COMPARING DO WHILE AND WHILE LOOPS

Document Summary:

Here we will be comparing two well-known loops: the while and do while loops. First, let's start by discussing the most fundamental differences in a code development environment (syntax wise). To demonstrate “while” and “do while” loops, I will depict their syntax on the following:

While loops Syntax:

```
while (condition) {
    // Block of code
}
```

Figure 1 (While loops)

Do while loops Syntax:

```
do {
   // Block of code
} while (condition);
```

Figure 2 (Do while loops)

It's clear in Figures 1 and 2 that we can find distinctions between “do while” and “while” loops (code wise). In while loops, the code will continuously be executed as long as the condition is true, it will only end when the condition is satisfied. As compared to “do while" loops, the condition occurs after the code has been executed at least once for each iteration/repetition; condition is verified after the code.

NOTE: In loops, when the condition is never satisfied or will never be true, it will lead us into an endless loop.

To become more transparent, let’s further enhance our understanding by providing you some examples.

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

Figure 3 (example of while loops)

As shown in Figure 3, the code initially states that while the value of a, which is equal to zero, is less than 10, keep printing the value as long as the condition is false. However, you will see that there is an increment code under the print statement, which means that the value of a (0) will continue to increase with each iteration. This will result in the printing of numbers 0-9.

NOTE: The word iteration initially means: repetition/loop.

Why does the output only print up to nine and not ten? It's because nine is less than ten. If you want to print from 0-10, you should have a condition of (a <=10) to get that output.

Here's a logical way to see how **while loops** work:

Step 1: Check the condition if true then proceed to step 2, or else exit.
Step 2: Execute the code as long as the condition is true.
Step 3: Repeat.

The loop will end when the condition is finally false.

NOTE: in while loops, the code inside it will only be executed as long as the condition is true.

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

Figure 4 (example of do while loops)

Here, you'll see a do while the loop, still the same code logic as Figure 3. (it will still print an output of 0-9). So, what's the difference here? The distinction here is that, likewise, the condition is not validated at the outset, rather the block of codes is executed at least once at the beginning before the condition is tested if either true or false. The process will repeat for each iteration.

Here's a logical way to see of how **do while loops** work:

Step 1: Execute the code once.
Step 2: Check the condition, if true then proceed or else exit.
Step 3: Repeat.

And so on and so forth until the value of a is equals to 9 since it's the highest point you could reach for this condition.
In a very simple way to compact this whole document in a very short sentence, in while loops, it will first verify the conditions before executing the code; they will continuously execute the code

as long as the condition is true. As compared to do while loops, the code block will be executed at the beginning before checking the state; the code will be executed once before checking the condition. This will occur on a continuous basis with each iteration.

>> END OF DOCUMENT <<