Assignment 2.4: Loops

This assignment will continue using the Scanner class to handle input from the terminal. Please refer to Assignment 2.1 for a refresher on this topic.

For Loops

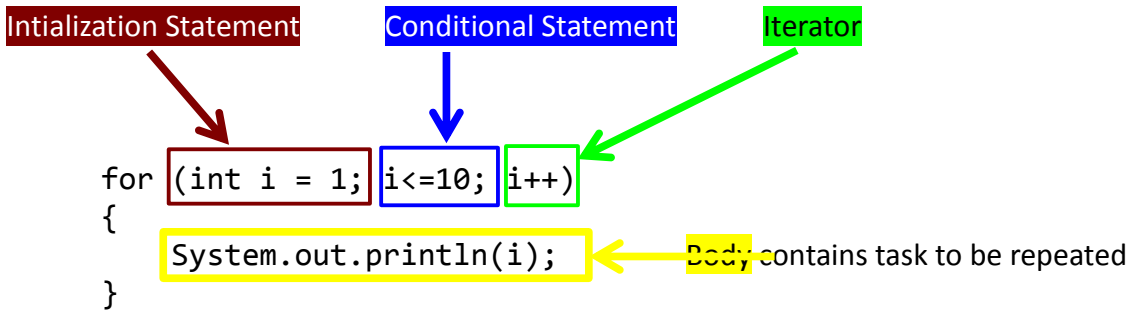
A for loop is used to repeat a set of instructions. Here is an example. Please read it carefully and try to figure out what it does before reading on.

Example 1:

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

Check your answer at the end

Parts of a For Loop



Initialization Statement

The Initialization Statement creates a variable to keep track of how many times we have repeated the loop, and what value we will start counting from. In this case, we call this variable “i” and we start counting from 1. The initialization statement only happens the first time we start the loop, not every time we go through the loop.

