Advanced Robot Commands

Edward Rusu

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

Table of contents:

Quick Summary

In Java, you can use multiple advanced robot commands to make your robot perform certain things. Advanced robot commands are any command beyond move, turnLeft, pickThing, and putThing. By using these commands together, your robot's potential greatly increases.

When to Use This / Avoid This

There are many things you can do with move, turnLeft, pickThing, and putThing, but as you continue writing java, you'll find that those four commands just don't cut it. This is where advanced robot commands come in. Advanced robot commands are a must for anyone writing java.

However, sometimes it's easier to use beginning level command to accomplish a certain task. As you continue writing java, you will develop a sense of when to use advanced robot commands.

Example of Usage

We will cover six useful advanced robot commands. They are:

canPickThing—We've all experienced the dreaded Crash! from telling our Robot to pick up something when there's nothing there. With canPickThing, you can make sure that there is something there for your Robot to pick up.

countThingsInBackpack—every Robot can carry Things. This command allows you to analyze the number of Things in the Robot's "backpack." Because there is no command "canPutThing," this command is often used with a true/false statement to get the same effect.

frontIsClear—Worse than our Robot crashing while trying to pick up something is crashing into a wall. frontIsClear allows you test whether or not there is a wall in your way. This is particularly useful in maps that randomly generate walls.

getAvenue—Humans often get lost. Robots are the same. Sometimes, a Robot just needs to know where it is. With this command, you can ask the Robot for its avenue and proceed from there.

getStreet—Equally important as getAvenue, getStreet allows to ask the Robot what street it's on. When combined with getAvenue, it becomes a powerful command.

getDirection—With so many turnLefts in the mix, getDirection is all important. If you ever get dizzy, just ask your Robot to getDirection, and you'll know which way you're facing.

*Please note: An answer will not be displayed on your screen unless you explicitly say so. If you write countThingsInBackpack, your robot will know how many things there are and be able to make decisions based on that, but a number will not appear telling you the number. The same is true for getDirection. You will not see a big flashing "WEST."

Follow along with the following code by opening advanced_robot_commands.java

Line	Code
1	import becker.robots.*;
2	
3	public class advanced_robot_commands extends Object
4	{
5	public static void main(String[] args)
6	{
7	City seattle = new City();
8	Robot jo = new Robot (seattle,0,0,Direction.EAST,0);
9	Robot mary = new Robot(seattle, 0, 3, Direction.NORTH,0);
10	new Thing(seattle,0,0);
11	new Wall(seattle,3,2,Direction.SOUTH);
12	
13	//that was all code to set up the city
14	
15	if(jo.canPickThing()) //safer way to pick things up
16	{
17	jo.pickThing();
18	}
19	
20	jo.move();
21	
22	if(jo.canPickThing()) //safer way to pick things up
23	{

```
24              jo.pickThing();
25          }
26
27          jo.move();
28
29          if( jo.countThingsInBackPack() > 0 )  //safer way to put things down
30          {
31              jo.putThing();
32          }
33
34          jo.turnLeft();
35          jo.turnLeft();
36          jo.turnLeft();
37          jo.move();
38
39          if(jo.countThingsInBackpack() > 0 )  //safer way to put things down
40          {
41              jo.putThing();
42          }
43
44          while( jo.frontIsClear() )  //safer way to move
45          {
46              jo.move();
47          }
48
49          jo.turnLeft();
50
51          while( jo.getAvenue() < mary.getAvenue()  //smarter way to move
52          {
53              jo.move();
54          }
55
56          if( jo.getStreet() == 3 )  //smarter way to move
57          {
58              if( jo.getAvenue() == 0 )  //smarter way to move
59              {
60                  jo.move();
61              }
62          }
63
64          if( jo.getDirection() == Direction.EAST  //smarter way to move
65          {
66              jo.move();
67          }
68      }
```

69	} //close program

Example Trace:

Notice that none of these commands are action commands. It doesn't help much to only ask the Robot if the front is clear because it won't actually make the Robot do anything (like move). Thus, these advanced robot commands are all used conditionally. If circumstance meet that condition (If countThingsInBackpack is greater than 0), then the Robot will do whatever is in the braces {}.

