Sofia University
Department of Mathematics and Informatics

Course : Applied OO Programming part 1

Date: June 8, 2020

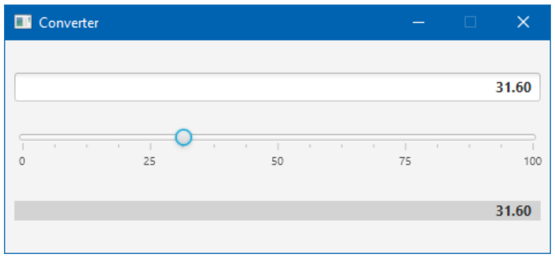
Student Name:

Lab No. 15

(JavaFX properties)

Problem 1

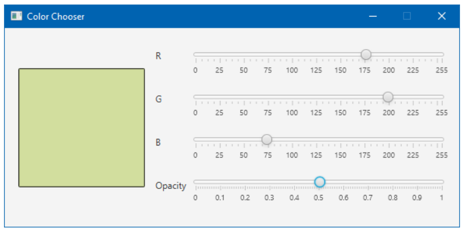
Write a JavaFX application that contains three elements- a **Textfield** at the top and a **Slider** control and a **Label** beneath it. You'd like to have the **Textfield** and the **Label** represent the value of the **Slider** such that when you move the **Slider** back and forth, the values in the **Textfield** and the **Label** changes to show the Slider's current value. Use databinding to implement this task.

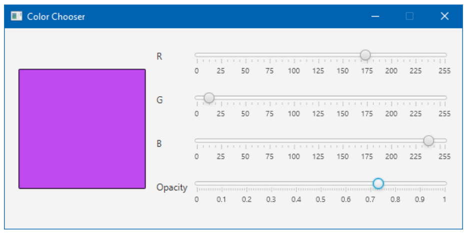


Use JavaFX properties and implement versions of Binding- using **Fluent API**, static methods of class **Bindings**, **low level binding** and handling the **Change** event of the textValue() property of the **Textfield** to solve the problem.

Problem 2a

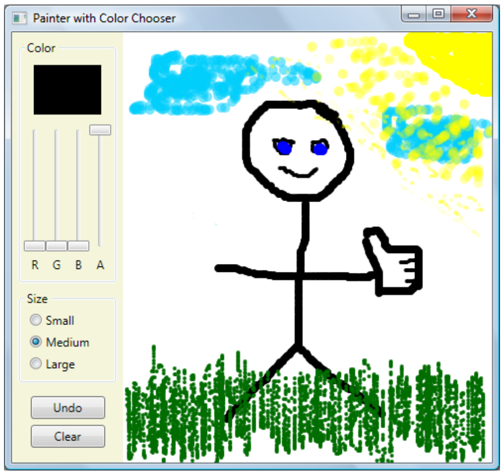
Write a JavaFX application that allows to choose a color using sliders for selecting Red, Green, Blue and Opacity values. Add a Rectangle to the application to visualize the color selection using binding of the fill color of the rectangle to values of the slider as shown on the figure below

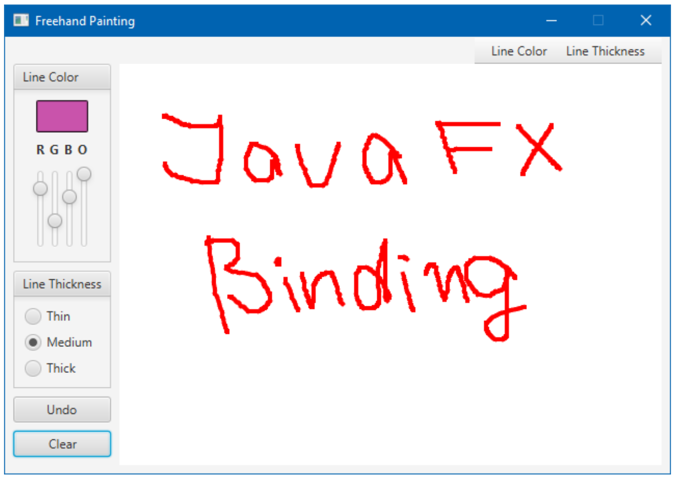




Problem 2b

Incorporate an RGBA color chooser into the Painter example (shown in the left side below ,*attached* to the Lab as Painter.rar) to look like the figure displayed on the right side below m**aking use of user-defined styles that target respectively the Slider and the Button types**. Apply the styles to the sliders and the buttons in the required GUI design style to make all the sliders and buttons (with **rounded corners**) look the same. Let the user select the brush color using the color chooser instead of the group of RadioButtons. Use **Expression Blend** for this purpose.





