

# Chapter 3: Putting Branches Into Your Program

## Putting Branches Into Your Program

Looking back at the part of the program design that's in Chapter 1, we know the program should do different conversions depending on which temperature type the user enters. But where did that temperature type come from?

In the original program description, before we could do conversions, we were supposed to ask our user for a temperature type as well as a temperature value. Once we have the type and the value, *then* we can do conversions correctly. So we need to add two parts to our program: one to get the temperature type, and another to do the correct conversions.

This time, we'll assume our users are *almost* perfect and don't make numeric mistakes. They'll never enter invalid temperatures like *hot* or *xyz*. But we will check to make sure they enter a valid temperature type, since we already need to check that value to decide which conversion to do. So we'll add one simple error message for invalid temperature types.

Getting the temperature type is the easier of the two additions to our program, so let's tackle that first. The original spec said to ask for the temperature type first, before the temperature value, so that's how we'll do it. Also, since we do our input and output in our `main()` method, that's where we'll need to make our first change.

Just to remind you of how our `main()` method looked, here's how we left it:

```

public static void main(String[] args) {
    double inputTemperature = 0.0;
    Scanner keyboard = new Scanner(System.in);
    Temperature t1 = new Temperature();
    Temperature t2;

    System.out.print("Enter a Fahrenheit temperature: ");
    inputTemperature = keyboard.nextDouble();
    System.out.println("You entered " + inputTemperature +
        " degrees Fahrenheit");
    t1.setDegreesFahrenheit(inputTemperature);
    System.out.println("which is " + t1.getDegreesCelsius() +
        " degrees Celsius");
    System.out.println("and " + t1.getDegreesKelvin() +
        " degrees Kelvin.");
    System.out.print("Enter another Fahrenheit temperature: ");
    inputTemperature = keyboard.nextDouble();
    System.out.println("You entered " + inputTemperature +
        " degrees Fahrenheit");
    t2 = new Temperature(inputTemperature);
    System.out.println("which is " + t2.getDegreesCelsius() +
        " degrees Celsius");
    System.out.println("and " + t2.getDegreesKelvin() +
        " degrees Kelvin.");
}

```

## The "Save As" Option



If you'd like to save this version of the program from Lesson 5 and make your Lesson 6 changes to a copy, open the project in BlueJ, go to the **Project** menu, and use the **Save As** option to save a copy of the project with a new name before continuing.

Since we're going to add code to our main() method, I'll remove the object t2 from the program so it won't be so long. That means deleting the last six statements in main(), which use t2, as well as the declaration for t2 at the top of the method.

Now my main() method looks like this:

```

import java.util.Scanner;


/**
 * TemperatureDriver runs and tests the Temperature class.
 *
 * @author Merrill Hall
 * @version 1.0
 */
public class TemperatureDriver {
    /**
     * main() reads two Fahrenheit temperatures and
     * displays their Celsius and Kelvin equivalents.
     */
    public static void main(String[] args) {
        double inputTemperature = 0.0;
        Scanner keyboard = new Scanner(System.in);
        Temperature t1 = new Temperature();

        System.out.print("Enter a Fahrenheit temperature: ");
        inputTemperature = keyboard.nextDouble();
        System.out.println("You entered " + inputTemperature +
            " degrees Fahrenheit");
        t1.setDegreesFahrenheit(inputTemperature);
        System.out.println("which is " + t1.getDegreesCelsius() +
            " degrees Celsius");
        System.out.println("and " + t1.getDegreesKelvin() +
            " degrees Kelvin.");
    }
}

```

To capture the type that the user enters, we'll need a variable to store it in. That value will be a character, so we'll use a String object to hold it. I'll name my variable `temperatureType` since that's what it will store. The first thing we'll add to the program is this new variable declaration and initialization at the start of our `main()` method:

`String temperatureType = "";`



```

Temperature t1 = new Temperature();
String temperatureType = "";

System.out.print("Enter a temperature type (C for Celsius, " +
    "F for Fahrenheit, K for Kelvin): ");
temperatureType = keyboard.next();

```

Now we have a place to store the type, so let's get it from our user. We should give our user a prompt to let him or her know what we expect. So after our declarations, but before we prompt the user for the temperature, let's add a statement to ask for the type:


```
System.out.print("Enter a temperature type (C for Celsius, " +  
    "F for Fahrenheit, K for Kelvin): ");
```

Next we need to read the input value from the Scanner object keyboard, as we did with the numeric temperature in the last lesson. But this time the input value will be a text string instead of a number.

