

## Lab Assignment #3

### Exercise on While Loop:

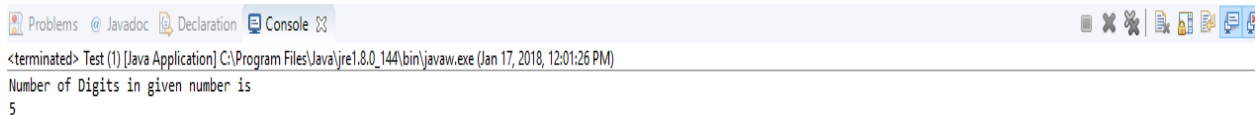
#### Objective-1 Use of While loop and operators

**Problem Statement:** Write a Java program that reads a positive integer and count the number of digits.

#### Source Code:

```
class ArmstrongExample{
    public static void main(String[] args) {
        int c=0,a,temp;
        int n=153;//It is the number to check armstrong
        temp=n;
        while(n>0)
        {
            a=n%10;
            n=n/10;
            c=c+1
        }
        System.out.println ("Number of digits in given number is"+c); }
}
```

#### Output:



The screenshot shows a Java IDE window with a console tab. The console output is as follows:

```
<terminated> Test (1) [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Jan 17, 2018, 12:01:26 PM)
Number of Digits in given number is
5
```

**Explanation:** In every iteration, when we take the remainder, it will give the least significant bit and store it in a variable. Then we divide the value by 10 to remove the least significant bit, since *int* will only store the integer part.

**Syntax:****while(Boolean expression)**

```
{  
    Statement1;  
    Statement2;  
    .....  
}
```

**Objective-2 Use of while loop**

**Problem Statement:** Write Java program to check if a number is Armstrong number or not using while loop.

**Note:** 371 is an Armstrong number because  $3*3*3+7*7*7+1*1*1=371$

**Objective-3 Use of While loop**

**Problem Statement:** Write a Java Program to print Fibonacci series up to a number n using While loop.

**Note:**  $\text{fib}(0)=0, \text{fib}(1)=1, \text{fib}(2)=1, \text{fib}(3)=2, \text{fib}(4)=3, \dots$

$\text{fib}(n)=\text{fib}(n-1)+\text{fib}(n-2)$  other than  $\text{fib}(0)$  and  $\text{fib}(1)$

# Exercise on do-while Loop

## Objective-4: Use of Do While Loop

**Problem Statement:** Write a Java Program to calculate factorial of numbers between 10 to 15 using do-while and while loop.

**Note:** Factorial (5)=5\*4\*3\*2\*1

### Source Code:

```
class Test{

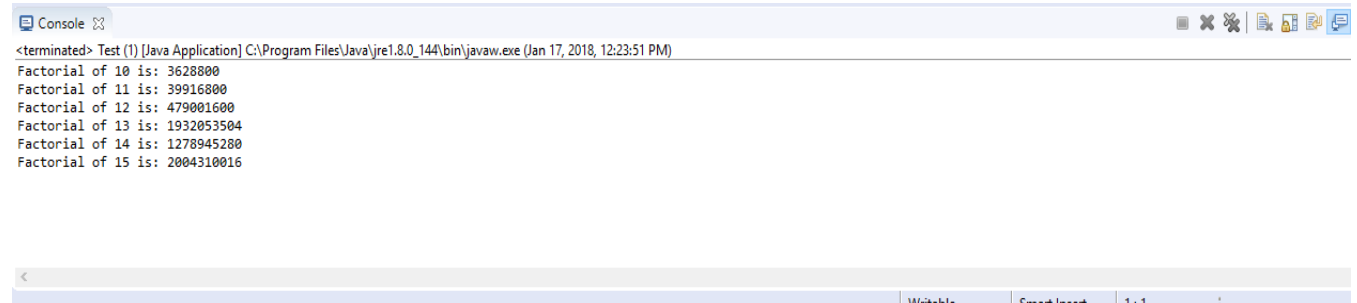
    public static void main(String args[]){
        int number=10;

        do
        {
            int i=1, fact=1;

            //It is the number to calculate factorial
            while(i<=number){
                fact=fact*i;
                i=i+1;
            }
            System.out.println("Factorial of "+number+" is: "+fact);
            number=number+1;
        }while(number<=15); }

}
```

### Output:



```
<terminated> Test (1) [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Jan 17, 2018, 12:23:51 PM)
Factorial of 10 is: 3628800
Factorial of 11 is: 39916800
Factorial of 12 is: 479001600
Factorial of 13 is: 1932053504
Factorial of 14 is: 1278945280
Factorial of 15 is: 2004310016
```

**Explanation:** factorial (n) = n\*n-1\*n-2\*.....\*1

### Syntax:

```
do{

    Statement1;

    Statement2;

    ....

} while(Boolean expression)
```

**Objective-5: Use of do while loop.**

**Problem Statement:** Write a java program to check whether number is strong number or not using do- while loop.

**Note:** *Strong number* is a special number whose sum of factorial of digits is equal to the original number.

$$145 = !1 + !4 + !5$$

$$!5 = 5 * 4 * 3 * 2 * 1 = 120$$

$$!4 = 4 * 3 * 2 * 1 = 24$$

$$!1 = 1$$

So 145 is a Strong Number.

