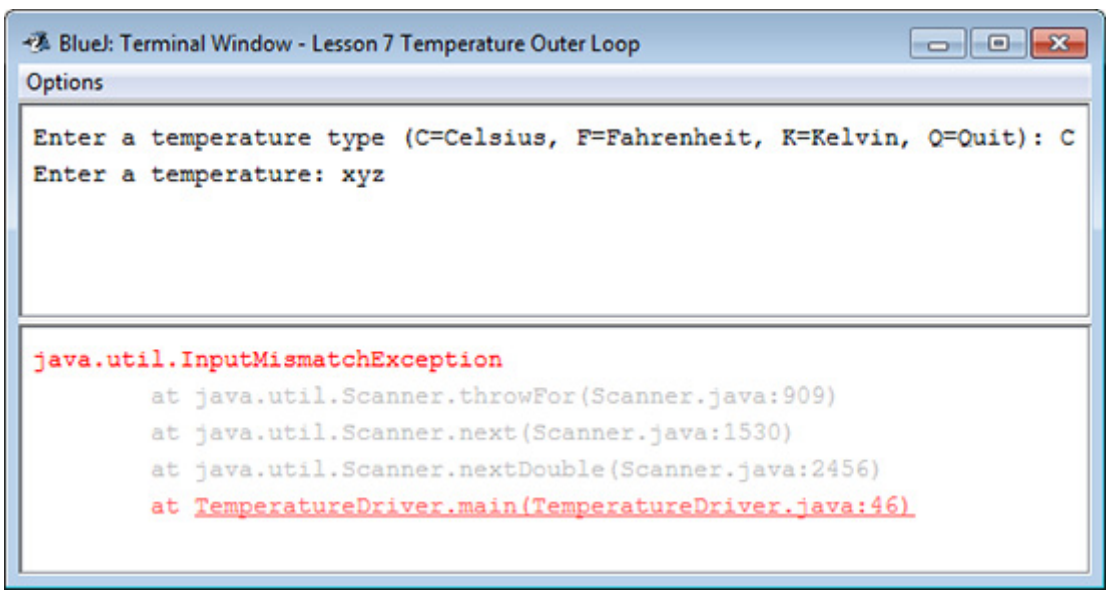
Chapter 3: More Loops

More Loops

We now have a temperature conversion program that will repeat itself until our user enters a Q to quit.

However, if you've made any mistakes while testing this program, you probably noticed that it doesn't do well with unexpected input. For example, if you put in anything besides the letters C or c, F or f, K or k, or Q or q, the program still asks for a temperature even though it doesn't really know what type the temperature is, and once it has a temperature, it doesn't do anything with it. That's not how we'd like it to behave.

Also, if you've entered anything besides a number when the program asked for a temperature, you've seen an even bigger problem: The program breaks! (Or in common programming terminology, it *crashes* or *blows up*.) We get an error that our program doesn't know how to handle. It looks something like this:



Exception error message

BlueJ tries to help by opening our source code and highlighting the line that caused the problem. In this case, the line it highlights is:

```
inputTemp = keyInput.nextDouble();
```

This line reads the numeric temperature value.

BlueJ also displays an error message at the bottom of the terminal window (as you can see in the image above). That message explains what caused our program to fail. In this case, the error reads *java.util.InputMismatchException*. When a Java program runs into situations it can't handle, it issues (or in Java terminology, it *throws*) what it calls *exceptions*. An *exception* is an event or an error that disrupts the flow of the program's instructions. The computer can't detect the problem at compile time, but the error stops the flow of the program at run time.

Java has many types of exceptions, and in future lessons, we'll discuss catching them and handling them. The *exception type* in the error (in this case it's *InputMismatchException*) is Java's attempt to tell us what went wrong.

Our exception was an input mismatch, which means that Java couldn't figure out how to make the input into the type of data it was looking for. Since our program was looking for a valid number, any input that's not a number will cause this error. You can see that I entered "xyz" in the window above to cause the exception.

One sign of a well-designed program is its ability to handle bad input without crashing. Our program doesn't do that yet, although our original program description told us what was supposed to happen: The program was supposed to circle back and ask for the input again.

So what we'll do next is add *input edits* to our program to make sure that the input we get is right, and if not, then to ask for it again. An input edit is program logic that makes sure our users are giving us good data. We'll start with an edit of the temperature type in this chapter and deal with the numeric temperature value in the next chapter.

There's More Than One Way to Solve a Problem

Note

We're going to take the simplest approach in this lesson, which is to put the edits directly into our driver program. In the next lesson, I'll show you a different approach that I think is better. I'm doing it the simpler way now because I need to introduce a few more topics before we can use the other technique.

How can we edit the temperature type? We need to tell the user about the mistake and ask for the input again. How do we do that? We need to go back to the start of our while loop, right after we asked for the temperature type.