Problem 3

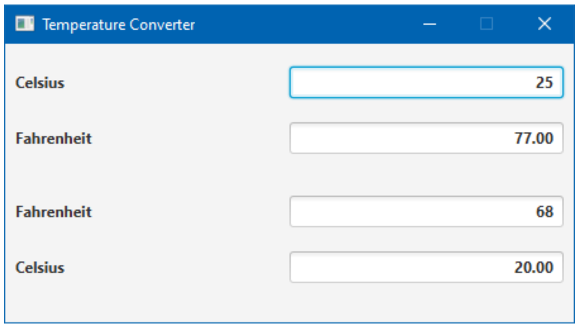
Write a JavaFX application that allows to input Fahrenheit temperature in a TextBox, computes and outputs in a Label its **celsius** equivalent concurrently with changes in Fahrenheit value using the calculation

celsius = 5.0 / 9.0 * (Fahrenheit **- 32);**

Allow also input of **celsius** temperature in a TextBox, computes and outputs in a Label its Fahrenheit the equivalent concurrently with changes in **celsius** value using the calculation

Fahrenheit **= 9.0 / 5.0 * celsius + 32;**

Use JavaFX properties and implement versions of Binding- using Fluent API, static methods of class Bindings, low level binding and ChangeListener to solve the problem.


Temperature Converter

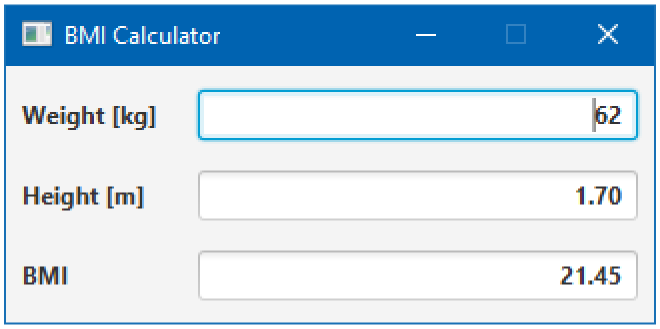
Celsius	25
Fahrenheit	77.00
Fahrenheit	68
Celsius	20.00

Problem 4

Write a JavaFX application to calculate the BMI given weight and height using the formula

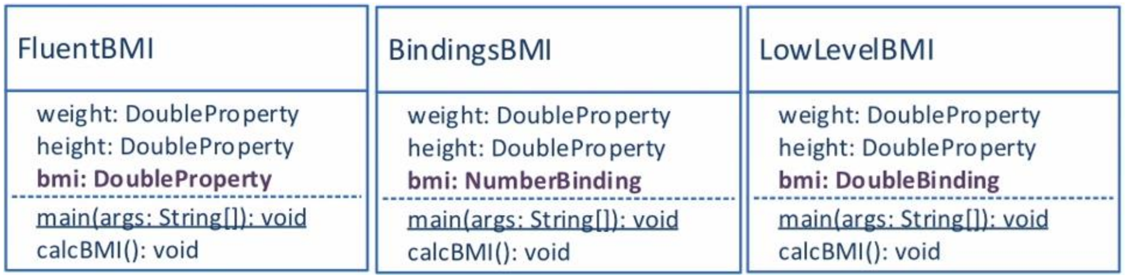
BMI = weight*10000/(height * height), where weight is measured in kg and height in meters

Use JavaFX properties for weight and height and use Binding to display the BMI coefficient concurrently with the change of weight and height input.


