The Basics of While loops In Java

By: Devon Harting & Michael Hesler and

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

Table of contents:

Quick Summary:

In Java Programming, you can use while loops to continually execute an expression if it evaluates as true. If the expression is not true the while loop will end and the program will proceed to the next line in the code.

When To Use This / Avoid This:

There are several situations where you would want to use a technique. For instance, in a counter is an empty space of memory in which a number can be inserted. Or it can be used in a series of commands that are repetitive.

There are also situations where you would want to avoid this technique. An example would be when you only need a command to execute one time.

Example Of Usage:

Let's say you want a robot to move five (5) times.

There are three steps:

Step one: create a while loop with arguments inside of the closed parenthesis.
Ex: `while ( counter < 5 )`

Step two: In the body of the while loop add commands with which the while loop will execute if in the event that the loop is correct. Anything can be put in the body of the while loop just as long as said commands are valid.

Step three: Make sure to open and close the loop with an open curly brace under the while loop argument a closing curly brace under the last command inside of the while loop.

The code for this example is as follows:

Line	Code
1.	`import becker.robots.*;`
3.	`class MultipleMovesRobot extends Robot`
4.	`{`
5.	`public MultipleMovesRobot (City c, int st, int ave, Direction dir, int num)`
6.	`{`
7.	`super(c, st, ave, dir, num);`
8.	`}`
9.	`//This line names the while loop that moves the robot five times`
10.	`public void moveFive()`
11.	`{`
12.	`int counter = 0;`
13.	`while ( counter < 5 ) //This line begins its definition`

```java
{
this.move();
counter = counter + 1;
System.out.println( "I have moved:" + counter + "times!"
);
}
}
}
public class moving_robot extends Object
{
public static void main(String[] args)
{
City The_Mushroom_Kingdom = new City();
MultipleMovesRobot bullet_bill = new MultipleMovesRobot
(The_Mushroom_Kingdom, 5, 0, Direction.EAST, 0);

bullet_bill.moveFive(); //This line calls the while loop.
bullet_bill.turnLeft();
bullet_bill.moveFive(); //This line calls the while loop.
}
}
```

Example Trace:

This shows that the robot has called a while command named moveFive and has moved five times or until its counter is less than or equal to 5. It then printed out the counter each time it moved added 1 to the counter. Afterward it then turns left. Finally it then moves five times again by calling the single command moveFive and then the program stops.

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	bullet_bill Street	bullet_bill Avenue	bullet_bill Direction	bullet_bill Backpack	TRUE FALSE	Counter	Print Statement
24	`public static void main(String[] args)`	-	-	-	-	-	-	-
26	`City The_Mushroom_Kingdom = new City();`	-	-	-	-	-	-	-
27	`MultipleMovesRobot bullet_bill = new MultipleMovesRobot (The_Mushroom_Kingdom, 5, 0, Direction.EAST, 0);`	5	0	EAST	0	-	-	-
29	`bullet_bill.moveFive() ;`	5	0	EAST	0	-	-	-
10	`public void moveFive()`	5	0	EAST	0	-	-	-
12	`int counter = 0;`	5	0	EAST	0	-	0	-
13	`while ( counter < 5 )`	5	0	EAST	0	TRUE	0	-
15	`this.move();`	5	1	EAST	0	-	0	-
10	`counter = counter + 1;`	5	1	EAST	0	-	1	-
12	`System.out.println( "I have moved:" + counter + "times!" );`	5	1	EAST	0	-	1	I have moved: 1 times!
13	`while ( counter < 5 )`	5	1	EAST	0	TRUE	1	-
15	`this.move();`	5	2	EAST	0	-	1	-
10	`counter = counter + 1;`	5	2	EAST	0	-	2	-
12	`System.out.println( "I have moved:" + counter`	5	2	EAST	0	-	2	I have moved: 2 times!

	+ "times!");							
13	while (counter < 5)	5	2	EAST	0	TRUE	2	-
15	this.move();	5	3	EAST	0	-	2	-
10	counter = counter + 1;	5	3	EAST	0	-	3	-
12	System.out.println("I have moved:" + counter + "times!");	5	3	EAST	0	-	3	I have moved: 3 times!
13	while (counter < 5)	5	3	EAST	0	TRUE	3	-
15	this.move();	5	4	EAST	0	-	3	-
10	counter = counter + 1;	5	4	EAST	0	-	4	-
12	System.out.println("I have moved:" + counter + "times!");	5	4	EAST	0	-	4	I have moved: 4 times!
13	while (counter < 5)	5	4	EAST	0	TRUE	4	-
15	this.move();	5	5	EAST	0	-	4	-
10	counter = counter + 1;	5	5	EAST	0	-	5	-
12	System.out.println("I have moved:" + counter + "times!");	5	5	EAST	0	-	5	I have moved: 5 times!
13	while (counter < 5)	5	5	EAST	0	FALSE	5	-
30	bullet_bill.turnLeft();	5	5	NORTH	0	-	-	-
10	public void moveFive()	5	5	NORTH	0	-	-	-
12	int counter = 0;	5	5	NORTH	0	-	0	-
13	while (counter < 5)	5	5	NORTH	0	TRUE	0	-
15	this.move();	4	5	NORTH	0	-	0	-
10	counter = counter + 1;	4	5	NORTH	0	-	1	-
12	System.out.println("I have moved:" + counter + "times!");	4	5	NORTH	0	-	1	I have moved: 1 times!
13	while (counter < 5)	4	5	NORTH	0	TRUE	1	-
15	this.move();	3	5	NORTH	0	-	1	-

