Basic “If” Statements

Authors: Beshoy Sarkis
Orion Steinbrueck

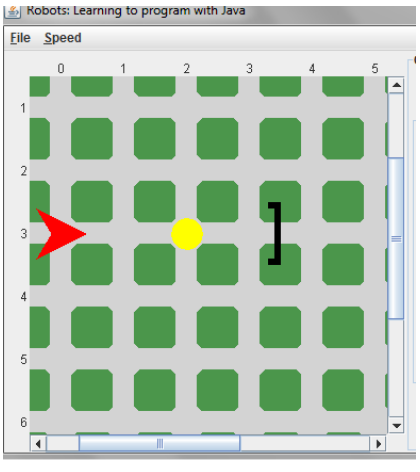
Assignment Definition and General Feedback by Michael Panitz
at Cascadia Community College (http://www.cascadia.edu)

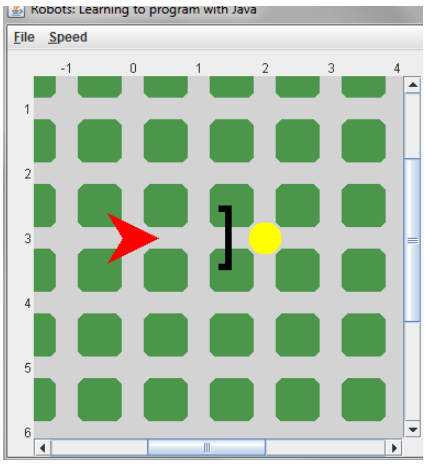
Table of Contents

Quick Summary:

There are times when you are writing code that you cannot use a move command or a pick command because you do not know if there is a wall in front of the robot or if there is a “thing” underneath a robot. However, there is a way to have a robot safely check to see if there is a wall in front of it or check to see if it can pick a “thing” and have it enforce the move or pick command only if it is safe. This can be achieved through the “if” statement.

When the “if” statement is called, the program asks a question and receives a true or false answer. A few examples of these questions are asking if the front of the robot is clear or if the robot can pick a thing.





When to Use and Avoid Method:

A good question to ask is when you should use this new command. An

example of when to use this command is when you need to pick a thing up but the position of the wall is generated randomly and you do not know if your front is clear.
(See Figure above to get a picture of the city that I am describing.)
An example of where you should NOT use this command is if you, as the programmer, know the layout of the city and the coordinates of the walls.
The statement is not a loop and can only ask the true or false question once. This means that it will only check if the question has a true or false answer once, then it will move forward to the other commands. Unlike the while loop the “if” command will not repeat itself.
Another area that you might want to use this command is if you wanted to place a thing while you moved but you do not want to put a thing where a thing is already there. You can do this by:

```
If(!canPickThing())
{
ian.putThing())
}
```

Note: This is a code snippet
By placing this command in your code, you guarantee that on your next move, you will not place a thing in an intersection where there is a thing already there.

Example of Usage

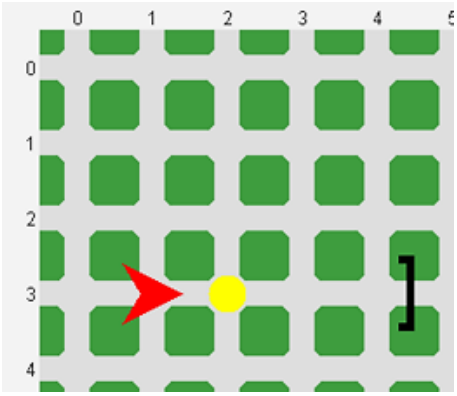
Here we will look at how to write the code into java. After creating the city, robot, walls, and things let us look at the bolded code below. The typed commands for having the robot check one space before to see if it is safe to move or not. Notice that in the line after the “if” statement there is a {. This tells java that if the if statement returns true, the commands that the robot should carry out is between the curly braces. After you are done typing the code for the “if" statement, be sure that you closed the curly brace with a closed curly brace}.
The same thing has been done with the “if you can pick up a thing” command.
We can also place the “if” statement in a public void. If you look at the first highlighted section in the sample code below, we define the safeMove in a public void. Make sure to place the curly braces after the public void and at the end of

the commands inside the public void. Another thing to notice is that instead of using the name of the robot, we use “this”. This is done because if you were to have multiple robots and you needed them to call this command, the “this” gets replaced with the name of the robot that called the safeMove command.

