

Odisee
DE CO-HOGESCHOOL

Java Fx

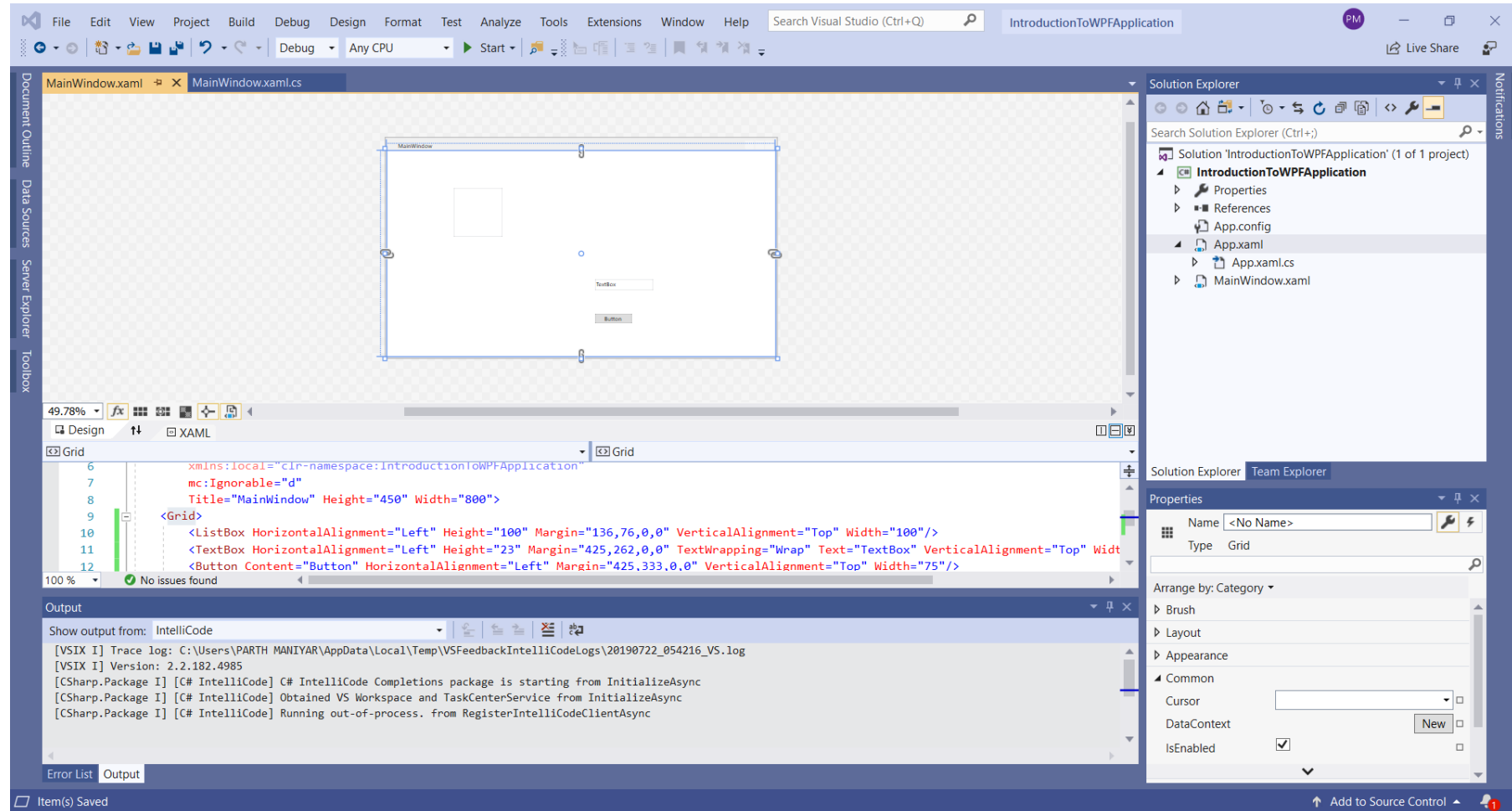


Maarten Troost – Jens Baetens

1.

Installatie en voorbeeld

Hoe maak je een GUI in C#?



Hoe maak je een GUI in C#?

```
<Window x:Class="TaxiApplication.WPF.MVVM.View.PaymentView"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:local="clr-namespace:TaxiApplication.WPF.MVVM.View"
mc:Ignorable="d"
WindowStartupLocation="CenterScreen"
ResizeMode="NoResize"
Height="600" Width="800">

<Grid Background="Beige">

    <Border BorderBrush="Black" BorderRadius="5" BorderThickness="1" HorizontalAlignment="Center" Height="100">

        <Image x:Name="imgPaymentType" Source="../../Images/PaymentOptions.png" HorizontalAlignment="Left" Height="100" Width="100" Margin="0,0,0,0" />

        <RadioButton x:Name="rdbVisa" HorizontalAlignment="Left" Height="18" Margin="227,160,0,0" VerticalAlignment="Top" />
        <RadioButton x:Name="rdbMasterCard" HorizontalAlignment="Left" Height="18" Margin="341,160,0,0" VerticalAlignment="Top" />
        <RadioButton x:Name="rdbMaestro" HorizontalAlignment="Left" Height="18" Margin="451,160,0,0" VerticalAlignment="Top" />
        <RadioButton x:Name="rdbBancontact" HorizontalAlignment="Left" Height="18" Margin="565,160,0,0" VerticalAlignment="Top" />

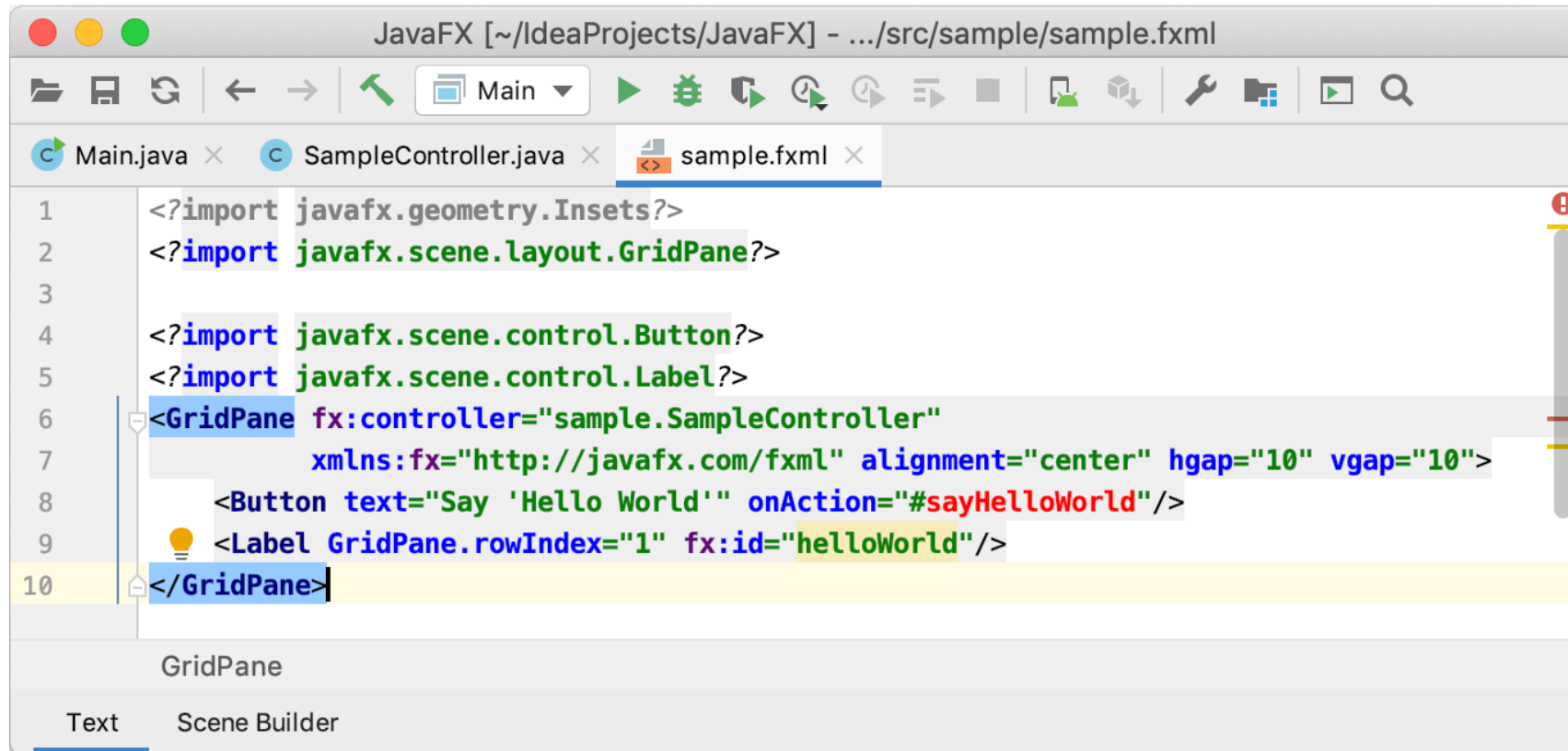
        <Label Content="Card" HorizontalAlignment="Left" Height="23" Margin="191,202,0,0" VerticalAlignment="Top" />
        <TextBox x:Name="txtCardType" HorizontalAlignment="Left" Height="23" Margin="324,202,0,0" TextWrapping="Wrap" />

        <Label Content="Account number" HorizontalAlignment="Left" Height="23" Margin="191,230,0,0" VerticalAlignment="Top" />
        <TextBox x:Name="txtAccountNumber" HorizontalAlignment="Left" Height="23" Margin="324,230,0,0" TextWrapping="Wrap" />

    </Border>

</Grid>
```

In Java via JavaFx

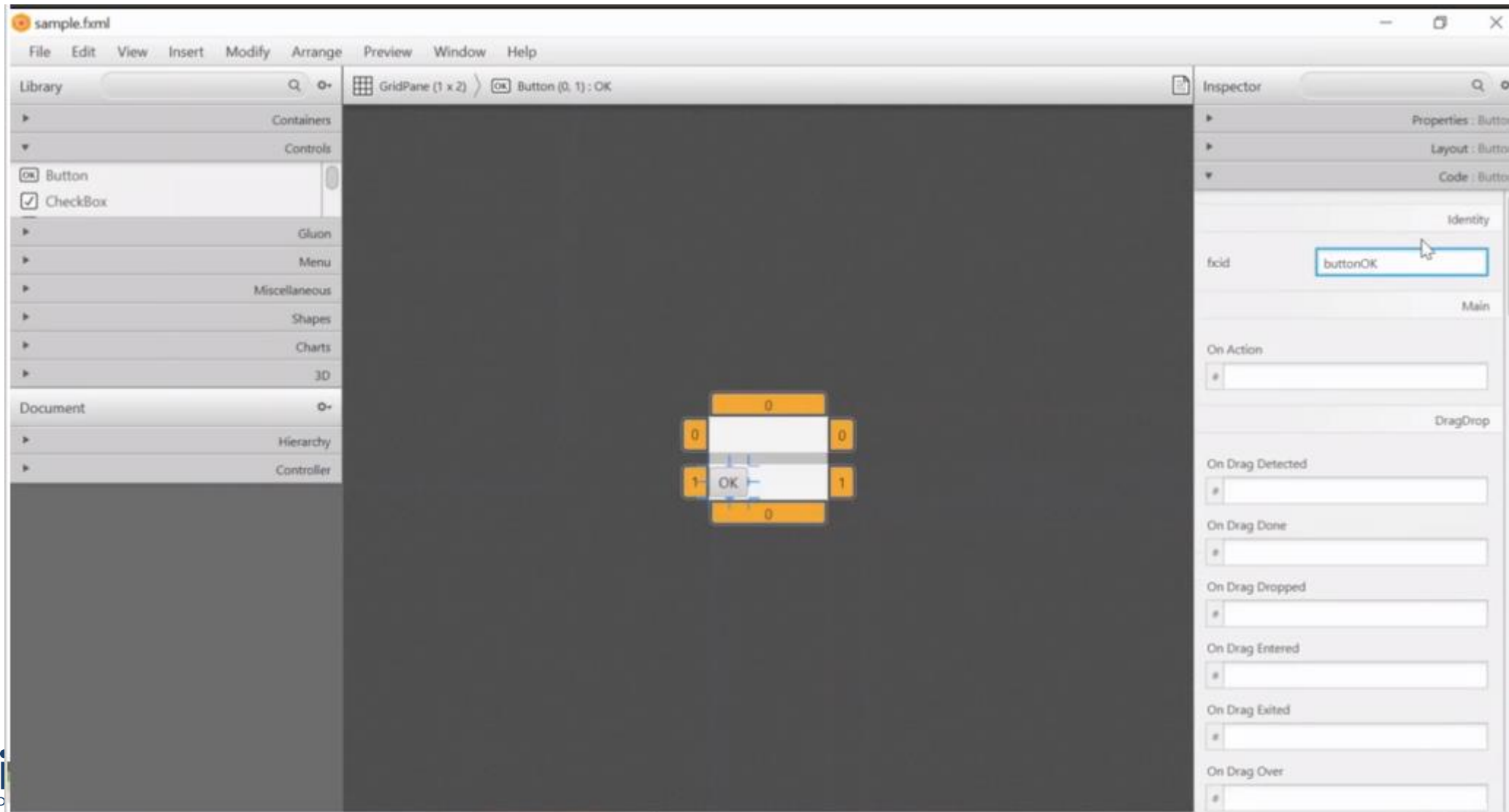


The screenshot shows an IDE window titled "JavaFX [~/IdeaProjects/JavaFX] - .../src/sample/sample.fxml". The window contains three tabs: "Main.java", "SampleController.java", and "sample.fxml". The "sample.fxml" tab is active, displaying the following XML code:

```
1 <?import javafx.geometry.Insets?>
2 <?import javafx.scene.layout.GridPane?>
3
4 <?import javafx.scene.control.Button?>
5 <?import javafx.scene.control.Label?>
6 <GridPane fx:controller="sample.SampleController"
7     xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10" vgap="10">
8     <Button text="Say 'Hello World'" onAction="#sayHelloWorld"/>
9     <Label GridPane.rowIndex="1" fx:id="helloWorld"/>
10 </GridPane>
```