The Scanner methods that programmers most often use to return String objects are `next()` and `nextLine()`. The `next()` method returns the next word from the input (after skipping white space), while the `nextLine()` method returns an entire input line, including any white space. Since the only thing that's supposed to be on the line is our temperature type, let's use `next()`.

Before I go on for so long that you forget what we're trying to do, let me show you the statement we're going to add to read our input. Add this right after the statement that asked for the temperature type:

```
temperatureType = keyboard.next();
```



```
System.out.print("Enter a temperature type (C for Celsius, " +  
    "F for Fahrenheit, K for Kelvin): ");  
temperatureType = keyboard.next();  
System.out.print("Enter a temperature: ");  
inputTemperature = keyboard.nextDouble();
```


The next line of our program prompts our user for a Fahrenheit temperature. Since the user can put in any temperature type now, we should remove the word *Fahrenheit* from the prompt. (I'll leave that to you; I won't bore you with a new version of the output statement.)

The next line is the input line that reads the temperature. That still works just the way we want it to, so we'll leave it alone and keep going.

The line after that is where we start doing things that we want to happen only if the temperature type is Fahrenheit. Last time we displayed the Fahrenheit temperature back to the user, stored and converted it, and then displayed the two converted values. That took four lines in our program. Now we want to change the program to only execute those lines if `temperatureType` contains F. So we need to check for that value before we execute those four lines.

To do that, let's add an if statement to check for the Fahrenheit type. We'll put it between the input line and the first line specific to Fahrenheit temperatures. It will look like this:

```
if (temperatureType.equals("F"))
```



```
System.out.print("Enter a temperature: ");  
inputTemperature = keyboard.nextDouble();  
if (temperatureType.equals("F")) {  
    System.out.println("You entered " + inputTemperature +  
        " degrees Fahrenheit");  
}
```

## What About == and !=?

You probably noticed that I explained Java's comparison operators in Chapter 2, but I didn't use any of them here. We'll get to use them, but let me explain why I didn't use them here.

In Java, character strings are objects, not primitive types. When Java compares objects using the `==` operator, it doesn't compare their contents to see if they contain the same data. Java's designers didn't want it to get bogged down trying to decipher and compare the complex internal structures of Java objects.

So the designers settled on a simple alternative. When Java compares two objects with the `==` operator, it compares their memory addresses. (We talked about memory addresses back in Lesson 3—they're how the computer tracks where it stored data and instructions.) If they have the same memory address, Java considers them equal and returns `true`. If not, Java returns `false`.

In our case `temperatureType` and the String literal `"F"` wouldn't both occupy the same memory address, and Java would always return `false` even if they both contained `"F"`.

Here's how most Java programmers express this: For Java classes whose objects will be compared to a method (a set of commands that you can use over and over), they name the method `equals()`, and they have the method return a boolean value of `true` if the contents of the two objects are equal and `false` if not.

The `String` class has that method, and that's what I've called it in this `if` condition. Since the method returns a boolean value, and that's what an `if` statement needs for its condition, I don't need to compare the value to anything else. I can use it as the complete condition.

We'll also need to put curly brackets around the four statements that make up our Fahrenheit logic. The curly brackets will tell Java to execute all four of those lines if the temperature is in Fahrenheit.

As another housekeeping step, we should indent the lines we put inside the curly brackets. Most Java programmers indent each block of code so the program logic will be easier to read. I indented each of them four spaces, which is a standard amount.

Our program now *reads* any type of temperature, but it will only *display* conversion information for Fahrenheit temperatures. In Chapter 4, we'll look at the changes we need to make to display conversion information for the other two types. In the meantime, here's what our `main()` method looks like at this point:

```
public static void main(String[] args) {  
    double inputTemperature = 0.0;  
    Scanner keyboard = new Scanner(System.in);  
    Temperature t1 = new Temperature();  
    String temperatureType = "";  
  
    System.out.print("Enter a temperature type (C for Celsius, " +  
        "F for Fahrenheit, K for Kelvin): ");  
    temperatureType = keyboard.next();  
    System.out.print("Enter a temperature: ");  
    inputTemperature = keyboard.nextDouble();  
  
    if (temperatureType.equals("F")) {  
        System.out.println("You entered " + inputTemperature +  
            " degrees Fahrenheit");  
        t1.setDegreesFahrenheit(inputTemperature);  
        System.out.println("which is " + t1.getDegreesCelsius() +  
            " degrees Celsius");  
        System.out.println("and " + t1.getDegreesKelvin() +  
            " degrees Kelvin.");  
    }  
}
```