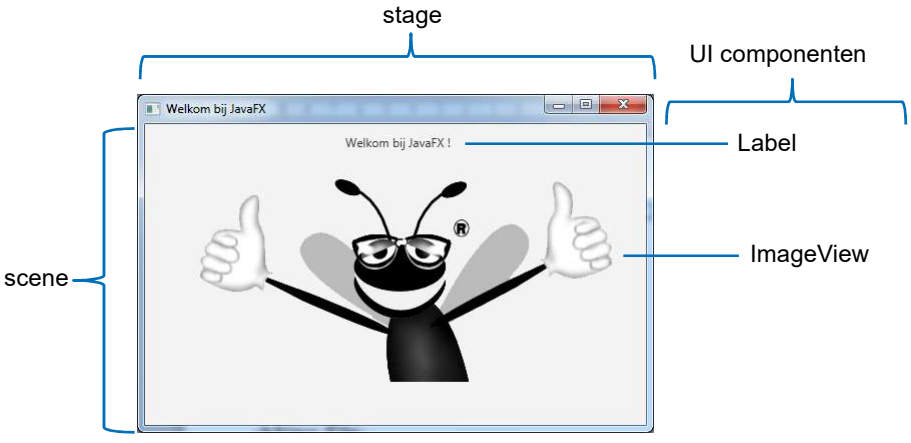
1.2 JavaFX applicatie in 3-lagen model

- FXVoorbeeld1



HoGent 5

1.3 Delen van het venster in JavaFX



HoGent 6

```java
package gui;
import javafx.scene.control.Label;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.Pane;

public class WelkomScherm extends Pane {

    public WelkomScherm() {
// Eerste grafische component: een label
        Label lblWelkom = new Label("Welkom bij JavaFX !");

// Tweede grafische component, een ImageView toont een Image op het scherm
        ImageView ivImage = new ImageView(
                new Image(getClass().getResourceAsStream("/images/bug.png"))
        );
// In dit voorbeeld gebruik we geen layout
// We geven de componenten een vaste positie mee
// Dit doen we met de methoden setLayoutX en setLayoutY
        lblWelkom.setLayoutX(200);
        lblWelkom.setLayoutY(10);
        ivImage.setLayoutX(50);
        ivImage.setLayoutY(50);
// Alle componenten worden verzameld in ons paneel
// componenten toevoegen aan een paneel
        this.getChildren().addAll(lblWelkom, ivImage);
    }
}
```

import uit **javafx**

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

Hc

7

```java
package main;

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

public class StartUp extends Application
{
    @Override
    public void start(Stage primaryStage)
    {
        WelkomScherm root = new WelkomScherm();

        Scene scene = new Scene(root, 500, 300);

        primaryStage.setScene(scene);

        primaryStage.setTitle("Welkom bij JavaFX");
        primaryStage.show();

    }

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

HoGent

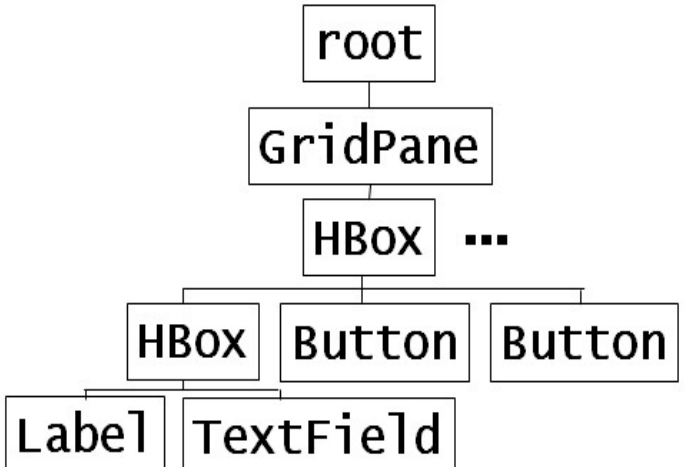
8

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.

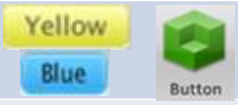
HoGent 9

1.4 Opbouw van een JavaFX applicatie

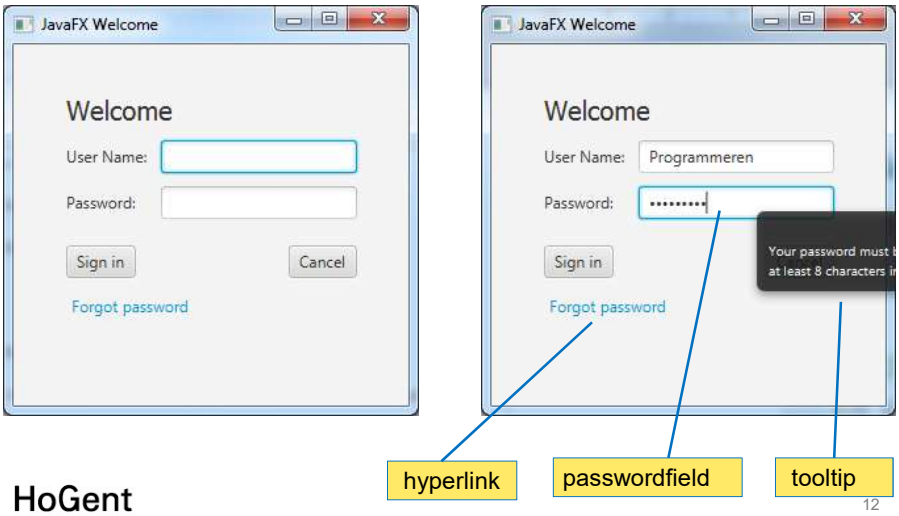


HoGent 10

2 Enkele componenten uit javafx.scene.control package

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

2.1 Een login pagina: FXVoorbeeld2



HoGent

12

