Debugging simple constructs

If statements
First of all, we might want to test if the program visits a branch of an if statement at all.

Click the gutter against the first line in the if block. A red circle appears: this is a line breakpoint. It will tell the debugger to suspend the program execution whenever this line is executed.

Click the **Debug** button that is located next to **Run** and **Run/Debug Configurations**. Apart from building, installing, and running the app (like **Run**), **Debug** will also attach a debugger.

The line with the breakpoint turns blue. It means that the program was just about to execute it, but it was suspended and now it waits for commands from you.

For now, we've just made sure that the program has executed the if in question. If it hadn't, then we wouldn't have stopped at this line, because the program would've never reached the line with the breakpoint.

Let's resume the program so that it can execute the remaining statements and terminate. You can do this by pressing **Resume** in the **Debug** tool window.

Loops
Just like with if statements, you can set a breakpoint inside a loop, and the program will suspend it each time it executes the line, that is, with each iteration of the loop.

```
void whatever() {
    char rangeStart = 'C';
    char rangeEnd = 'Y';
    char findLetter = 'Q';
    for (char c = rangeStart; c <= rangeEnd; c++) {
        if (c == findLetter) {
            System.out.printf("Character %s is within range %s-%s",
                    findLetter, rangeStart, rangeEnd);
            return;
        }
    }
    System.out.printf("Character %s is not within range %s-%s",
            findLetter, rangeStart, rangeEnd);
}
```

1. As in the previous example, set a breakpoint somewhere in the loop.

2. Right-click the breakpoint, and specify the condition: c == 'H'. The condition is just a boolean expression that must evaluate to true in order for the program to stop at this breakpoint. Note that the program evaluates the condition in the context of the code that the breakpoint is set in. For example, the condition c == 'H' will not work for a breakpoint outside the loop, because the variable c is not visible there.

- tart debugging by either clicking Debug or attaching a debugger to the existing process.
- The program stops in the loop. In contrast with the previous example, the debugger has not suspended the program every time it hit the breakpoint. We didn't stop until the loop was checking the letter H.