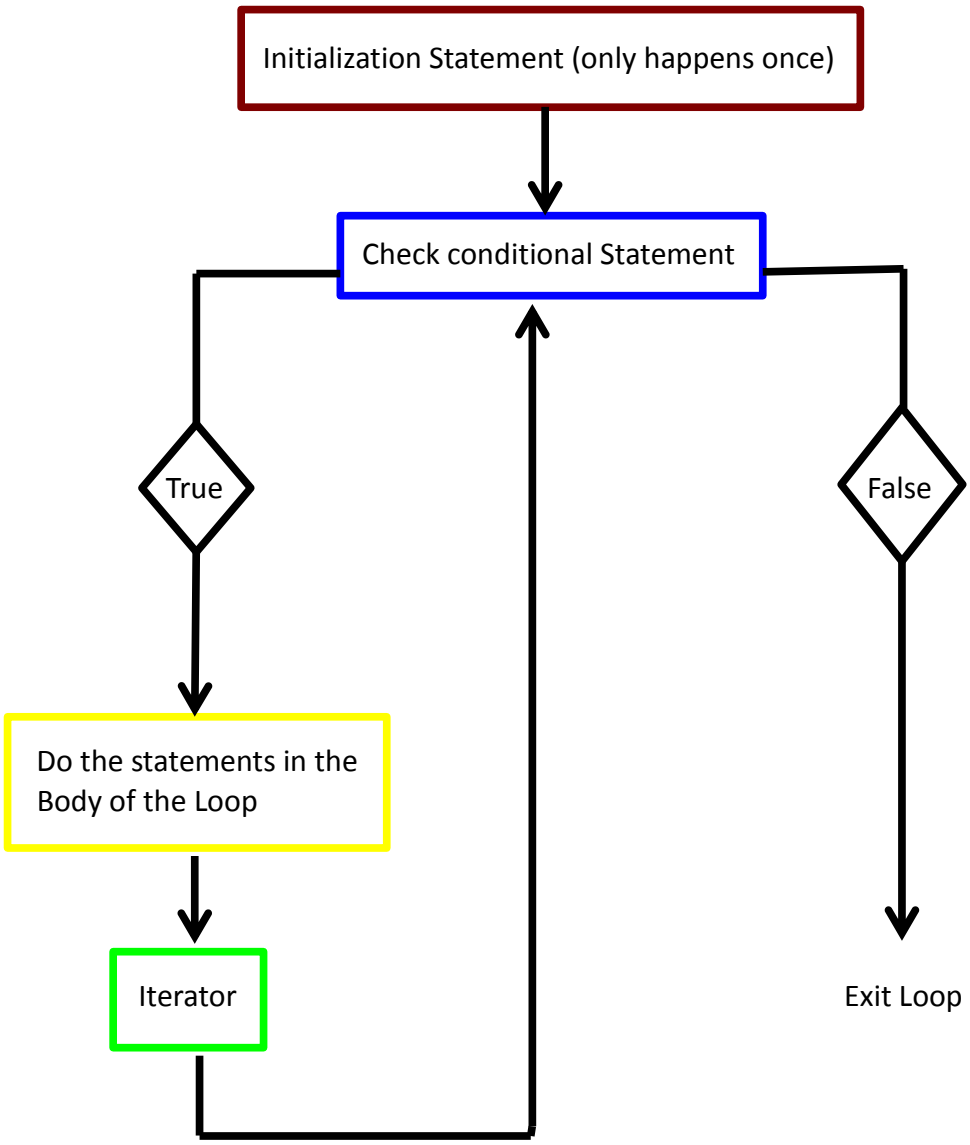
Conditional Statement

The Conditional Statement is a boolean (true or false) expression that determines if we repeat the loop or not. The statement evaluates to true, the task will be repeated another time. If it is false, the loop will stop. The Conditional Statement is checked every time it repeats the loop, including the FIRST time through the loop. So, it is possible that the body of the loop doesn't even get to run once.

Iterator

The Iterator is how we are counting as we go through the loop. The code `i++` is short for "add 1 to i", which means we are counting by ones. You could replace this with other statements, such as `i--` to count down by 1 or `i = i * 2` to double i every time.

How A Loop Works in Detail



Using this flowchart, we can see that the output from our example loop will be: 12345678910

While Loops

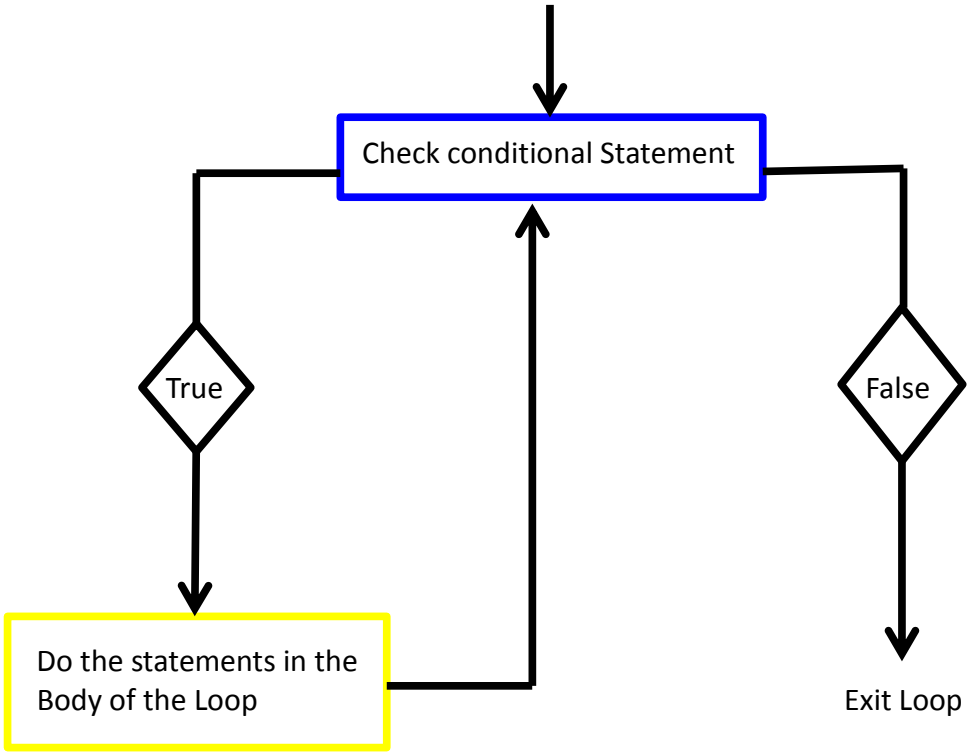
While loops are the most used loop in Java. It is the simplest and most versatile loop.

```
while (true)
{
      System.out.print("!");
}
```

Conditional Statement

Body contains task to be repeated

As long as the conditional statement is true, the loop will repeat the statements in the body of the loop. The flow chart for this kind of loop looks like this:



If the conditional statement is false at the beginning of the loop, then the loop will **not** execute the statements in the body of the loop even once.