First let's look at how to make sure we get one of the valid values: Q, C, F, or K. We want to keep asking for a type until we get one of those. When I say "keep asking," I'm implying that we need to repeat an action, which will require another loop. This one will be inside the loop we already have, since that outer loop already controls converting temperatures until the user tells us to quit. This new *inner loop* (also called a *nested loop*) will make sure that every time we ask for a type, we'll get a good one.

Near the start of that outer loop, right after we read the user's input, I want to add an inner loop that will repeat until we get a good input value. The condition for this loop will be similar to the one for our outer loop, which repeats until the boolean value changes. We're going to use another boolean variable to control this loop too, since this time we have even more conditions to check.

This time I'll name my boolean goodType. A value of true will indicate that we have a valid temperature type, and a value of false will tell us we don't yet. So it'll start out false. Here's the declaration I'll add to the top of the main() method:

```
boolean goodType = false;
```

Since I may process multiple temperatures in our outer loop, I also need to reset the boolean to false every time I process a new temperature. So I'll set up that assignment and the basic loop structure like this, right after the statement that reads the temperature type and stores it into temperatureType:

```
// variable declarations, outer loop, and prompt are here
temperatureType = keyInput.next();
goodType = false;
while ( ! goodType) {
}
```

That sets up the condition to control our new loop. But what needs to go in it?
I need an if statement to either tell me I have a good temperature type by setting goodType to true or give my user an error message and ask for a new type. I have four letters to check in the if, and since I want to allow uppercase and lowercase letters, I'll use the String method equalsIgnoreCase() again.

If our user entered a valid type, all we need to do is set goodType to true, and the loop will end. If the type isn't one we want, we need an error message, a request for a new value, and a statement to read that value before we go back to the top of the loop. You may choose different text for your error messages, but my statements look like this:

```
if (temperatureType.equalsIgnoreCase("Q") ||
        temperatureType.equalsIgnoreCase("C") ||
        temperatureType.equalsIgnoreCase("F") ||
        temperatureType.equalsIgnoreCase("K")) {
    goodType = true;
}
else {
    System.out.println("Invalid temperature type!");
    System.out.println("The type must be C, F, K, or Q.");
    System.out.print("Please enter the temperature type again: ");
    temperatureType = keyInput.next();
}
```

If you put these lines into the inner loop, you'll be ready to try this. You should be able to repeat the request for temperature type until you get a good one. Try it a few times and see.
Here's an example of what it might look like:

Text equivalent start.

Terminal Window displaying example output:

```
Enter a temperature type (C= Celsius, F= Fahrenheit, K= Kelvin, Q=Quit): xyz
Invalid temperature type!
The type must be C, F, K or Q.
Please enter the temperature type again: 99
Invalid temperature type!
The type must be C, F, K or Q.
Please enter the temperature type again: f
Enter a temperature: 32
You entered 32.0 degrees Fahrenheit
which is 0.0 degrees Celsius
and 273.15 degrees Kelvin.
Enter a temperature type (C for Celsius, F for Fahrenheit, K for Kelvin): q

Program ended.
```

Text equivalent stop.

That looks better, doesn't it?

Once your program is working to this point, go on to the next chapter. That's where we'll add our last loop, which will make sure we have a valid number for a temperature.

In case you have any problems, here's a link to my complete TemperatureDriver class at this point so you can compare it to yours:

Hide answer

```
import java.util.Scanner;
/**
 * TemperatureDriver runs and tests the Temperature class.
 *
 * @author Merrill Hall
 * @version 3.0
 */
public class TemperatureDriver {
    /**
     * main() reads a temperature type and value, then
     * converts it to the other two temperature scales.
     */
    public static void main(String[] args) {
        double inputTemperature = 0.0;
        Scanner keyInput = new Scanner(System.in);
        Temperature t1;
        String temperatureType = "";
        boolean moreTemperatures = true;
        boolean goodType = false;

        while (moreTemperatures) {
            System.out.print("Enter a temperature type (C=Celsius, " +
                "F=Fahrenheit, K=Kelvin, Q=Quit): ");
            temperatureType = keyInput.next();
            goodType = false;
            while ( ! goodType) {
                if (temperatureType.equalsIgnoreCase("Q") ||
                        temperatureType.equalsIgnoreCase("C") ||
                        temperatureType.equalsIgnoreCase("F") ||
                        temperatureType.equalsIgnoreCase("K")) {
                    goodType = true;
                }
                else {
                    System.out.println("Invalid temperature type!");
                    System.out.println("The type must be C, F, K, or Q.");
                    System.out.print("Please enter the temperature type again: ");
                    temperatureType = keyInput.next();
                }
            }
            if (temperatureType.equalsIgnoreCase("Q")) { // quit
                moreTemperatures = false;
                System.out.println("\nProgram ended.");
```

```
        }
        else {
            System.out.print("Enter a temperature: ");
            inputTemperature = keyInput.nextDouble();
            t1 = new Temperature(temperatureType, inputTemperature);

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