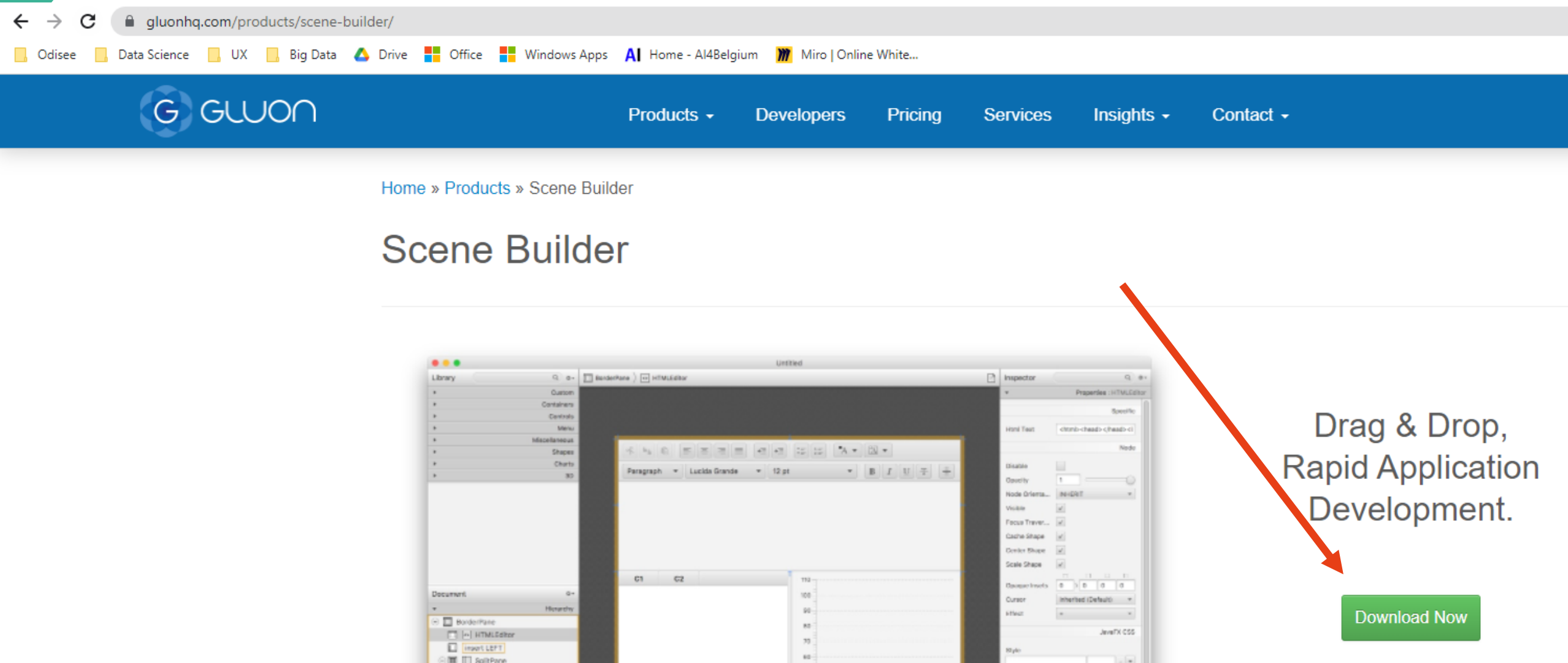
The code is color-coded: XML tags are in blue, attributes and values are in green, and text content is in black. A yellow highlight is under the closing tag of the `<GridPane>` element on line 10. A red exclamation mark icon is visible in the top right corner of the editor area. At the bottom of the window, there are two tabs: "Text" and "Scene Builder".

Externe GUI om GUI's te bouwen voor Java Fx



Hoe installeren?

<https://gluonhq.com/products/scene-builder/>



The screenshot shows the Gluon website's 'Scene Builder' product page. The browser's address bar displays 'gluonhq.com/products/scene-builder/'. The website's navigation bar includes links for Products, Developers, Pricing, Services, Insights, and Contact. The main heading reads 'Scene Builder'. Below this, a preview of the Scene Builder application is shown. The application interface features a 'Library' on the left with categories like Custom, Containers, Controls, Menu, Miscellaneous, Shapes, and Charts. The central workspace is titled 'Untitled' and contains a 'Paragraph' widget. On the right, an 'Inspector' panel shows properties for the selected widget, including 'Html Text', 'Node', 'Opacity', 'Node Orientation', 'Visible', 'Focus Traversal', 'Cache Shape', 'Center Shape', 'Scale Shape', 'Group Insets', 'Cursor', and 'Style'. A red arrow points from the text 'Drag & Drop, Rapid Application Development.' to a green 'Download Now' button.

Home » Products » Scene Builder

Scene Builder

Drag & Drop,
Rapid Application
Development.

Download Now

Hoe installeren?

Download Scene Builder

Scene Builder **18.0.0** was released on **March 31, 2022**.

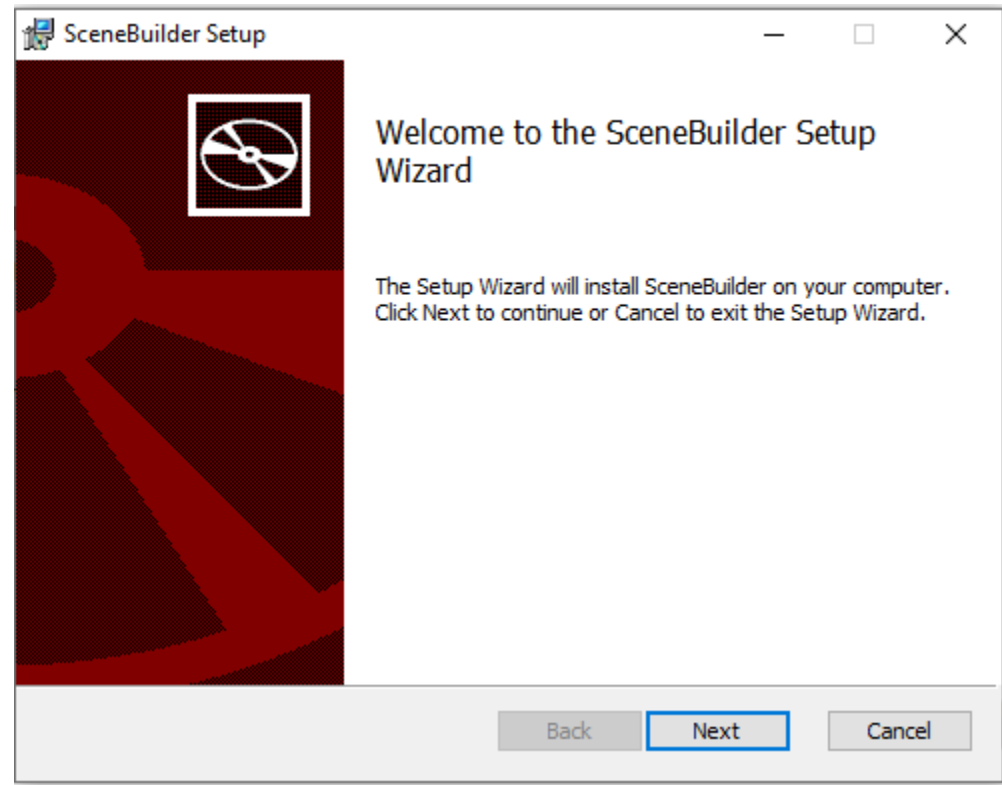
*You can use this Scene Builder version together with **Java 11 and higher**.*

Product	Platform	Download
Scene Builder	Windows Installer	Download
Scene Builder	Mac OS X dmg (Intel)	Download
Scene Builder	Mac OS X dmg (Apple Silicon)	Download
Scene Builder	Linux RPM	Download
Scene Builder	Linux Deb	Download
Scene Builder Kit info	Jar File	Download

Kies de juiste versie

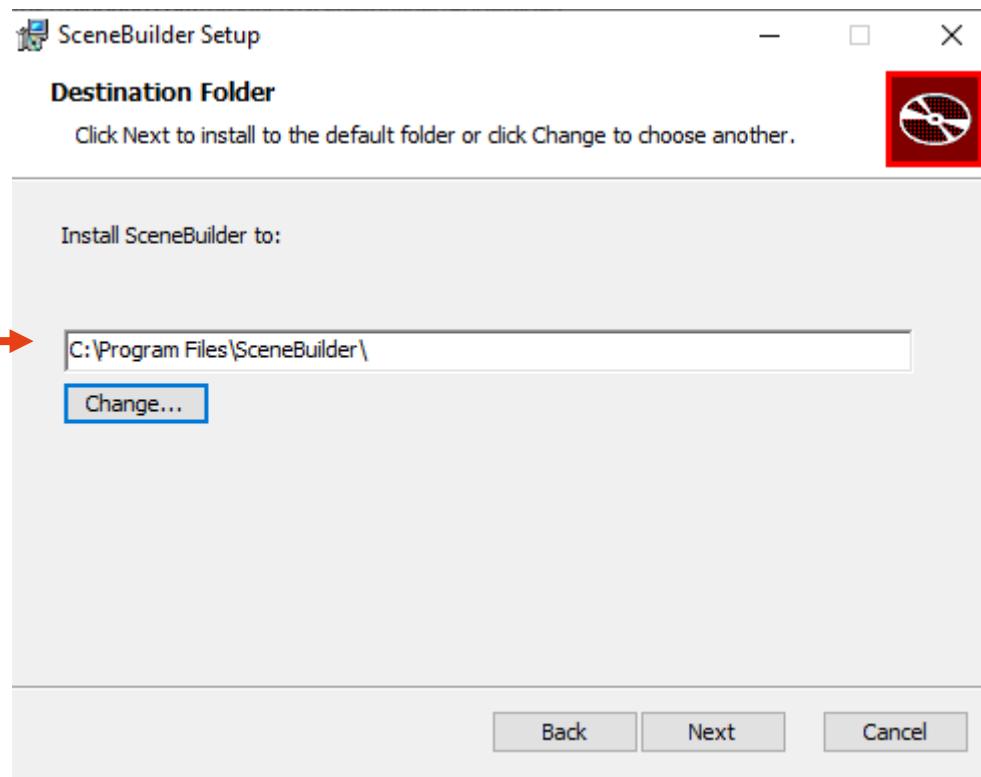


Hoe installeren?

