For Loops:

Will Tanna

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

Table of contents:

Summary:

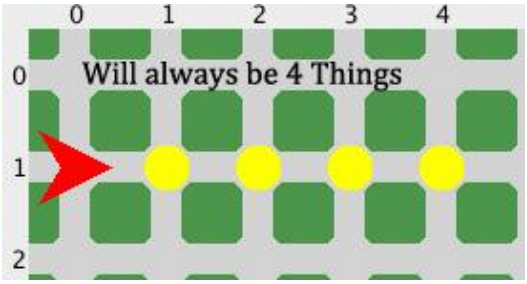
There are many variations of loops that you can do in Java. One of those variations is the for loop. A for loop is used when you know how many times you want to use a certain command. Unlike the while command where it keeps going unless a command comes out true or false. ~~The for loop command bypasses the asking question portion of the loop and goes right to the commands given~~. It uses a counter command to count how many times the main command has been used.

When to Use and Avoid this Method:

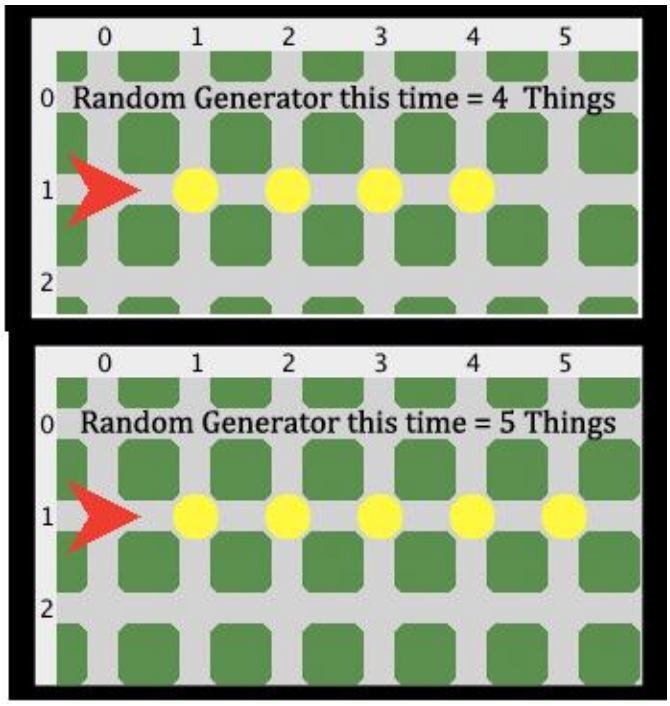
There are many different variations of loops in Java. How do you know which one to use? You should ask your self what kind of program you are working with or creating. If you know the exact integer a certain command is, then the for loop will work out best for you. The great thing about the for loop is that instead of relying on questions being asked for the robot you are telling it a direct command that it needs to follow a certain amount of times

However, there are some scenarios where using the for loop is bad. When you have walls or Things that are randomly generating in different locations. You won't know the exact integer value because it is always changing. Another reason why not to use the for loop would be for if the robot needs to be asked a question. If a Robot needs to move down a hallway then turn around, the while loop can make a command for just that situation. You can program the robot to walk down the hallway then ask itself if it can keep moving forward or turn around so it does not crash. If you were to do a for loop command and the integer was too high the Robot could hit a wall down the hallway and break.

When to Use:



When To Avoid:



Example Of Usage:

Now we are going to learn how to write this loop in Java. Lets say for example you have a robot that just got pushed and when getting pushed all his Things came out of his backpack. Now he needs to move forward and pick those Things up to put them back in his backpack.

One way to type this is (in this program the robot is named Brian):

```
brian.move();
brian.pickThing();
brian.move();
brian.pickThing();
brian.move();
```

```
brian.pickThing();
brian.move();
brian.pickThing();
```

Instead of typing all of that, you can save yourself a lot of time by using the for loop command. The for loop command can create a loop where it will repeat the commands "brian.move();" and "brian.pickThing();" a however many times you want or need. This is how you create the for loop command.

You first start for the for loop command be typing "for ("

Second, you need to create an integer for the loop. You do this by typing "int" right after the (we just put in. After "int" we want to put a space and type "counter = 0;". What we have so far is: "for (int counter = 0;" What this means is we have an integer that is starting at 0. The "counter" can be named anything. But for this program we will leave it as counter. There needs to be a ";" after 0 that way Java knows that is when that command ends. A

After the semicolon we must add the integer the counter will count up too, or equal too. What this will look like is: counter < 4;. This shows that the counter will be true if the counter is less then 4. When Brian moves 4 times he is no longer less then the counter, which means he will stop.

