Logical branching tutorial

In this tutorial, you'll write some boolean expressions and use them in conditional statements to branch your code depending on the values of the program's variables. You'll write a program that tells the user how to prepare for their day based on weather conditions.

By the end of this tutorial, you'll have written code that:

- Combines values, variables, and operators into boolean expressions.
- Uses boolean expressions in `if` statements to control the flow of your program logic.
- Uses `if` / `else if` constructs to make more complex decisions about program flow.

Getting started

To get started, follow these steps:

1. Import the project into IntelliJ IDEA.
2. Once the project is open, navigate to the project folder `src/main/java/com/techelevator`.
3. Open the `Tutorial.java` file by double-clicking on the filename.

You'll see the starter code for this project. All of your work will be in the `main` method of this class.

Step One: Review the starting code

Before starting the tutorial, review the starting code to understand what it does:

```
// Step One: Greet the user
System.out.println("*************************************");
System.out.println("*** Welcome to the Weather Station ***");
System.out.println("*************************************");
```

This code prints a welcome message to the user on the console.

Now run the code to see how it behaves. To run your program, click `Run` on the menu bar across the top of the IntelliJ IDEA window, then click `Run...` from the menu that's displayed. Finally, click on `Tutorial` in the dialog that appears.

In the window at the bottom of your screen, you can see your program output. There's a line indicating the command that was used to run your program, followed by your program's output:

```
*************************************
*** Welcome to the Weather Station ***
*************************************
```

Step Two: Declare and initialize program variables

When writing a new program, you must think about the data the program will store and act upon. This program recommends certain actions for the user to take based upon the weather conditions. Those weather conditions are represented by variables in the program.

For this program, you'll check the expected high temperature and likelihood of precipitation for the day, so you'll create variables to store that information.

Under the comment that begins with `// Step Two:`, add these lines of code:

```java
int highTemperatureF = 55;
boolean precipitationExpected = true;
final int FREEZING_TEMPERATURE_F = 32;
```

- `highTemperatureF` stores the expected high temperature for today. The `F` signifies that the temperature is represented as Fahrenheit.
- `precipitationExpected` stores whether it's likely to rain or not.
- `FREEZING_TEMPERATURE_F` is a *constant* to represent the freezing point. You could code `32` into your program every place you need this value, but giving it a name makes your program easier to understand.

Step Three: Report weather conditions to the user

You'll start by reporting today's expected conditions to the user. First, you'll print the temperature, then whether precipitation is expected. If precipitation is expected, you'll write one message. If not, you'll write a different one.

Under the `// Step Three:` comment, add this code:

```java
System.out.println("Expected high temperature: " + highTemperatureF + "\u00B0F.");
// Degrees character in Unicode: \u00B0
if (precipitationExpected) {
    System.out.println("The chance of precipitation is high.");
} else {
    System.out.println("The chance of precipitation is low.");
}
```

The `if` statement requires a *boolean expression* inside the parentheses. Remember that a boolean expression is a combination of values, variables, and operators that are evaluated and produce a boolean result. Since the variable `precipitationExpected` is of type `boolean`, it's in fact a simple boolean expression.

After adding this code, run your program again. This time, you can select `Run 'Tutorial'` from the `Run` menu.

You'll see this in the output window:

```
*************************************
*** Welcome to the Weather Station ***
```

```
*************************************
Expected high temperature: 55°F.
The chance of precipitation is high.
```

Step Four: Determine if the user should bring a coat

Now you'll advise the user on how to prepare for their day. If the temperature won't reach 65°F degrees today, you'll advise the user to bring a coat.

Under `// Step Four:`, add this code:

```java
if (highTemperatureF < 65) {
    System.out.println("* Don't forget to bring a coat today.");
}
```

Run your program again. You can again use the menu option `Run > Run 'Tutorial'`. Or you can try the keyboard shortcut `Shift-F10`.

Your output now looks like this:

```
*************************************
*** Welcome to the Weather Station ***
*************************************
Expected high temperature: 55°F.
The chance of precipitation is high.
* Don't forget to bring a coat today.
```

> If you want, change the value in `highTemperatureF` to a number greater than 65, and run your program again. Did you see the coat advice this time?

