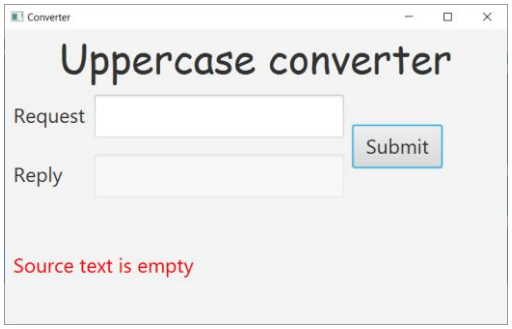
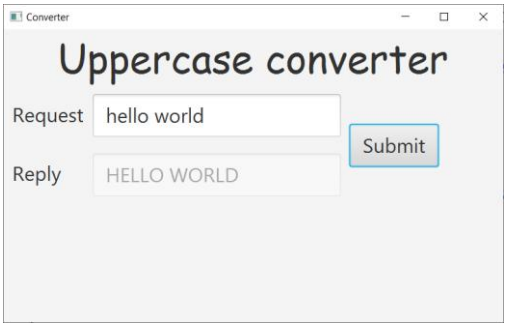
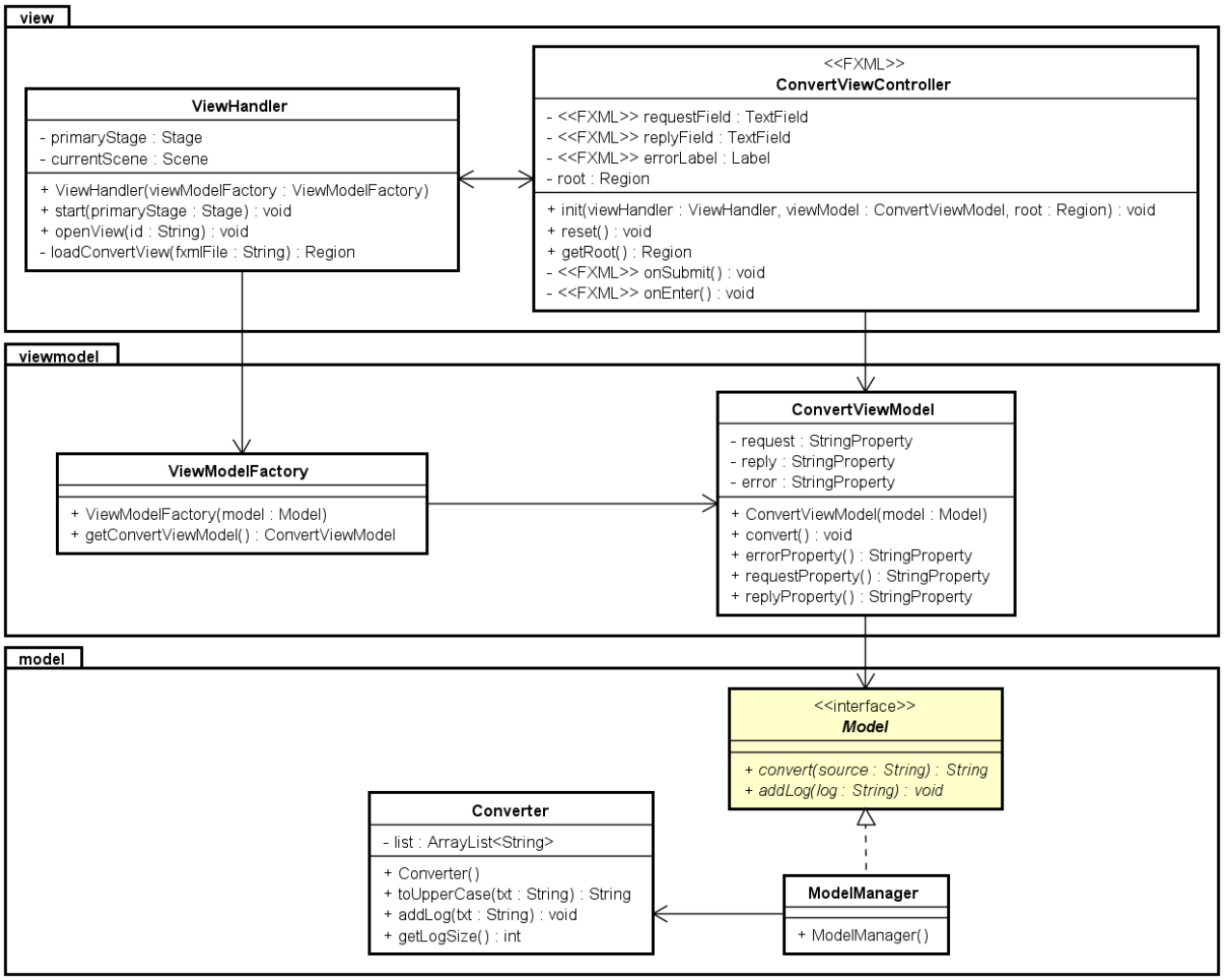
Exercise: MVVM - Uppercase

