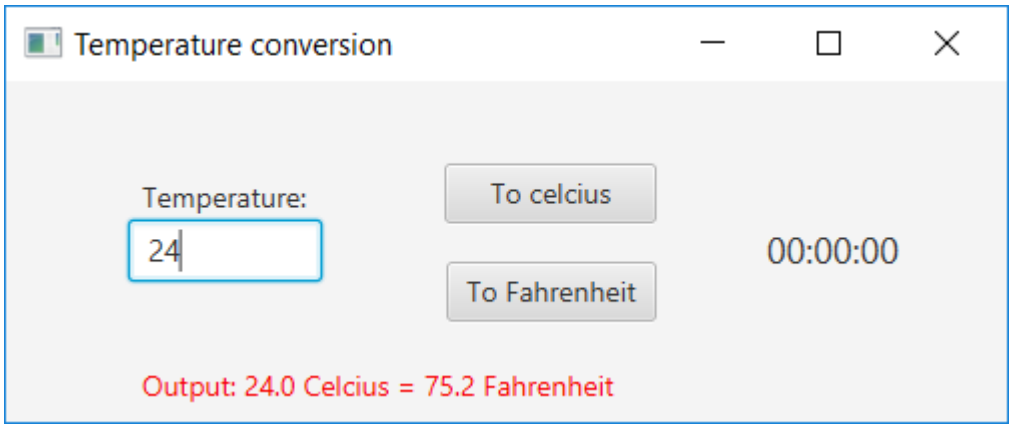
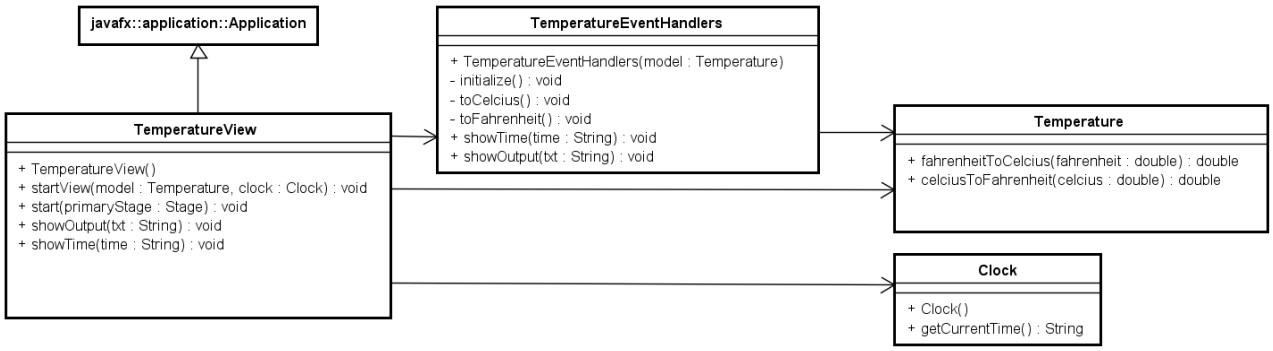
MAKE ONLY ONE OF THE EXERCISES – EITHER THE JAVAFX-VERSION OR THE SWING-VERSION

JavaFX-version: Exercise (see below)

The uploaded file consists of a Java application including a GUI implemented in JavaFX. When you run the program, the follow window is shown:






```
public class Main
{
   public static void main(String args[])
   {
      Temperature model = new Temperature();
      Clock clock = new Clock();
      TemperatureView view = new TemperatureView();
      view.startView(model, clock);

      System.out.println("MAIN ENDED");
   }
}
```

JavaFX-version: Part 1 – a separate GUI thread

As soon as you start up the view the main thread freezes – not until you close the GUI will you see the text "MAIN ENDED" in the console. The reason is that method `startView` in class `TemperatureView` has a statement that calls the launch method: `Application.launch(this.getClass());`

Step 1: Create a new class implementing `Runnable` with a run method having only one statement:

```
Application.launch(view.getClass());
```

Note that the constructor needs a reference to class `TemperatureView` to be stored in an instance variable.

Step 2: Replace the similar statement in method `startView` in class `TemperatureView` with the following: 1) create an object from the Runnable class you just implemented 2) create a Thread with the Runnable object you just created as an argument to the constructor, and 3) start the thread

Run the program and observe that the main thread ends before the GUI is terminated.

JavaFX-version: Part 2 – updating the time (in the console) every second

Step 1: Create a class `Timer` implementing `Runnable` with a run method doing the following in an infinite loop: 1) sleep for 1000 milliseconds and 2) print the current time to the console (using method `getCurrentTime` in class `Clock` and thus, you need the Clock as a parameter to the constructor).

