Lesson 2.3: While Loops, Iterations, and For Loops

Mimi Duong
Rosalba Rodriguez

Java Crash Course

January 6th, 2015

Do Now

Take 2 - 3 minutes to pull up your homework from last night.

If you got stuck last night, don't worry. We are going to go over a possible solution and walk through the code.

CLICK HERE for the solution.

Algorithm

Definition:

A set of steps to follow to complete a task

ex:

take in a number

add 10 to it 5 times

print out the number

while()

Your first *flow control* statement!

Flow control: go through an algorithm fully, then start over again (depending)

Syntax:

```
while(<condition is true?>) {
    run the code in this block...
}
```

while() Example

What do you think this chunk of code does?

```
int a = 0;
while (a < 10) {
    System.out.println(a);
}
```

while() Example

See what the code does:

- Create a new class called **whileLoopEx**
- Insert the previous code into your class
- Compile and run!

while() Example

The **stopping condition** is what causes the while() loop to terminate (finish)

In this case, it's when it is no longer true that $a < 10$.

We need a **loop variant** here - something that changes each time the loop iterates (runs through) - so that we get closer to the stopping condition over time.

while() Example: Fixed!

Functioning code:

```
int a = 0;
while (a < 10) {
  System.out.println(a);
  a = a + 1; // loop variant: 'a' changes over
      time
}
```

Playing with while()

- Replace

```
a = a + 1;
```

  with

```
a++;
```

- change the while() condition

```
a < 10
```

  to

```
a <= 10
```

Classwork 8

Write a program to print all positive integers up to 100.

Extra: if you finish the above, write a program to display all **even**, positive integers up to 84.

Iterations

What do you think the following code block will produce?

```
int i = 0;
  int j = 1;
  int a = 10;
  while(i < j) {
      System.out.println(a); {
      i++;    // (same as i = i + 1;)
      j++;
      a;
  }
```

Tracing through an Iterative Algorithm

Using a table is a great way to keep track of what's happening in an iterative algorithm.

while() loop algorithm: while(i < j) {i++; j++; a−}

We will trace together on the board.

Classwork 9

Task 1: $n + (n - 1) + \ldots + 3 + 2 + 1$
Task 2: $n^2 + (n - 1)^2 + \ldots + 3^2 + 2^2 + 1^2$

With a partner, **on paper**, figure out how to write a while() loop to accomplish the first task, where n is any number.
Hints:

- as before, have an *accumulator* variable like **sum**
- consider what *changes* with every term
- **then** figure out the stopping condition

Once you and your partner agree on a solution, show one of the instructors. If we tell you to, try to code the algorithm in Java. Then compile, run, and check if you get correct answers. Repeat for Task 2.

For Loops

For loops are a translation of while() loops. The only difference is that for loops use an iterator.

General syntax:

```
for (int variableName = value; <running codition
    using variableName>; incrementation/decrementation
    to reach stopping condition) {
     ....
}
```

Live Coding

Follow along as we translate a previous while() loop into a for loop

Classwork 10

Write a class called **printStars**. In this class, you have to use a for loop to print a single row of stars. You will be creating a *static method*, that when you call and use the number of stars the user input.

ex:

```
How many stars would you like?
5
*****
```

Nested For Loops

Cool feature to for loops:

you can nest them!

This means that you can have one for loop inside of another for loop.

Nested For Loops

General syntax:

```
for ( ... ) {
    for ( ... ) {
    <insert code to evaluate here>
     }
}
```

Nested For Loops

Using nested for loops is very similar to using regular, single for loops.

Follow the iteration on the board for the following code:

```
for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
        System.out.println("*");
    }
}
```