```
package gui;

import ...

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

    public RegistreerScherm()
    {
       // Aligneert 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 gebruiker hiervoor method reference
        btnSignIn.setOnAction(this::buttonPushed);

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

// We koppelen een event handler aan de hyperlink
// We gebruiken hiervoor een anonieme 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

HoGent 17

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

HoGent 18

- Programmeur moet 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**
    → cfr. Scenebuilder in het volgende hoofdstuk
  - Met een **lambda expressie**
    → zie OO Programmeren III
  - Met een **anonieme innerklasse**

HoGent 19

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
```

HoGent 20

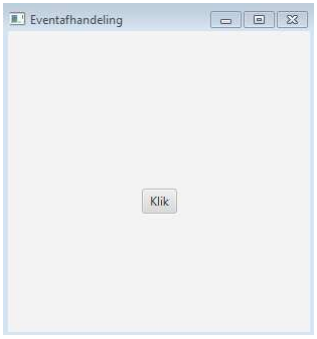
- Anonieme innerklasse:

```
88  // We koppelen een event handler aan de hyperlink
89  // We gebruiken hiervoor een anonieme innerklasse
            linkForgot.setOnAction(new EventHandler<ActionEvent>()
91          {
92              @Override
                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

HoGent 21

3.3 Van publieke klasse tot anonieme innerklasse




```
x = 101.0
y = 76.0
x = 50.0
y = 213.0
x = 240.0
y = 236.0
x = 226.0
y = 73.0
x = 77.0
y = 61.0
```

Zie voorbeeld: DemoEventAfhandeling

HoGent 22

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();


    }
}
```

HoGent 23

```
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);
    }

}
```

HoGent 24

```java
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");
    }
}
```

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

HoGent 25

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.

HoGent 26

```
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());
    }
```

HoGent 27

```
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());
    }
}
```

HoGent 28

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.


```
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());
            }
        });
    }
}
```

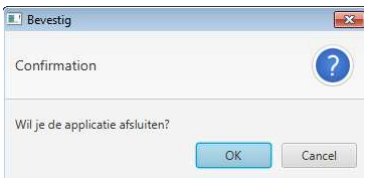
3.3 D. Eventafhandeling in de anonieme innerklasse: WindowEvent (versie 1)

```
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)

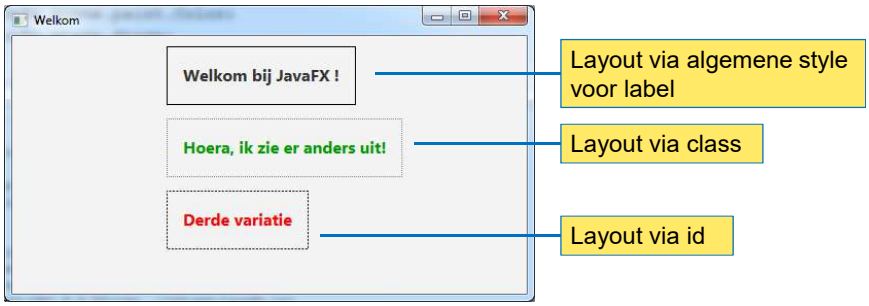
```
primaryStage.setOnCloseRequest(new EventHandler<WindowEvent>()
        {

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

            }
        }
        );
```



HoGent

4 CSS: FXVoorbeeld3

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



HoGent 33

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 rode draad deel 2.