All for loops can also be written as whle loops. For example, this for loop:

```
for (int count = 1; count < 11; count++)
{
      System.out.println("Count is: " + count);
}
```

Can be written as the following while loop:

```
int count = 1;
while (count < 11)
{
      System.out.println("Count is: " + count);
      count++;
}
```

Take some time to trace through this loop and understand how the two loops work the same.

As you can see, **for loops** are simpler for repeating a task a set number of times. However, **while loops** can be like "forever if" loops in scratch. They will continue to repeat something until the condition is false.

Here is an example of a while loop that does not repeat a task a set number of times, but keeps doing the task until some condition becomes false.

```
1    public void whileTest()
2    {
3        Scanner sc = new Scanner(System.in);
4
5        String answer = "Mike";
6
7        while (answer.equals("Mike"))
8        {
9            System.out.print("What's your name, Mike? ");
10           answer = sc.next();
11       }
12   System.out.print("Hello, " + answer);
13   }
```

A sample of the output of the program is as follows:

```
What's your name, Mike?  Mike
What's your name, Mike?  Mike
What's your name, Mike?  Mike
What's your name, Mike?  Bob
Hello, Bob
```

Think about the answer to these question about the above loop. If you don't know the answers, ask for help!

On line 5, we initialize the variable answer to "Mike". What would have happened if we set it to "Bob" ?

How does Java know when to stop the loop and move on to the next instructions?

You can implement an infinite loop using the while statement as follows:

```
while (true)
{
    // your code goes here
}
```

Do While Loops

Do while loops are exactly like while loops, except the condition is checked at the end. This means the loop will always execute **at least once**. Here is an example of a do while loop that keeps generating random numbers between 1 and 20 (like a d20 die). It asks the user if they want to roll the die again, and repeats as long as they don't say no.

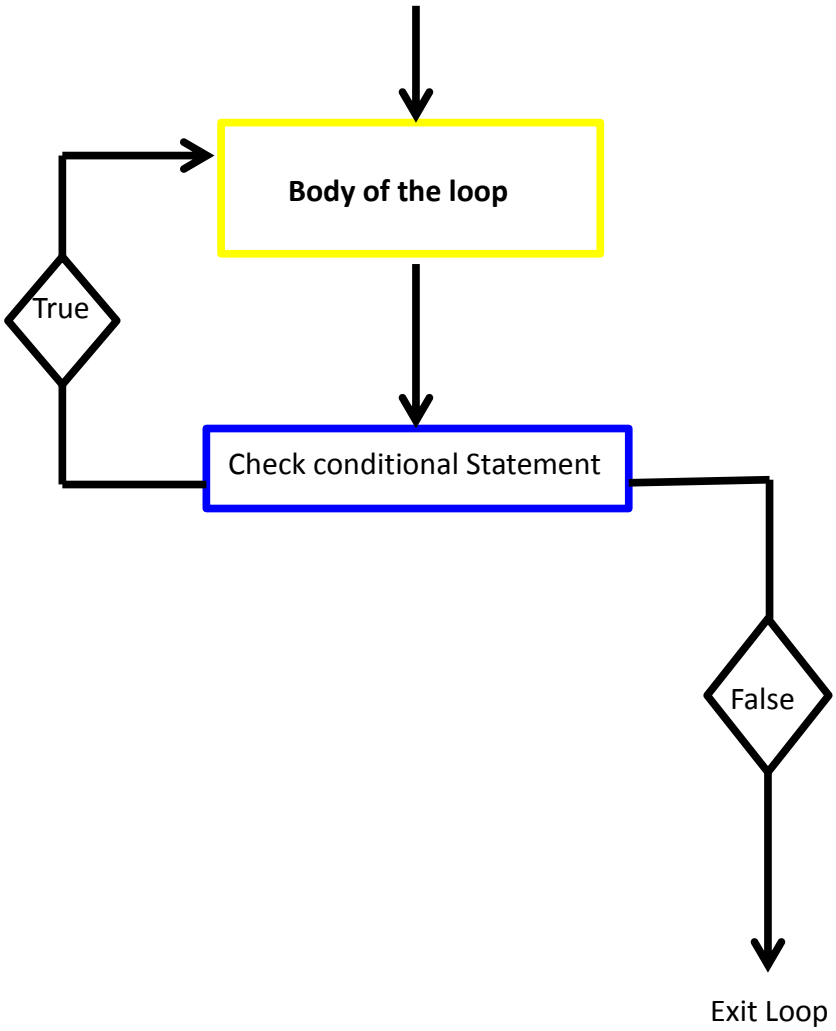
```
class D20()
{
   public static void main (String[] args)
   {
       double magicNumber;
       String answer;
       Scanner sc = new Scanner (System.in);

       do
       {
           magicNumber = Math.random();
           magicNumber = (int) (magicNumber * 20 + 1);
           System.out.println("You rolled a " + magicNumber);
           System.out.print("Again? [y,n]");
           answer = sc.next();
       } while(!answer.equals("n"));

       System.out.println("Good bye!");
   }
}
```

