Title: Logical operators: NOT “!”

By Adrian Hunter

Assignment Definition and General Feedback by Michael Panitz
at Cascadia Community College (http://www.cascadia.edu)

Table of contents:

(Required sections have a * after them in the above list)

Quick Summary:

In Java, you can use if/else or while statements to run a certain set of instructions if that statement is true. However, we can use the symbol “!” to create the opposite statement. Using this symbol to create a NOT statement allows us to make a program do what we want it to do in an easier way.

While statements will be covered in depth later in this book.

When To Use This / Avoid This:

You can use this technique to do something instead of using an else statement after an if statement to do what you want. If you know that you want the robot to do something specific if a condition is false, then you can use “!”. Also, else statements cannot be used after while statements, so using “!” allows you to do something as long as something is false.

There is no situation where you would need to be sure to avoid NOT statements, but be sure that you know when to use them. Using a NOT statement when you could use a standard conditional statement can make it difficult to get the robot to do what you want it to do.

Example of Usage:

Let's say you want to something particular when it reaches a wall.

There are three main steps:
First, determine what it is you want your robot to do. In our example, when the robot reaches the wall on the far left, you want it to turn left and face south.

Second, you begin writing the code just like you would a normal if/else or while statement. However, since this is a NOT statement you will add a "!" after the open parenthesis and before your actual statement.

Third, you write out the steps you want the robot to take if the condition is met like you would a normal if/else or while statement. In this example, we want the robot to turn left and face south once it reaches the wall.

Line	Code
1.	`import becker.robots.*;`
2.	
3.	
4.	`public class With_A_NOT_Operator extends Object`
5.	`{`
6.	`   public static void main(String[] args)`
7.	`   {`
8.	`      City toronto = new City();`
9.	`      Robot jo = new Robot(toronto, 1, 5, Direction.WEST, 0);`
10.	`      new Wall(toronto, 1, 1, Direction.WEST);`
11.	
12.	`      jo.move();`
13.	`      jo.move();`
14.	`      jo.move();`
15.	`      jo.move();`
16.	`      if (!jo.frontIsClear())`
17.	`      {`
18.	`         jo.turnLeft();`
19.	`      }`
20.	
21.	`   }`
22.	`}`

Example Trace:

In a nutshell, this technique allows you to do something if or as long as something is not true.

In order to go over these details more thoroughly, here is a (partial) trace of the above program, with some additional explanation afterwards.

Line #	Program Statement	jo St #	jo Ave #	Jo Direction	jo Back pack	Wall #1	True/False
6	`public static void main(String[] args)`	-	-	-	-	-	-
8	`City toronto = new City();`	-	-	-	-	-	-
9	`Robot jo = new Robot(toronto, 1, 5, Direction.WEST, 0);`	1	5	West	0	-	-
10	`new Wall(toronto, 1, 1, Direction.WEST);`	1	5	West	0	(1,1,W)	-
12	`jo.move();`	1	4	West	0	(1,1,W)	-
13	`jo.move();`	1	3	West	0	(1,1,W)	-
14	`jo.move();`	1	2	West	0	(1,1,W)	-
15	`jo.move();`	1	1	West	0	(1,1,W)	-
16	`if (!jo.frontIsClear())`	1	1	West	0	(1,1,W)	True
18	`jo.turnLeft();`	1	1	South	0	(1,1,W)	-

You'll notice that the trace starts at beginning of the **main** function, on line 6, like normal. It proceeds like a normal program until line 16. Previously, once you had arrived at the wall you would have had to use a conditional statement that said, “if the front is clear, then do something, otherwise (else) turn left,” to get the desired effect. However as you can see in line 16 you can use “!” to create a NOT statement that says, “if the front is NOT clear, then do something,” and create the same effect.

Syntax Explanation:

Note on syntax of command: The syntax for a NOT statement is exactly the same as a standard if/else or while statement. The only difference between the two is the “!” after the open parenthesis and right before your condition.

Let's start with the program as it's written here.

Line #	Program Source Code
1.	`import becker.robots.*;`
2.	
3.	
4.	`public class Without_A_NOT_Operator extends Object`
5.	`{`
6.	`   public static void main(String[] args)`
7.	`   {`
8.	`      City toronto = new City();`
9.	`      Robot jo = new Robot(toronto, 1, 5,` `          Direction.WEST, 0);`
10.	`      new Wall(toronto, 1, 1, Direction.WEST);`
11.	
12.	`      jo.move();`
13.	`      jo.move();`
14.	`      jo.move();`
15.	`      jo.move();`
16.	`      if (jo.frontIsClear())`
17.	`      {`
18.	`         jo.move();`
19.	`      }`
20.	`      else`
21.	`      {`
22.	`         jo.turnLeft();`
23.	`      }`
24.	
25.	`   }`
26.	`}`

Below, you can see the finished program with the differences highlighted in yellow, so it's easy to see what's been added/changed).

Line #	Program Source Code
1.	`import becker.robots.*;`
2.	
3.	
4.	`public class With_A_NOT_Operator extends Object`
5.	`{`
6.	`   public static void main(String[] args)`
7.	`   {`
8.	`      City toronto = new City();`

```
9.          Robot jo = new Robot(toronto, 1, 5,
                Direction.WEST, 0);
10.         new Wall(toronto, 1, 1, Direction.WEST);
11.
12.         jo.move();
13.         jo.move();
14.         jo.move();
15.         jo.move();
16.         if (!jo.frontIsClear())
17.         {
18.            jo.turnLeft();
19.         }
20.
21.     }
22. }
```

Since you have added the "!" before the condition, you no longer need the else statement to get the desired effect. Now, Java runs through the program and when it arrives at line 16 asks if the front of the robot is NOT clear. If that is true, the program will run the commands within the statement. If that is false, it will skip those commands and continue with the rest of the program.

Help With The Logic:

This is good to use whenever you know you want to do something when a condition is false. If you only use standard if statements, but know that you want to do something if that condition is false, then you can use a NOT statement instead of an else statement.

You cannot use else statements with while statements. This makes the "!" operator your only option if you want to do something as long as something is false.

Important Details:

- A NOT statement is not a perfect replacement for every else statement. You may have a situation where you want to do one thing or, if you can't do that, do something else. In that case, a NOT statement is not appropriate and you need to use an if/else statement.

- You may follow a NOT statement with an else statement. As mentioned in Examples of Usage, a NOT statement works just like a standard if statement, the only difference being that you want your condition that follows the "!" operator to be false.

Mistakes People Commonly Make (And How to Fix Them):

- **Quick Name Of Mistake:** Missing "!"
  Detailed Example Of Error: You know that you want to do something if a condition is false, but forget to put in the "!" operator. If this happens you may encounter a logic error.
  Detailed Example Of Fix: (unless it's obvious): Check your conditional statements and find the one that needs the "!" operator.

See the previous section about if/else statements and following sections about while statements for potential errors in syntax.

Licensing

Plagiarism

If you believe that some or all of this document infringes on your intellectual property (i.e., part or all of this document is copied from something you've written) please immediately contact Mike Panitz at Cascadia Community College (perhaps using the Faculty And Staff Directory at http://www.cascadia.edu/pages/searchtemplate.aspx)