

COMP 110/L Lecture 14

Maryam Jalali

Sides adapted from Dr. Kyle Dewey

Outline

- **Loops**
 - `while`
 - `for`
 - `do...while`
- **Shorthand variable updates**

Loops

Motivation

Some computations need to be performed multiple times
We need a way of repeating code!

Motivation

Some computations need to be performed multiple times

Question: given only $+$, how can $*$ be implemented?

$$3 * 4$$

Motivation

Some computations need to be performed multiple times

Question: given only $+$, how can $*$ be implemented?

$$3 * 4$$

$$3 + 3 + 3 + 3 \text{ (or } 4 + 4 + 4)$$

Motivation

Some computations need to be performed multiple times

Question: given only $+$, how can $*$ be implemented?

$$3 * 4$$

$$3 + 3 + 3 + 3 \text{ (or } 4 + 4 + 4)$$

$$12$$

Motivation

Some computations need to be performed multiple times

Question: given only $+$, how can $*$ be implemented?

$$3 * 4$$

$$3 + 3 + 3 + 3 \text{ (or } 4 + 4 + 4)$$

$$12$$

$$A * B$$

Motivation

Some computations need to be performed multiple times

Question: given only $+$, how can $*$ be implemented?

$$3 * 4$$

$$3 + 3 + 3 + 3 \text{ (or } 4 + 4 + 4)$$

$$12$$

$$A * B$$

Add A to itself B times
(with some extra rules)


```
public static int  
multiply(int a, int b) {  
    ...  
}
```

```
public static int  
multiply(int a, int b) {  
    switch(b) {  
        case 0:  
            return 0;  
        case 1:  
            return a;  
        case 2:  
            return a + a;  
        case 3:  
            return a + a + a;  
        ...  
    }  
}
```

Enter while

Intuition: while a condition is true, execute the given code.
Condition checked, all code executed, condition checked...

Three essential components

- An *initialization* statement that specifies how the loop begins
- A *continuation* (or *termination*) condition that specifies whether the loop should continue to execute or terminate
- An *iteration* statement that makes progress toward the termination condition

Enter while

Intuition: while a condition is true, execute the given code.
Condition checked, all code executed, condition checked...

```
int x = 0;
while (x < 10) {
    System.out.println(x);
    x = x + 1;
}
```

Example:

`WhileXLessThan10.java`

Revisiting Multiplication:

`MultiplyWithWhile.java`

while Caveat

Counterintuitively, it does **not** exactly mean:
“while condition is true”

while Caveat

Counterintuitively, it does **not** exactly mean:
“while condition is true”

```
int x = 0;
while (x < 5) {
    System.out.println("hi");
    x = 10;
    System.out.println("bye");
}
```

while Caveat

Counterintuitively, it does **not** exactly mean:
“while condition is true”

```
int x = 0;  
while (x < 5) { Condition only checked here  
    System.out.println("hi");  
    x = 10;  
    System.out.println("bye");  
}
```

Prints:

hi

bye

A Pattern Emerges

- Many loops commonly:
 - Do some sort of initialization
 - Check some sort of condition
 - Update some variables on each iteration
- Special type of loop for this: `for`

for Loops

for Loops

```
int x = 0;
while (x < 10) {
    System.out.println(x);
    x = x + 1;
}
```

for Loops

```
int x = 0; Initialization
while (x < 10) {
    System.out.println(x);
    x = x + 1;
}
```


for Loops

```
int x = 0; Initialization
while (x < 10) { Condition check
    System.out.println(x);
    x = x + 1;
}
```

for Loops

```
int x = 0; Initialization
while (x < 10) { Condition check
    System.out.println(x);
    x = x + 1; Variable update
}
```

for Loops

```
int x = 0; Initialization
while (x < 10) { Condition check
    System.out.println(x);
    x = x + 1; Variable update
}
```

```
for (int x = 0; x < 10; x = x + 1) {
    System.out.println(x);
}
```

for Loops

```
int x = 0; Initialization
while (x < 10) { Condition check
    System.out.println(x);
    x = x + 1; Variable update
}
```

Initialization

```
for (int x = 0; x < 10; x = x + 1) {
    System.out.println(x);
}
```

for Loops

```
int x = 0; Initialization
while (x < 10) { Condition check
    System.out.println(x);
    x = x + 1; Variable update
}
```

Initialization

Condition check

```
for (int x = 0; x < 10; x = x + 1) {
    System.out.println(x);
}
```

for Loops

```
int x = 0; Initialization
while (x < 10) { Condition check
    System.out.println(x);
    x = x + 1; Variable update
}
```

Initialization	Condition check	Variable update
<pre>for (int x = 0; x < 10; x = x + 1) { System.out.println(x); }</pre>		

Example:

`ForXLessThan10.java`

Revisiting Multiplication:

`MultiplyWithFor.java`

Same Condition Caveat

Condition is only checked at the start of the loop.

Increment is only done at the end of the loop.

Same Condition Caveat

Condition is only checked at the start of the loop.

Increment is only done at the end of the loop.

```
for (int x = 0; x < 5;) {  
    System.out.println("hi");  
    x = 10;  
    System.out.println("bye");  
}
```

Same Condition Caveat

Condition is only checked at the start of the loop.

Increment is only done at the end of the loop.

Condition only checked here

```
for (int x = 0; x < 5;) {  
    System.out.println("hi");  
    x = 10;  
    System.out.println("bye");  
}
```

Prints:

hi

bye

for vs. while

- Sometimes `for` is more appropriate, sometimes `while`
- Either will work in any situation where a loop is needed
- In general you use a for loop when you know how many (even a variable number of) iterations you are going to execute
- In general you use a while loop when you don't know (up front) how many iterations you will execute

do . . . while Loops

Like a `while` loop, but the condition is checked at the end.

`do . . . while` always executes at least once, unlike `while`.

do . . . while Loops

Like a `while` loop, but the condition is checked at the end.

`do . . . while` **always** executes at least once, unlike `while`.

```
int x = 0;
do {
    System.out.println(x);
    x = x + 1;
} while (x < 10);
```

Example:

```
DoWhileXLessThan10.java
```

Multiplication with do . . . while

Conversion to do . . . while would be **incorrect**

Multiplication with do...while

Conversion to do...while would be **incorrect**

```
public static int  
multiply(int a, int b) {  
    int result = 0;  
    while (b > 0) {  
        result = result + a;  
        b = b - 1;  
    }  
    return result;  
}
```

Multiplication with do...while

Conversion to do...while would be **incorrect**

```
public static int  
multiply(int a, int b) {  
    int result = 0;    Won't be true  
    while (b > 0) { if b initially was 0  
        result = result + a;  
        b = b - 1;  
    }  
    return result;  
}
```

Shorthand Variable Updates

Motivation

We very often update variables in loops

Motivation

We very often update variables in loops

```
x = x + 1;  
b = b - 1;  
result = result + a;
```

Motivation

We very often update variables in loops

```
x = x + 1;  
b = b - 1;  
result = result + a;
```

```
x++ OR ++x  
b-- OR --b  
result += a;
```

Motivation

We very often update variables in loops

```
x = x + 1;  
b = b - 1;  
result = result + a;
```

```
x++ OR ++x  
b-- OR --b  
result += a;
```

Saves some typing, very commonly used.