JAVA

Loop’s while, do and for

Repetition Statements

- *Repetition statements* allow us to execute a statement multiple times
- Often they are referred to as *loops*
- Like conditional statements, they are controlled by boolean expressions
- Java has three kinds of repetition statements:
  - the *while loop*
  - the *do loop*
  - the *for loop*
- The programmer should choose the right kind of loop for the situation

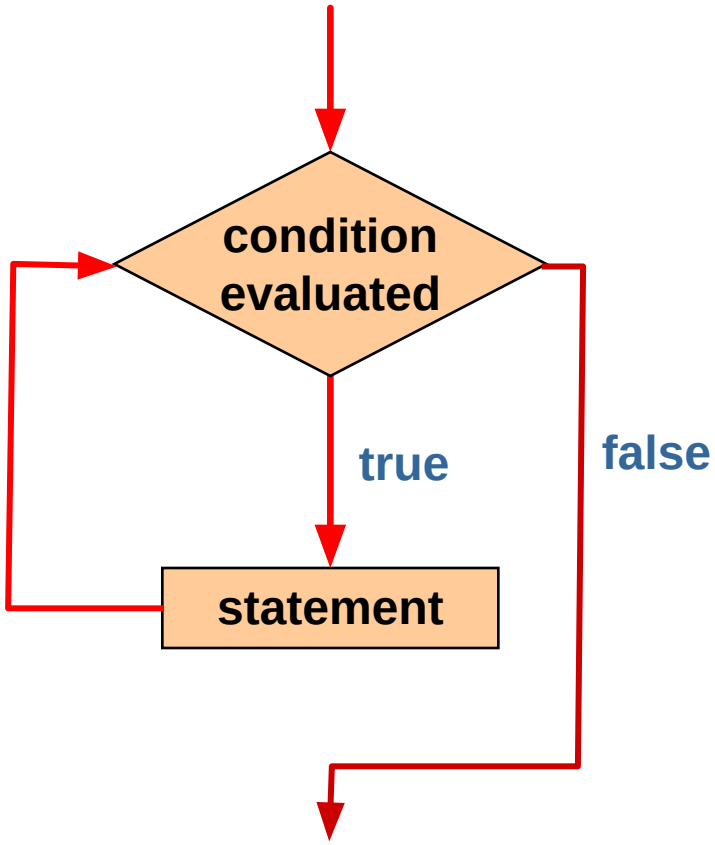
The while Statement

- A *while statement* 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
- A loop can be used to maintain a *running sum*
- A *sentinel value* is a special input value that represents the end of input
- See Average.java Listing 5.8
- A loop can also be used for *input validation*, making a program more *robust*
- See WinPercentage.java Listing 5.9

Nested Loops

- Similar to nested `if` statements, loops can be nested as well
- That is, 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++;
}
```

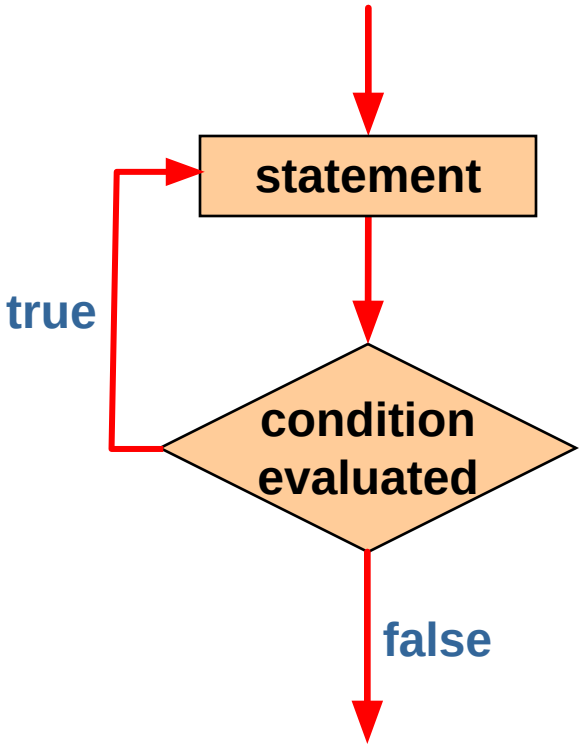
The do Statement

- A *do statement* has the following syntax:

```
do
{
    statement;
}
while ( condition );
```

- The `statement` is executed once initially, and then the `condition` is evaluated
- The statement is executed repeatedly until the condition becomes false

Logic of a do Loop



The do Statement

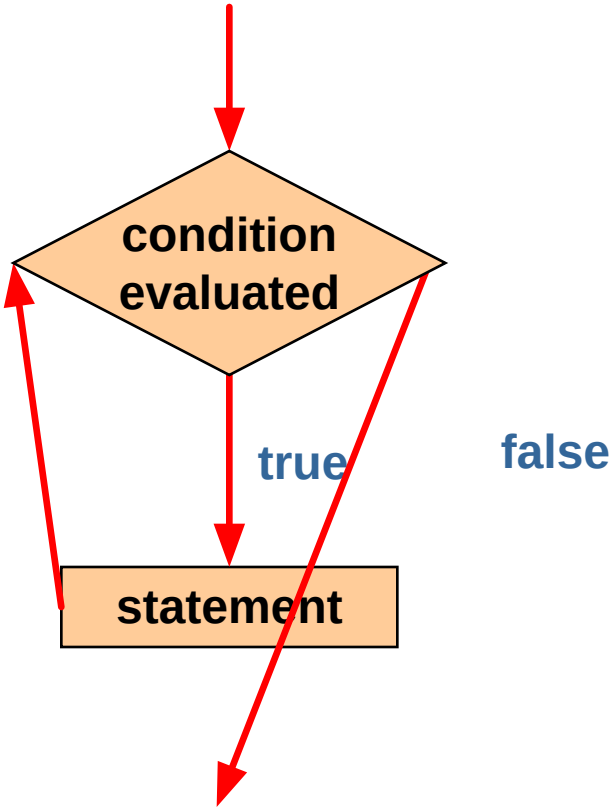
- An example of a `do` loop:

```
int count = 0;
do
{
   count++;
   System.out.println (count);
} while (count < 5);
```

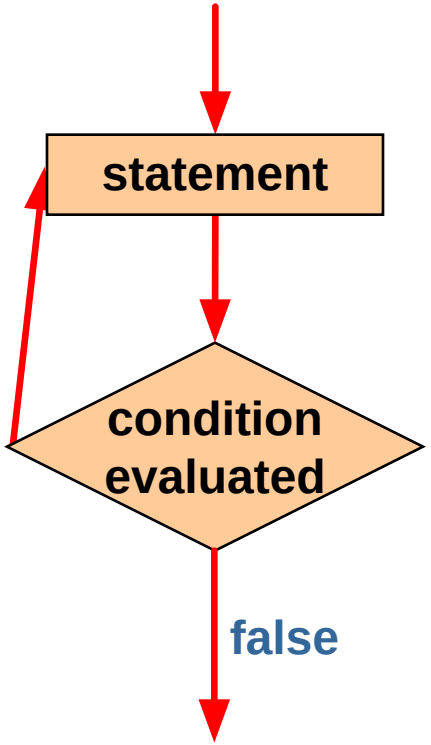
- The body of a `do` loop executes at least once

Comparing while and do

The while Loop



The do Loop



The for Statement

- A *for statement* has the following syntax:

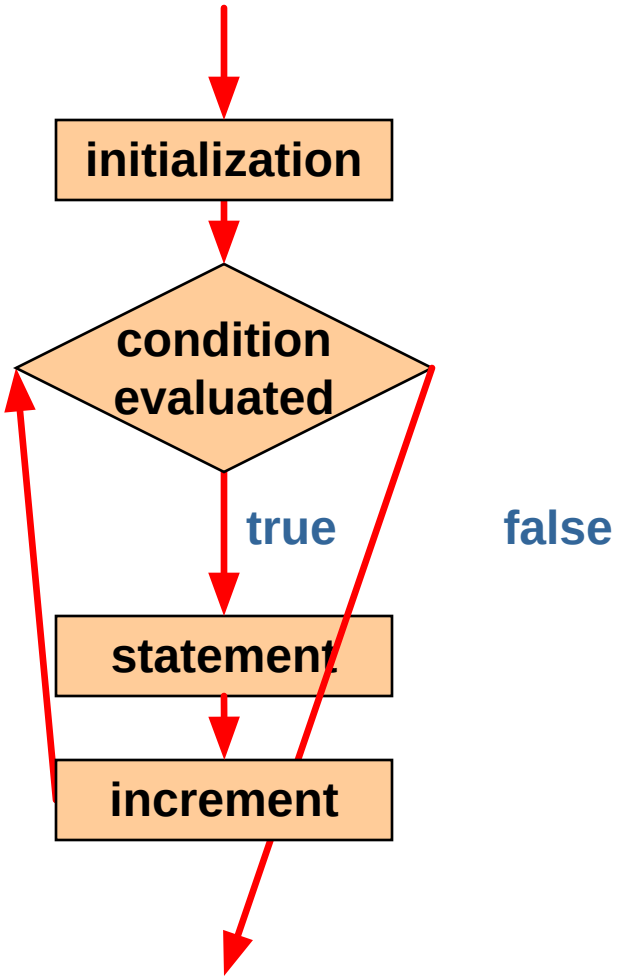
The *initialization* is executed once before the loop begins

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

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

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

Logic of a for loop



The for Statement

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

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

The for Statement

- 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 Statement

- The increment section can perform any calculation

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

- A `for` loop is well suited for executing statements a specific number of times that can be calculated or determined in advance
- See Stars.java Listing 5.14

The for Statement

- Each expression in the header of a `for` loop is optional
- If the initialization is left out, no initialization is performed
- If the condition is left out, it is always considered to be true, and therefore creates an infinite loop
- If the increment is left out, no increment operation is performed