10	`counter = counter + 1;`	3	5	NORTH	0	-	2	-
12	`System.out.println( "I have moved:" + counter + "times!" );`	3	5	NORTH	0	-	2	I have moved: 2 times!
13	`while ( counter < 5 )`	3	5	NORTH	0	TRUE	2	-
15	`this.move();`	2	5	NORTH	0	-	2	-
10	`counter = counter + 1;`	2	5	NORTH	0	-	3	-
12	`System.out.println( "I have moved:" + counter + "times!" );`	2	5	NORTH	0	-	3	I have moved: 3 times!
13	`while ( counter < 5 )`	2	5	NORTH	0	TRUE	3	-
15	`this.move();`	1	5	NORTH	0	-	3	-
10	`counter = counter + 1;`	1	5	NORTH	0	-	4	-
12	`System.out.println( "I have moved:" + counter + "times!" );`	1	5	NORTH	0	-	4	I have moved: 4 times!
13	`while ( counter < 5 )`	1	5	NORTH	0	TRUE	4	-
15	`this.move();`	0	5	NORTH	0	-	4	-
10	`counter = counter + 1;`	0	5	NORTH	0	-	5	-
12	`System.out.println( "I have moved:" + counter + "times!" );`	0	5	NORTH	0	-	5	I have moved: 5 times!
13	`while ( counter < 5 )`	0	5	NORTH	0	FALSE	5	-

The trace starts at beginning of the **main** function, on line 24, like normal. It proceeds until line 29 where the robot bullet_bill calls our while loop named moveFive. This causes the program to jump up to line 13 where, as long as the counter is less than 5, it will move once and add 1 to the counter, and print out “I have moved X times”. If the counter exceeds 5, line 13 will evaluate as false and the while loop will end. The program will then go to the next line which is line 30 and execute a left turn. Line 31 calls our while loop again so the program goes back to line 13 and the while loop moveFive will execute again. It is worth noting that the counter goes back to 0 each time the while loop ends..

Syntax Explanation:

Note on syntax of command:

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.*;`
3.	`class MultipleMovesRobot extends Robot`
4.	`{`
5.	`public MultipleMovesRobot (City c, int st, int ave, Direction dir, int num)`
6.	`{`
7.	`super(c, st, ave, dir, num);`
8.	`}`
9.	`//moves the robot five times`
10.	`public void moveFive()`
11.	`{`
12.	`int counter = 0;`
13.	`while ( counter < 5 )`
14.	`{`
15.	`this.move();`
16.	`counter = counter + 1;`
17.	`System.out.println( "I have moved:" + counter + "times!" );`
19.	`}`
20.	`}`
21.	`}`
22.	`public class moving_robot extends Object`
23.	`{`
24.	`public static void main(String[] args)`
25.	`{`
26.	`City The_Mushroom_Kingdom = new City();`

```
27. MultipleMovesRobot bullet_bill = new
    MultipleMovesRobot (The_Mushroom_Kingdom, 5, 0,
    Direction.EAST, 0);

29. bullet_bill.moveFive();
30. bullet_bill.turnLeft();
31. bullet_bill.moveFive();
32. }
33. }
```

The while loop is the preferred method to use whenever you have series of repeating commands. Short, succinct, concise programs are preferred because they are easier to read and debug. To begin, name the command. In this case public **void** moveFive( )

It is essential that you mark the beginning of the command definition with an open curly brace found on line 11 (**{**), and we mark the definition's end with the matching close curly brace (**}**) on line 21.

It is easy to forget the closing brace so you are highly encouraged to put in both, immediately, so that you don't forget either one.

For now, all the commands that you create will start with "**public void**". "**public**" means that any part of the program is allowed to make use of the command and "**void**" means that this command will do work, but won't produce any particular calculation.

Then (on lines 12 through 17), we put the various commands that we wish to do INSIDE the while command (i.e., between the opening & closing curly braces).

Finally (on lines 29 and 31) we invoke the command bullet_bill.moveFive();

Help With The Logic:

It is a good thing to use while loops when you have a situation where you need to repeat a series of commands more than once. In robotic code for instance if you have a robot and you want him to move as long as he can pick something up. You would then create a while loop that says "as long as there is something to pick up then move".

While loops are also useful in many situations where you need to execute a series of commands more than once.

```
    public static void main(String[] args)
    {
        // There's more in the file, but it's left out in this
example..

        While (this.canPickThing());
       {
         bullet_bill.move();
       }
    }
}
```

Important Details:

Point #1;

While loops must be evaluated to a true statement. If the statement is not true the while loop ends.

Point #2;

If you want the while loop to evaluate an untrue statement you can use the ! (exclamation point) as a negative predicate thereby making a false statement true.

Point #3

The statements must be enclosed in opening and closing curly brackets.

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

Quick Name Of Mistake: Intent error

Detailed Example Of Error: Make sure that the while loop does exactly what you want it to do. For example the code down below will move your robot four times because it states that while the number in the counter is less than 5 it will move but if it is at 5 not to move.

Detailed Example Of Fix: In order to remedy this situation make sure to add an equal sign to your argument in order to move it the full five spaces. EX: while(counter<=5).

```
public void moveFive()
{
int counter = 0;
while ( counter < 5 ) //This line begins its definition
{
this.move();
counter = counter + 1;
System.out.println( "I have moved:" + counter + "times!" );
}
```

Detailed Example Of Error: **int counter = 6;**

counter is greater than 5

Detailed Example Of Fix:

int counter = 0; **set the counter to 0 instead of 6**

Quick Name Of Mistake: Misplacement of {'s
Detailed Example Of Error: If the {'s are not done properly, the command will not compile or the command will not be able to be found.
Detailed Example Of Fix: (unless it's obvious): Highlight the {'s so that the pair can be identified.

Licensing

Plagiarism