34

5.1 BorderPane

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

HoGent 35

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.

HoGent 36

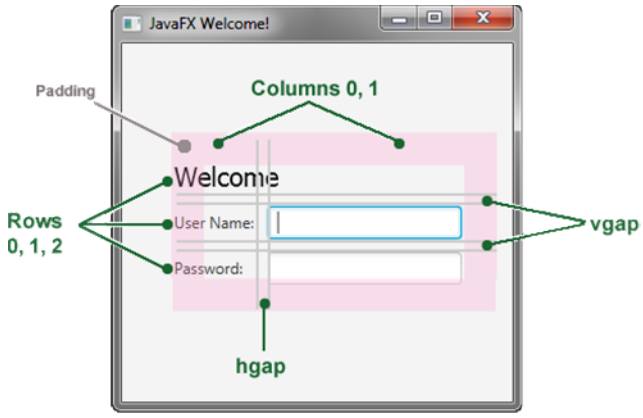
5.3 StackPane

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

HoGent 37

5.4 GridPane

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



HoGent 38

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 grootst nodige. Of zelf instellen via prefTileWidth en prefTileHeight!

HoGent

39

5.6 AnchorPane

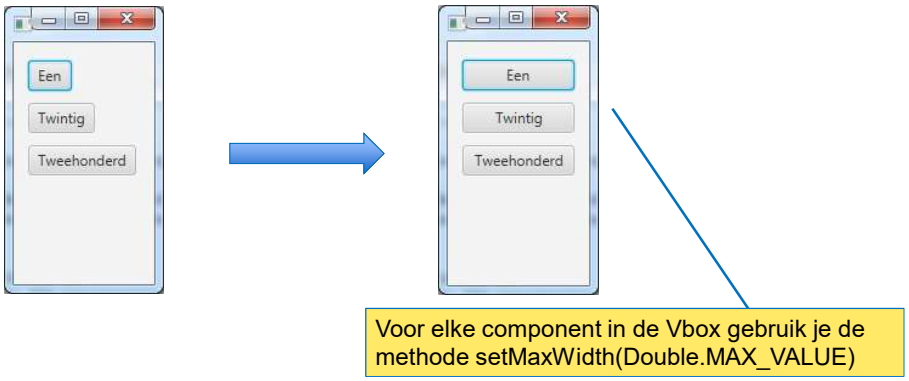
In een AnchorPane kunnen er ankerpunten geplaatst worden bij de top, left, right, center en bottom van de layout.

HoGent

40

5.7 Grootte van de nodes

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



HoGent 41

5.8 Uitlijning van de nodes

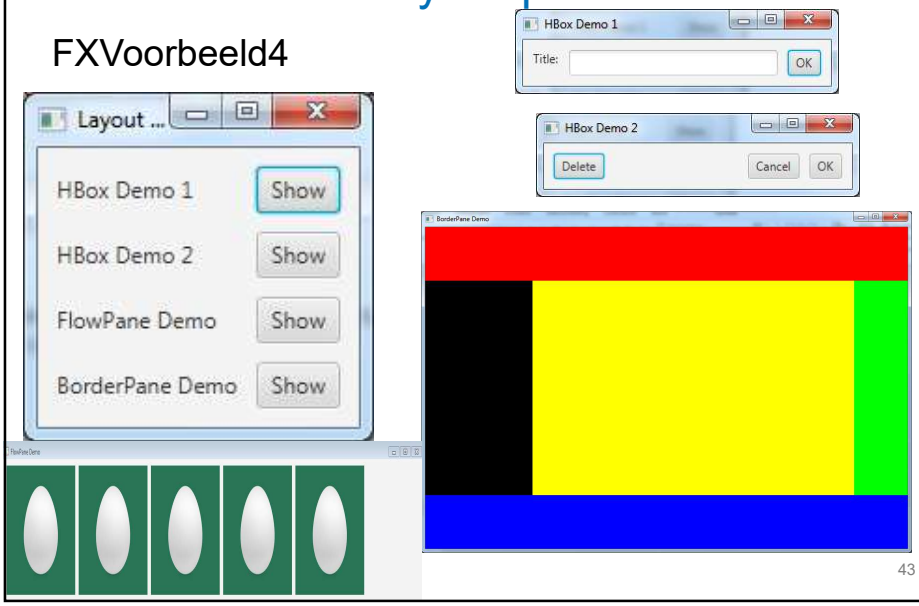
Gebruik de methode setAlignment(Pos value)