Step Five: Determine if the user should bring an umbrella

Next, you'll advise the user to bring an umbrella if conditions warrant it. If precipitation is expected, the user should bring an umbrella. However, if the temperature stays below freezing, they won't need one since umbrellas are ineffective in snowy weather. So, you'll need to test whether precipitation is expected *and* the temperature is above freezing.

Under `// Step Five:`, add this code:

```java
boolean isAboveFreezing = highTemperatureF > FREEZING_TEMPERATURE_F;
if (precipitationExpected && isAboveFreezing) {
    System.out.println("* Be sure to pack an umbrella!");
}
```

In this code, you first created a boolean variable `isAboveFreezing` to store the result of a boolean expression `highTemperatureF > FREEZING_TEMPERATURE_F`. You did this because later in the program, you'll need to know whether the temperature is above freezing. So you're testing the condition once and storing the result.

When you run the program, your output looks like this:

```
*************************************
*** Welcome to the Weather Station ***
*************************************
Expected high temperature: 55°F.
The chance of precipitation is high.
* Don't forget to bring a coat today.
* Be sure to pack an umbrella!
```

Step Six: Refine the advice on wearing a coat

In Step Four, you determined whether a user should bring a coat. Upon further reflection, you decided to be more specific on what type of coat the user should wear.

You'll make this decision based on the expected high temperature:

- If the temperature won't get above 0°F, tell the user to bundle up.
- If the temperature will be above 0°F but not above freezing, advise a winter coat.
- If the temperature will be above freezing but below 65°F, advise a light jacket.
- If the temperature will be 65°F or higher, don't advise the user on wearing a coat.

Under `// Step Six:`, add this code:

```java
if (highTemperatureF <= 0) {
    System.out.println("* Bundle up! Wear everything you have!");
} else if (!isAboveFreezing) {
    System.out.println("* Please wear a warm winter coat!");
} else if (highTemperatureF < 65) {
    System.out.println("* Take along a light jacket.");
}
```

Note that you tested for temperatures *not above* freezing by testing the `isAboveFreezing` variable you set earlier, but used the `!` (NOT) operator to get its "opposite" value.

Also note that if the temperature gets to 65°F, *none* of the print statements run, and the user isn't advised to bring a coat.

When you run your program, you'll see output similar to this:

```
*************************************
*** Welcome to the Weather Station ***
*************************************
```

```
Expected high temperature: 55°F.
The chance of precipitation is high.
* Don't forget to bring a coat today.
* Be sure to pack an umbrella!
* Take along a light jacket.
```

> If you want, you can remove the code you added in Step Four so you aren't duplicating your coat advice.

Step Seven: Advise the user on temperature variation

Finally, you'll advise the user to dress in layers if a dramatic temperature swing is expected today.

To do this, you'll introduce another program variable, lowTemperatureF, and then check the difference between today's high and low. If there's more than a 20°F swing, advise the user to dress in layers.

Under // Step Seven:, add this code:

```
int lowTemperatureF = 34;
if (highTemperatureF - lowTemperatureF > 20) {
    System.out.println("* Dress in layers today, as the temperature may swing
dramatically.");
}
```

Look at the if condition. How many expressions do you see?

In this condition, there are two expressions: one *arithmetic expression* and one *boolean expression*. The arithmetic expression highTemperatureF - lowTemperatureF is evaluated first, resulting in an integer value.

Next, this result is used in the boolean expression result_of_arithmetic_expression > 20, where result_of_arithmetic_expression represents highTemperatureF - lowTemperatureF, to test if the difference is more than 20°F. As you continue your coding journey, you'll find this "embedding" or *nesting* of expressions is common and can get quite complex.

Run your program and you'll see output like this:

```
************************************
*** Welcome to the Weather Station ***
************************************
Expected high temperature: 55°F.
The chance of precipitation is high.
* Don't forget to bring a coat today.
* Be sure to pack an umbrella!
* Take along a light jacket.
* Dress in layers today, as the temperature may swing dramatically.
```

Next steps

Go back to your code and change values for the high temperature, low temperature, and precipitation variables to represent different daily conditions. After each change, run the program, and see how your program output changes based on the starting data.