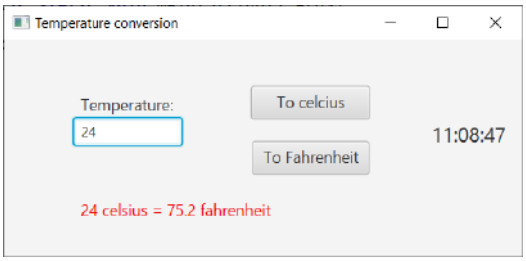
A JavaFX temperature Converter using the MVVM pattern

In this exercise, you are going to transform your solution to the temperature converter exercise (*the exercise converting to Celsius and Fahrenheit and having a clock)*, such that this follows the MVVM pattern.



Step 1: Create a class `TemperatureViewModel` (a ViewModel class for the ViewController)

- Make 3 `StringProperty` variables (for input temperature, output temperature, and time). Initialise all 3 in the constructor (to `SimpleStringProperty`).
- Make get methods for each of the `StringProperty` instance variables.
- Include an instance variable of the model (`TemperatureModel`) and initialise it to a parameter variable given in the constructor.
- Include two `void` methods without any parameters, one for converting to Celsius and one for converting to Fahrenheit. In these methods, make a try catch block in which you 1) convert the value in input temperature property to a double (`Double.parseDouble`), 2) call the model method to get the temperature, 3) convert this into a string, 4) set it to the output temperature property and 5) empty the input temperature by setting its value to null. In the catch block, you set the value of the output temperature (which in the view is also used as an error label) into a proper string, e.g. “Error in input”
- Make this class responsible for the `ObservableClock`, i.e.
    - implement `PropertyChangeListener` and in the `propertyChange` method, set the time property to the string value of the time – and wrap it into a `Platform.runLater` (because it is going to be used in a JavaFX-view).
    - Create and start the `ObservableClock` thread in the constructor (move the statements from the view into the `init` method of the ViewModel)

Step 2: Create a class `ViewModelFactory`

- One instance variable, an object of type `TemperatureViewModel`
- A constructor taking a `TemperatureModel` and creating the `TemperatureViewModel` object.
- A getter for the `TemperatureViewModel` instance variable.

Step 3: Update class `TemperatureViewController`

- Change the model instance variable into an instance variable of type `TemperatureViewModel`. Update the `init` method to take the ViewModel as parameter. (No direct connection to the model any longer)
- Update the methods `toCelsius` and `toFahrenheit` to be one-statement methods just calling the corresponding methods in the ViewModel.

- Remove all connections to the `ObservableClock` class, 1) no `ObservableClock` instance variable, 2) no creations of a thread in the `init` method, 3) no `PropertyChangeListener` interface being implemented, and 4) no `PropertyChange` method (delete it)
- Create the binding to ViewModel properties in the `init` method. Bind (single direction) the two labels to their corresponding ViewModel properties and make a bidirectional binding between the text field and the input temperature property in the ViewModel.

Step 4: Update class `ViewHandler`

- Replace the model instance variable with an instance variable of type `ViewModelFactory` and update the constructor to take an argument of the factory type too.
- Update the private method `loadTemperatureView` to call the `init` method with the proper arguments. In this case, call the get method in the factory to get the ViewModel object needed.

Step 5: Update class `MyApplication`

- Add a statement to create the `ViewModelFactory` object (just after creating the model)
- Pass the `ViewModelFactory` object to the View

Step 6: Run the `main` method in class `Main`

- Try to convert to Celsius and convert to Fahrenheit
- Test that you get an error message trying to convert with wrong input, i.e. when the text field is empty and when it contains letters instead of digits.
- Does the time label update every second?