Line	Code
1	import becker.robots.*;
2	class Robot1 extends Robot
3	{
4	public Robot1(City c, int st, int ave, Direction dir, int num)
5	{
6	super(c, st, ave, dir, num);
7	}
8	public void safeMove( )
9	{
10	If (this.frontIsClear( ))
11	{
12	this.move( );
13	}
14	}}
15	public class Decision extends Object
16	{
17	public static void main(String[] args)
18	{
19	City toronto = new City();
20	Robot1 bob = new Robot1 (toronto, 3, 1, Direction.EAST, 0);
21	new Thing(toronto, 3, 2);
22	new Wall(toronto, 3, 4, Direction.EAST);
23	// new Wall(toronto, 2,3, Direction.WEST);
24	If (bob.frontIsClear( ))
25	{
26	bob.move( );
27	}
28	If (canPickThing( ))
29	{
30	bob.pickThing();
31	}
32	bob.move();
33	If (canPickThing( ))
34	{

```
35          bob.pickThing();
36      }
37  bob.turnLeft();
38  bob.move();
39    }
40  }
```

The code above produces the city on the right.



Example Trace

Now that we looked at how to write the code, let us look at how we trace it.

Line #	Program Statement	Bob St #	Bob Ave #	bob Dir	Bob Back pack	Thing	Wall	True/False
12	City toronto = **new** City();	-	-	-	-	-	-	-
13	RobotThatMakesDecisions bob = new RobotThatMakesDecisions (toronto, 3, 1, Direction.EAST, 0);	3	1	East	-	-	-	-
14	**new** Thing(toronto, 3, 2);	3	1	East	-	3,2	-	-
15	**new** Wall(toronto, 3, 4, Direction.EAST);	3	1	East	-	3,2	3,4	
16	**If (bob.frontIsClear( ))**	3	1	East	-	3,2	3,4	True
18	**bob.move( );**	3	2	East	-	3,2	3,4	-
20	**If (bob.canPickThing( ))**	3	2	East	-	3,2	3,4	True
22	**bob.pickThing();**	3	2	East	1	-	3,4	-
24	bob.move();	2	2	North	1	-	3,4	-
25	**If (canPickThing( ))**	2	2	North	1	-	3,4	False
29	bob.turnLeft();	2	2	West	1	-	3,4	-
30	bob.move();	1	2	West	1	-	3,4	-

Notice that when tracing an if statement, the command of moving or picking up a thing doesn't occur until the actual move or pick thing command is called. It is asking a question that will return back a true or false. Then depending on the answer, the next command will be decided. For example, the last "if can pick thing command" returned false and thus we did not apply the commands in that "if" statement.

Syntax Explanation

When writing this command in the code above, there are a few things that you have to watch out for. First thing to notice is that we have to write the name of the robot in the "if" statement, unless it is in a public void. If you look at the code above, it is stated that the name of the robot in both the "move" and the "pick thing" commands.

Second thing to watch out for are the use of the curly braces. In order for your code to be successful, you must be organized in the placement of your curly braces. Notice how they are lined up with each other so that you can easily see which curly brace pairs with the other. You can indent the curly braces by clicking on the left side of one and clicking the "tab" key once.

Also, the capitalization of words is essential for your code to work. The "if" must be lowercase and have a pair of parenthesis after it. And in that pair of parenthesis, you place question.

Finally, the fourth thing to look out for is the normal parentheses in the "if" command. After the "if", you place an open parenthesis then write the command. Afterwards you make open then closed parentheses like you would in a normal command. Finally you end the command with a closing parenthesis.

Help With the Logic

If you are still not sure how this command works, you should read the following paragraph.

Let's say that there is a city which randomly generates its walls, things, and robot locations. When writing code for this city to make the robot go from point A to

point B, you cannot simply place a move command because the robot might move into a wall and crash. A way of getting around that is by placing an “if” statement. This statement asks the robot a question; in this case it asks whether the front is clear to move. This question returns back a true or a false answer. If the answer is true, then the program executes the commands within the { } after the statement. If the answer returns false, then the program moves on to the command after the “if” statement and skips the commands within the { }.

Licensing

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License

Plagiarism

If you believe that some or this entire document infringes on your intellectual property (i.e., part or this entire 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)