The purpose of this exercise is to create an MVVM application with the ability to convert a string to uppercase.





Implement the MVVM application exactly as shown in the class diagram below (description is given on the next page)



Model:

- Interface `Model` and classes `ModelManager` and `Converter` are all given in appendix

ViewModel (class ConvertViewModel):

Implement `ConvertViewModel`

- `Model` instance variable
- Three `StringProperty` instance variables, `request`, `reply` and `error`
- A constructor setting the `Model` instance variable to the parameter variable and initializing the three `StringProperty` variables instances of `SimpleStringProperty`
- Three get methods for each of the `StringProperty` instance variables
- A method `convert` doing the following
    - In a try catch block, call method `convert` from the `Model` variable using the String from the `request` property as argument. Store the result in the String of the `reply` property (`reply.set(…)`) and set the String in the `error` property to a `null` string.
    - If you catch an exception, set the String in the `error` property to the exception message

View:

FXML file

FXML file `ConvertView.fxml` is given in appendix. Create the file in the view package (and copy the contents given)

ConvertViewController class

Implement class `ConvertViewController`

- An @FXML annotated instance variable `requestField` of type `TextField`
- An @FXML annotated instance variable `replyField` of type `TextField`
- An @FXML annotated instance variable `errorLabel` of type `Label`
- An instance variable `root`, of type `Region`. Instance variables for `viewModel` (`ConvertViewModel`) and `ViewHandler`
- An `init` method setting all non-FXML instance variables and perform a binding to viewModel properties the following way
    - The textproperty of the `requestField` is bound bidirectional to the `viewModel request` property
    - The textproperty of the `replyField` is bound to the `viewModel reply` property
    - The textproperty of the `errorLabel` is bound to the `viewModel error` property
- Getter for the `root` instance variable
- A `reset` method with an empty body (instead of deleting the method, it may come in handy if you late add more windows)
- An @FXML annotated method `onSubmit` simply calling convert in the `viewModel`
- An @FXML annotated method `onEnter` doing the same (e.g. call `onSubmit`)
- Note that the annotated instance variables and methods has to be named exactly as given in the FXML file

Implement class `ViewHandler`

- Instance variables for `Stage`, `Scene`, `ConvertViewController` and `ViewModelFactory`
- Constructor setting `ViewModelFactory` and creating the `Scene` (with a new empty `Region` as argument)
- A method `start` setting `Stage` and calls a method to open the window
- A method `openView` calling the private method to load the view, setting scene and title, and showing the view.
- A private method `loadConvertView` loading the FXML file and returning the root node. The method gets the JavaFX-Controller (a `ConvertViewController` instance variable) and calls the `init` method.

