# Computer Science I

**Chapter 4 Notes**

## Loops

- *Loops* allow us to execute a statement multiple times
- Like if statements, they are controlled by boolean expressions.
- Java has three kinds of loops:
  - the *while loop*
  - the *for loop*
  - the *do loop* *(this one is not covered in CS 172)*
- The programmer should choose the right kind of loop for the situation.

## THE WHILE LOOP

- A *while loop* has the following syntax:

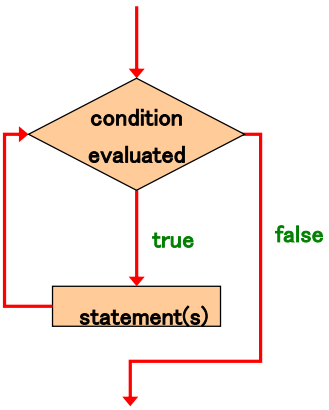
```
while ( condition )
    statement;
```

If the condition is true, the statement is executed.

Then the condition is evaluated again, and if it is still true, the statement is executed again.

The statement is executed repeatedly until the condition becomes false.

## LOGIC OF A WHILE LOOP



## THE WHILE STATEMENT

An example of a while statement:

```
int count = 1;
while (count <= 5)
{
   System.out.println (count);
   count++;
}
```

If the condition of a while loop is false initially, the statement is never executed.

Therefore, the body of a while loop will execute zero or more times.

## THE WHILE STATEMENT

- Let's look at some examples of loop processing.

1. A loop can be used to calculate a sum.
2. A loop can be used to count items in the input.
3. A *sentinel value* is a special input value that represents the end of input.
4. A loop can be used to print a sequence.

## Infinite Loops

- The body of a `while` loop eventually must make the condition false.
- If not, it is called an *infinite loop*, which will execute forever (until the user interrupts the program).
- This is a common logical error.
- You should always double check the logic of a program to ensure that your loops will terminate normally.

## Infinite Loops

- An example of an infinite loop:

```
int count = 1;
while (count <= 25)
{
   System.out.println (count);
   count = count - 1;
}
```

This loop will continue executing until interrupted.

In JGrasp, there is an End button in the bottom portion of the screen.

## Nested Loops

- Similar to nested `if` statements, loops can be nested as well.
- The body of a loop can contain another loop.
- For each iteration of the outer loop, the inner loop iterates completely.

## Nested Loops

- How many times will the string `"Here"` be printed?

```
count1 = 1;
while (count1 <= 10)
{
    count2 = 1;
    while (count2 <= 20)
    {
        System.out.println ("Here");
        count2++;
    }
    count1++;
}
```

**The outer loop repeats 10 times.**
**Every time the outer loop repeats, the inner loop will repeat 20 times.**
**10 * 20 = 200**

## THE FOR LOOP

- A *for loop* has the following syntax:

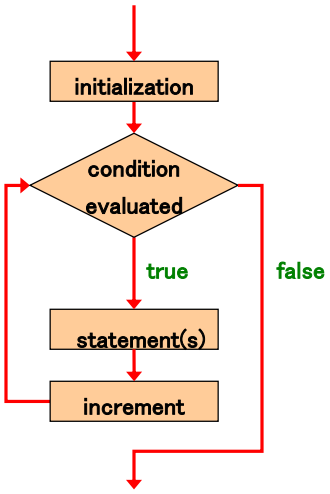
The *initialization* is executed once before the loop begins

The *loop repeats if* the condition is true.

```
for ( initialization ; condition ; increment )
   statement;
```

The *increment* portion is executed AFTER the statement(s) in the body of the loop.

## LOGIC OF A FOR LOOP



## THE FOR LOOP

- A `for` loop is functionally equivalent to the following `while` loop structure:

```
initialization;
while ( condition )
{
   statement;
   increment;
}
```

## THE FOR LOOP

- An example of a `for` loop:

```
for (int count=1; count <= 5; count++)
   System.out.println (count);
```

The initialization section can be used to declare a variable.

Like a while loop, the condition of a for loop is tested prior to executing the loop body.

Therefore, the body of a for loop will execute zero or more times.

## THE FOR LOOP

- The increment section can perform any calculation

```
for (int num=100; num > 0; num -= 5)
   System.out.println (num);
```

A for loop is best when you know, in advance, how many times you want the loop to execute.