Chapter 4: Converting Other Temperature Types

Converting Other Temperature Types

To convert all three temperature types, we still have several changes to make (not necessarily in this order):

1. Our TemperatureDriver class has conditional logic in the main() method to process Fahrenheit values. We need to add similar logic for Celsius and Kelvin numbers.

2. Temperature already has methods to retrieve Celsius and Kelvin temperatures, and we need to repeat that logic for Fahrenheit values.

3. We already have a method in Temperature to store and convert Fahrenheit temperatures. We'll need to repeat similar logic for Celsius and Kelvin.

4. Once we have "set" methods in Temperature for all three types, we'll revisit our constructors, and then we'll be done with the program for this lesson!

Let's do the easiest of those changes first: adding a method to Temperature to retrieve the Fahrenheit value. It will look very much like the getDegreesCelsius() and getDegreesKelvin() methods, but it'll use the Fahrenheit variable instead. I placed this method right after the other two get methods in my class.

```
public double getDegreesFahrenheit() {
    return degreesFahrenheit;
}
```

That wasn't too bad, was it? Now let's add methods for storing and converting Celsius and Kelvin temperatures into our class. They'll both look similar to setDegreesFahrenheit(), but we'll need to change their conversion formulas. If you look back at the original program description, you'll find the formulas we need:

1. The formula for converting Celsius to Fahrenheit is $F = C \times 9 / 5 + 32$.

2. The formula for converting Celsius to Kelvin is $K = C + 273.15$.

3. The formula for converting Kelvin to Celsius is $C = K - 273.15$.

4. The formula for converting Kelvin to Fahrenheit is $F = (K - 273.15) \times 9 / 5 + 32$.

If you copy the method setDegreesFahrenheit() and use the above formulas, you can create two new methods that look like this:

```
public void setDegreesCelsius(double degrees) {
    degreesCelsius = degrees;
    degreesFahrenheit = degreesCelsius * 9.0 / 5.0 + 32.0;
    degreesKelvin = degreesCelsius + 273.15;
}

public void setDegreesKelvin(double degrees) {
    degreesKelvin = degrees;
    degreesCelsius = degreesKelvin - 273.15;
    degreesFahrenheit = degreesCelsius * 9.0 / 5.0 + 32.0;
}
```

Now that we have our "set" methods, let's reconsider how we want our constructors to work.

In the last lesson we had two constructors: a default that set up an object with a default Fahrenheit value of zero, and one that set our Fahrenheit value to whatever value was passed as an argument in the call. But now we want to be able to set more than Fahrenheit temperatures, so what should our default type and value be? Does it even make sense to have a default constructor? I'm not so sure it does.

So I'm going to illustrate a class with only one constructor and no default constructor. That means that users of this class won't be able to create a Temperature object until they know what type of temperature they are using and what its value is. That makes sense, doesn't it? Is there any point in creating a Temperature object without a value to convert?

To set up our new constructor, I'm going to delete both constructors from Temperature and replace them with this single constructor:

```
/**
* This constructor for Temperature sets the temperature
* values to the value in degrees, based on type
*
* @param type temperature scale to use
* @param degrees degrees Fahrenheit
*/
public Temperature(String type, double degrees) {
    if (type.equalsIgnoreCase("C"))
        setDegreesCelsius(degrees);
    else if (type.equalsIgnoreCase("F"))
        setDegreesFahrenheit(degrees);
    else if (type.equalsIgnoreCase("K"))
        setDegreesKelvin(degrees);
}
```

This constructor decides which set method to call based on the value passed in the parameter type. It adds two new coding wrinkles to our constructor.

- It's the first constructor or method we've seen that has two parameters. The first one is a String object that gives us the type of temperature, and the second is a double value with the temperature itself. Java methods can have as many parameters as are needed.

- Instead of the equals() method, I used equalsIgnoreCase(). I'm guessing that the name will tell you what the difference is, but I wanted to point it out so you know how to compare strings when case doesn't matter. So this statement would return true:

if ("A".equalsIgnoreCase("a"))

That takes care of our Temperature class. But without a default constructor, we need to change our main() method to call the new one. So instead of having this line in our Temperature object:

Temperature t1 = new Temperature();

we'll change it to this:

Temperature t1;

Then we'll add this line after we have both our temperature type and value:

t1 = new Temperature(temperatureType, inputTemperature);

Now we just have to add conditional logic to our main() method to handle Celsius and Kelvin temperatures. The easiest way to do that is to copy our Fahrenheit logic and modify it for Celsius and Kelvin values. And we might as well change the comparisons to ignore case at the same time too. Try doing that yourself before you continue, and then compare what you added to what I added to my main() method.

Hide answer

```
if (temperatureType.equalsIgnoreCase("C")) {
    System.out.println("You entered " + inputTemperature +
            " degrees Celsius");
    t1.setDegreesCelsius(inputTemperature);
    System.out.println("which is " + t1.getDegreesFahrenheit() +
            " degrees Fahrenheit");
    System.out.println("and " + t1.getDegreesKelvin() +
            " degrees Kelvin.");
}

if (temperatureType.equalsIgnoreCase("K")) {
    System.out.println("You entered " + inputTemperature +
            " degrees Kelvin");
    t1.setDegreesKelvin(inputTemperature);
    System.out.println("which is " + t1.getDegreesCelsius() +
            " degrees Celsius");
    System.out.println("and " + t1.getDegreesFahrenheit() +
            " degrees Fahrenheit.");
}
```

Hmmm. Do you see anything redundant here? Since our constructor call now sets the temperature type and value, we don't need to call the separate set methods any more. So we can delete the calls to setDegreesFahrenheit(), setDegreesCelsius(), and setDegreesKelvin().

With that, we're finally done! (Except for compiling and testing, of course.) If you have trouble either compiling or testing, you can compare your code to my version in Chapter 5.