- 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
- FXVoorbeeld2: this.setAlignment(Pos.CENTER);

HoGent 42

5.9 Demo van layout panelen

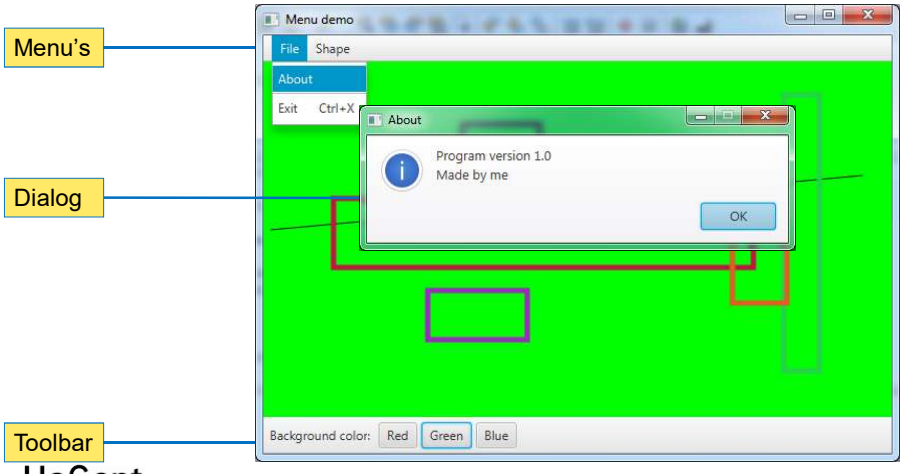
FXVoorbeeld4



43

6 Menu's, Toolbar en Dialog

FXVoorbeeld5



HoGent

44

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

HoGent 45

Voorbeeld van een Menu

```
MenuBar menuBar = new MenuBar();
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();
    }
});
```

HoGent 46

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);
```

HoGent 47

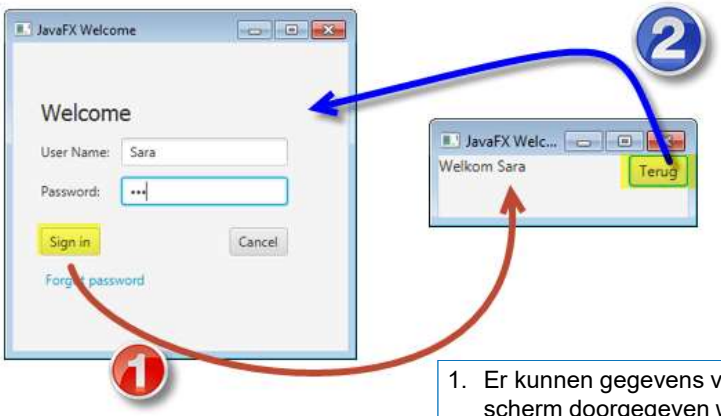
6.3 Dialog

- Vanaf JDK 8u40 (beschikbaar sinds maart 2015) zijn eenvoudige Dialogs en Alerts opgenomen in JavaFX 8
- Voorheen was men aangewezen op *openjfx-dialogs-1.x.x.jar* van ControlsFX
- Voorbeeld:

```
public void aboutClicked (ActionEvent e) {

    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("About");
    alert.setHeaderText(null);
    alert.setContentText("Program version 1.0\nMade by me");
    alert.showAndWait();
}
```

HoGent 48

7.1 Wisselen van scherm (FXVoorbeeld7)



1. Er kunnen gegevens van het ene scherm doorgegeven worden naar het andere.
2. Je kan terugkeren naar het scherm zoals het was, d.w.z. met de ingevulde gegevens.

HoGent

7.1 Wisselen van scherm (FXVoorbeeld7)