**Objective-6: Use of do while loop.**

**Problem Statement:** Write a Java program to find all prime factors of a number using do-while loop.

**Note:**  $45 = 1 * 3 * 3 * 5$

So prime factors of 45 is 1,3,5

# Exercise on For Loop

## Objective-7: Use of for loop.

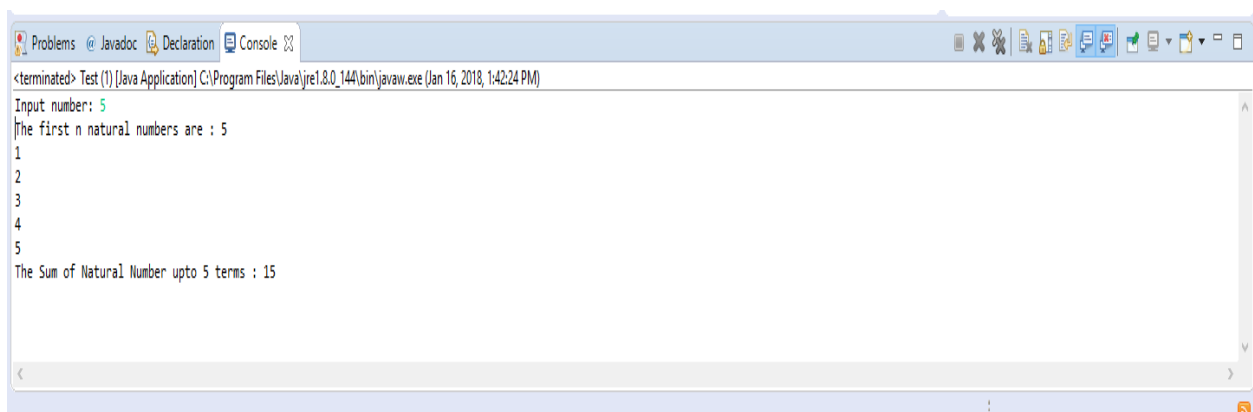
**Problem Statement:** Write a program in Java to display 20 terms of natural numbers and their sum.

**Problem Statement:** Program that takes input as n natural numbers and print their sum on output Screen.

### Source Code:

```
public class Exercise11 {  
    public static void main(String[] args)    {  
        inti, sum=0;    {  
        System.out.println("The first n natural numbers are :);  
        for(i=1;i<=20;i++) {  
        System.out.println(i);  
            sum+=i;}  
        System.out.println("The Sum of 20 Natural Number are" +sum);  
    }  
}
```

### Output



**Explanation:** Here we have used a simple for loop demo to print sum of natural numbers in its each iteration. For that inside for loop, the value of i in each iteration itself is the natural number starting from 1 to range(20), and we keep on adding into sum variable to get the final sum up-to 20 terms.

## Syntax

**for(initialization; boolean expression; update)**

```
{  
    Statement1;  
    Statement2;  
    ....  
}
```

### **Objective-8: Use of For Loop**

**Problem Statement:** Write a program in Java to display the 20 terms of odd natural number and their sum.

**Note:** For example 5 terms of odd natural numbers are-1,3,5,7,9

### **Objective-9: Use of for loop**

**Problem Statement:**Generate all the prime numbers between 1 and 1000 using for loop.

**Note:** A prime number is an integer greater than 1 that is evenly divisible by only 1 and itself. For example, 2, 3, 5, and 7 are prime numbers, but 4, 6, 8, and 9 are not.

### **Objective-10: Use of for loop**

**Problem Statement:** Write a program in Java to display the pattern like right angle triangle with a number.

```
1  
12  
123  
1234  
12345  
123456  
1234567  
12345678  
123456789  
12345678910
```

**Objective-11: Use of for loop**

**Problem Statement:** Write a Java program to display the number rhombus structure.

The expected output is

```
  1
 212
32123
4321234
543212345
65432123456
7654321234567
65432123456
543212345
4321234
32123
212
1
```

## Miscellaneous:

1. An interesting problem in number theory is sometimes called the “necklace problem.” This problem begins with two single-digit numbers. The next number is obtained by adding the first two numbers together and saving only the ones digit. This process is repeated until the “necklace” closes by returning to the original two numbers. For example, if the starting two numbers are 1 and 8, twelve steps are required to close the necklace: 1 8 9 7 6 3 9 2 1 3 4 7 1 8

Create a Necklace application that prompts the user for two single-digit integers and then displays the sequence and the number of steps taken. The application output should look similar to:

***Enter the first starting number: 1***

***Enter the second starting number: 8 1 8 9 7 6 3 9 2 1 3 4 7 1 8***

2. Let  $d(n)$  be defined as the sum of proper divisors of  $n$  (numbers less than  $n$  which divide evenly into  $n$ ).

If  $d(x) = y$  and  $d(y) = x$ , where  $x \neq y$ , then  $x$  and  $y$  are an amicable pair and each of  $a$  and  $b$  are called amicable numbers.

For example, the proper divisors of 220 are 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 and 110; therefore  $d(220) = 284$ . The proper divisors of 284 are 1, 2, 4, 71 and 142; so  $d(284) = 220$ . Evaluate all pair of the amicable numbers less than 1000.