Step 2: Update method `startView` in class `TemperatureView` with the following: 1) create an object from the Runnable class you just implemented 2) create a Thread with the Runnable object you just created as an argument to the constructor, and 3) start the thread

Run the program and observe that the current time is shown in the console.

Update to **step 2**: You may have noticed that the clock is still printed after the GUI is terminated. Therefore mark the thread as a daemon thread (calling `Thread` method `setDaemon(true)`). Observe the result.

JavaFX-version: Part 3 – a clock updating the GUI every second

Step 1: Create a new class `ShowTimeUpdater` implementing `Runnable` with a run method having only one statement

```
view.showTime(time);
```

The constructor needs a reference to class `TemperatureView` (called `view` in the above statement) and to a string containing the current time (called `time` in the above statement).

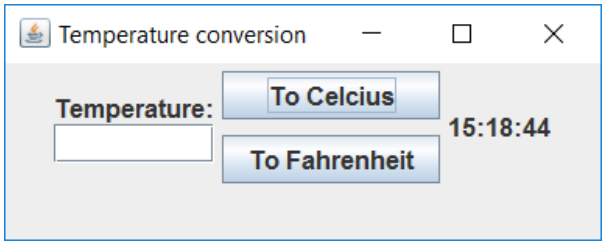
Step 2: Update the class `Timer` you implemented in part 2 such that you instead of printing to the console now uses the class from step 1 in the following way:

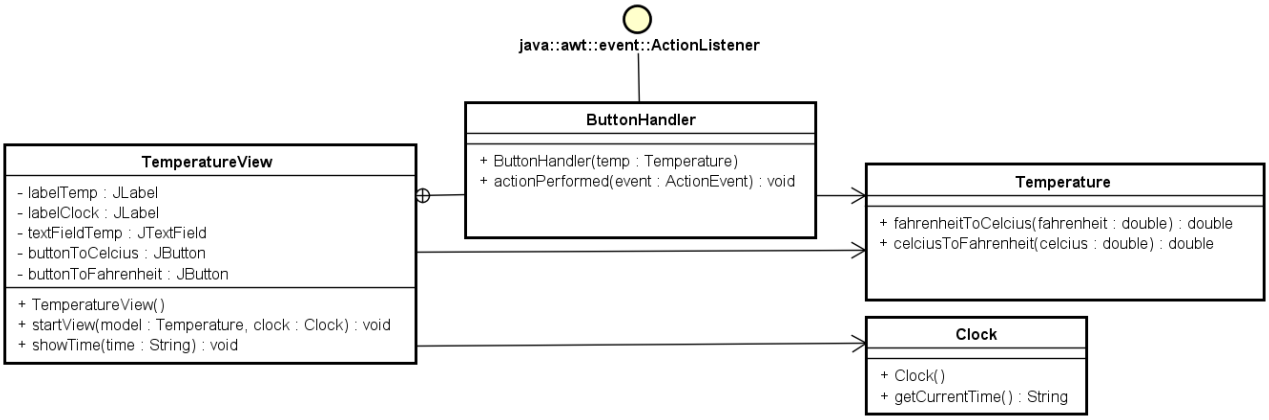
```
String time = clock.getCurrentTime();
Platform.runLater(new ShowTimeUpdater(view, time));
```

You now need both the clock and the view as parameters to the constructor. Run it and observe the result.

Swing-version: Exercise (see below)

The uploaded file consists of a Java application including a GUI implemented in Java Swing. When you run the program, the follow window is shown:






```
public class Main
{
   public static void main(String args[])
   {
      Temperature model = new Temperature();
      Clock clock = new Clock();
      TemperatureView view = new TemperatureView();
      view.startView(model, clock);

      System.out.println("MAIN ENDED");
   }
}
```

Swing-version: a clock updating the GUI every second

Step 1: Create a class `Timer` implementing `Runnable` with a run method doing the following in an infinite loop: 1) sleep for 1000 milliseconds and 2) get the current time (from class `Clock` method `getCurrentTime`) and show it to in the view (calling method `showTime` in the view). Let the constructor take the view and the clock as parameters.

Step 2: Update method `startView` in class `TemperatureView` with the following: 1) create an object from the Runnable class you just implemented 2) create a Thread with the Runnable object you just created as an argument to the constructor, and 3) start the thread

Run the program and observe that the current time is shown in the console.

Update to **step 2**: You may have noticed that the clock is still printed after the GUI is terminated. Therefore mark the thread as a daemon thread (calling `Thread` method `setDaemon(true)`). Observe the result.