Java loops

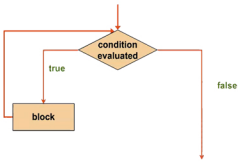
The `while` loop

- If the *condition* is true, the *block* is executed
- Then the condition is evaluated again, and if it is still true, the block is executed again
- The block is executed repeatedly until the condition becomes false
- The condition is evaluated at the *start* of each iteration.
- When the condition evaluates to false, the block is *not* executed. Control passes to the next statement *after* the block.

Syntax

```
while ( condition )
      block
```

FlowChart



The `for` loop

- A `for` loop has the following syntax

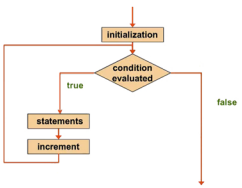
The *initialization* is executed once before the loop begins

The *block* is executed until the *condition* becomes false

```
for ( initialization ; condition ; increment )
{
    statements
}
```

The *increment* portion is executed at the end of each iteration

FlowChart



Example of `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

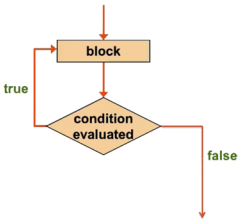
The `do... while` loop

- A *do... while loop* has the following syntax:

```
do
    block
while ( condition );
```

- The *block* is executed once initially, and then the *condition* is evaluated
- The statement is executed repeatedly until the condition becomes false
- The condition is evaluated at the *end* of each iteration
- When the condition becomes *false*, control passes to whatever comes next.
- *note semi-colon at end of the loop*

FlowChart



The `do...while` loop

- An example of a `do...while` loop

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

Note the semi-colon here

- In a `do...while` loop the statement is first executed and then the condition is evaluated
- Therefore, the body of a `do..while` loop will execute at least once

Extra: Comparing For loop with While loop

`while` and `for` loop equivalence

while loop

```
int count=0;
int sum=0;

while (count < max)
{
    int newInt =   scan.nextInt();
    sum = sum + newInt;
    count++;
}
```

for loop

```
int sum=0;

for (int count=0; count < max; count++)
{
        int newInt =scan.nextInt();
        sum = sum + newInt;
}
```

Every `for` loop can always be rewritten as a `while` loop

`for` loops are typically preferred for any form of "counting" (i.e., where the number of iterations is known)