Title: If/ Else Statements

Kyle Dyson

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 have the program do something if certain conditions are met. You can also tell the program what to do if these conditions are not met. These are called If/Else statements. If statements allow you to as

When To Use This / Avoid This:

There are several situations where you will want to use this technique. You really want to use this technique for what its for. If you want the robot to, say, check if theres a wall infront of it and what it should do if thats true, you would use if/ else statements. Any situation that requires the robot "checking" if somethings true or not is the best situation to use an if/ else statement.

There are several situations where you will want to avoid this technique. Any sort of loop is where you'll want to avoid these. Technically you can have a loop if you repeat if/ else statemnets, but that can take up a lot of lines of code. For those situations, you would most likely want to stick to while loops.

Example Of Usage:

Let's say you want the robot to check to see if theres a wall in front of it, and if there is it should turn left. If there is not a wall we want it to move.

There are ___4____ steps:

First, we write an if statement asking if the space in front of the robot is clear

Second, we write what we want the robot to do if it's clear (move)

Third, we add an else statement for the situation if the front is not clear

fourth, we add what we want the robot to do in that situation(turn left).

Line	Code
1.	Import becker.robots.*;
2.	
3.	Public class starting_template extends object
4.	{
5.	Public static void main(string[] args)
6.	{
7.	City toronto = new City();
8.	Robot Jo = new robot(toronto, 3, 0, Direction.EAST, 0);
9.	New wall(toronto, 3, 1, Direction.EAST);
10.	
11.	jo.move();
12.	if(jo.frontIsClear())
13.	{
14.	jo.move();
15.	}
16.	else
17.	{
18.	Jo.turnLeft();
19.	}
20.	if(jo.frontIsClear())
21.	{
22.	jo.move();
23.	}
24.	else
25.	{
26.	Jo.turnLeft();
27.	}
28.	}
29.	}

30.	

Example Trace:

In a nutshell, this technique allows you to

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 Backpack	Jo Dir	Wall	true/false			
7	City toronto = new City();	-	-	-	-	-	-			
8	Robot Jo = new robot(toronto, 3, 0, Direction.EAST, 0);	3	0	0	E	-	-			
9	New wall(toronto, 3, 1, Direction.EAST);	3	0	0	E	3,1,E	-			
11	jo.move();	3	1	0	E	3,1,E	-			
12	if(jo.frontIsClear())	3	1	0	E	3,1,E	F			
16	else	3	1	0	E	3,1,E	T			
18	Jo.turnLeft();	3	1	0	N	3,1,E	-			
20	if(jo.frontIsClear())	3	1	0	N	3,1,E	T			
22	jo.move();	2	1	0	N	3,1,E	-			

Syntax Explanation:

Note on syntax of command:

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

Line #	Program Source Code
1.	Import becker.robots.*;
2.	
3.	Public class starting_template extends object
4.	{
5.	Public static void main(string[] args)
6.	{

7.	City toronto = new City();
8.	Robot Jo = new robot(toronto, 3, 0, Direction.EAST, 0);
9.	New wall(toronto, 3, 1, Direction.EAST);
10.	
11.	jo.move();
12.	
13.	

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.	Public class starting_template extends object
4.	{
5.	Public static void main(string[] args)
6.	{
7.	City toronto = new City();
8.	Robot Jo = new robot(toronto, 3, 0, Direction.EAST, 0);
9.	New wall(toronto, 3, 1, Direction.EAST);
10.	
11.	jo.move();
12.	if(jo.frontIsClear())
13.	{
14.	jo.move();
15.	}
16.	else
17.	{
18.	Jo.turnLeft();
19.	}
20.	if(jo.frontIsClear())
21.	{
22.	jo.move();
23.	}

```
24.  else
25.  {
26.  Jo.turnLeft();
27.  }
28.  }
29.  }
```

Help With The Logic:

This is good to use whenever you want the program to check a certain variable, and if that variable is true then to do something. The else part is good to use when you want something to happen if the condition is not met.

Important Details:

- Point #1: When you are tracing programs with if else statements, you need to remember that these statements return true or false. So when tracing them, you need a true false column that identifies if the statement is true or false.

- Point #2: the else part is not mandatory, you can have just if statements. You would want to do that in a situation where you just want to check a condition and if its met do something, but you don't have something specific you want it to do if the condition is not met.

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

- **Quick Name Of Mistake: variable mistakes**
  Detailed Example Of Error: when using if/else statements, you will sometimes accidently use the wrong variable, or misspell it and will be given an error.
  Detailed Example Of Fix: (unless it's obvious): make sure you correctly spell the variable name and use the correct variable.

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)