Read over this code carefully. Trace through it in your mind and make sure you ask questions if it does not make sense. Consider the following questions: Can you rewrite this as a while loop? If so, how? Can you rewrite this as a for loop? If so, how?

Flow Chart for Do While Loops:



EXERCISE 1:

What is the output of this loop?

```
for (i = 100; i >= 10; i--)
{
     System.out.println("i is: " + i);
}
```

Rewrite this for loop as a while loop.

EXERCISE 2:

What is the output of this loop?

```
for (i = 1; i <= 10; i++)
{
     for (j = 1 ; j <= i; j++)
     {
          System.out.print(j + " ");
     }
     System.out.println("");
}
```

EXERCISE 3:

Write class RunningTally that repeatedly asks the user to enter an integer until the users gives a negative integer. At that point, the program prints out the total of all the numbers that the user entered (not including the negative). For example:

```
Let's add some numbers! [Type a negative number to quit]

Add:  4
Add:  6
Add:  2
Add:  -1

Your total is 12.
```

Note: Although the total is displayed at the end, it is important to update the total every time the user enters a new number.

EXERCISE 4:

Write class Bar that asks a user for a String char and an int x. The program then prints out a row of chars that is as long as x. For example, your program would look like this:

```
What character do you want your bar made of? *
How long do you want your bar? 7

Here's your bar!

*******
```

EXERCISE 5:

Write class Triangle that asks a user for a String char and an int x. Then the program prints out a right triangle of chars that is as tall as x. For example:

```
What character do you want your triangle made of? *
How tall do you want your triangle? 5

Here's your triangle!

*
**
***
****
*****
******
```

This will require a nested for loop (a for loop inside a for loop).

EXERCISE 6:

Cut and paste your old D6 program into this project. Change your code so that it is contained within a loop. Change it so it runs like the following output:

```
How many D6's do you want to roll? 3

You rolled: 2 5 4
Total: 11

Again? [y,n]  y

How many D6's do you want to roll? 5

You rolled: 1 2 1 3 4
Total: 11

Again? [y,n]  n
Good-bye!
```

EXERCISE 7:

Write class GuessingGame that simulates a the "Guess What Number I'm Thinking Of" game. In this game, the computer picks a random number between 1 and 100. Then, the method will begin a loop, asking the user to guess the number. After each guess, the method will tell the user whether the

guess was too high or too low, and then repeat the process. If the user guesses correctly, the loop will stop and the method will then display a congratulatory message and tell you how many tries it took you to guess the correct answer. Sample output is as follows:

```
I've picked a random number between 1 and 100.  Try to guess it!
What is your guess?  50
Too high ...
What is your guess?  25
Too high ...
What is your guesss?  12
Too low ...
What is your guess?  16
You've guessed my number! Good job!  It only took you 4 tries.
```