BMI Calculator

Weight [kg]	62
Height [m]	1.70
BMI	21.45

Implement versions of Binding- using Fluent API, static methods of class Bindings low level binding and using ChangeListener.



Problem 5

Short questions:

1. Can you create an object of `IntegerProperty` using `new IntegerProperty(3)`? If not, what is the correct way to create it?
2. Consider

```
1  public static void main(String[] args) {
2        DoubleProperty d1 = new SimpleDoubleProperty(1);
3        DoubleProperty d2 = new SimpleDoubleProperty(2);
4        d1.bind(d2);
5        System.out.println("d1 is " + d1.getValue()  + " and d2 is " +
         d2.getValue());
6        d2.setValue(70.2);
7       System.out.println("d1 is " + d1.getValue() + " and d2 is " +
        d2.getValue());
         }
```

What will the output if line 4 is replaced by `d1.bind(d2.multiply(2))`?

What will the output if line 4 is replaced by `d1.bind(d2.add(2))`?

Problem 6a

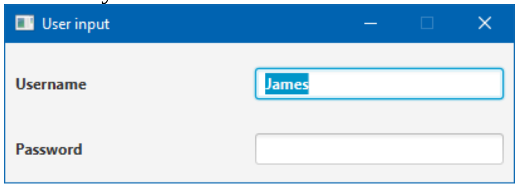
Create class User with datamembers `lastname` and `password` of type `StringProperty`. Implement `password` property using the conventional procedure and property `lastname` using best practices, where the JavaFX property is created on demand. Add a general purpose constructor to initialize both properties `String` values. Add a default constructor purpose constructor to initialize both properties String values to `password` = “007” and `lastname`= “James”.
Create class UserTest, where run the following test scenario:

- Create `StringProperty lname`
- Create instance of class `User` as `password` = “007” and `lastname`= “Bond”
- Bind bidirectionally `lname` to `lastname`
- Set the value of `lastname` property to “007”
- Set the value of `lname` to “New agent”
- Print output on standard the values of `lastname` property and the value of `lname`

Execute the program and explain the printout

Problem 6b

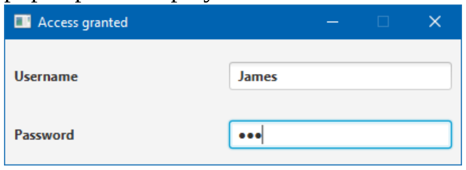
Create a JavaFX FXML application allowing to input `lastname` and `password` for an instance `user` of User created by default.


User input

Username: James

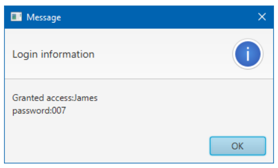
Password:

Add `Alert messageBox` and `BooleanProperty accessGranted` to the Controller inialized to `false`. Bind unidirectionally `lastnameProperty` of `user` to the `textProperty` of the `username Textfield`. Implement a handler for the ACTION event for the `password Textfield`, so that the title of the window changes to "Access granted" and "No access" when the client hits the Enter key and respectively, succeded or failed to enter the correct password (in this case “007”). Moreover, in case of correct password input an Alert message box pops up and displays the user `lastname` and `password`


Access granted

Username: James

Password: •••

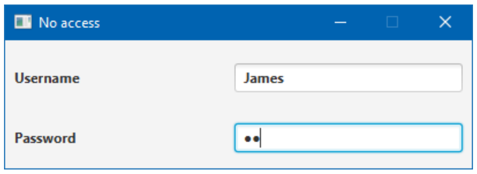

Message

Login information

Granted access:James
password:007

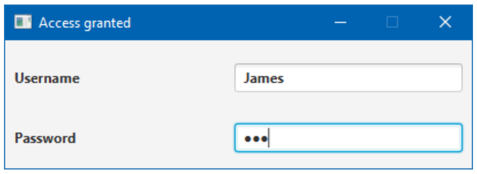
OK

Add a `ChangeListener` to the textProperty of the `password Textfield`, so that `accessGranted` is set to `true` or `false`, when respectively, the new `password` entry matches or differs from the user `password` Additionally, the `accessGranted` is set to `true` or `false` the window changes to "Access granted" or "No access"