To start the application

- A class extending Application, with a `start` method creating `Model`, `ViewModelFactory`, `ViewHandler` – and starting the view:

```
import javafx.application.Application;
import javafx.stage.Stage;
import model.Model;
import model.ModelManager;
import view.ViewHandler;
import viewmodel.ViewModelFactory;

public class MyApplication extends Application
{
  public void start(Stage primaryStage)
  {
    Model model = new ModelManager();
    ViewModelFactory viewModelFactory = new ViewModelFactory(model);
    ViewHandler view = new ViewHandler(viewModelFactory);

    view.start(primaryStage);
  }
}
```

- A class with the main method:

```
import javafx.application.Application;

public class UppecaseMain
{
  public static void main(String args[])
  {
    Application.launch(MyApplication.class);
  }
}
```

Appendix: Source code

Interface Model

```
package model;

public interface Model
{
  String convert(String source) throws Exception;
  void addLog(String log);
}
```

Class ModelManager

```
package model;

public class ModelManager implements Model
{
  private Conterter converter;

  public ModelManager()
  {
    this.converter = new Conterter();
  }

  @Override
  public synchronized String convert(String source)
  {
    String reply = converter.toUpperCase(source);
    addLog("Converting: " + source);
    return reply;
  }

  @Override
  public synchronized void addLog(String log)
  {
    String logValue = converter.getLogSize() + ": " + log;
    converter.addLog(logValue);
  }
}
```

Class Converter

```
package model;

import java.util.ArrayList;

public class Converter
{
  private ArrayList<String> logList;

  public Converter()
  {
    this.logList = new ArrayList<>();
  }

  public String toUpperCase(String txt)
```

```
  {
    return txt.toUpperCase();
  }

  public void addLog(String txt)
  {
    logList.add(txt);
  }

  public int getLogSize()
  {
    return logList.size();
  }
}
```

FXML file

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<VBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="351.0" prefWidth="600.0"
xmlns="http://javafx.com/javafx/10.0.1" userData="Uppercase converter"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="view.ConvertViewController" >
   <children>
      <HBox alignment="TOP_CENTER">
         <children>
            <Label text="Uppercase converter">
               <font>
                  <Font name="Comic Sans MS" size="48.0" />
               </font>
            </Label>
         </children>
      </HBox>
      <HBox alignment="CENTER_LEFT" prefHeight="100.0" prefWidth="200.0"
spacing="10.0">
         <children>
            <VBox spacing="20.0">
               <children>
                  <Label prefHeight="50.0" prefWidth="86.0"
text="Request">
                     <font>
                        <Font size="24.0" />
                     </font>
                  </Label>
                  <Label layoutX="10.0" layoutY="10.0" prefHeight="50.0"
prefWidth="86.0" text="Reply">
                     <font>
                        <Font size="24.0" />
                     </font>
                  </Label>
               </children>
            </VBox>
```

```
            <VBox spacing="20.0">
               <children>
                  <TextField fx:id="requestField" onAction="#onEnter">
                     <font>
                        <Font size="24.0" />
                     </font>
                  </TextField>
                  <TextField fx:id="replyField" disable="true"
layoutX="10.0" layoutY="10.0">
                     <font>
                        <Font size="24.0" />
                     </font>
                  </TextField>
               </children>
            </VBox>
            <Button mnemonicParsing="false" onAction="#onSubmit"
text="Submit">
               <font>
                  <Font size="24.0" />
               </font>
            </Button>
         </children>
         <VBox.margin>
            <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
         </VBox.margin>
      </HBox>
      <HBox prefHeight="120.0" prefWidth="579.0">
         <children>
            <Label fx:id="errorLabel" prefHeight="125.0"
prefWidth="580.0" text="ErrorLabel" textFill="RED" wrapText="true">
               <font>
                  <Font size="24.0" />
               </font>
            </Label>
         </children>
         <VBox.margin>
            <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
         </VBox.margin>
      </HBox>
   </children>
</VBox>
```