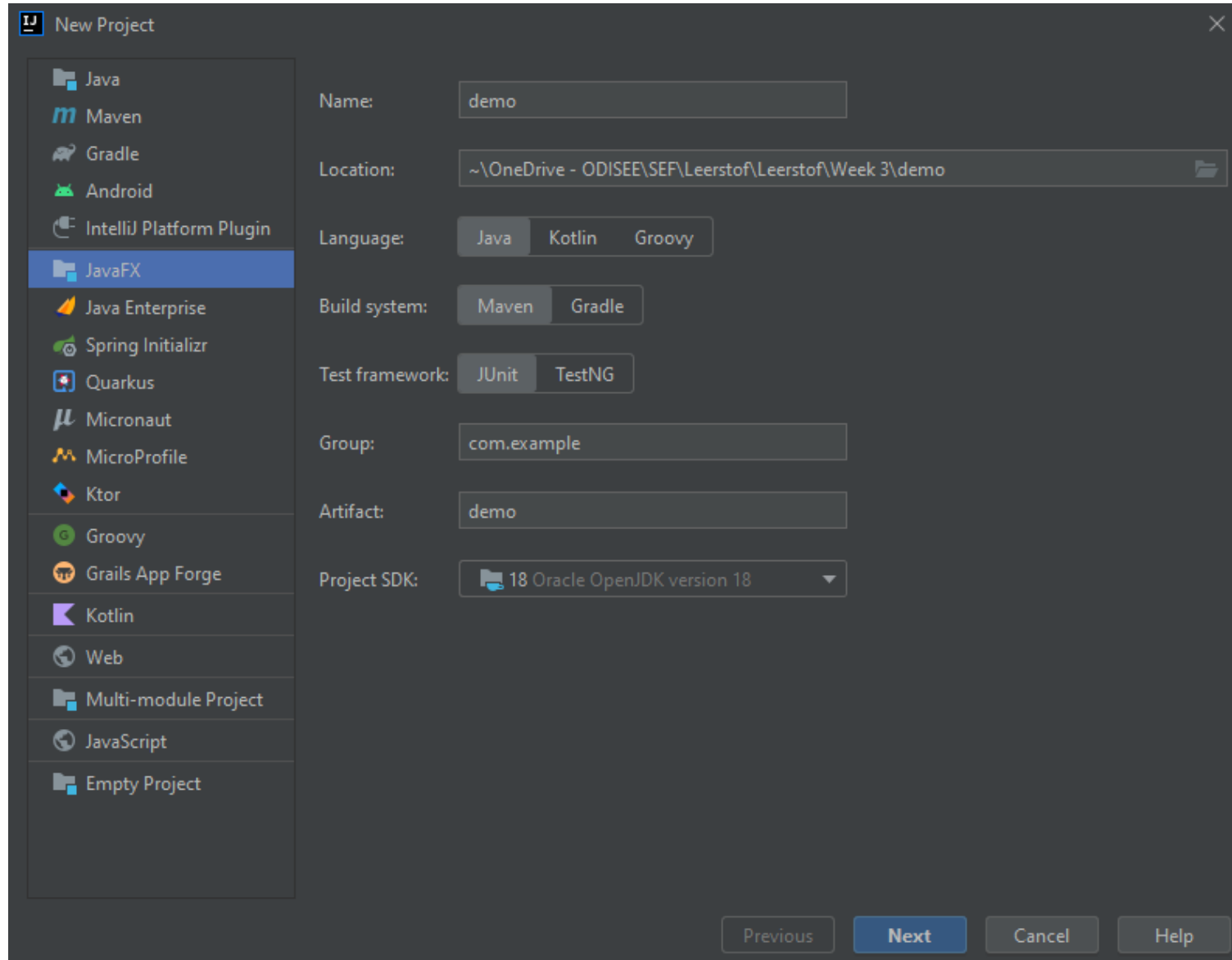


Hoe installeren?

Onthoud dit pad!



Maak nu een JavaFx Project in IntelliJ



Hello world - applicatie

■ Start applicatie

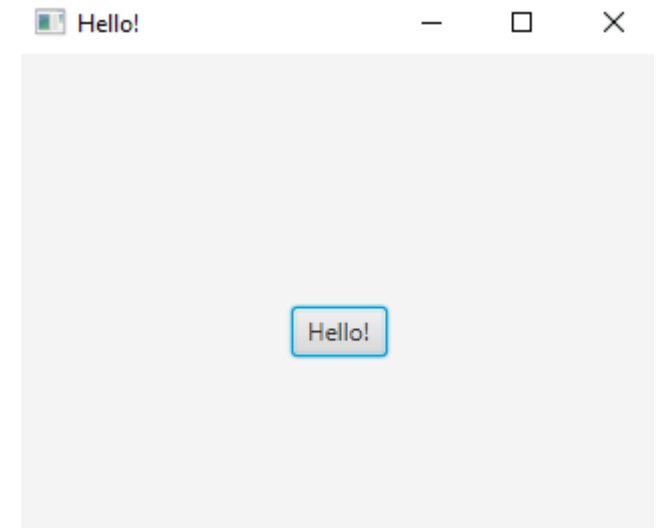
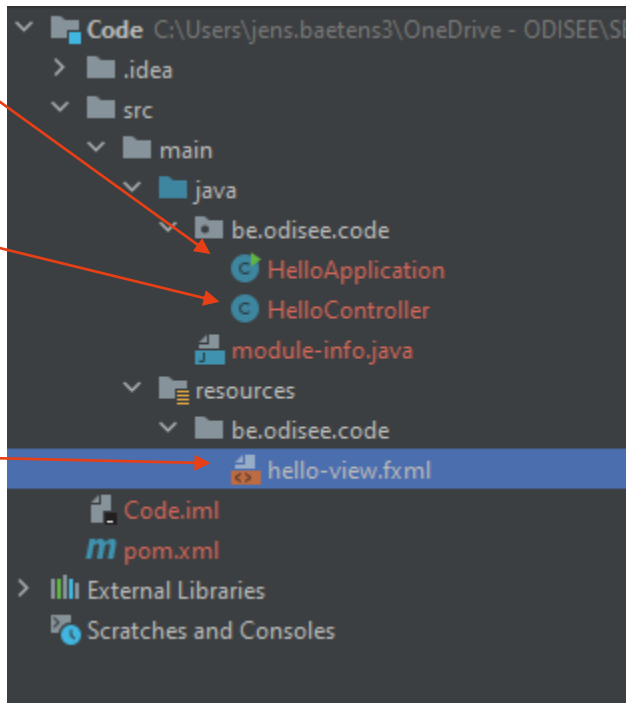
- Bevat main-functie

■ Controller

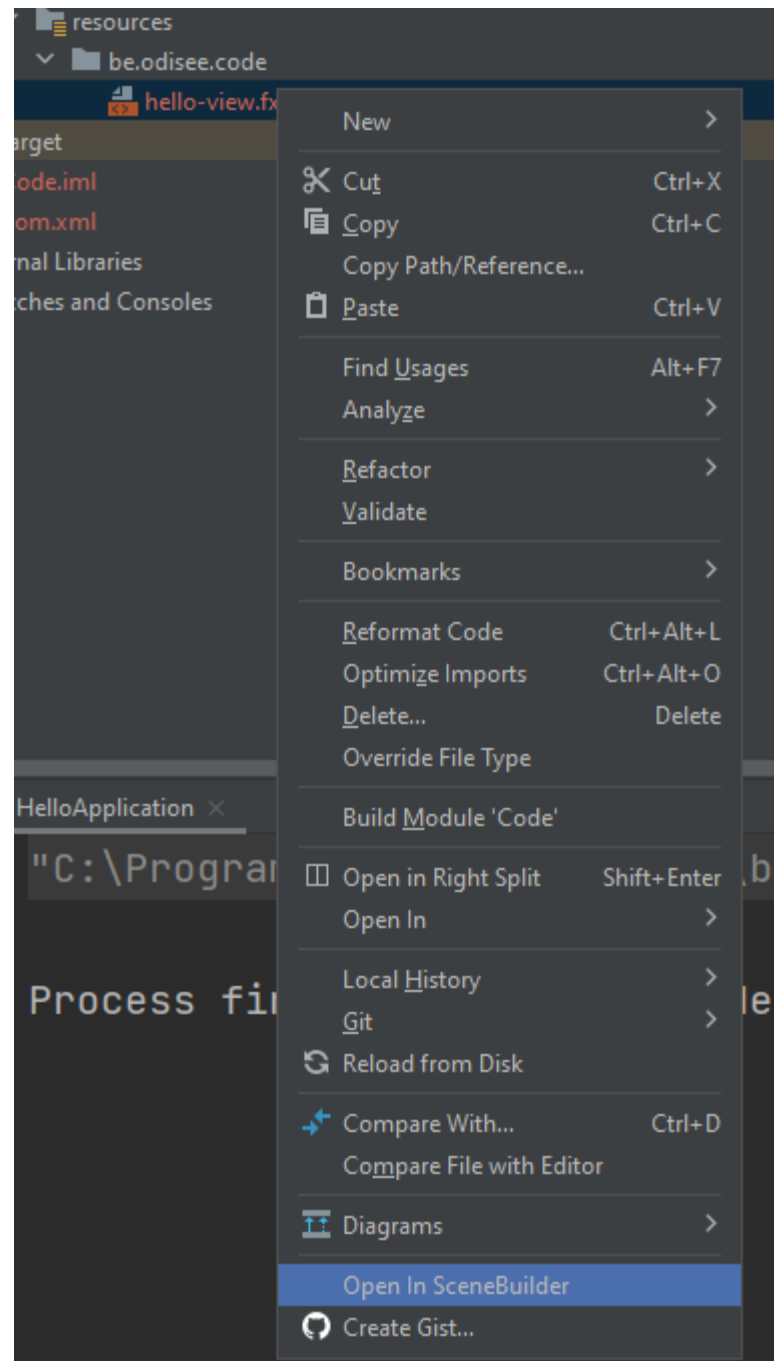
- Code-behind (.xaml.cs)
- 1 per view/fxml file