```
// 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();
}
```

HoGent

7.1 Wisselen van scherm (FXVoorbeeld7)

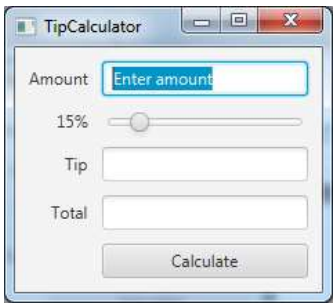
```
private void buildGui() {
    lblWelkom = new Label("Welkom " + inlognaam);
    btnTerug = new Button("Terug");
    Region spring = new Region();
    HBox.setHgrow(spring, Priority.ALWAYS);
    this.getChildren().addAll(lblWelkom,spring,btnTerug);

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

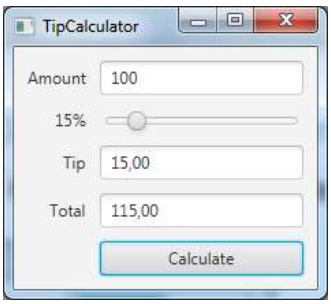
login is het attribuut waarin het vorige scherm wordt bijgehouden (doorgegeven via constructor)

HoGent

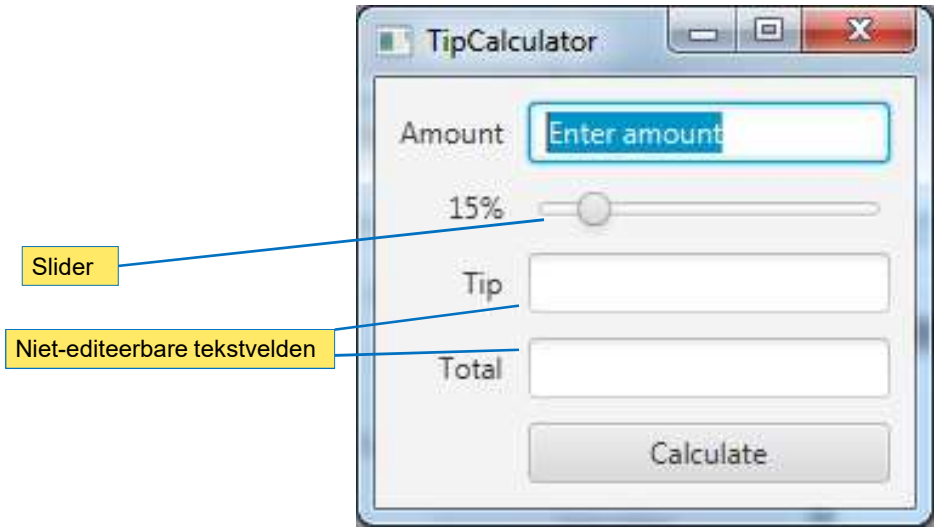
7.2 Tipcalculator (FXVoorbeeld6)

- 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



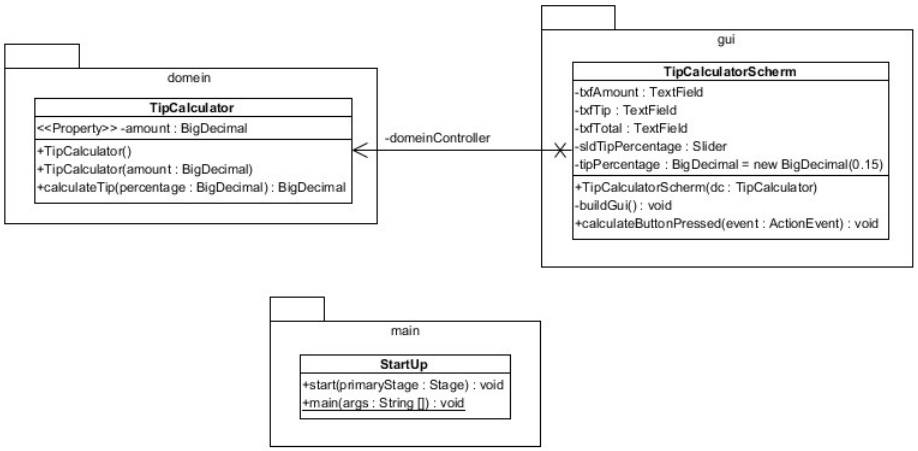


HoGent



HoGent 53

UML:



HoGent 54

- Niet- editeerbare tekstvelden:

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

- Event afhandeling bij Slider:

```
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))));
        }
    }
);
```

HoGent 55

8 Oefening JavaFX_RodeDraad – DEEL 1

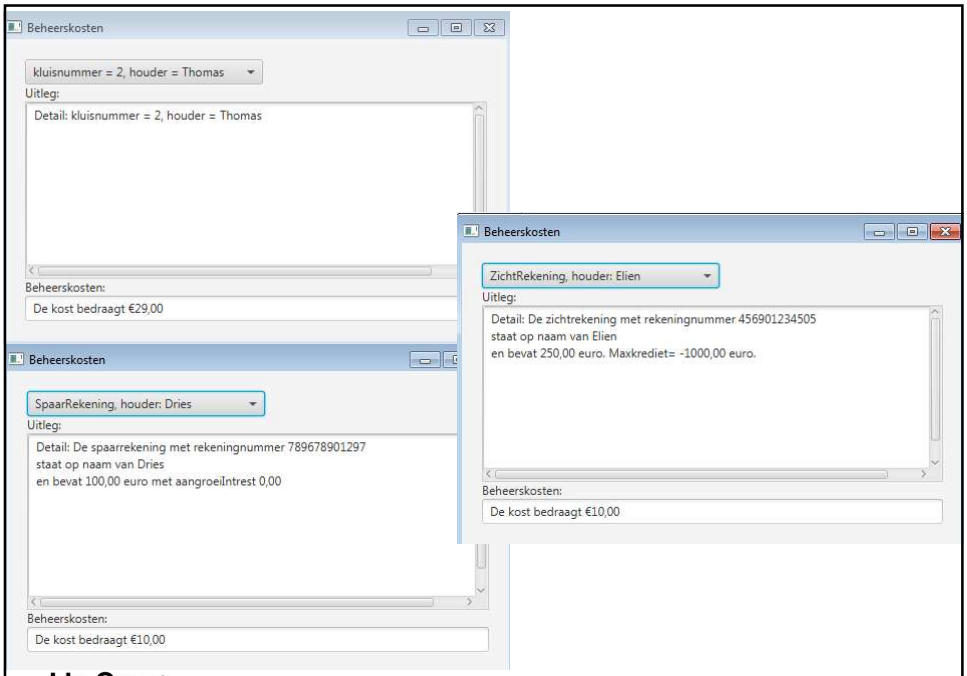
- Gegeven:

  zie Chamilo: JavaFX_RodeDraadStartversie_Deel1

  breid Oefening H10: Rode draad – deel 2 (zie H10) uit met een grafische user interface.

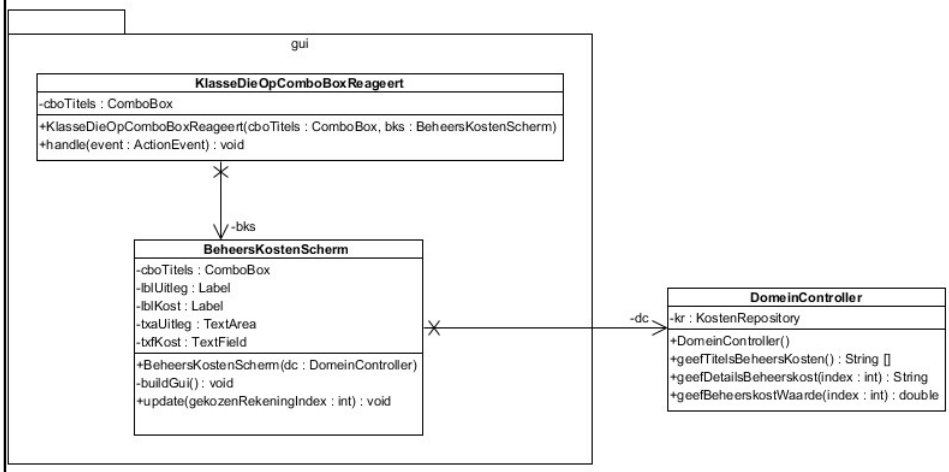
- Gevraagd: vul de gui-klasse BeheerskostenScherm aan volgens de UML en de richtlijnen.
  - Een combobox, gevuld met titels van items, wordt weergegeven. Een item is een object van een implementatie-klasse van de interface Beheerskosten.
  - De gebruiker kan een item selecteren uit de combobox om de details van het item en de beheerskosten weer te laten geven in een textarea en een textfield

