Logical operators: NOT (!)

Written by: Jordan Teater and Nick Vitulli

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

Table of contents:

- Quick Summary
- When To Use (or Avoid) This
- Example Of Usage
- Example Trace
- Syntax Explanation

Quick Summary:

In Java, you can use operators which can turn around a statement to create the opposite of what a statement is asking. such as the “if front is clear” can easily become “if front is not clear”

When To Use This / Avoid This:

There are several situations where you will want to use this technique. This technique can be used if you know you want a certain outcome when something happens. Lets say a robot is moving along a street and reaches a wall. Instead of using an “if the front is clear command” you can use the NOT operator and say “if the front is ‘not’ clear” then do this command. This logical operator is also commonly used in search engines. while researching something if you don't want to have some of the results show up using not can limit what you are looking for so you can narrow your search. An example would be typing in the search engine “Patriots not new England” because if you are researching Patriots from the American Revolution you don't want the football team showing up every other link.

There are several situations where you will want to avoid this technique such as if you know the exact details of what is going to happen. If you know the robot is going to have a wall in front of him then there is no point in making an “if the front is ‘not’ clear” because you already know the outcome. Making a NOT statements will just add extra work if you know what the outcome is already going to be.

Example Of Usage:

Let's say you want the robot karel to turn around and move when he reaches a wall.

There are three steps:
First create the city with a wall at 3, 2 facing east and a robot (in this example we will name him karel) at 3, 0 facing East, note the robot is 2 spaces from the wall.

Second, move the robot so he will reach the wall

Third, make a NOT statement so when the front is not clear it can turn around and move away from the wall.

Line	Code
1.	import becker.robots.*;
2.	public class Starting_Template extends Object
3.	{
4.	public static void main(String[] args)
5.	{
6.	City Seattle = new City();
7.	Robot karel = new Robot(Seattle, 3, 0, Direction.EAST, 0);
8.	new wall (Seattle, 3, 2, Direction.EAST);
9.	
10.	while(karel.frontIsClear())
11.	{
12.	karel.move();
13.	}
14.	if(! karel.frontIsClear())
15.	{
16.	karel.turnLeft();
17.	karel.turnLeft();
18.	karel.move();
19.	}
20.	
21.	
22.	
23.	

24.	
25.	
26.	
27.	
28.	
29.	
30.	
31.	
32.	
33.	
34.	
35.	
36.	
37.	
38.	
39.	
40.	

Example Trace:

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

Line#	code	karel Str # Ave#	karel direction	Wall	True/ False
7	Robot karel = new Robot(Seattle, 3, 0, Direction.EAST, 0);	3,0	E	-	-
8	new wall (Seattle, 3, 2,	3,0	E	(3,2,E)	-

	Direction.EAST);				
10	while(karel.frontIsClear())	3,0	E	(3,2,E)	T
12	karel.move();	3,1	E	(3,2,E)	-
10	while(karel.frontIsClear())	3,1	E	(3,2,E)	T
12	karel.move();	3,2	E	(3,2,E)	-
10	while(karel.frontIsClear())	3,2	E	(3,2,E)	F
14	if(! karel.frontIsClear())	3,2	E	(3,2,E)	T
16	karel.turnLeft();	3,2	N	(3,2,E)	-
17	karel.turnLeft();	3,2	W	(3,2,E)	-
18	karel.move();	3,1	W	(3,2,E)	-
END	PROGRAM				

Syntax Explanation:

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

1.	import becker.robots.*;
2.	public class Starting_Template extends Object
3.	{
4.	public static void main(String[] args)
5.	}
6.	City Seattle = new City();
7.	Robot karel = new Robot(Seattle, 3, 0, Direction.EAST, 0);
8.	new wall (Seattle, 3, 2, Direction.EAST);
9.	
10.	while(karel.frontIsClear())
11.	{
12.	karel.move();
13.	}
14.	
15.	
16.	karel.turnLeft();
17.	karel.turnLeft();
18.	karel.move();
19.	

20.	
21.	
22.	
23.	
24.	
25.	
26.	
27.	
28.	
29.	

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

1.	import becker.robots.*;
2.	public class Starting_Template extends Object
3.	{
4.	public static void main(String[] args)
5.	}
6.	City Seattle = new City();
7.	Robot karel = new Robot(Seattle, 3, 0, Direction.EAST, 0);
8.	new wall (Seattle, 3, 2, Direction.EAST);
9.	
10.	while(karel.frontIsClear())
11.	{
12.	karel.move();
13.	}
14.	if(! karel.frontIsClear())
15.	{
16.	karel.turnLeft();
17.	karel.turnLeft();
18.	karel.move();

19.	}
20.	
21.	
22.	
23.	
24.	
25.	
26.	
27.	
28.	
29.	
30.	
31.	
32.	
33.	
34.	
35.	
36.	
37.	
38.	

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)