■ View

- Bevat Gui-elementen
- Layout
- .xaml in C#

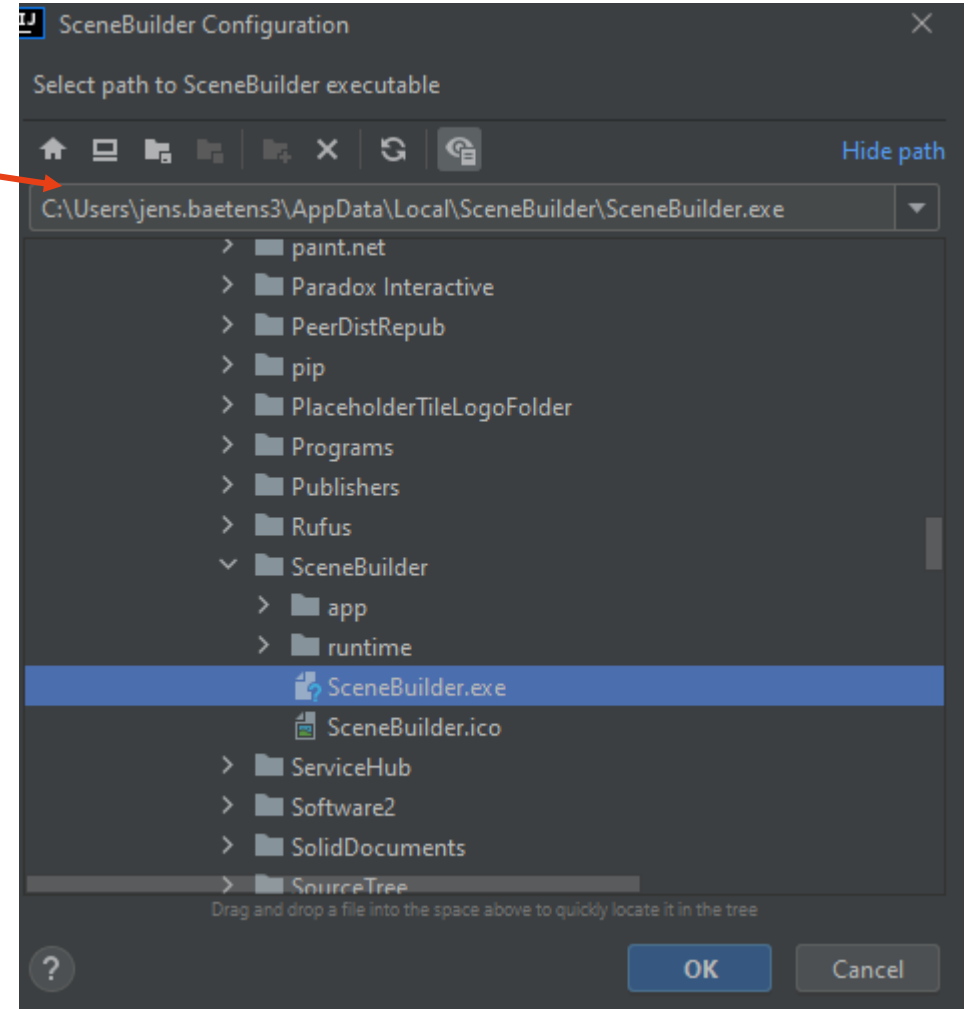


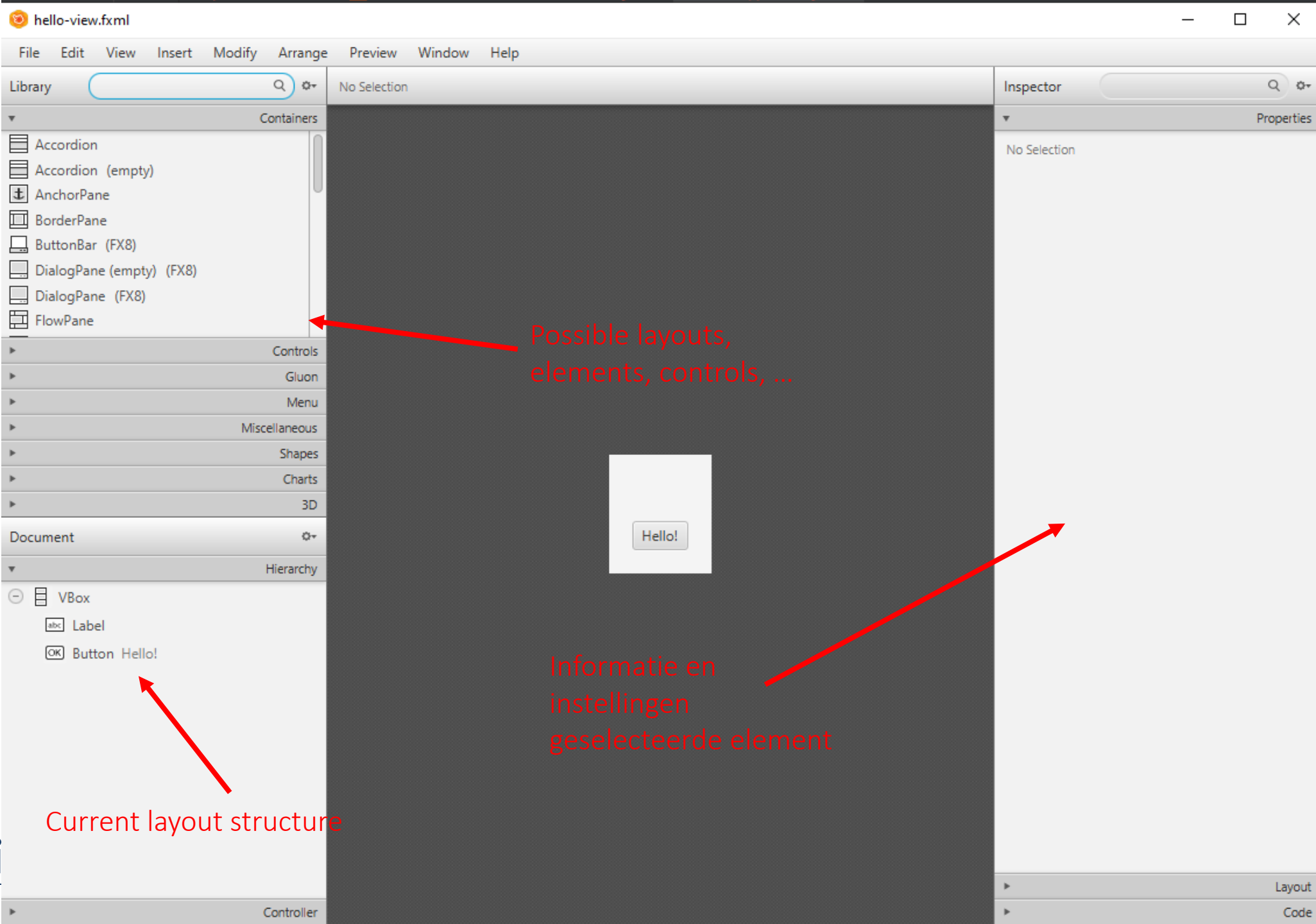
Open fxml file in SceneBuilder



Open fxml file in SceneBuilder

- Selecteer .exe waar het geïnstalleerd is





Soorten elementen

▣ Containers

- Breng structuur aan je layout: VBox, Hbox, GridPane, TabPane, ...

▣ Controls

- Basis UI elementen: Labels, Buttons, TextFields, Sliders, ...

▣ Menu

- Controls voor een menubar bovenaan je scherm aan te maken en te configureren

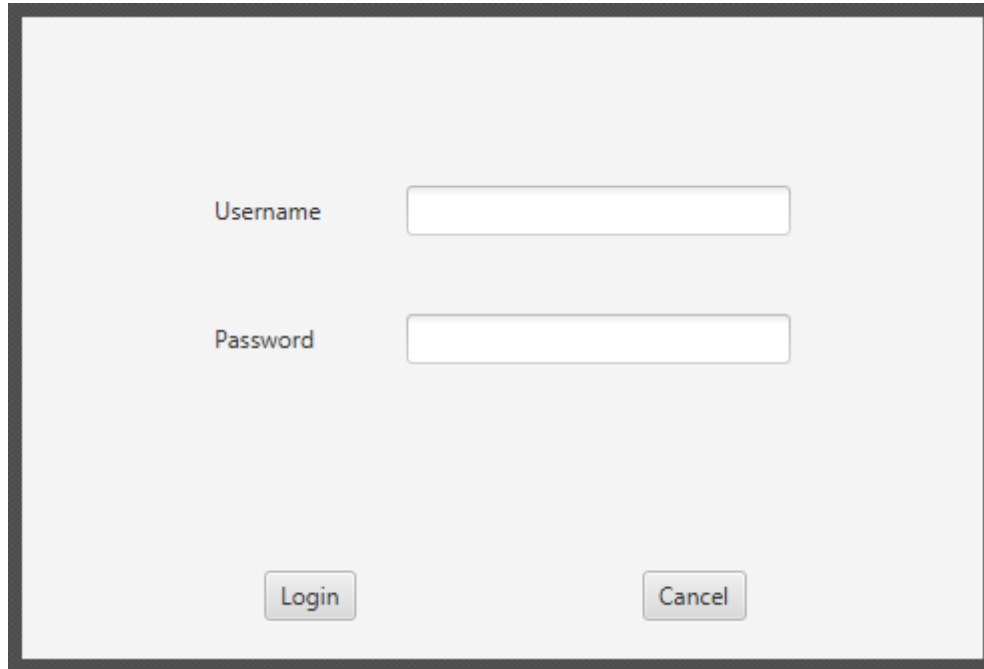
▣ Miscellaneous, Shapes, Charts, 3D

- Meer geavanceerde UI-elementen

Hoe aanpassen layout: LoginWindow

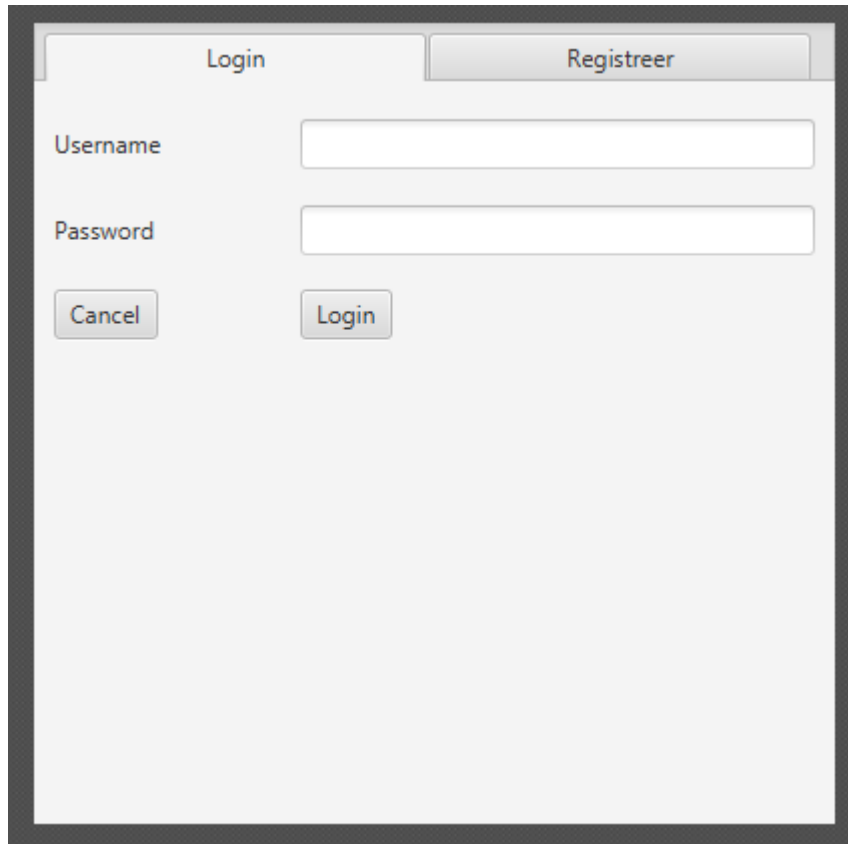
- ▣ Verwijder alle elementen
- ▣ Maak een grid aan van 5 op 5
- ▣ Zet preferred width en height op 320 en 480 pixels
- ▣ Voeg 2 buttons toe op onderste rij met tekst login en cancel
 - ▬ Zorg dat ze centraal staan in hun kolom
- ▣ Vervolledig het scherm met labels en textvelden toe te voegen voor username en password
 - ▬ Textvelden zijn twee kolommen breed

LoginWindow resultaat



```
<GridPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="320.0" prefWidth="480.0" />
<columnConstraints>
  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
</columnConstraints>
<rowConstraints>
  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
</rowConstraints>
<children>
  <Button mnemonicParsing="false" text="Login" GridPane.columnIndex="1" GridPane.halignment="CENTER" GridPane.rowIndex="4" />
  <Button mnemonicParsing="false" text="Cancel" GridPane.columnIndex="3" GridPane.halignment="CENTER" GridPane.rowIndex="4" />
  <Label text="Username" GridPane.columnIndex="1" GridPane.rowIndex="1" />
  <Label text="Password" GridPane.columnIndex="1" GridPane.rowIndex="2" />
  <TextField GridPane.columnIndex="2" GridPane.columnSpan="2" GridPane.rowIndex="1" />
  <TextField GridPane.columnIndex="2" GridPane.columnSpan="2" GridPane.rowIndex="2" />
</children>
</GridPane>
```

Oefening maak het volgende scherm na



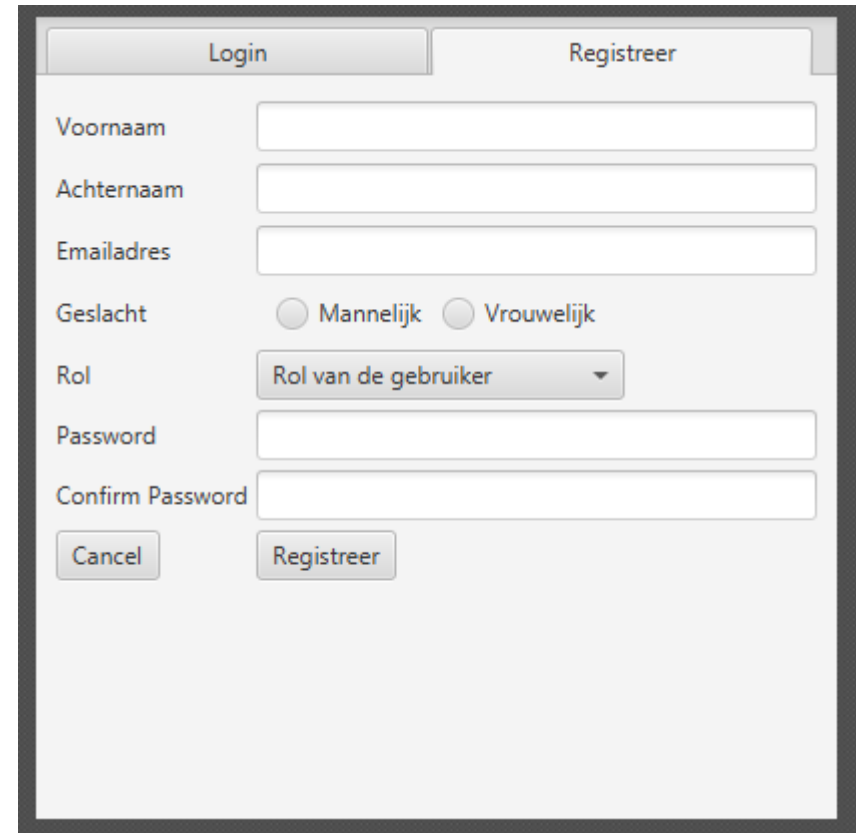
A mockup of a login form. It features two tabs at the top: 'Login' (selected) and 'Registreer'. Below the tabs, there are two input fields: 'Username' and 'Password'. At the bottom, there are two buttons: 'Cancel' and 'Login'.

Login Registreer

Username

Password

Cancel Login



A mockup of a registration form. It features two tabs at the top: 'Login' and 'Registreer' (selected). Below the tabs, there are several input fields: 'Voornaam', 'Achternaam', 'Emailadres', 'Geslacht' (with radio buttons for 'Mannelijk' and 'Vrouwelijk'), 'Rol' (with a dropdown menu showing 'Rol van de gebruiker'), 'Password', and 'Confirm Password'. At the bottom, there are two buttons: 'Cancel' and 'Registreer'.

Login Registreer

Voornaam

Achternaam

Emailadres

Geslacht ☐ Mannelijk ☐ Vrouwelijk

Rol

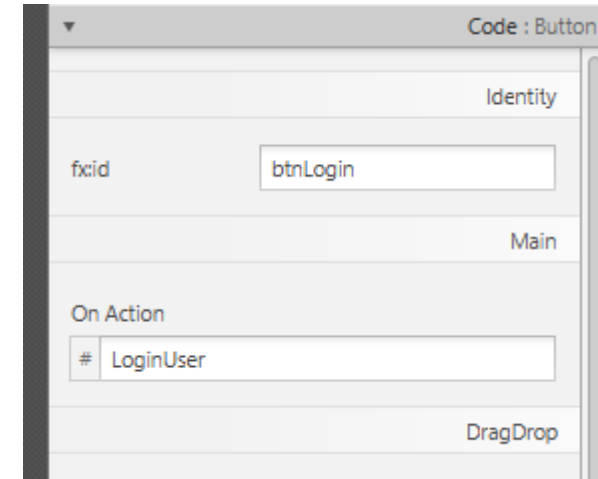
Password

Confirm Password

Cancel Registreer

Hoe gebeurt de koppeling tussen UI-elementen en code

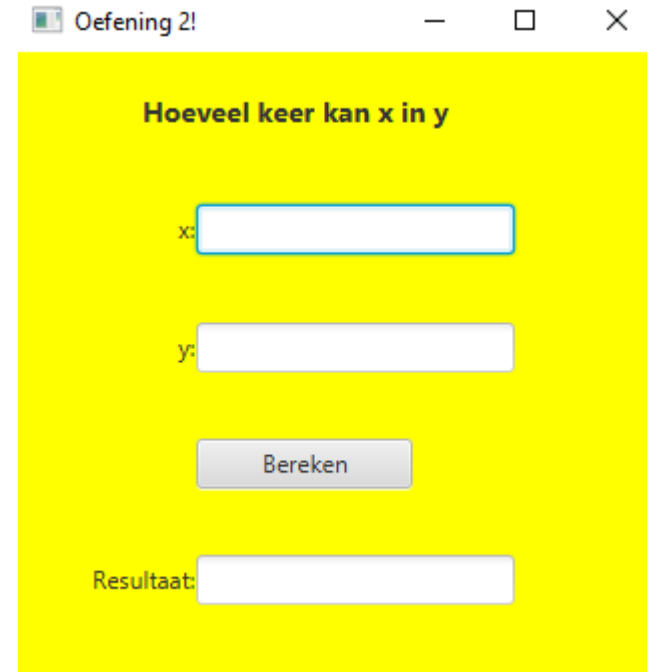
- Elk element kan een fx:id property hebben
 - Zie code tab aan de rechterkant
 - Ook eventhandlers kunnen ingesteld worden
- Link in code gebeurt aan de hand van dit id en de annotatie @FXML



```
public class DefeningLoginController {  
    @FXML  
    private TextField txtUsername;  
  
    @FXML  
    private Button btnLogin;  
  
    @FXML  
    protected void loginUser() {  
        String username = txtUsername.getText();  
    }  
}
```

Oefening 2 - java

- Maak een java class aan met naam GeheleDeling
 - Implementeer een method welke de deling berekent.
- Maak in de resources folder een nieuwe fxml file aan
 - Noem deze geheleDeling.fxml
 - Laat het view zo goed mogelijk lijken op de figuur hiernaast



The screenshot shows a JavaFX application window titled "Oefening 2!". The window has a yellow background and contains the following elements:


- A title bar with standard window controls (minimize, maximize, close).
- A header text: "Hoeveel keer kan x in y".
- Two input fields: "x:" and "y:", each followed by a text box.
- A button labeled "Bereken" (Calculate).
- A result label "Resultaat:" followed by a text box.

Oefening 2 – fxml

■ geheleDeling.fxml

- Zorg ervoor dat de fx:controller verwijst naar een klasse met naam GeheleDeling Controller
- Zorg ervoor dat de textvelden en de buttons een fx:id hebben
- De actie van de knop is: bereken

■ Pas de HelloApplication klasse aan zodat deze het nieuwe geheleDeling.fxml venster start



Oefening 2!

Hoeveel keer kan x in y

x:

y:

Bereken

Resultaat:



2.

Software architecturen

Software architectuur

- ▣ Structuur van een software systeem
- ▣ Blauwdruk voor de te ontwikkelen applicatie
- ▣ Moeilijk aan te passen na start ontwikkeling
- ▣ Keuzes voor structuur, taal, ... worden bepaald door de vereisten
 - Bvb: Voor de space shuttle moest de applicatie snel en betrouwbaar zijn

Waarom eerst de architectuur vastleggen?

- ▣ Maakt een begrijpbare abstractie van een complex systeem
- ▣ Basis om het gedrag te analyseren zonder code te schrijven
- ▣ Basis om hergebruik van keuzes en elementen te detecteren
- ▣ Ondersteunen van design keuzes om ontwikkeling en onderhoud van een applicatie te beïnvloeden
- ▣ Basis om communicatie over de applicatie te ondersteunen
- ▣ Reduceren van risico's en verbeteren slaagkansen
- ▣ Reduceren van de ontwikkelkosten

SOLID – principes van software architectuur

- ▣ S – Single Responsibility Principle
- ▣ O – Open-closed principle
- ▣ L – Liskov Substitution principle
- ▣ I – Interface Segregation Principle
- ▣ D – Dependency inversion principle



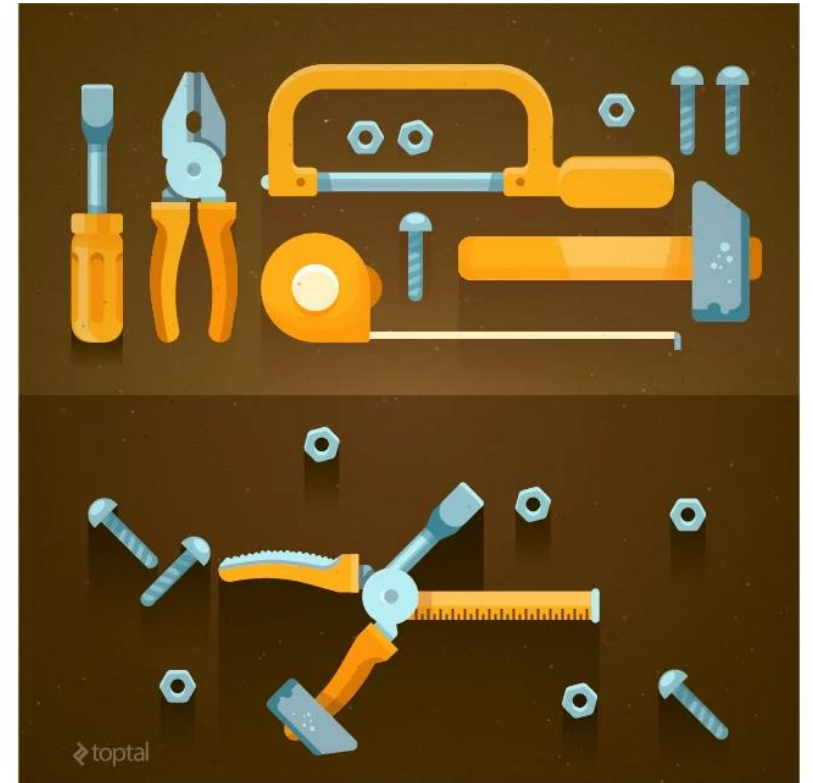
Robert C. Martin (aka Uncle Bob)

<https://www.youtube.com/watch?v=7EmboKQH8IM&list=PLUxsZVpqZTNShoypLQW9a4dEcffsoZT4k>



SOLID – principes van software architectuur

- ▣ S – Single Responsibility Principle
 - ▬ A class should have only one job
 - ▬ Bvb niet het berekenen van waarden en output genereren in 1 klasse
- ▣ O – Open-closed principle
- ▣ L – Liskov Substitution principle
- ▣ I – Interface Segregation Principle
- ▣ D – Dependency inversion principle



SOLID – principes van software architectuur

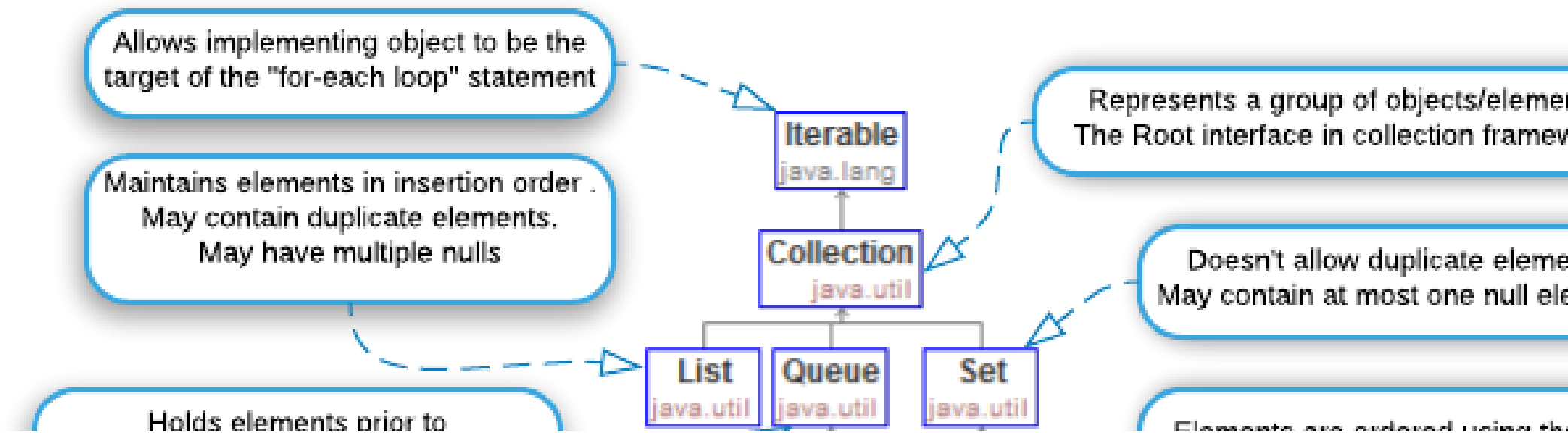
- ▣ S – Single Responsibility Principle
- ▣ O – Open-closed principle
 - Open for extension, closed for modifications
 - Bvb: uitbreidingen/wijzigingen aan algoritmes door dependency injection (interfaces)
- ▣ L – Liskov Substitution principle
- ▣ I – Interface Segregation Principle
- ▣ D – Dependency inversion principle

SOLID – principes van software architectuur

- ▣ S – Single Responsibility Principle
- ▣ O – Open-closed principle
- ▣ L – Liskov Substitution principle
 - A derived class should be substitutable for their base class
 - Resulteert in meer hergebruik van code en een lossere koppeling
- ▣ I – Interface Segregation Principle
- ▣ D – Dependency inversion principle

SOLID – principes van software architectuur

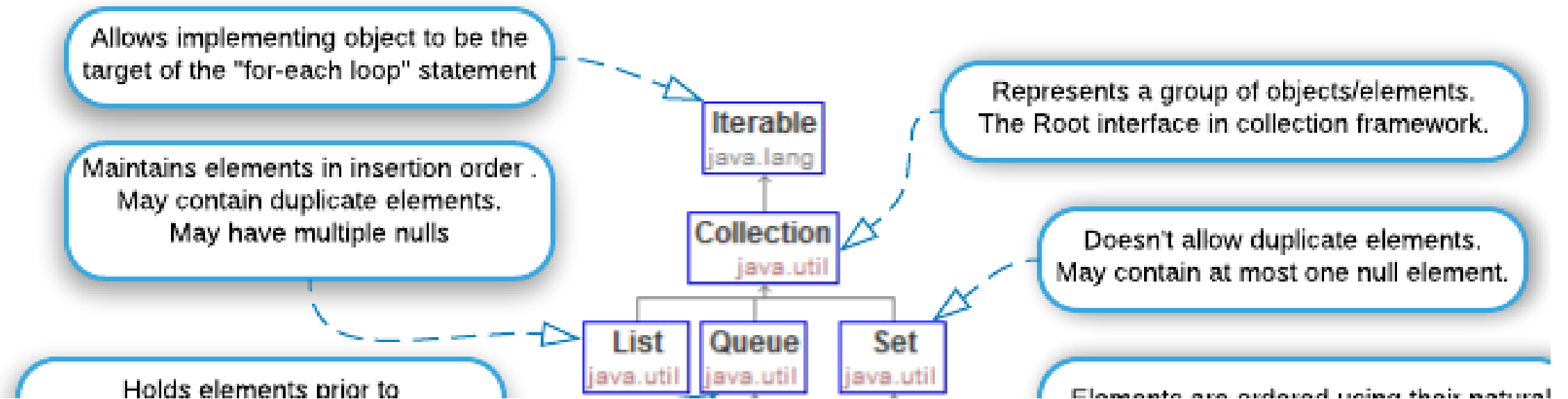
- S – Single Responsibility Principle
- O – Open-closed principle
- L – Liskov Substitution principle
- I – Interface Segregation principle
 - A client should only use the methods that it needs
- D – Dependency Inversion principle



SOLID – principes van software architectuur

■ I – Interface Segregation Principle

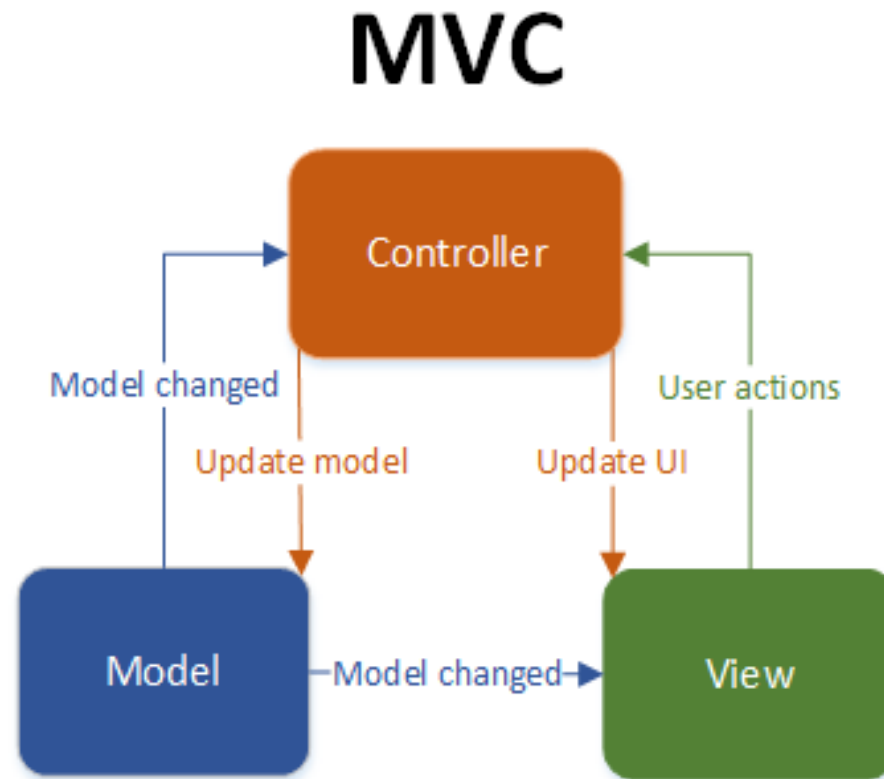
- A client should not be forced to depend on methods they do not use



SOLID – principes van software architectuur

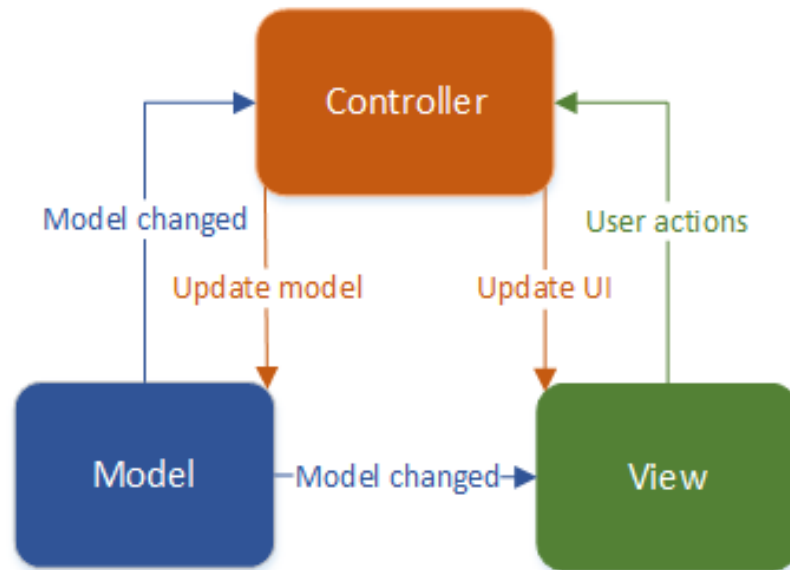
- ▣ S – Single Responsibility Principle
- ▣ O – Open-closed principle
- ▣ L – Liskov Substitution principle
- ▣ I – Interface Segregation Principle
- ▣ D – Dependency inversion principle
 - ▣ A higher level module should not depend on a lower level module but on abstractions of it

Model – View - Controller

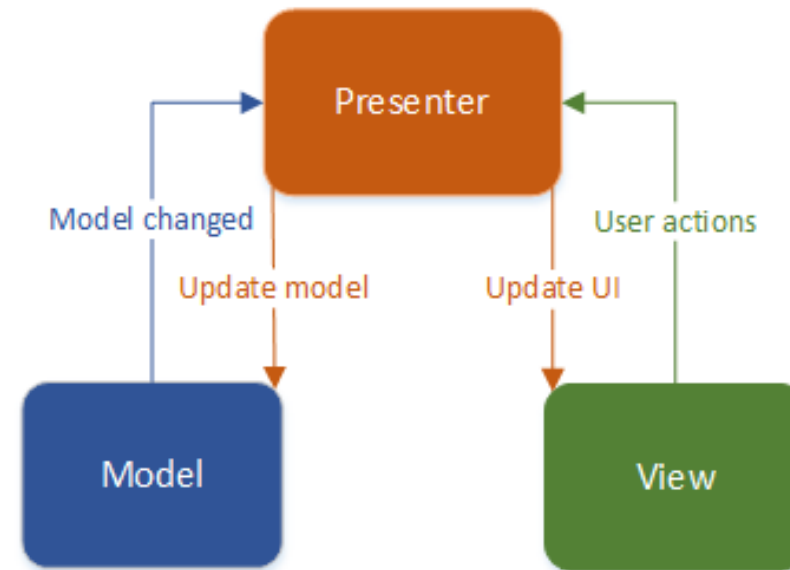


Model – View - Presenter

MVC



MVP





Voordelen MVC / MVP

- ▣ Eenvoudiger om om nieuwe schermen toe te voegen
- ▣ Vermijd complexiteit door applicatie te verdelen in meerdere MVC/MVP eenheden
- ▣ Laat toe om logisch gerelateerde acties te groeperen in een controller



Nadelen MVC / MVP

- ▣ Business logica is nog verwoven met de UI-elementen
- ▣ Moeilijk om unit tests te schrijven
- ▣ Hogere complexiteit van de applicatie
- ▣ Inefficiëntie van de data
 - Zelf data van het model in het view te plaatsen
- ▣ Ouder concept



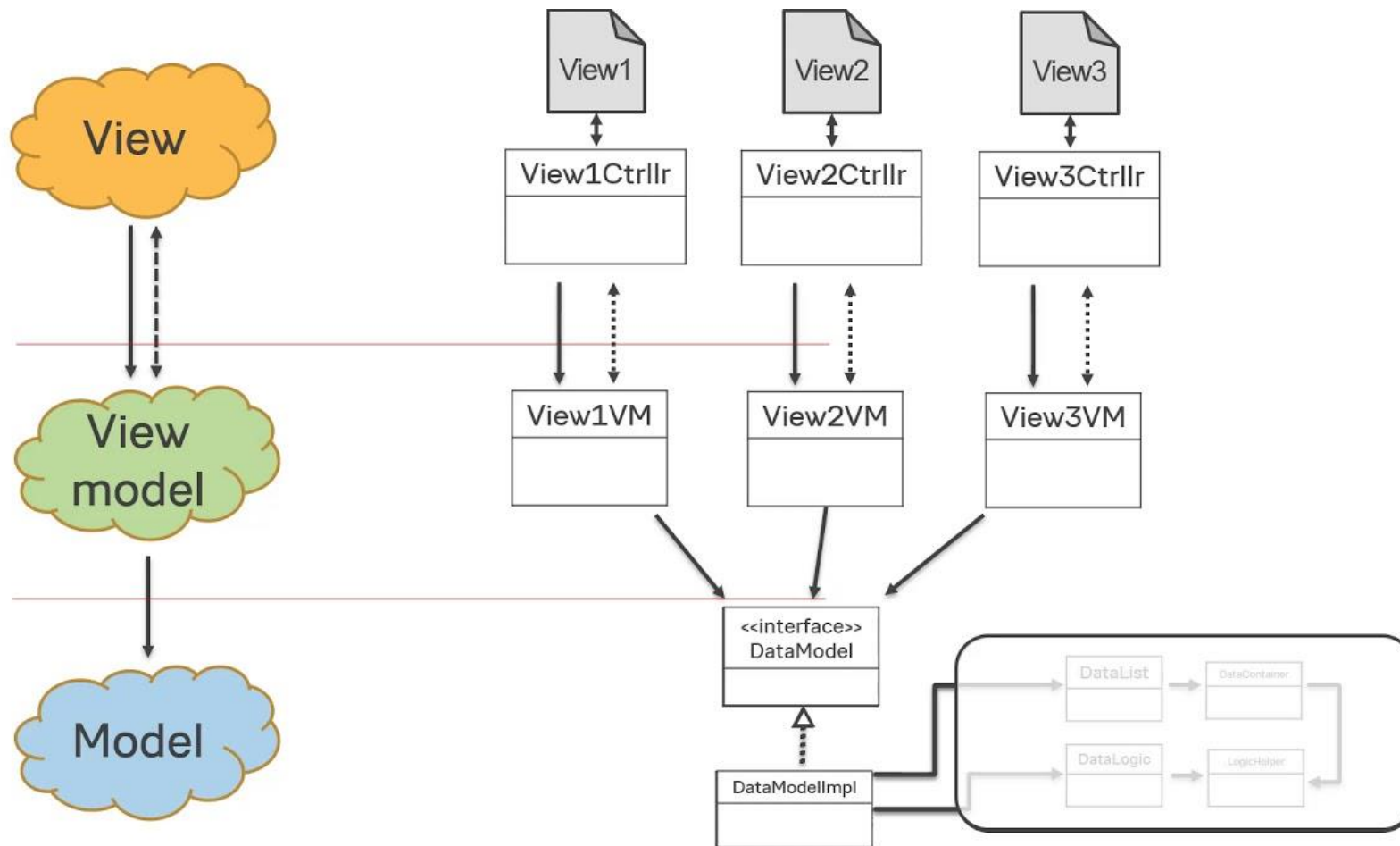
MVVM

- ▣ Verder scheiding tussen View/UI en business logica
 - Via commands en databinding
- ▣ Het view is de entry point van de applicatie
- ▣ Code/UI updates worden gedreven door events ipv functiecalls
- ▣ Maakt het mogelijk om business logica te testen zonder kennis van de UI
- ▣ Test Driven Development kan eenvoudig gebruikt worden
- ▣ Loosely coupled architecture



MVVM





<https://www.youtube.com/watch?v=mLzMI8wrHdY>

Oefening MVC

- ▣ Vertrek vanaf de login applicatie.
- ▣ Model – View – Controller : welk onderdelen zie je in de login app? Waar?
- ▣ Toe te voegen functionaliteit:
 - ▬ controle of de ingegeven login + password correct is
 - ▬ Afhankelijk van de controle een success of failure boodschap
- ▣ Welke functionaliteit hoort in welke laag (Model View Controller)?
- ▣ In welke class?