HoGent 56



HoGent 57

UML:



HoGent 58

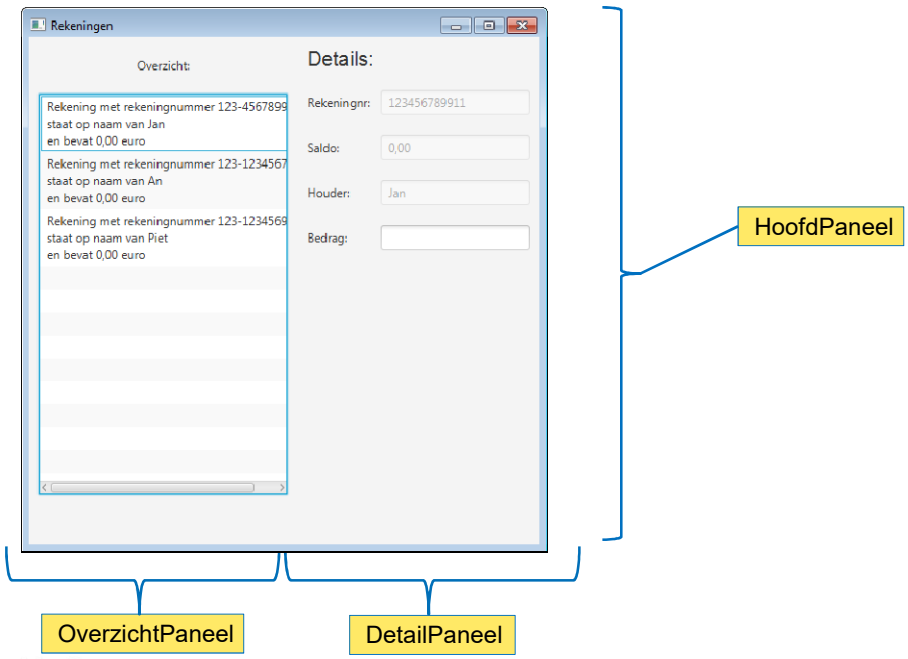
8 Oefening JavaFX_RodeDraad – DEEL 2

- Gegeven:

  Een klasse Rekening (JavaFX_RodeDraadStartversie_Deel2):
  - Attributen: rekeningNr (long), saldo (double) en houder (String)
  - Methoden: 2 constructoren, get- en setmethoden, stort, haalAf, schrijfBedragOverNaar en toString

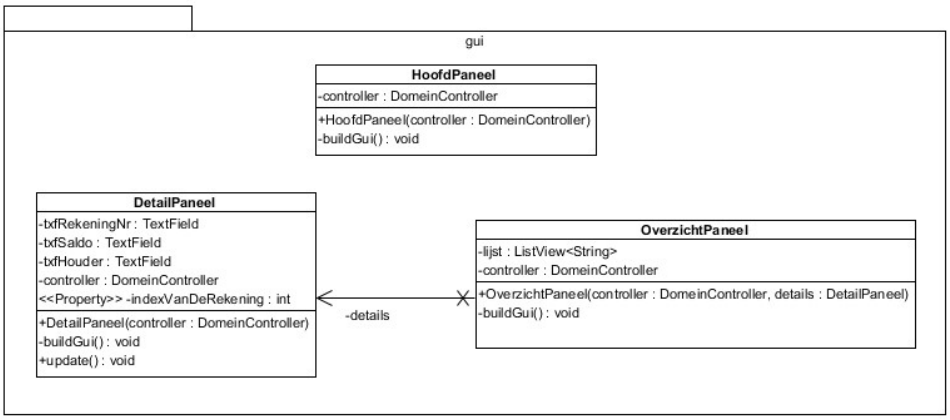
- Gevraagd:
  - Maak een overzicht van alle bestaande rekeningnummers in een OverzichtPaneel
  - Maak een DetailPaneel dat de details toont van de geselecteerde rekening. Wanneer een andere rekening geselecteerd wordt in het OverzichtPaneel dan wordt het DetailPaneel aangepast.
    Via het veld “bedrag” kan je geld storten op de geselecteerde rekening.

HoGent 59



HoGent 60

UML:



HoGent 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://fxexperience.com/controlsfx/
- http://www.java2s.com/
- http://code.makery.ch/

HoGent 62