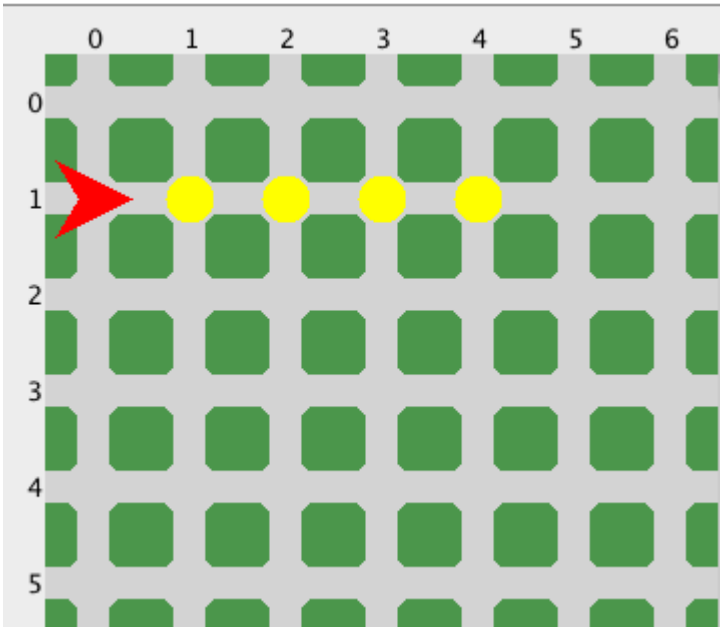
Once the counter integer is done, we need a command that will actually count the counter. We do this by typing: "counter = counter +1)". What this command means is that for every time the statements are done it will count 1 towards the counter. You can put any number where the 1 is, but if you are trying to count just one statement at a time, there really is no need to go above 1. You must end the command with a parenthesis so Java knows that's when the integer counter is done, and you can go onto the statements. Just to clarify, the statements are the lines of code that Brian will take to move and pick the Things up.

Once you get all of the code lined up and set. This is an example of what it would look like. For the sake of the example we will bold the counter portion of the code.

Line	**Code**
1	import becker.robots.*;
2	public class For_Loop_Example extends Object
3	{
4	public static void main(String[] args
5	{
6	City tokyo = new City();
7	NewRobotType brian = new NewRobotType(tokyo, 1, 0, Direction.EAST, 0);
8	new Thing(tokyo, 1, 1);
9	new Thing(tokyo, 1, 2);
10	new Thing(tokyo, 1, 3);
11	new Thing(tokyo, 1, 4);

12	**for (int counter = 0; counter < 4; counter = counter +1)**
13	{
14	**brian.move();**
15	**brian.pickThing();**
16	}
17	}
18	

This is what we will get with the code we just typed:



Example Trace:

Now that the code is all typed up and ready to run. We will now trace the program to show what the counter command is doing to Brian and the four Things. What this command will do is move Brian four times while picking up each thing. Once he has done that, he will stop.

Line #	Program Statement	Brian St #	Brian Ave #	Brian Direc	Brian Back pack	Thing location 1	Thing location 2	Thing location 3	Thing location 4	Wall	Counter	True/False
4	public static void main(String[] args)	-	-	-	-	-	-	-	-	-	-	-
6	City tokyo = new	-	-	-	-	-	-	-	-	-	-	-

	City();											
7	NewRobotType brian = new NewRobotType(tokyo, 1, 0, Direction.EAST, 0);	1	0	E	0	-	-	-	-	-	-	-
8	new Thing(tokyo, 1, 1);	1	0	E	0	(1,1)	-	-	-	-	-	-
9	new Thing(tokyo, 1, 2);	1	0	E	0	(1,1)	(1,2)	-	-	-	-	-
10	new Thing(tokyo, 1, 3);	1	0	E	0	(1,1)	(1,2)	(1,3)	-	-	-	-
11	new Thing(tokyo, 1, 4);	1	0	E	0	(1,1)	(1,2)	(1,3)	(1,4)	-	-	-
12	for (int counter =0; counter < 4; counter = counter +1)	1	0	E	0	(1,1)	(1,2)	(1,3)	(1,4)	-	0	True
14	brian.move();	1	1	E	0	(1,1)	(1,2)	(1,3)	(1,4)	-	0	-
15	brian.pickThing();	1	1	E	1	-	(1,2)	(1,3)	(1,4)	-	0	-
12	`counter = counter +1`	1	1	E	1	-	(1,2)	(1,3)	(1,4)	-	1	-
12	for (int counter =0; counter < 4; counter = counter +1)	1	1	E	1	-	(1,2)	(1,3)	(1,4)	-	1	True
14	brian.move();	1	2	E	1	-	(1,2)	(1,3)	(1,4)	-	1	-
15	brian.pickThing();	1	2	E	2	-	-	(1,3)	(1,4)	-	1	-
12	`counter = counter +1`	1	2	E	2	-	-	(1,3)	(1,4)	-	2	-
12	for (int counter =0; counter < 4; counter = counter +1)	1	2	E	2	-	-	(1,3)	(1,4)	-	2	True
14	brian.move();	1	3	E	2	-	-	(1,3)	(1,4)	-	2	-
15	brian.pickThing();	1	3	E	3	-	-	-	(1,4)	-	2	-
15	`counter = counter +1`	1	3	E	3	-	-	-	(1,4)	-	3	-
12	for (int counter =0; counter < 4; counter = counter	1	3	E	3	-	-	-	(1,4)	-	3	True

