Title: Counting While Loops

Erik Boling, Panitz, Bit115, Winter Quarter 2011

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

Table of contents:

Quick Summary:

We have already seen the use of a while loop and the use of variables in java. Now we can combine the two to keep track of how many times the while loop is being run. This can help us create a command that will execute a given number of tasks, or keep track of how many events are happening.

When To Use This / Avoid This:

You want to use this technique when you find yourself wanting to count how many times something is happening, like the number of times you have moved or how many things you have picked up. You can even use this to limit the number of times you execute an event, for example stopping after you've picked up a given number of things.

You would NOT want to use a while counting loop if you're only using it for a task that isn't very repetitive or also if you don't necessarily need to keep track of any variables.

Example Of Usage:

Let's say you want your robot to move forward 7 spaces, but you don't want to type out the move command 7 separate times, you can use a while counting loop to help simplify your program

There are 4 steps:
First: Create the variable **counter**, which will keep track of the number of times the robot moves. This is done on **line 10**.

Second: Create the **while loop**, and put a **Boolean Expression** (a statement with a greater than, less than, or equal to sign) inside the while argument to tell the robot how many times it should move. We can think of the Boolean Expressions as a way of limiting how many times the loop will run.
In this case the **Boolean Expression** argument on **line 11** is, while the counter is less than 7, do what's in the while loop. We want the Robot to move 7 times, so we set the **counter** equal to 0, and say when it reaches 7, stop the loop.

Third: Inside the while loop, now you tell the Robot what you want it to do, which is move. This can be seen on **line 13**.

Fourth: So far we have created the counter to keep track of the movements, created the while loop to say how many movements we want, and had the Robot move. Now as the final part of the loop, we tell the counter to record that we've just moved one space. This can be seen on **line 14**. This is the bit of code that actually counts, if it wasn't for this, the counter would keep returning that it was 0, and the Robot would move forever.

Line	Code
1.	`import becker.robots.*;`
2.	
3.	`public class Counting_While_Loop extends Object`
4.	`{`
5.	`    public static void main(String[] args`
6.	`    {`
7.	`     City toronto = new City();`
8.	`     Robot Jo = new Robot(toronto, 3, 0, Direction.EAST, 0`
9.	
10.	`       int counter = 0;`
11.	`       while ( counter < 7 )`
12.	`       {`
13.	`          Jo.move();`
14.	`          counter = counter + 1;`
15.	`       }`
16.	`    }`
17.	`}`
18.	

Example Trace:

In a nutshell, this technique allows you to create a while loop to perform a repetitive task, and keep track of how many times the program is completing the task. This while loop uses a

variable to keep track of how many times the program has gone through the loop. Every time the while loop starts, it checks to see if the **counter** is less than 7, and if that is true then that means the Robot has moved less than 7 times, so it should continue with the loop!

Line #	Program Statement	Jo St #	Jo Ave #	Jo Direction	Jo Backpack contents	Counter	True/False
5	`public static void main(String[] args)`	-	-	-	-	-	-
7	`City toronto = new City();`	-	-	-	-	-	-
8	`Robot Jo = new Robot(toronto, 3, 0, Direction.EAST, 0);`	3	0	East	0	-	-
10	`int counter = 0;`	3	0	East	0	0	-
11	`while ( counter < 7 )`	3	0	East	0	0	True
13	`Jo.move();`	3	1	East	0	0	-
14	`counter = counter + 1;`	3	1	East	0	1	-
11	`while ( counter < 7 )`	3	1	East	0	1	True
13	`Jo.move();`	3	2	East	0	1	-
14	`counter = counter + 1;`	3	2	East	0	2	-
11	`while ( counter < 7 )`	3	2	East	0	2	True
13	`Jo.move();`	3	3	East	0	2	-
14	`counter = counter + 1;`	3	3	East	0	3	-
11	`while ( counter < 7 )`	3	3	East	0	3	True
13	`Jo.move();`	3	4	East	0	3	-
14	`counter = counter + 1;`	3	4	East	0	4	-
11	`while ( counter < 7 )`	3	4	East	0	4	True
13	`Jo.move();`	3	5	East	0	4	-
14	`counter = counter + 1;`	3	5	East	0	5	-
11	`while ( counter < 7 )`	3	5	East	0	5	True
13	`Jo.move();`	3	6	East	0	5	-
14	`counter = counter + 1;`	3	6	East	0	6	-
11	`while ( counter < 7 )`	3	6	East	0	6	True
13	`Jo.move();`	3	7	East	0	6	-
14	`counter = counter + 1;`	3	7	East	0	7	-
11	`while ( counter < 7 )`	3	7	East	0	7	False
	END OF PROGRAM						

The program starts in main on line 5. Line 7 and 8 create the City and Robot. **Line 10** is important; it creates the variable that will be used to record the number of times the Robot has

moved. **Line 11** states that, while the counter variable is less than 7, do whatever is inside the loop. Next, **line 13** tells the Robot to move forward once. Line 14 is the counting part of the program, it adds one to the **counter** variable, which is our way of keeping track of how many times the Robot has moved. After line 13, the program jumps back up to line 11 and repeats the process until the **counter** is equal to 7, in which case the statement is no longer true, so the while loop quits.

It is important to note that we could have used different values to keep track of the counting loop.
For example, we could have said on **line 10:** `int counter = 14;`,
and then changed the argument to on **line 11:** to `while ( counter > 7 )`
and then changed **line 14** to `counter = counter - 1;`,
and the program would still move the Robot over 7 spaces. The only difference would be that the variable would count down from 14 and stop once it was no longer greater than 7.

With that said, it is important to note that having easy-to-read syntax is a very important part of programming. So although you could set up your code however you want, it is best to start at zero and count up to a final number just so what you are trying to do is as clear as possible.

Syntax Explanation:

It doesn't matter too much where the counting while loop is placed, it can be inside a method or inside of main. The syntax is the same for the other while loops that have been used before.

Let's start with the program as it's written here. See how we have to put a `Jo.move();` every time we want one more move

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

Below, you can see the finished program with the differences highlighted in yellow, so it's easy to see what's been added/changed). The clutter of 7 move commands has been replaced with one counting while loop.

Line #	Program Source Code
1.	`import becker.robots.*;`
2.	`public class Counting_While_Loop extends Object`
3.	`{`
4.	`    public static void main(String[] args)`
5.	`    {`
6.	`        City toronto = new City();`
7.	`        Robot Jo = new Robot(toronto, 3, 0, Direction.EAST, 0);`
8.	`        int counter = 0;`
9.	`        while ( counter < 7 )`
10.	`        {`
11.	`            Jo.move();`
12.	`            counter = counter + 1;`
13.	`        }`
14.	`    }`
15.	`}`

The basic syntax of a counting while loop is

```
int pick_a_variable_name = whatever_interger_you_want;

while(pick_a_variable_name is ==, >, <, ≤, ≥, or != final_number)
{
    whatever_commands_you_want
    pick_a_variable_name = pick_a_variable_name + new_number

}
```

Help With The Logic:

This is a little different from the while loops that we have already learned. If you want the Robot to move to the end of a hall, it's better to do something like;

```
while (Robot.frontIsClear())
{
Robot.move
}
```

however if you want to keep track of how many times something is happening (which will be more useful later when we learn how to print messages into the console), or if you want a shortcut for doing a movement a specific number of times, then you should use a counting while loop

Important Details:

- Make sure that you have created a counter to keep track of the event, and make sure that you don't forget to increase/decrease the counter so it serves its purpose!

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

- **Quick Name Of Mistake:** Intent Errors with Counting
  Detailed Example Of Error: You want the loop to turn a specific number of times, however when you run the program it doesn't run as many times as you want.
  Detailed Example Of Fix: You can fix this by tracing through the program, and making sure that your counter variable is at the correct starting value and that the **Boolean Expression** in your while loop's argument is set up so that the program stops after the desired number of events.

Licensing

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

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)