No access

Username: James

Password: ••

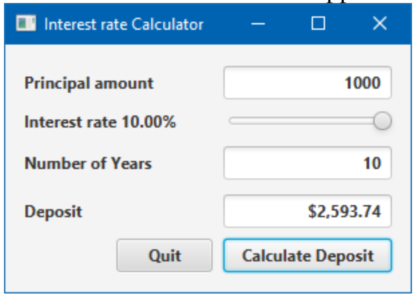

Access granted

Username: James

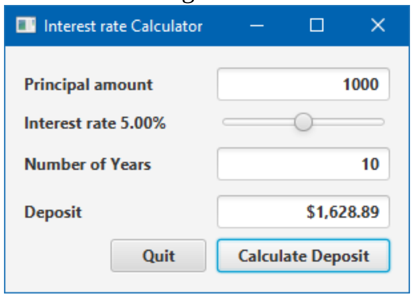
Password: •••

Problem 7

Update the sample FXML application InterestRateCalculatorSlider given in Lectures(InterestRateCalculatorSlider.rar) to use JavaFX property binding and display the total deposit once the client has input principal amount, interest rate and number of years (without having to click a button)
InterestRateCalculatorSlider application without binding

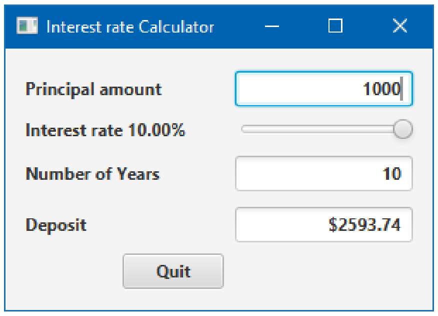


Principal amount 1000
Interest rate 10.00%
Number of Years 10
Deposit $2,593.74

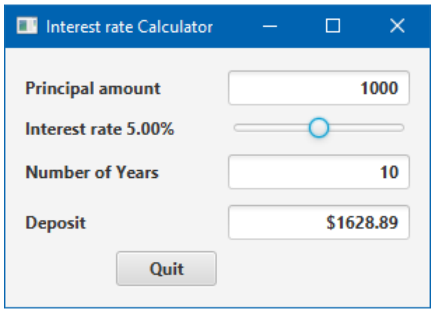


Principal amount 1000
Interest rate 5.00%
Number of Years 10
Deposit $1,628.89

InterestRateCalculatorSlider application with binding



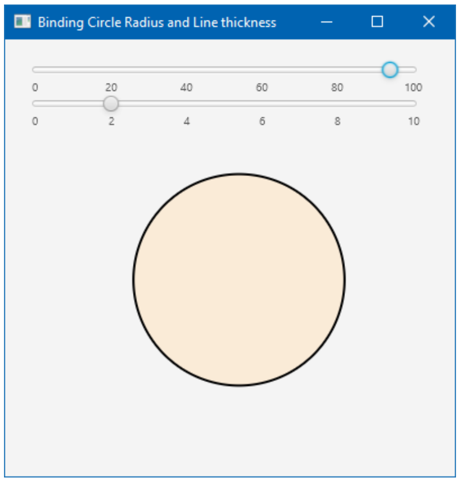
Principal amount 1000
Interest rate 10.00%
Number of Years 10
Deposit $2593.74

