Name ______________________________________________________________________________

# *for-loops*

| Your Tasks (Mark these off as you go) |
|---|
| ☐ Interpret a *for-loop* |
| ☐ Write a *for-loop* to print a word backwards |
| ☐ Write a for-loop to print a number backwards |
| ☐ Write nested *for-loops* to create shapes on a grid |
| ☐ Receive credit for this lab guide |

## ☐ Interpret a *for-loop*

As we learned previously, one of the most important control structures in Java is the *for-loop*. A *for-loop* is a block of code that is repeated with certain rules about how to start, increment, and end the process.

Consider a loop that sums all the numbers in a given range. To do this we need a starting point and an ending point. We also need to know how to increment each step along the way.

- start = 0
- incrementing = +1
- stop = 100

The problem of summing all the number in a given range can be solved with the following for loop.

```
int sum = 0;

for(int n= 0; n < 100; n++{

    sum += n;

}
```

| Write a for-loop that prints all the even numbers in a given range. |
|---|
|  |

When writing for-loops keep in mind the following

1. Initializing expression. The initializing expression can be any integer.

2. Control expression. The control expression indicates how long to continue looping. This is a boolean expression. As long as the expression is true, the loop will continue.

    **Warning**: There is something really bad that can happen here. You must write your code so as to ensure that this control statement will eventually become false, thus causing the loop to terminate. Otherwise you will have an endless loop which will crash your program.

3. Step expression. The step expression tells us how our variable should change each time through the loop. In this case we are incrementing j each time. However, other possibilities could include,

    n - -        n = n + 4        n = n * 3        etc.

    In the example above, n++ occurs at the bottom of the loop.

Indicate the output for each of the following,

```
int j = 0;

for(int g = 1;g < 10;g++){

                j++;
}

System.out.println(j);
```

```
int s = 1;

for(int j = 4;j >= 0;j--){

                s = s + j;
}

System.out.println(s);
```

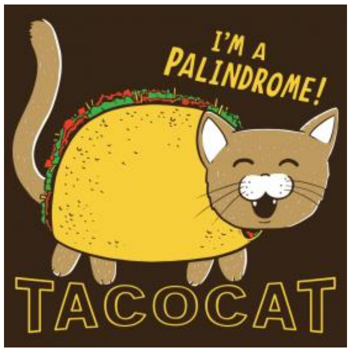
```
int i = 0;
for(int i = 10;i > 0;i--){

                i *=-3;
}

System.out.println(i)
```

## ☐ Write a for-loop to write a word backwards

Now that we understand a bit about for-loops, lets consider how a for-loop could be applied to write a word or phrase backwords. Consider the following word



| | | | | | | |
|---|---|---|---|---|---|---|
| T | A | C | O | C | A | T |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

To get the last letter, the second to the last letter, the third to the last letter, etc of the word we can use the following approach,

```
String word = "TACOCAT";

String wordL = word.length();
String lastLetter = word.substring(wordL – 1, wordL);//last letter
String secondToLast = word.substring(wordL – 2, wordL – 1);//second to last
String thirdToLast = word.substring(wordL – 3, wordL – 2);//third to last
```

| |
|---|
| Consider the word declared above. Write a for-loop that could be used to store the reversal of the word in String wordR. Then print wordR to the consol. |
| |

Now that you have the reversed word stored in its own variable, we can determine whether or not or word is a palindrome.

| |
|---|
| Write code that could be used to determine whether or not *word* and *wordR* are palindromes. If they are print to the consol “Palindrome!”, otherwise print “No!”. |
| |

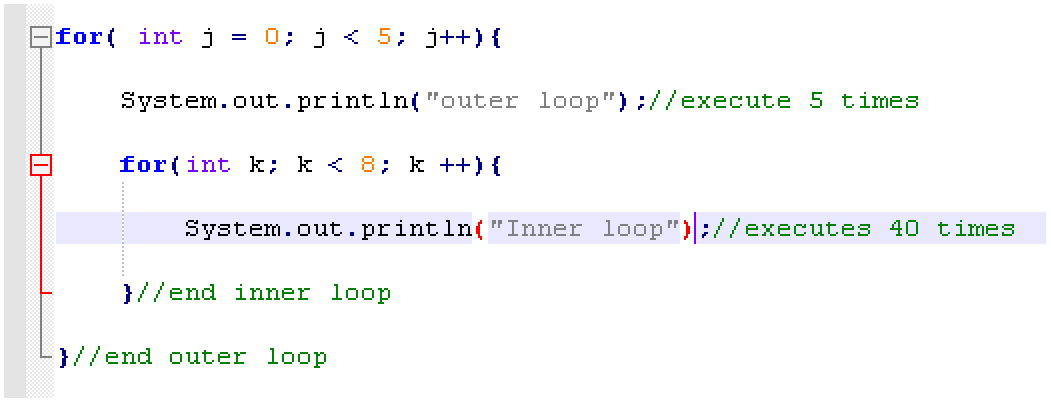
## ☐ Write a *for-loop* to print a number backwards

In a previous lesson you wrote code to print out a number backwards. The problem however was that the number could only be four digits. *for-loops* can be used to fix this issue and expand the problems to any number of digits (within the range of an integer of course)

| |
|---|
| Consider the number 1234567892 write a *for-loop* that could be applied to determine whether the number is the same whether it is written forwards or backwards. In other words, to check if the number is a palindrome! You rfor-loop must reverse the number then store the reversed number as an int. |
| |

## ☐ Write nested *for-loops* to create shapes on a grid

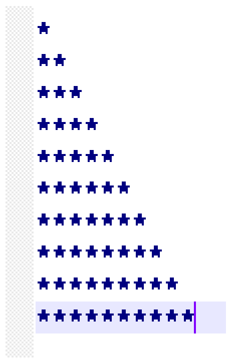
Nested loops is the term used when one loop is placed inside another as in the following example,



```
for( int j = 0; j < 5; j++){
    System.out.println("outer loop");//execute 5 times
    for(int k; k < 8; k ++){
        System.out.println("Inner loop");//executes 40 times
    }//end inner loop
}//end outer loop
```

In the example above, the inner loop executes 8 times for each of the five iterations of the outer loop.  Therefore, the code inside the inner loop will execute 40 times.

| Write nested for-loops that could be used to create the following patterns | |
|---|---|
| **********<br>**********<br>**********<br>**********<br>**********<br>**********<br>**********<br>**********<br>**********<br>********** | |
| **********<br>*********<br>********<br>*******<br>******<br>*****<br>****<br>***<br>**<br>* | |



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

## ☐ Receive Credit for this lab guide

Submit this portion of the lab to Pluska to receive credit for the lab guide. Once received, your completed code challenges will also be graded and will count towards your final lab grade.