

Lab - For Loop

Tutorial Lab 1: Using the For Loop

Copy the code below into the text editor on the left. Then click on the TRY IT button to see the resulting output and ++Code Visualizer++ link (below) to see how the program runs line by line.

```
for (int x = 0; x < 11; x++) {  
    if (x % 2 == 0) {  
        System.out.println("Even");  
    }  
    else {  
        System.out.println("Odd");  
    }  
}
```

[Code Visualizer](#)

Program Summary

1. The `for` loop runs through all the values of the variable `x` from 0 to 10 as specified in the loop header.
2. For each value of `x`, an expression is evaluated using a conditional `if` statement.
3. If `x` modulo 2 evaluates to 0, then print Even with a newline character.
4. If `x` modulo 2 *does not* evaluate to 0, then print Odd instead with a newline character.

Lab - While Loop

Tutorial Lab 2: The While Loop

Copy the code below into the text editor on the left. Then click on the TRY IT button to see the resulting output and ++Code Visualizer++ link (below) to see how the program runs line by line.

```
int counter = 0;
while (counter < 10) {
    System.out.println(counter);
    counter = counter + 1;
}
System.out.println("while loop ended");
```

[Code Visualizer](#)

Program Summary

1. A counter variable is initialized to keep track of how many times the loop will be executed.
2. The loop will run as long as counter is less than 10.
3. Each time the loop runs, the integer value of counter is printed to the screen.
4. The value of counter is then incremented by 1.
5. When counter reaches 10, the boolean expression no longer evaluates to true and the program will exit the loop.
6. Before the program terminates, a statement is printed to the screen, indicating that the while loop has ended.
7. Recall that the while loop must have an exit condition. By incrementing the counter variable, we ensure that the loop will eventually end. If we do not increment counter in this loop, we will create an *infinite* loop because counter will never reach 10 or greater.

Lab - Break Statement

Tutorial Lab 3: Breaking from the While Loop

Copy the code below into the text editor in the upper left panel. Then click on the TRY IT button to run the resulting program in the Terminal in the lower left panel.

▼ What does `Scanner sc = new Scanner(System.in);` do?

The `Scanner sc = new Scanner(System.in);` command records what a user enters on the screen and stores that information in the scanner object `sc` as an **input stream**.

▼ What do `try` and `catch` do?

`try` checks to see if `sc`, the input stream, entered by the user can be converted to a double using `Double.parseDouble(sc.next())`. If `sc` can be converted to a double, the system will do so and store that double into the variable `input`. Then it will add `input` to the running sum, `result`. On the other hand, if `sc` cannot be converted to a double, `catch` will ignore the error message and then print the current sum.

```
double result = 0;
double input;
Scanner sc = new Scanner(System.in);

while (true) {
    System.out.print("Enter a number to add to sum. ");
    System.out.println("Or enter a non-number to quit and
        calculate sum.");
    try {
        input = Double.parseDouble(sc.next());
        result = result + input;
    }
    catch (NumberFormatException ignore) {
        System.out.println("Sum = " + result);
        break;
    }
}

sc.close();
```

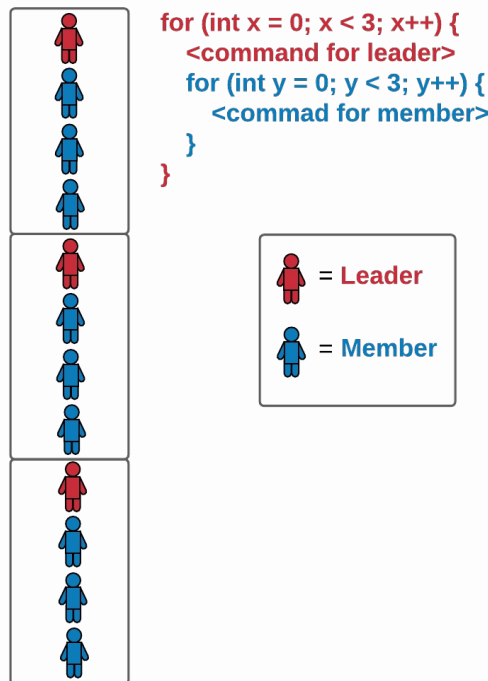
Program Summary

1. Declare the variable `result` and initialize it to `0`. `result` will store the total of the summation.
2. Declare the variable `input`. `input` will store the information that the user enters as a double.
3. Create a scanner object that records the information the user enters as an input stream.
4. Next we set up a `while` loop with `true` as the expression in the loop header. We do this because we want the loop to continue running and storing information from the user. Since we don't know how much information the user will enter, a `while` loop is best for the situation.
5. The user is prompted to enter some information and that information will be converted to a double. Then it is stored in the variable `input` which was declared earlier.
6. If the information was converted and stored into `input` successfully, the value in `input` will be added to the value in `result`, our total summation.
7. If the information was *not* converted and stored into `input` successfully, then the program will ignore the error and print out the total summation `result`. Then the program will exit the `while` loop.

Lab Challenge - Loop Patterns

Nested Loop Example

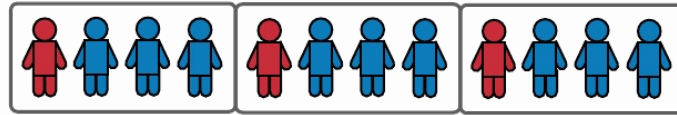
One of the benefits of nested loops is that they can be used to construct complex patterns. Imagine a classroom full of students and they are distributed evenly into smaller groups and asked to form a single line with their groups. The outer loop is like the group leader (represented in **red** and **L**) and the inner loop is like the rest of the group members (represented in **blue** and **M**).



.guides/img/NestedLoopExample

```
for (int x = 0; x < 3; x++) {  
    System.out.println("L");  
    for (int y = 0; y < 3; y++) {  
        System.out.println("M");  
    }  
}
```

What is the pattern described by the above example? There are 3 leaders and **each** leader has 3 members. However, note that the example shows the students standing in a *vertical* line. What if you want to arrange the students in a *horizontal* line like this instead?



`.guides/img/NestedLoopHorizontal`

By removing `\n` from the `System.out.println` commands from the code above, you can accomplish this task. Alternatively, you can also make use of an `if` and `else` statement instead of a nested loop. Both ways will produce the same result.

```
for (int x = 0; x < 3; x++) {  
    System.out.print("L");  
    for (int y = 0; y < 3; y++)  
    {  
        System.out.print("M");  
    }  
}
```

```
for (int x = 0; x < 12; x++)  
    if ((x == 0) || (x == 4) || (x == 8))  
    {  
        System.out.print("L");  
    }  
    else {  
        System.out.print("M");  
    }
```

Nested For Loop Challenge

challenge

Assignment:

For this challenge, you will use your knowledge of **patterns**, **conditionals**, and **nested for loops** to produce the following output:

```
XOXOXOXOX
OXO
OXO
XOXOXOXOX
OXO
OXO
XOXOXOXOX
OXO
OXO
```

Requirement:

Your program **must** include **at least two** for loops, one nested within another, in order to receive credit. In addition, you are only allowed to use, **at most, two** `System.out.print()` or `System.out.println()` statements.

▼ Hint

You should start by determining a pattern that repeats itself. One noticeable pattern is:

```
XOXOXOXOX
OXO
OXO
```

Try creating that particular pattern first, then iterate that pattern by using a loop.