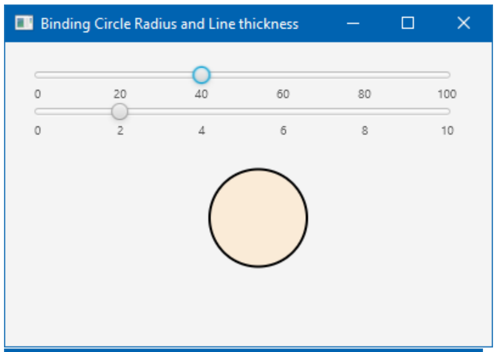


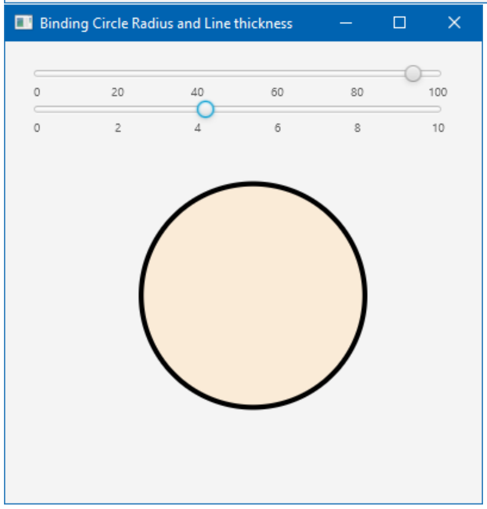
Principal amount 1000
Interest rate 5.00%
Number of Years 10
Deposit $1628.89

Problem 8

Write a JavaFX application that displays a circle centered in the middle of he window. Bind the coordinates of the circle so that the circle remains in the center of the window whenever the window is resized.
Add two sliders to the Scene and bind the radius and the line thickness of the circle to the sliders.



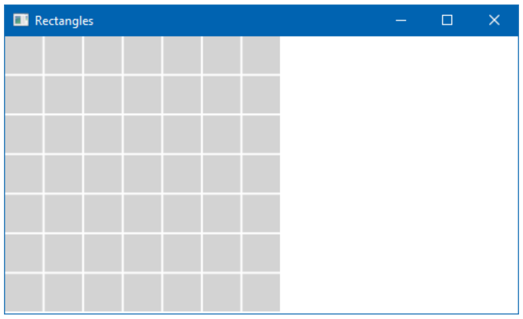


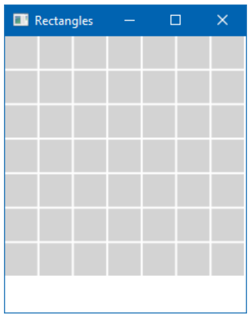


Problem 9

Write a JavaFX application create a 7x7 grid of rectangles that will try to fill the maximum possible area of a stage during resizing. You need to create three bindings for that:

- One to calculate which side of the window is smaller now (given below)
- One, applied to all rectangles, to choose the rectangles' positions (xProperty and yProperty) correctly
- One, applied to all rectangles, to change the rectangles' sizes (heightProperty, widthProperty) accordingly





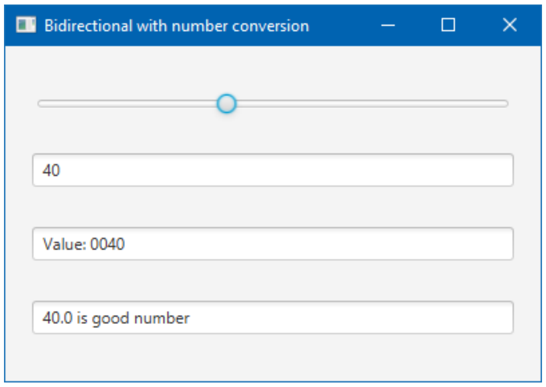
Hint: Use

```
NumberBinding minSide = Bindings
                                .min(root.heightProperty(), root.widthProperty())
                                .divide(count);
//or
```

```
NumberBinding minSide = Bindings
        .when(root.heightProperty().lessThan(root.widthProperty()))
        .then(root.heightProperty())
        .otherwise(root.widthProperty())
        .divide(count);
```
where `root` is the root of the Scene graph and `count` is 7(the number of rectangles)

Problem 10

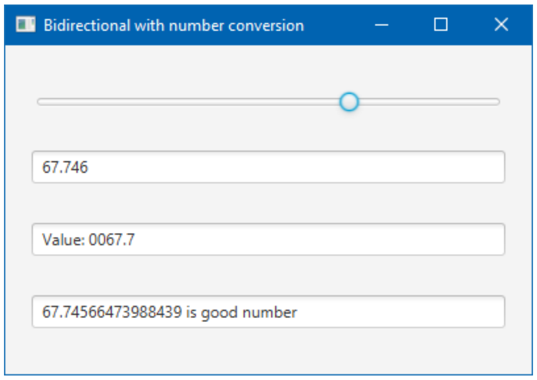
Create a BiDirectional binding between a JavaFX `Slider` and a `Textfield`. Note, their properties are o different type, use `NumberStringConverter` to adapt property types to each other as well as apply proper formatting



40

Value: 0040

40.0 is good number



67.746

Value: 0067.7

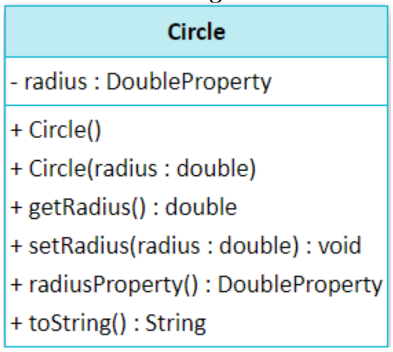
67.74566473988439 is good number

Hint: Use
```
txtOutput.textProperty().bindBidirectional(slider.valueProperty(),
                                           new NumberStringConverter());
```
or instead of `NumberStringConverter` use
```
new StringConverter<Number>() {
    @Override
    public String toString(Number number) {
        return number + " is good number";
    }

    @Override
    public Number fromString(String string) {
        return Double.valueOf(string.split(" ")[0]);
    }
});
```

Problem 11a

Create the following class :



Circle
- radius : DoubleProperty
+ Circle() + Circle(radius : double) + getRadius() : double + setRadius(radius : double) : void + radiusProperty() : DoubleProperty + toString() : String

Using the `Circle` class, create a `main`() that does the following

1. Construct a `Scanner` so you can get some user input and construct a default `Circle` object.
2. Create a `NumberBinding` expression that calculates the circumference of the circle, and a second `NumberBinding` expression that calculates the area of the circle.
   Tip: you can chain multiple binding methods together, and you can pass these binding methods primitive values.
3. Write a loop that repeatedly asks the user to enter a new radius. After a radius is entered, set the circle object's radius to the user-entered value and then print the values of the two binding expressions.
4. Test your program

Modify question 1 as follows: create `main`() method a string binding expression that formats the circumference as "Circumference: c.cc" (where c.cc is the actual circumference, formatted to 2 decimal places) and concatenates it to the formatted area as "Area: a.aa" (where a.aa is the actual area formatted to to decimal places. Don't forget a new-line in between (add the new-line to the area formatting, don't add a new `concat`() just for the new-line, that would be inefficient)!

Finally, create a JavaFX application that allows the user to enter the radius of a circle in a text field. Add 2 labels: one will hold the circle's circumference and the other will hold the circle's area. When the value of the radius text box changes, the labels should show the updated circumference and area. Use your new Circle class with the radius property.

Problem 11b

Create the following `class` :

Class: Inventory
Data Members: - id : String - name : String - qoh : int - rop : int - sellPrice : double
Method Members: + Inventory() + Inventory(id : String, name : String, sellPrice : double) + Inventory(id : String, name : String, qoh : int, rop : int, sellPrice : double) + getId() : String + setId(id : String) : void + getName() : String + setName(name : String) : void + getQoh() : int) + setQoh(qoh : int) : void + getRop() : int + setRop(rop : int) : void + getSellPrice() : double + setSellPrice(price : double) : void + toString() : String

The `Inventory ID` stored in the id member must be unique for every single object and must be implemented as specially-formed String that consists of 3 upper-case letters , followed by a dash, followed by 6 digits with leading zeros. For example, "`ABC-001234`"

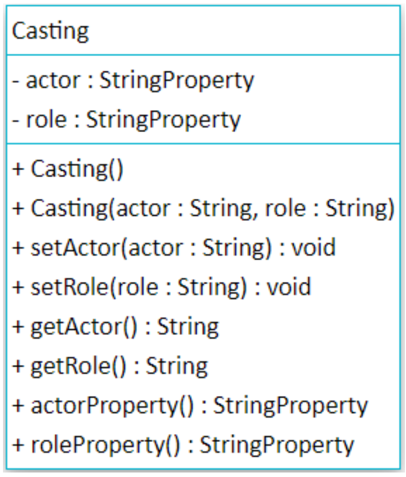
The BooleanExpression class models a `boolean` binding expression. If you looked at the documentation for your property classes, you might have noticed that there are several methods that return `BooleanBinding` objects (child of `BooleanExpression`).

Using your Inventory class, add the following code to a `main`() method:

1. Construct a `Scanner` so you can get some user input.
2. Construct an inventory object with all 5 values, use whatever valid values you like. Make sure your Q.O.H is greater than your R.O.P.
3. Construct a `BooleanExpression` that binds a relational expression for Q.O.H less than or equal to R.O.P (the `qoh` and `rop` properties). If you're not sure of the binding method to use, check the `IntegerProperty` class (tip - there are relevant methods in one of the parent classes).
4. Prompt the user to enter a new quantity on hand, and set the value to your inventory object's Q.O.H.
5. Write an if statement: `if` the boolean expression contains a false value, display the message, "`You need to order more`!".
6. Test your program

Problem 11c

Create the following `class` :



Casting

- actor : StringProperty
- role : StringProperty

+ Casting()
+ Casting(actor : String, role : String)
+ setActor(actor : String) : void
+ setRole(role : String) : void
+ getActor() : String
+ getRole() : String
+ actorProperty() : StringProperty
+ roleProperty() : StringProperty

Create a `main`() method that performs the following tasks:

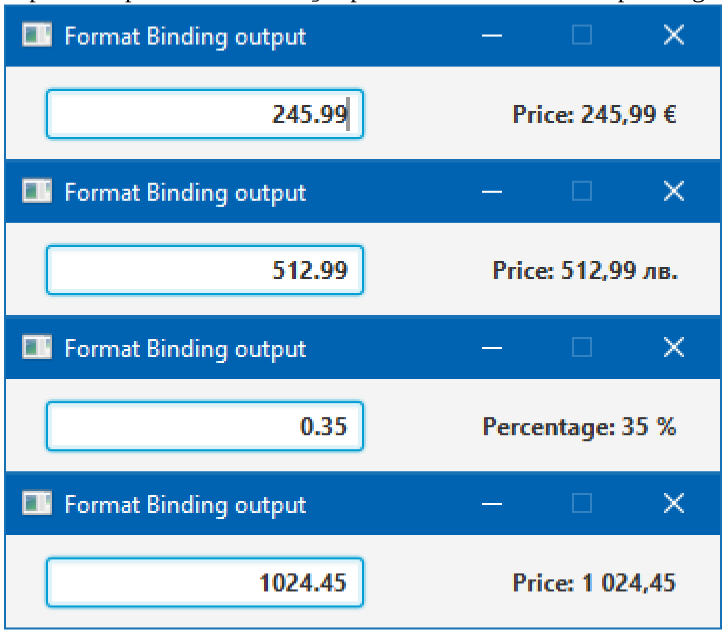
1. Construct a default instance of the ` class.
2. Create a string binding expression that binds the actor property and the role property as the string expression "[actor] playing the role of [role]."
3. Print the string binding expression value and explain why you see the output that appears.
4. Set the value of the casting object's actor property to any actor name you wish.
5. Print the string binding expression value. Why is there a "null" in the output?
6. Set the value of the role object's role property to any TV or movie role you wish.
7. Print the string binding expression value. You should see the correct output.
8. Test your program

Problem 12

Create a UniDirectional binding between a JavaFX `Textfield` and a `Label`. The `Label` must display the `Textfield` input formatteed as:

a) The number format for a given `Locale`
b) The currency format for a given `Locale`
c) The percentage format for a given `Locale`

Write a helper method `tryParseDouble(String value)` to validate input of a double value in the `Textfield` using a regular expression. In case the `String` input in the `Textfield` matches the regular expression parse the `String` input and return the corresponding double value, otherwise return 0.


Format Binding output
245.99
Price: 245,99 €
Format Binding output
512.99
Price: 512,99 лв.
Format Binding output
0.35
Percentage: 35 %
Format Binding output
1024.45
Price: 1 024,45