The Java programming language contains three loops. Loops are blocks of code designed to repeat a process until an exit condition is reached. The three loops are the *for* loop, the *while* loop and the *do while* loop. Both the *for* loop and the *while* loop are entry control loops, meaning they check for a Boolean value to an expression prior to entering the loop. The syntax of a *for* loop is as follows:

```
for (expression 1;       // sets a variable for iteration of the loop— this variable is evaluated only the first time the loop is
                         // run, and can be local to the loop only
     expression 2;       // Boolean condition that must be met to enter the loop

     expression 3)       // increments/decrements variable for eventual loop exit—runs after each time expression 4 executes
{

     expression 4        // body of code to execute while loop condition evaluates to true.
}
```

A *for* loop is useful when you want to specify how many times as body of code needs to be ran.
But what about when you need a loop to run for as many times as necessary until an exit condition is reached?
Then we could turn to a *while* loop. *While* loops execute for as long as a Boolean value evaluates to true. The syntax of a *while* loop is as follows:

```
while (expression 1)    // Boolean condition to evaluate before the loop can be entered
{
        expression 2    // body of code to execute if the Boolean evaluates to true; should contain a modification to a
                        // variable for eventual loop exit
 }
```

What if you want a loop that will always run at least once? This is a case for the *do while* loop. The *do while* loop is called an exit control loop, because it evaluates a Boolean condition at the *end* of the loop instead of at the *beginning*, so it always will execute once, even if the condition is false. The syntax of a *do while* loop is as follows:

```
do
{
expression 1              // body of code to be executed
}
while (expression 2);   // Boolean condition evaluated after expression 1 that determines whether to reenter the loop
```

Sources:
https://www.geeksforgeeks.org/loops-in-java/
https://www.w3schools.com/java/java_for_loop.asp