CS193E: Assignment 1-A
Temperature Converter

Due Date

This assignment is due by **5:00 PM, April 13**.

Assignment

This assignment requires some easy Objective-C programming and illustrates how objects that you create can be used inside Interface Builder.

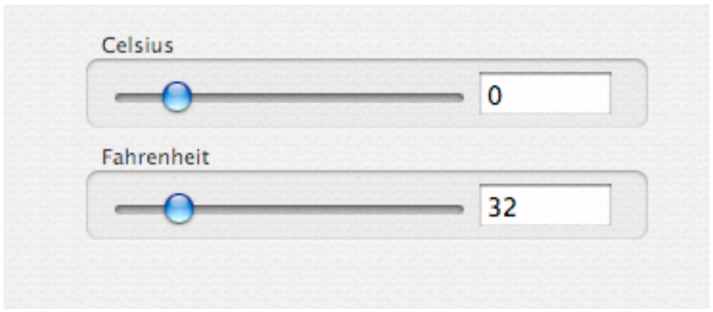
The goal of this assignment is to familiarize you with the Cocoa development tools and work-flow, and to introduce some fundamental concepts of object interaction in Cocoa.

IMPORTANT: The assignment description below is a high level description of the assignment. For this first assignment, also be sure to use the accompanying 'walkthrough' handout which is a largely step by step guide through building this application. The walkthrough also contains enough Obj-C guidance to complete the assignment.

Create a new Cocoa application in Xcode.

Open the application's MainMenu.nib file with Interface Builder. Add a slider and a text field to the main window. Select them and group them in a box (look at the menus `Layout→Make subviews of→Box`). Change the box's label to "Celsius". Do the same thing again, but label the box "Fahrenheit".

The goal is to wire everything up so that entering a Celsius temperature using the Celsius slider or text field causes the Fahrenheit slider and text field to display the corresponding Fahrenheit temperature, and vice versa.



Of course, the Celsius slider and text field should stay in sync with each other, and similarly for the Fahrenheit slider and text field.

In Interface Builder, add a new subclass of NSObject called "Converter".

Add outlets "celsiusSlider", "celsiusTextField", "fahrenheitSlider" and "fahrenheitTextField", and an action "convert:" to the Converter class, and create the files for the Converter class. Instantiate a Converter object and wire it up to the interface elements.

In Xcode, implement the "convert:" method in your Converter class and build your app. Once complete, changing any one user-interface element should update the other three.

The Celsius slider should range from at least 0˚C to 100˚C; the Fahrenheit slider should range from at least 32˚F to 212˚F.

Testing

Make sure the following work in your application (these are the things we will test to determine your grade):

0. Your project should build without errors or warnings.
1. When the Celsius slider is moved, the Celsius text field updates dynamically with the value of the slider, and the Fahrenheit slider and text field update dynamically with the corresponding Fahrenheit temperature.
2. When the Fahrenheit slider is moved, the Fahrenheit text field updates dynamically with the value of the slider, and the Celsius slider and text field update dynamically with the corresponding Celsius temperature.
3. When a value is entered in the Fahrenheit text field, the Fahrenheit slider updates with the value of the text field, and the Celsius slider and text field update with the corresponding Celsius temperature.
4. When a value is entered in the Celsius text field, the Celsius slider updates with the value of the text field, and the Fahrenheit slider and text field update with the corresponding Fahrenheit temperature.

Hints

Treat the Converter object as your controller: it should do the work to get values from the inputs and set the output fields appropriately. (Note that in this case, we don't really have an explicit model.)

To convert Celsius into Fahrenheit: multiply by 9, divide by 5, and add 32.

A handy way to convert mentally from Celsius to Fahrenheit (and impress your friends): double the temperature, subtract 10%, and add 32.

Troubleshooting

It's easy to forget to make connections in Interface Builder between all of your objects — so easy, in fact, that it's probably the most common pitfall for new Cocoa programmers. Unfortunately, forgetfulness is compounded by obscurity: it's a not easy to check all of the connections and make sure that they exist and go to the right spot with the correct outlet or target/action. (In this assignment, however, checking the connections is pretty straightforward.)

If you have a missing target/action connection (from a button, for example), clicking the button will appear do nothing. In this case, since there's no target for the button's action, in fact the button *is* doing nothing — more precisely, no message is sent when you click the button. You can diagnose this by setting a breakpoint on the action in the debugger to see whether it's invoked.

If you have a missing outlet connection (to an output text field, for example), the user interface may appear to respond to user actions — for example, messages may be sent when buttons are clicked — but nothing may appear in the output text field. This is an easy case to diagnose: you can usually just check the object with the outlet in the debugger. If the outlet is nil, the connection wasn't made.

Extra Credit

As usual, keep in mind that extra credit only applies if the required behavior outlined above is working. Don't spend time on extra credit until you've got the basics down.

• Æsthetics matter: a particularly attractive, innovative, or clever design of the interface is always appreciated. (If you want consideration for extra credit in this case, you must let us know when you submit your project.)

• Add other temperature scales. For scientists, Kelvin (abbreviated K, not ˚K) would be an interesting addition; for students of scientific history, degrees Réamur or degrees Rankine (˚R) would be appreciated.

• Change the colors of the text fields to visually indicate the temperature: the text should be red for hot temperatures and blue for cold ones. (This is a little advanced for this point in the course, so don't despair if you don't know where to start: look at the `NSColor` documentation, particularly the method `blendedColorWithFraction:ofColor:`. You'll also want to look at the `NSTextField` documentation and the felicitously named method `setTextColor:`.)