Variabelen en datatypes



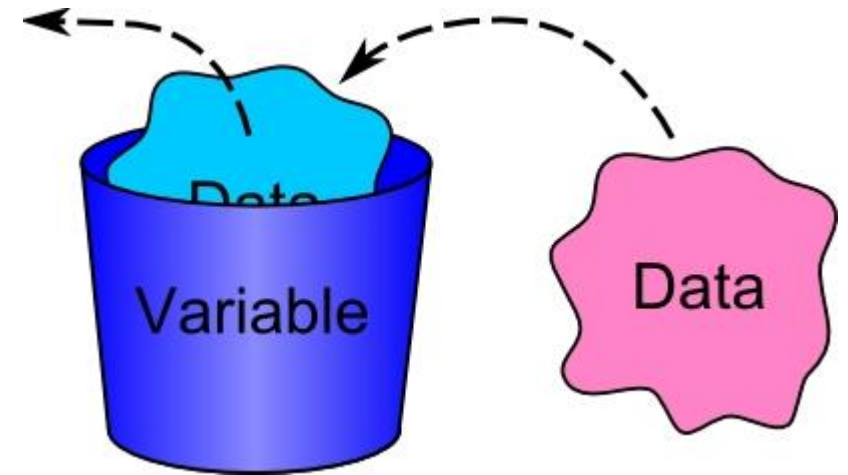
Maarten Troost – Jens Baetens

Variabele? Wasda?

In een programmeertaal gebruiken we variabelen om wisselende data in op te slaan.

```
int var=5;
```

Bij het uitvoeren van een programma, waar is die variabele op een fysieke computer?
Hoe worden meerdere variabelen van elkaar onderscheiden?



Variabele? Wasda?

- Een **variabele** is een **geheugenplaats** die via een naam wordt benaderd. De naam van de variabele is niets anders dan een **symbool** voor het adres in het geheugen.
 - ▬ Geheugenplaatsen = RAM, SSD, HDD, ...
 - ▬ Een plaats in het geheugen = een adres in het geheugen
vb 0x0003C4
 - ▬ Na compilatie geen namen meer maar enkel adressen

Variabelen in java

- ▣ Een waarde toekennen aan een variabele betekent dat we gegevens stoppen in het geheugen waarnaar de variabele verwijst. In Java gebruiken we hiervoor het “=”-teken ook de assignment operator genoemd.

Vb. `int var = 6;`

- ▣ Waarden vergelijken doen we met “==” (en “!=”), comparison operators voor primitieve types, de `.equals` methode voor objecten.

```
if(var == 6) {  
    String str=new String("meh");  
    if(str.equals("beeeh")) {
```

```
@Override  
public boolean equals(Object obj) {
```

Operatoren – prioriteit en associativiteit

Precedence	Operator	Associativity	Operator
1	++	Right	Pre/post increment
	--	Right	Pre/post decrement
	+, -	Right	Unary plus or minus
	~	Right	Bitwise complement
	!	Right	Logical complement
	(cast)	Right	Cast
2	*, /, and %	Left	Multiplication, division, and modulus
3	+ and -	Left	Addition and subtraction
	+	Left	String concatenation
4	<<	Left	Left shift
	>>	Left	Right shift and sign fill
	>>>	Left	Right shift and zero fill
5	<, <=, >, >=	Left	Logical
	InstanceOf	Left	Type comparison
6	== and !=	Left	Equality and inequality
7	&	Left	Bitwise and Boolean AND
8	^	Left	Bitwise and Boolean XOR
9		Left	Bitwise and Boolean OR

Operatoren – prioriteit en associativiteit

Precedence	Operator	Associativity	Operator
10	&&	Left	Boolean AND
11		Left	Boolean OR
12	?:	Right	Conditional
13	=	Right	Assignment
	+=, -=, *=, /=, and %=	Right	Compound

Bron: Reese - OCA, Java SE7 Programmer Study Guide - Packt 2012

For operators, associativity means that when the same operator appears in a row, then which operator occurrence we apply first. In the following, let `q` be the operator

`a q b q c`

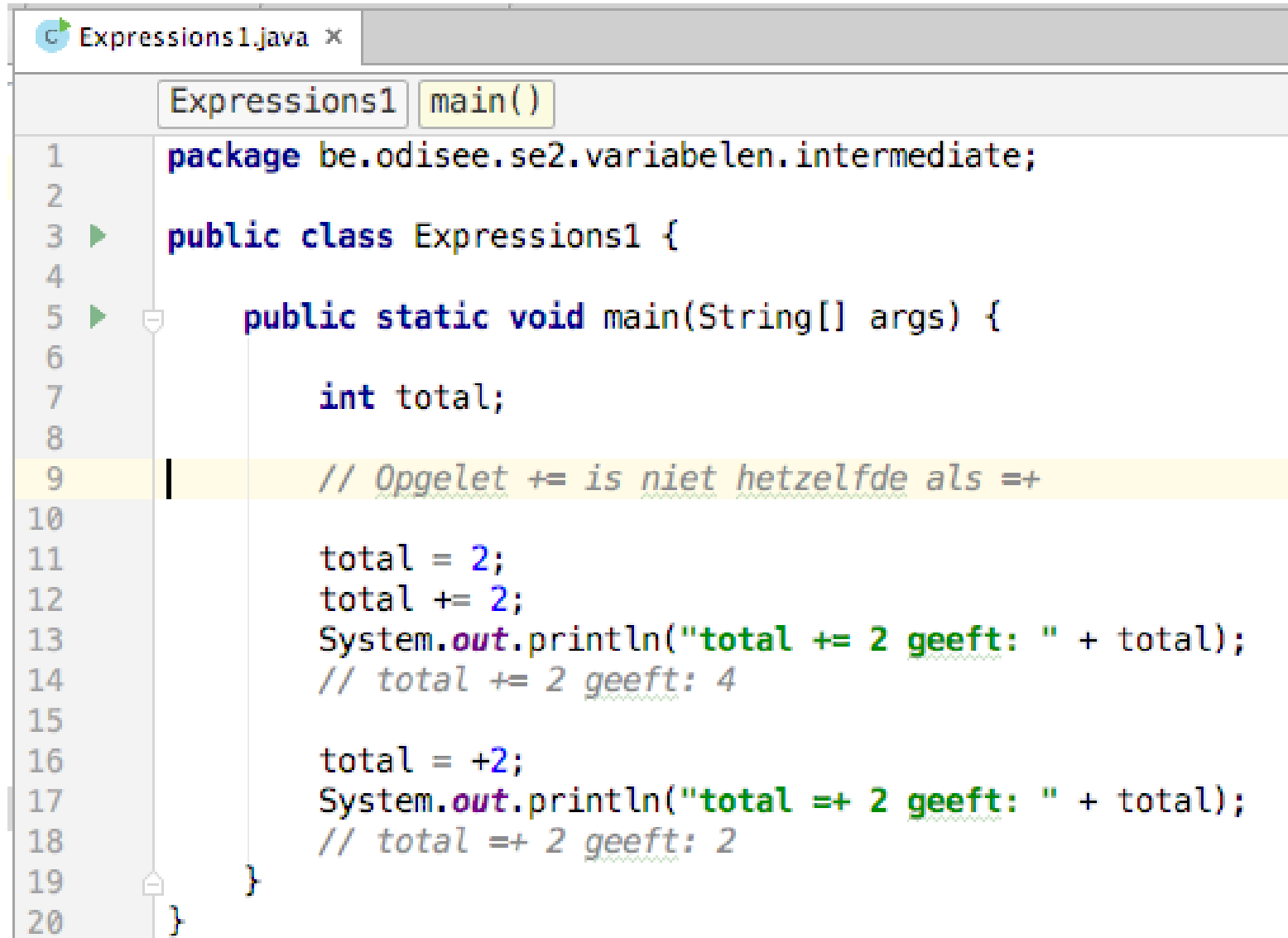
If `q` is left associative, then it evaluates as

`(a q b) q c`

And if it is right associative, then it evaluates as

`a q (b q c)`

Operatoren – oefening: leg uit wat de uitkomst is

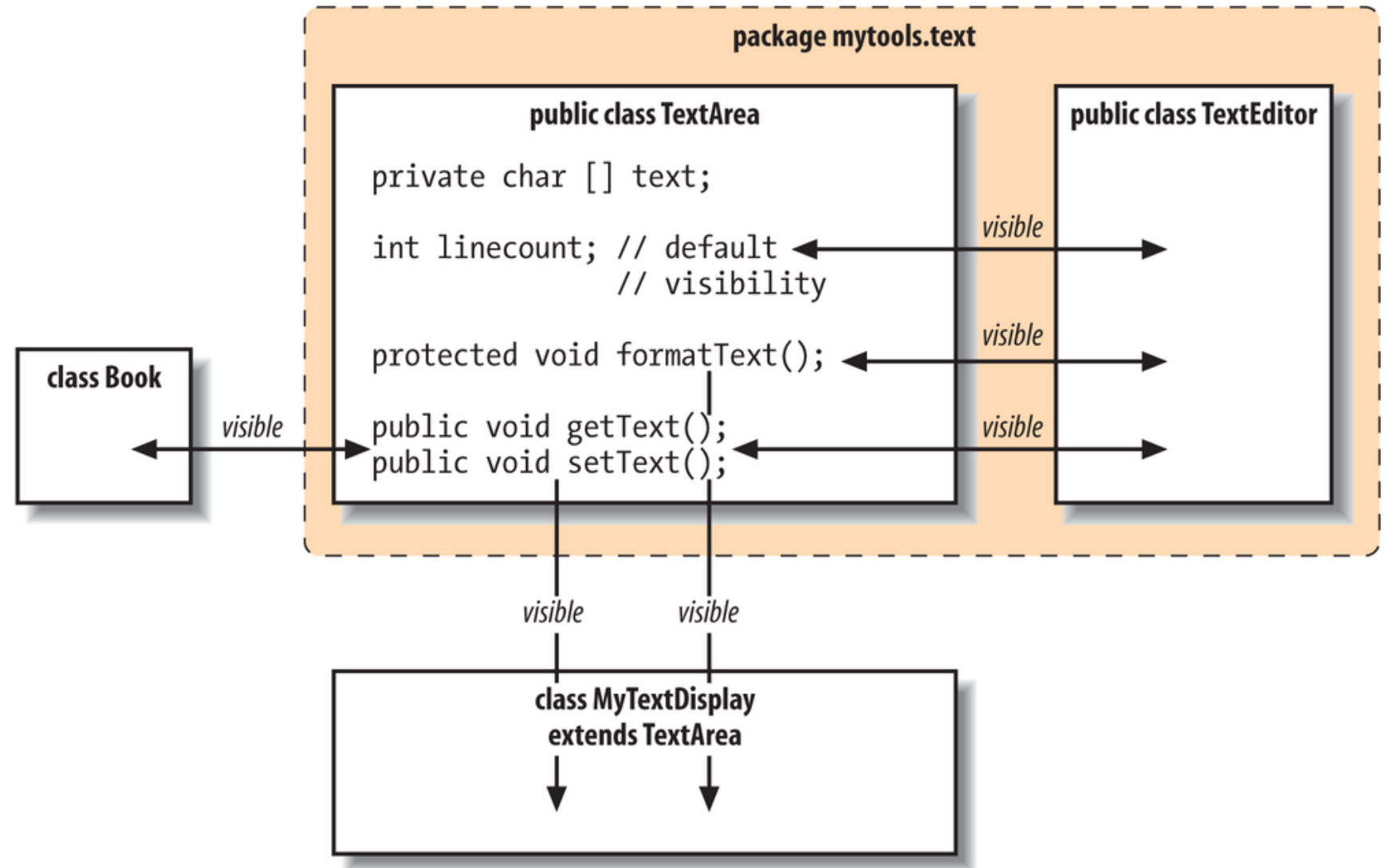


```
Expressions1.java x
Expressions1 main()
1 package be.odisee.se2.variabelen.intermediate;
2
3 public class Expressions1 {
4
5     public static void main(String[] args) {
6
7         int total;
8
9         // Opgelet += is niet hetzelfde als +=
10
11         total = 2;
12         total += 2;
13         System.out.println("total += 2 geeft: " + total);
14         // total += 2 geeft: 4
15
16         total = +2;
17         System.out.println("total += 2 geeft: " + total);
18         // total += 2 geeft: 2
19     }
20 }
```


Visibility

Access modifiers

- public
overal zichtbaar
- private
eigen class only
- protected
eigen class
+ derived classes
+ eigen package
- package (=default)
eigen class
+ eigen package



Bron: O'REILLY

Visibility

▣ Waarom visibility?

- ▬ Information hiding: beschermen van gedrag (methods) en data (at run time) tegen wijzigingen. Enkel invloed van beperkte bron toestaan = voorspelbaarder gedrag
- ▬ Code locality: Alle code over X moet bij elkaar staan

▣ Wanneer welke access modifier?

- ▬ private
Zo veel mogelijk
- ▬ protected
Indien subclassing en relevant voor extenties
- ▬ package (=default)
Indien samenwerking met andere classes (aan hetzelfde onderwerp = package)
- ▬ public



To be static or not to be static

▣ static

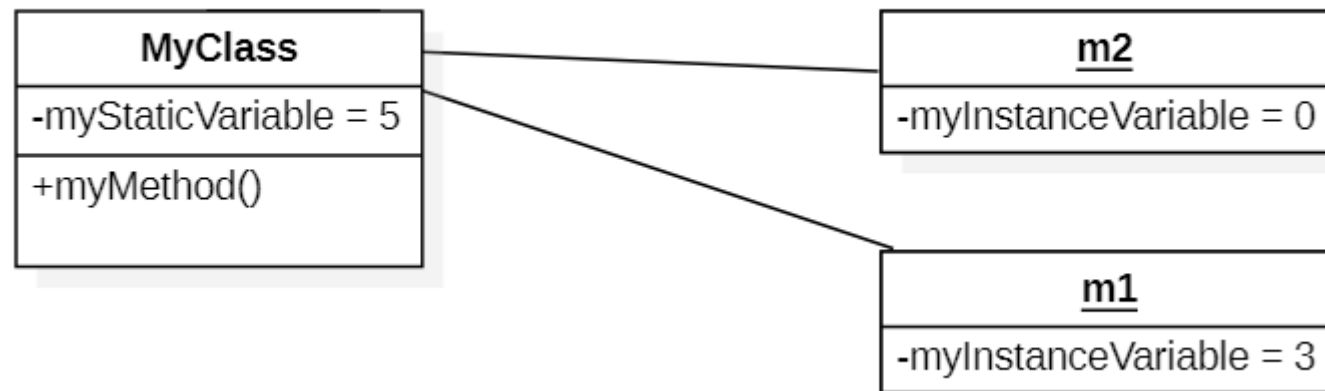
- ▬ Variabele gedeeld door alle objecten in de klasse
- ▬ Variabele van de klasse, niet van het object


private static int *myStaticVariable*;

MyClass.*myStaticVariable*++;

private int *myInstanceVariable*;

MyClass m1=new MyClass();
m1.*myInstanceVariable*++;





```
/** Stelt een fysieke teller voor welke aan de deur staat en telt hoeveel personen binnen en buitengaan.  
Bedient door mensen. */
```

```
public class ToegangsControleStaticExample {
```

```
/** Deze variable onthoudt hoeveel fysieke tellers (= instances van deze class) er gemaakt zijn. */
```

```
private static int aantalTellers=0;
```

```
ToegangsControleStaticExample() {
```

```
aantalTellers++; //Ctor: nieuwe teller in gebruik
```

```
}
```

```
public static void verwijderTeller() {
```

```
aantalTellers--; //teller wordt uit gebruik gehaald
```

```
}
```

```
public static boolean zijnErNogActieveTellers() {
```

```
return aantalTellers > 0; //zijn er nog tellers in gebruik
```

```
}
```

```
/** onthoudt het aantal mensen dat door 1 deur zijn binnengetreden (en niet zijn buitengetreden)
```

```
* 1 aantal per fysieke teller */
```

```
private int aantal = 0;
```

To be static or not to be static

■ Wanneer static gebruiken?

▬ Variabelen

- Constanten (naam constante in HOOFDLETTERS)
vb `Integer.SIZE = 32;`
- Hoeveel objecten zijn er van mijn class? Nodig als er maar 1 instance mag zijn.
vb maar 1 connectie met de database

▬ Methods

- Als een method niet afhankelijk is van een instance variabele
vb `Integer.getInteger(String name)` en `Integer.getInteger(String name, int defaultValue)`

Constanten

▣ final variable

- ▬ Laat niet toe de waarde te wijzigen na initialisatie
- ▬ Kan in de constructor ingesteld worden, niet in een method

```
class myFinalExampleClass {  
    /* Kan ingesteld worden bij declaratie of in constructor  
       NIET IN BEIDE TEGELIJK! */  
    private final int myVariable=5;  
    myFinalExampleClass(int startWaarde) {  
        myVariable=startWaarde;  
    }  
}
```

Constanten

■ Wanneer final gebruiken?

- ▬ Als een waarde niet meer gewijzigd mag worden
- ▬ Constanten
- ▬ Zo veel mogelijk: het vermijdt onbedoelde wijzigingen

```
public int machtsverheffing(int grondtal, final int exponent) {  
    int result=1;  
    for(int i=exponent; i>0 ; i--) {  
        grondtal*=grondtal; //foute code maar geen syntax error  
        exponent--; //foute code en syntax fout  
    }  
    return result;  
}
```

final object != final class variables

- ▣ Opgelet! Het final keyword laat niet toe om de waarde van de variabele te wijzigen naar een *ander* object.

Wel toegelaten is het wijzigen van de (niet final) variabelen van het final object.

```
private class MySubClass {  
    1 usage  
    public int var=5;  
}  
2 usages  
final MySubClass sub=new MySubClass();  
public void someMethod() {  
    sub=new MySubClass(); //mag niet want final  
    sub.var=6; //mag wel want var is niet final  
}
```


Oefening equality

```
@Override  
public boolean equals(Object obj) {
```

- Maak een class aan met minstens 2 instance variabelen (type vrij te kiezen). Implementeer de equals method.
Test of de == operator en equals de te verwachten waarden returnen.
Gebruik TDD

```
final Account ac1= new Account(35,"Draak");  
final Account ac2= new Account(35,"Draak");  
final Account ac3= new Account(666,"Boss");  
System.out.printf("Zijn deze account identiek? %b",ac1.equals(ac2));  
System.out.printf("Zijn deze account identiek? %b",ac1.equals(ac3));  
final Account ac4=ac1;  
System.out.printf("Zijn het dezelfde objecten? %b",ac1==ac2);  
System.out.printf("Zijn het dezelfde objecten? %b",ac1==ac4);
```

Defining equality opoassing test class

```
@Test void equals_identical_returnsTrue() {  
    //Arrange  
    Account a1=new Account(3,"Goblin");  
    Account a2=new Account(3,"Goblin");  
    //Act  
    boolean result=a1.equals(a2);  
    //Assert  
    assertTrue(result);  
}
```

```
@Test void  
equals_notIdenticalLevel_returnsFalse() {  
    Account a1=new Account(3,"Goblin");  
    Account a2=new Account(4,"Goblin");  
    assertFalse(a1.equals(a2));  
}
```

```
@Test void  
equals_notIdenticalCreature_returnsFalse() {  
    Account a1=new Account(3,"Goblin");  
    Account a2=new Account(3,"Roc");  
    assertFalse(a1.equals(a2));  
}
```

Defining equality overriding implementation code

```
public class Account {  
    private final int level;  
    private final String creature;  
    public Account(int level, String creature) {  
        this.level = level;  
        this.creature = creature;  
    }  
    @Override  
    public boolean equals(Object obj) {  
        Account acc=(Account)obj;  
        return acc.level==this.level && acc.creature.equals(this.creature);  
    }  
}
```

Enum(eratie)

As an example, which is better?

```
/** Counts number of foobangs.
 * @param type Type of foobangs to count. Can be 1=green foobangs,
 * 2=wrinkled foobangs, 3=sweet foobangs, 0=all types.
 * @return number of foobangs of type
 */
public int countFoobangs(int type)
```

versus

```
/** Types of foobangs. */
public enum FB_TYPE {
    GREEN, WRINKLED, SWEET,
    /** special type for all types combined */
    ALL;
}

/** Counts number of foobangs.
 * @param type Type of foobangs to count
 * @return number of foobangs of type
 */
public int countFoobangs(FB_TYPE type)
```

Enum(eratie)

A method call like:

```
int sweetFoobangCount = countFoobangs(3);
```

then becomes:

```
int sweetFoobangCount = countFoobangs(FB_TYPE.SWEET);
```

In the second example, it's immediately clear which types are allowed, docs and implementation cannot go out of sync, and the compiler can enforce this. Also, an invalid call like

```
int sweetFoobangCount = countFoobangs(99);
```

is no longer possible.

Bron: <http://stackoverflow.com/questions/4709175/what-are-enums-and-why-are-they-useful>

Enum(eratie)

- ▣ Enum's zijn zelf te ontwerpen "datatypes" welke een opsomming bevatten van de mogelijke waarden.
- ▣ Is een soort van lite class. Kan ook methods en constructors bevatten.

```
private enum Richtingen {NOORD, OOST, WEST, ZUID };  
private void useIt() {  
    Richtingen richting=Richtingen.OOST;  
    switch(richting) {  
        case NOORD -> System.out.println("Onward!");  
        case OOST  -> System.out.println("Take a right");  
        case ZUID  -> System.out.println("Please make a U-turn.");  
        case WEST  -> System.out.println("Neen, de andere rechts");  
    }  
}
```

Enum(eratie)

```
private enum Richtingen {  
    NOORD(0), OOST(90), ZUID(180), WEST(270);  
    private final int graden;  
    Richtingen(int graden) {  
        this.graden=graden;  
    }  
    public int getGraden() { return graden; }  
};
```

```
public static void main(String[] args) {  
    Richtingen mijnRichting;  
    mijnRichting=Richtingen.WEST;  
    System.out.println("De richting is "+mijnRichting.getGraden()+" graden.");  
}
```

Enum – Wanneer en hoe te gebruiken?

- ▣ Indien een variabele maar enkele toegelaten waarden heeft
- ▣ Vaak gedefinieerd in een class omdat de enum verbonden is aan de implementatie ervan.
- ▣ public indien gebruikt door een publieke functie als return of param waarde
- ▣ Naam enum begint met Hoofdletter. Namen van de waarden in HOOFDLETTERS.

```
private enum Richtingen {NOORD, OOST, WEST, ZUID };
```


Enum – Oefening opgave

▣ Basic:

- ▬ Stel de dagen van de week voor: maandag .. zondag
- ▬ Maak een functie nextDay welke de volgende dag teruggeeft.
- ▬ Denk na over access modifiers, final, ...

▣ Uitdaging:

- ▬ Maak ook een prevDay
- ▬ En een getDayOfWeek welke een int X returned voor de Xde dag van de week
- ▬ Vermijd code duplicatie

Enum – Oefening oplossing basic

```
/** Represents the days of the week of the Gregorian calendar */
private enum WeekDays {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY ;

    /** @return the next day of the week */
    public WeekDays nextDay() {
        switch(this) {
            case MONDAY: return TUESDAY;
            case TUESDAY: return WEDNESDAY;
            case WEDNESDAY: return THURSDAY;
            case THURSDAY: return FRIDAY;
            case FRIDAY: return SATURDAY;
            case SATURDAY: return SUNDAY;
            default: return MONDAY;
        }
    }
}
```

Enum – Oefening oplossing uitdaging

```
private enum WeekDay {
    MONDAY(1),
    TUESDAY(2),
    WEDNESDAY(3),
    THURSDAY(4),
    FRIDAY(5),
    SATURDAY(6),
    SUNDAY(7)
;
    private final int dayOfWeek;
    /** The number of the day starting with
    monday = 1 */
    public int getDayOfWeek() {
        return dayOfWeek;
    }
}
```

```
WeekDay(final int dayOfWeek) {
    this.dayOfWeek=dayOfWeek;
}
/** translates number into to WeekDays */
private static WeekDay toWeekDay(final int day)
{
    switch(day) {
        case 1: return MONDAY;
        case 2: return TUESDAY;
        case 3: return WEDNESDAY;
        case 4: return THURSDAY;
        case 5: return FRIDAY;
        case 6: return SATURDAY;
        default: return SUNDAY;
    }
}
```

Enum – Oefening oplossing uitdaging

```
/**
 * calculates the next day of the week
 * @return the next day of the week
 */
public WeekDay nextDay() {
    return toWeekDay(dayOfWeek%7+1);
}

/**
 * calculates the previous day of the week
 * @return the previous day of the week
 */
public WeekDay prevDay() {
    return toWeekDay(((dayOfWeek-2)%7)+1);
}
```

```
public static void main(String[] args) {
    final WeekDay d= WeekDay.MONDAY;
    final WeekDay e= d.nextDay();
    System.out.println("Next day is "+e);
    System.out.println("Prev day is "+d.prevDay());
}
```

Huiswerk / Opdracht





Toledo leerobjecten

- ▣ Onder cursusdocumenten -> Leerobjecten week 2
- ▣ Neem het door, beluister de filmpjes

In te dienen oefening

- ▣ De opdracht staat op github classroom en kan bereikt worden via Toledo > opdrachten > week 3
- ▣ Dien de code in door het te pushen naar github
 - eventueel via IntelliJ git integratie
 - of externe git applicatie zoals github desktop
- ▣ Vergeet ook het leerverslag niet toe te voegen als **pdf** in de git repository