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 backpack	True or False	Mary St	Mary Ave	Thing	Wall
5	*program starts	-	-	-	-	-	-	-	-	-
7	City seattle = new City();	-	-	-	-	-	-	-	-	-
8	Robot jo = new Robot (seattle,0,0,Direction.EAST,0);	0	0	E	0	-	-	-	-	-
9	Robot mary = new Robot(seattle, 0, 3, Direction.NORTH,0);	0	0	E	0	-	0	3	-	-
10	new Thing(seattle,0,1);	0	0	E	0	-	0	3	(0,1)	-
11	new Wall(seattle,3,2,Direction.SOUTH);	0	0	E	0	-	0	3	(0,1)	(3,2,S)
15	if(jo.canPickThing())	0	0	E	0	False	0	3	(0,1)	(3,2,S)
20	jo.move();	0	1	E	0	-	0	3	(0,1)	(3,2,S)
22	if(jo.canPickThing())	0	1	E	0	True	0	3	(0,1)	(3,2,S)
24	jo.pickThing();	0	1	E	1	-	0	3	-	(3,2,S)
27	jo.move();	0	2	E	1	-	0	3	-	(3,2,S)
29	if(jo.countThingsInBackpack() > 0)	0	2	E	1	True	0	3	-	(3,2,S)
31	jo.putThing();	0	2	E	0	-	0	3	(0,2)	(3,2,S)
34	jo.turnLeft();	0	2	N	0	-	0	3	(0,2)	(3,2,S)
35	jo.turnLeft();	0	2	W	0	-	0	3	(0,2)	(3,2,S)
36	jo.turnLeft();	0	2	S	0	-	0	3	(0,2)	(3,2,S)
37	jo.move();	1	2	S	0	-	0	3	(0,2)	(3,2,S)
39	if(jo.countThingsInBackpack() >	1	2	S	0	False	0	3	(0,2)	(3,2,S)

	0)									
44	while(jo.frontIsClear())	1	2	S	0	True	0	3	(0,2)	(3,2,S)
46	jo.move();	2	2	S	0	-	0	3	(0,2)	(3,2,S)
44	while(jo.frontIsClear())	2	2	S	0	True	0	3	(0,2)	(3,2,S)
46	jo.move();	3	2	S	0	-	0	3	(0,2)	(3,2,S)
44	while(jo.frontIsClear())	3	2	S	0	False	0	3	(0,2)	(3,2,S)
49	jo.turnLeft();	3	2	E	0	-	0	3	(0,2)	(3,2,S)
51	while(jo.getAvenue() < mary.getAvenue())	3	2	E	0	True	0	3	(0,2)	(3,2,S)
53	jo.move();	3	3	E	0	-	0	3	(0,2)	(3,2,S)
51	while(jo.getAvenue() < mary.getAvenue())	3	3	E	0	False	0	3	(0,2)	(3,2,S)
56	if(jo.getStreet() == 0)	3	3	E	0	True	0	3	(0,2)	(3,2,S)
58	if(jo.getAvenue() == 0)	3	3	E	0	False	0	3	(0,2)	(3,2,S)
64	if(jo.getDirection() == Direction.EAST)	3	3	E	0	True	0	3	(0,2)	(3,2,S)
66	jo.move();	3	4	E	0	-	0	3	(0,2)	(3,2,S)

Don't get caught up in all the true/false statements. Focus on what the Robot is doing. When you ask canPickThing, the answer is true because jo is standing over the thing. Because the answer is true, jo picks the thing up. When you ask countThingsInBackpack, we see the answer is 1. But it's not worth anything to know that there is 1 thing in his backpack. Because we put it in a conditional statement, however, he will proceed to putting something down. This is the substitute for the missing command canPutThing. In some cases, the answer is false. When we ask getStreet == 3 and getAvenue == 0, we see that jo is not on 3,0 but on 3,3. Note, *the program still returned his street and avenue,* but because it only matched one of our conditions, not both of them, nothing happened. Take a moment to make sense of the other ones. Remember, don't get caught up in the *if* and *while* statements. Focus on what each advanced robot command does.

Syntax Explanation:

Note on syntax of command: Because these commands are defined in the becker.robot file, you must write them exactly as they appear above, i.e. make sure you capitalize the right letters.

Let's focus specifically on one of our advanced robot commands to work out what all the syntax means

We can enhance the code:

jo.putThing();

to be a safer way of putting things down:

```
if( jo.countThingsInBackpack() > 0 )
{
    jo.putThing();
}
```

Notice the code that is highlighted. This is the stuff we add to make our robot crash-proof when he puts something down. On the first line, we write an if statement. An if statement sets a condition. That condition is specified in the parenthesis. The word *if* must be lowercase. You must start the condition with an open parenthesis. (For more information, see the section on if statements). Write the robot's name, in this case *jo*. Then write a period and the advanced robot command. In this example, we use countThingsInBackpack(). Please note that the command must be spelled and capitalized exactly as written here. countThingsInBackpack() looks inside the robots backpack and returns a number. You do not see that number (unless you specify to), but your computer knows what it is. By using > 0, we specify that the number of things in the robot's backpack must be greater than zero. We could have said == 0, < 0, >24, and really any combination of inequalities and numbers, but for the sake of this command, we use > 0. The computer compares the number of things in the robot's backpack to the number we specify. If the is backpack carries more items than our number, than the robot proceeds to do what is defined in the braces. (Note: you must have open and closed brace around any command you want to test against your if statement.) However, if the number in his backpack is less than zero, the computer skips anything in the braces, keeping the robot from trying to put anything down when he doesn't have anything to put.

Help With The Logic

Note that working with advanced robot commands makes everything you do with the Robot safer and smarter. With these advanced robot commands, the robot first checks to make sure it can do what you're asking without crashing.

Try it Yourself

Open try_it_yourself.java and see if you can enhance your robot using advanced robot commands.

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)