For Loops

Cory Tsai

Taylor Philpott

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 create loops to repeat a given set of commands. A for loop is used when you know how many times you want that set of commands to be repeated. It can be used to replace a while loop, when you want the loop to be repeated a specific number of times instead of having it repeat or not depending on whether or not certain queries (questions) return true or false. For loops look cleaner than while loops, and it is often faster and easier to tell what they do from just a quick glance.

When To Use This / Avoid This:

There are many situations where you will want to use this technique. Whenever you know exactly how many times you want to repeat a set of actions, a for loop is a good choice because it is able to count instead of relying on asking a question about the current state of the program.

You will want to avoid this technique when you're unsure of how many times the loop needs to be repeated. For example, you shouldn't use a for loop when a robot is starting at a random distance away from a wall, and it needs to walk to the wall and then stop before it crashes. In that situation, a while loop would be better because it would be able to test something after each loop is called, so it will choose when to stop after some variable has been satisfied.

Example Of Usage:

Let's say you want the robot to move one intersection forward, drop one thing, and then repeat those actions two more times. Rather than typing:

```
karel.move();
karel.putThing();
karel.move();
karel.putThing();
karel.move();
karel.putThing();
```

You could instead use a for loop to have the program repeat the commands “karel.move(); karel.putThing();” as many times as desired. The following program demonstrates one way to do that (with the new/important stuff **bold-faced and in a larger font**)

There are five steps:

First, create the for loop by typing “for (”.

Second, create a new integer, and assign it the value zero by typing “int counter = 0;”. The word “counter” can be replaced by any combination of letters, without spaces. The standard format for creating a name that consists of more than one word is to capitalize the first letter of each new word in the name, starting after the first word. For example, if you wanted to name the counter “number of times robot moves and puts thing”, you would name it “numberOfTimesRobotMovesAndPutsThing”.

Third, assign the counter the number of times you want it to repeat the loop. Since the counter is starting at zero, and it will add one to the value of the counter each time the loop is completed, type the number of times you wish the loop to be repeated. For example, if you want the loop to be repeated 3 times, you would type “`numberOfMovesAndDrops < 3;`”.

Fourth, you must tell the program how to adjust the counter after each time the loop is completed. If you want the program to add one to the counter after each run through the loop, type “counter ++)”, where the word “counter” is replaced by the name you assigned to your integer in step two.

Fifth, in the lines directly below the loop counter setup, type the commands you want to have called each time the loop is called. You must type these in between “curly braces” {}, so the program recognizes that these are the commands it must repeat each time the loop is called.

Line	Code
1.	import becker.robots.*;
2.	
3.	public class For_Loops_Demo extends Object
4.	{
5.	public static void main(String[] args)
6.	{
7.	City wallville = new City(6, 6);
8.	Robot rob = new Robot (wallville, 0, 0, Direction.EAST, 10);
9.	
10.	**for (int numberOfMovesAndDrops = 0; numberOfMovesAndDrops < 3; numberOfMovesAndDrops ++)**
11.	**{**
12.	**rob.move();**
13.	**rob.putThing();**
14.	**}**
15.	}
16.	}

Example Trace:

In a nutshell, this technique allows you to create a loop that will repeat a specified number of times. When the program runs, it will repeat this set of commands and add one to the counter each time it is repeated. It will then stop once the counter reaches the "goal" number of actions.

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	rob St #	rob Ave #	rob Dir	rob Back pack	Thing locations	Wall	counter	true/ false
5	public static void main(String[] args)	-	-	-	-	-	-	-	
7	City wallville = new City(6, 6);	-	-	-	-	-	-	-	
8	Robot rob = new Robot (wallville, 0, 0, Direction.EAST, 10);	0	0	E	10	-	-	-	
10	for (int numberOfMovesAndDrops = 0; numberOfMovesAndDrops < 3; numberOfMovesAndDrops ++)	0	0	E	10	-	-	0	true
12	rob.move();	0	1	E	10	-	-	0	
13	rob.putThing();	0	1	E	9	(1,0)	-	0	
10	for (int numberOfMovesAndDrops = 0; numberOfMovesAndDrops < 3; numberOfMovesAndDrops ++)	0	1	E	9	(1,0)	-	1	true
12	rob.move();	0	2	E	9	-	-	1	
13	rob.putThing();	0	2	E	8	(1,0) (2,0)	-	1	
10	for (int numberOfMovesAndDrops = 0; numberOfMovesAndDrops < 3; numberOfMovesAndDrops ++)	0	2	E	8	(1,0) (2,0)	-	2	true
12	rob.move();	0	3	E	8	(1,0) (2,0)	-	2	
13	rob.putThing();	0	3	E	7	(1,0) (2,0) (3,0)	-	2	
10	for (int numberOfMovesAndDrops = 0; numberOfMovesAndDrops < 3; numberOfMovesAndDrops ++)	0	3	E	7	(1,0) (2,0) (3,0)	-	3	false

You'll notice that the trace starts at beginning of the **main** function, on line 5, like normal. On line 10, where the for loop is created, the counter is set to zero and the “goal” number is set to three. This causes the program to repeat lines 12 and 13 three more times, each time adding one to the counter and then eventually stopping because the counter is no longer less than three.

Syntax Explanation:

Note on syntax of integer: the name of the integer can be nearly anything, with a few formatting restrictions. The name should contain no spaces, and capital letters should be used to designate different words, with the first word being lower case or an underscore between the words.

Let's start with the program as it's written here. Notice how there are only two commands that are repeated three times in a row.

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

Below, you can see the finished program with the differences **in bold** as well as 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 For_Loops_Demo extends Object`

```
4.   {
5.       public static void main(String[] args)
6.       {
7.           City wallville = new City(6, 6);
8.           Robot rob = new Robot (wallville, 0, 0,
     Direction.EAST, 10);
9.
10.              for (int numberOfMovesAndDrops = 0;
     numberOfMovesAndDrops < 3;
     numberOfMovesAndDrops ++)
11.              {
12.               rob.move();
13.               rob.putThing();
14.              }
15.      }
16.  }
```

The three sets of repeated commands have now been replaced. Notice how the word "for" is in a purple text. This is because Java recognizes it, but it will only do so if spelled correctly and left UNcapitalized.

Inside the parenthesis, the word "int" has also been changed to a purple text, because java recognizes that it is creating an integer. The "= 0" assigns the integer a value of zero. "numberOfMovesAndDrops" is assigned as he integer name. This particular for loop is designed to call the commands it contains while the counter "numberOfMovesAndDrops" is smaller than the number three. It is also set up to add increment (increase) the value of the counter by one, which is shown as "numberOfMovesAndDrops ++"

Directly following line 10 is line 11, which contains nothing but an opening curly brace symbol "{". This tells java that what follows will be the lines that need to be called each time the loop is called. This loop is designed to call the commands "rob.move(); rob.putThing();" each time the loop is repeated. After these commands are listed, a closed curly brace "}" is typed on line 14, to tell java that this is the end of the series of commands that must be called each time the loop is called.

Licensing