	+1)											
14	brian.move();	1	4	E	3	-	-	-	(1,4)	-	3	-
15	brian.pickThing();	1	4	E	4	-	-	-	-	-	3	-
12	`counter = counter +1`	1	4	E	4	-	-	-	-	-	4	-
12	for (int counter =0; counter < 4; counter = counter +1)	1	4	E	4	-	-	-	-	-	4	False

As you can see, line 4-11 is the code to set up the city and the main details of the program. After that the program runs the commands of the robot. Line 12 is the start of the counter. It is set to repeat the command 4 times. The statement for line 12 will continue to be true until the counter hits 4. When it hits 4 it is no longer less then 4. Therefore the statement turns to False.

Syntax Explanation:

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

Line #	Program Source Code
1.	`import becker.robots.*;`
2.	`public class For_Loop_Example extends Object`
3.	`  {`
4.	`    public static void main(String[] args)`
5.	`    {`
6.	`      City tokyo = new City();`
7.	`      Robot brian = new Robot(tokyo, 1, 0, Direction.EAST, 0);`
8.	`      new Thing(tokyo, 1, 1);`
9.	`      new Thing(tokyo, 1, 2);`
10.	`      new Thing(tokyo, 1, 3);`
11.	`      new Thing(tokyo, 1, 4);`
12.	
13.	`      brian.move();`
14.	`      brian.pickThing();`
15.	`      brian.move();`
16.	`      brian.pickThing();`
17.	`      brian.move();`
18.	`      brian.pickThing();`
19.	`      brian.move();`
20.	`      brian.pickThing();`
21.	`    }`
22.	`  }`

23.	

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.	`public class For_Loop_Example extends Object`
3.	`{`
4.	`public static void main(String[] args)`
5.	`{`
6.	`City tokyo = new City();`
7.	`Robot brian = new Robot(tokyo, 1, 0, Direction.EAST, 0);`
8.	`new Thing(tokyo, 1, 1);`
9.	`new Thing(tokyo, 1, 1);`
10.	`new Thing(tokyo, 1, 1);`
11.	`new Thing(tokyo, 1, 1);`
12.	`for (int counter =0; counter < 4; counter = counter +1)`
13.	`{`
14.	`brian.move();`
15.	`brian.pickThing();`
16.	`}`
17.	`}`
18.	`}`
19.	

The code we typed in yellow is the new method we learned. We went from a long line of code to just a few lines. By using a counter command you can immensely shorten your program, but there are some things you need to watch out for.

Here is the code we wrote:

12	**for (int counter = 0; counter < 4; counter = counter +1)**
13	{
14	**brian.move();**
15	**brian.pickThing();**
16	}

If you notice the “for” is in purple; same with the “int”. The reason why these are purple is because Java recognizes these commands. It tells Java that these words have

specific meaning and perform a certain action. The “for” is for the loop we are trying to do. The “int” is used to show that we do in fact have an integer in our statement.

If you notice, the two purple words are lower case. This is very important because they must always be lower case. Along with the two purple words all the starting letter of every word needs to be lowercase. For example we used “counter” as what our counter would be called. It is lower case as you can see. If we wanted to name the counter, Brian moves and picks things up. We would write it as “brianMovesAndPicksThingsUp”.

When you first look at the code it can be a little confusing on which “counter” does what in the program. The very first counter with the “=0” is the counter to designate what number the counter is starting. In this case we are starting at 0. The next counter with the “< 4” means that we are counting until we get to 4. The for loop will be true every time the number is less then 4. Once it hits 4 it comes back false and the loop still stop. The last 2 counters, “counter = counter +1” is the command for actually counting the for loop. Once on loop has been done, this bit of code will count 1 number towards the counter. Eventually adding up to the number that we designated in the second counter command.

After the main for loop counter code has been put into place, we must put the statements in. It is very important that you put curly braces in, to state that the statements in between them are the ones the for loop will do. In between the curly braces is where we put our statements that we want to repeat. In this case it is “brian.move();” and “brian.pickThing();”. Once we type these commands we must end the for loop with one more curly brace like on line 16. Once that last curly brace is put into place, we are done with the loop.

Plagiarism: