**Day-01**

**20-02-2025**

**============**

**Loop Statements:**

**============**

-> Also called as "Iterative Statements".

-> Loop is a repetition.

-> When we need to execute the same amount of code again and again based on the condition, we can use "loop statements".

Ex: 5! ==> 5 X 4 X 3 X 2 X 1 ==> 120

n X n - 1 X n-2 X n-3 X n-4 ==> result

initialization = n

condition = (n >= 1)

update = n-1

Ex: 1 to 10

initialization = 1

condition = (s <= 10) or (s < 11)

update = s + 1

s < 11

{s = 1

print(s)} // 1

{s = s+1

print(s)} // 2

s = s+1

print(s) // 3

-> There are three types of loop statements:

1) for loop statement

2) while loop statement

3) do while loop statement

Here:

for, while and do ==> keywords

-> To work with loop statements:

there are three things are required.

1) Initialization

2) Condition

3) Update

-> Initialization: describe the starting point for loop.

-> Condition: until which value, the initialized value need to update

-> Update : describe that how the initialized value can change or update.

**Loop Entry Control Statements**

**====================**

-> When the loop body execution want to start, first the control can check the condition.

-> If the condition is "true", then it can enter into loop body for the execution.

-> If the condition is "false", then it cannot enter into loop body.

Ex: for, while

**Loop Exit Control Statements**

**===================**

-> Without evaluating the condition, the control can enter into the loop body for execution.

-> After the execution of the loop body for one time, the control can check the condition.

-> If the condition is "true", the control can stay in the loop body for the execution as repeat.

-> If the condition is "false", the control will come out from the loop body.

Ex: do while

**for loop:**

**======**

Syntax:

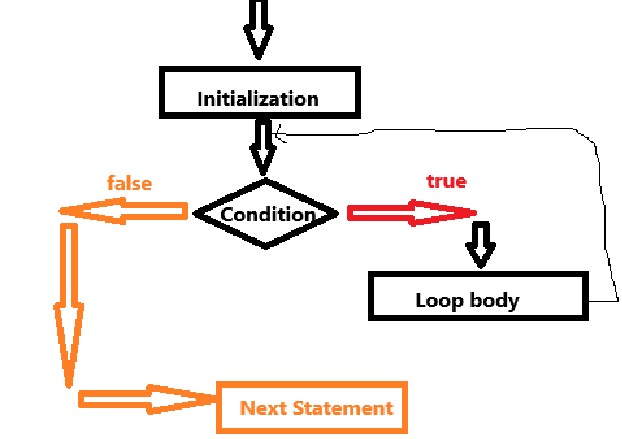
for(initialization; condition; update){

loop body

which includes one or more than one statements.

}

next statement;



Examples:

=========

1)

class ForLoopImplementation{

public static void main(String[] args)

{

for(int i = 1;i <= 5;i = i + 1)

{

System.out.println("i = "+i);

}

}

}

**Local Variable:**

**==========**

-> The variable which can be defined inside the method

and allowed to be access within the same method. That variable is called as "local variable".

**Block Variable:**

**==========**

-> The variable which can be defined within the block

and allowed to access within the same block is called as "block variable".

class ForLoopImplementation{

public static void main(String[] args)

{

//String name = "ravi";

for(int i = 1;i <= 5;i = i + 1)

{

System.out.println("i = "+i);

}

System.out.println("i = "+i);

//System.out.println(name);

}

//System.out.println(name);

}

==============================

class ForLoopImplementation{

public static void main(String[] args)

{

int i = 1;

for(;i <= 5;i = i + 1)

{

System.out.println("i = "+i);

}

System.out.println("i = "+i);

}

}

-> Initialization for the for loop can allowed to write above the for the loop or within the for loop also.

If we can write the above the for loop,

within the for loop, we need to avoid to write initialization but mandatory to write semi-colon.

-> condition of the for loop is always allowed to write within for only.

-> update statement of the for loop also allowed to write inside of the loop body.

Syntax:

initialization;

for(; condition ;){

loop body

update;

}

class ForLoopImplementation{

public static void main(String[] args)

{

int i = 1;

for(;i <= 5;)

{

System.out.println("i = "+i);

i = i + 1;

}

System.out.println("i = "+i);

}

}

=================================

class ForLoopImplementation{

public static void main(String[] args)

{

for(int I = 1;I <= 5; I = I + 1);

{

System.out.println(“Hello”);

System.out.println(“Today we are learning about for loop.”);

}

}

}

-> In the above code, the for head is acting as a statement because it was ended with semi-colon.

In this case this program can behave as normal java program that means it can make the sequential execution.

**Day\_02**

**21-02-2025**

Q: Is it possible to define the for loop by defining the condition in outside of the for header?

Syntax:

for(initialization; condition; update){

for loop body;

}

class ForPractice{

public static void main(String[] args)

{

int I = 1; // initialization

for(;😉{

if(I > 5){

break;

}

System.out.println(“Hello”);

I = I + 1;

}

}

}

class ForPractice{

public static void main(String[] args)

{

int I = 1; // initialization

for(;😉;{

if(I > 5){

break;

}

System.out.println(“Hello”);

I = I + 1;

}

}

}

Here:

break is defined in outside of the for loop, so, we can get “compile-time” error.

class ForWithTwoVariables{

public static void main(String[] args){

for(int I = 1,j = 10;i<=j; i++,--j){

System.out.println(i+j);

}

}

}

/\* WAP USING FOR LOOP IN JAVA TO FIND THE SUM OF N NATURAL NUMBERS.\*/

// using formula

// n\*(n+1)/2

import java.util.Scanner;

class SumOfNaturalNumbers{

public static void main(String[] args)

{

Scanner s = new Scanner(System.in);

System.out.println(“Enter number of natural numbers:”);

int n = s.nextInt();

int result = n \* (n+1)/2;

System.out.println(“The Sum of “+n+” number of natural numbers is = “+result);

}

}

/\* WAP USING FOR LOOP IN JAVA TO FIND THE SUM OF N NATURAL NUMBERS.\*/

// using loop

import java.util.Scanner;

class SumOfNaturalNumbers{

public static void main(String[] args)

{

Scanner s = new Scanner(System.in);

System.out.println(“Enter number of natural numbers:”);

int n = s.nextInt();

int sum = 0;

for(int I = 1;I <= n;i++){

sum = sum + I;

}

System.out.println(“The sum of natural numbers = “+sum);

}

}

// WAP TO FIND THE FACTORIAL OF THE NUMBER.

import java.util.Scanner;

class Factorial{

public static void main(String[] args)

{

Scanner s = new Scanner(System.in);

System.out.println(“Enter the value for finding the factorial:”);

int n = s.nextInt(); // 6

long fact = 1;

for(int I = n; I >= 1; I = I – 1){

fact = fact \* I; // 6 30 120 360 720 720

}

System.out.println(“The Factorial of the given number”+n+” is = “+fact);

}

}

// WAP TO FIND THE MULTIPLICATION TABLE OF THE GIVEN NUMBER

import java.util.Scanner;

class MultiplicationTable{

public static void main(String[] x)

{

Scanner s = new Scanner(System.in);

System.out.println(“Enter the value:”);

int n = s.nextInt();

int product = 1;

for(int I = 1;I <= 10;I = i+1)

{

product = n \* I;

System.out.println(n+” X “+i+” = “+product);

}

}

}

Assignment:

1. WAP FOR FINDING THE SUM OF EVEN NATURAL NUMBERS AND ODD NATURAL NUMBERS SEPARATELY.

**Day-03**

**22-02-2025**

**=================**

/\* WAP TO FIND THE SUM OF EVEN NATURAL NUMBERS AND ODD NUMBERS INDIVIDUALLY.

HINT:

NATURAL NUMBERS ==> 1 TO N

EVEN NATURAL NUMBERS ==> 2, 4, 6, 8, ...

ODD NATURAL NUMBERS ==> 1, 3, 5, 7, ....

\*/

// Solution-1:

import java.util.Scanner;

class SumEvenOdd{

public static void main(String[] args)

{

Scanner scan = new Scanner(System.in);

System.out.println("Enter the count for natural numbers:");

int n = scan.nextInt();

int seven = 0,sodd = 0;

System.out.println("Even Natural Numbers are:");

for(int i = 1; i <= n; i = i + 1){

if(i % 2 == 0){

// System.out.print(i+"\t");

seven = seven + i; // 2 6 12 ...

}

}

System.out.println("The sum of Even natural numbers = "+seven);

System.out.println("Odd Natural Numbers are:");

for(int i = 1; i <= n; i = i + 1){

if(i % 2 != 0){

// System.out.print(i+"\t");

sodd = sodd + i;

}

}

System.out.println("The Sum of Odd natural numbers = "+sodd);

}

}

// Solution-2:

import java.util.Scanner;

class SumEvenOdd{

public static void main(String[] args)

{

Scanner scan = new Scanner(System.in);

System.out.println("Enter the count for natural numbers:");

int n = scan.nextInt();

int seven = 0,sodd = 0;

for(int i = 1; i <= n; i = i + 1){

if(i % 2 == 0){

seven = seven + i; // 2 6 12 ...

}

else{

sodd = sodd + i;

}

}

System.out.println("The sum of Even natural numbers = "+seven);

System.out.println("The sum of Odd natural numbers = "+sodd);

}

}

=======================================

/\* WAP TO CHECK WHETHER THE GIVEN NUMBER IS PRIME NUMBER OR NOT

Prime number: When a number is having factors as 1 and itself called as "Prime number".

Least prime number is '2'

Even prime number is '2'

factor: if the given number can be divided with any other number, that other number should be the factor for the given number.

Ex: 4 % 2 == 0 ==> 2 is the factor for 4

\*/

Solution-1:

=======

import java.util.Scanner;

class PrimeNumberCheck{ // class begins

public static void main(String[] args){ // main starts

Scanner scan = new Scanner(System.in);

System.out.println("Enter a value to check the prime number or not:");

int n = scan.nextInt();

boolean result = primeNumber(n);

if(result == true){

System.out.println("The Given number "+n+" is a prime number");

}

else{

System.out.println("The Given number "+n+" is not a prime number");

}

} // main ends

public static boolean primeNumber(int num){ // udm starts

int count = 0;

for(int i = 2; i < num; i = i + 1){ // for starts

if(num % i == 0){ // if starts

count = count + 1;

}// if ends

} // for ends

if(count == 0){ // if starts

return true;

} // ends

else{ // else start

return false;

}//end

}// end

} // class ends

Solution-2:

========

import java.util.Scanner;

class PrimeNumberCheck{ // class begins

public static void main(String[] args){ // main starts

Scanner scan = new Scanner(System.in);

System.out.println("Enter a value to check the prime number or not:");

int n = scan.nextInt();

primeNumber(n);

} // main ends

public static void primeNumber(int num){ // udm starts

boolean flag = true;

for(int i = 2; i <= num/2; i++){

if(num % i == 0){

flag = false;

break;

}

}

if(flag == true){

System.out.println("The Given number "+num +" is Prime number");

}

else{

System.out.println("The Given number "+num+" is not Prime number");

}

}// end

} // class ends

Solution-3:

========

import java.util.Scanner;

class PrimeNumberCheck{ // class begins

public static void main(String[] args){ // main starts

Scanner scan = new Scanner(System.in);

System.out.println("Enter a value to check the prime number or not:");

int n = scan.nextInt();

primeNumber(n);

} // main ends

public static void primeNumber(int num){ // udm starts

boolean flag = true;

for(int i = 2; i <= Math.sqrt(num); i++){

if(num % i == 0){

flag = false;

break;

}

}

if(flag == true){

System.out.println("The Given number "+num +" is Prime number");

}

else{

System.out.println("The Given number "+num+" is not Prime number");

}

}// end

} // class ends

Assignment:

===========

1) WAP TO CHECK WHETHER THE GIVEN NUMBER IS PERFECT NUMBER OR NOT.

The sum of all the factors by excluding itself of the given number is equals to given number is called as "Perfect number".

Hint:

=====

6 ==> 1,2,3,6

1 + 2 + 3 = 6

**Day-04**

**24-02-2025**

**===============**

// WAP TO CHECK WHETHER THE GIVEN NUMBER IS PERFECT NUMBER OR NOT.

Hint: The sum of factors of the number (excluding itself) is equals to given number is called as "Perfect Number".

Solution-1:

=======

import java.util.Scanner;

class PerfectNumber{

public static void main(String[] args)

{

Scanner s = new Scanner(System.in);

System.out.println("Enter the number:");

int number = s.nextInt();

perfectNumberCheck(number);

}

public static void perfectNumberCheck(int n)

{

int sumFactors = 0;

for(int i = 1; i<n; i = i+1)

{

if(n % i == 0){

sumFactors = sumFactors + i;

}

}

if(sumFactors == n){

System.out.println("The Given number "+n+" is perfect number");

}

else{

System.out.println("The Given number "+n+" is not perfect number");

}

}

}

Solution-2:

=======

import java.util.Scanner;

class PerfectNumber{

public static void main(String[] args)

{

Scanner s = new Scanner(System.in);

System.out.println("Enter the number:");

int number = s.nextInt();

perfectNumberCheck(number);

}

public static void perfectNumberCheck(int n)

{

int sumFactors = 0;

for(int i = 1; i<=n/2; i = i+1)

{

if(n % i == 0){

sumFactors = sumFactors + i;

}

}

if(sumFactors == n){

System.out.println("The Given number "+n+" is perfect number");

}

else{

System.out.println("The Given number "+n+" is not perfect number");

}

}

}

/\* WAP TO FIND AND DISPLAY FIBONACCI SERIES UPTO THE GIVEN LENGTH.

HINT:

=====

In Fibonacci series,

the first two terms are:

term1 = 0

term2 = 1

term3 = term1 + term2

0 1 1 2 3 5 8 13 ....

\*/

import java.util.Scanner;

class FibonacciSeries{

public static void main(String[] args)

{

Scanner s = new Scanner(System.in);

System.out.println("Enter the total length of the Fibonacci series:");

int length = s.nextInt();

fibonacci(length);

}

public static void fibonacci(int l)

{

int term1 = 0, term2 = 1;

if(l == 1){

System.out.print(term1 + "\t");

}

else if(l == 2){

System.out.print(term1 + "\t");

System.out.print(term2 + "\t");

}

else{

System.out.print(term1 + "\t");

System.out.print(term2 + "\t");

for(int t = 3; t <= l; t = t + 1)

{

int term3 = term1 + term2;

term1 = term2;

term2 = term3;

System.out.print(term3+"\t");

}

}

}

}

**General Exercise:**

**============**

Problem-1:

=======

class Dummy{

public static void main(String[] args)

{

for(int i = 1; i <= 4; i = i + 1){

System.out.println(i);

}

for(i = 5; i <= 10; i = i + 1){

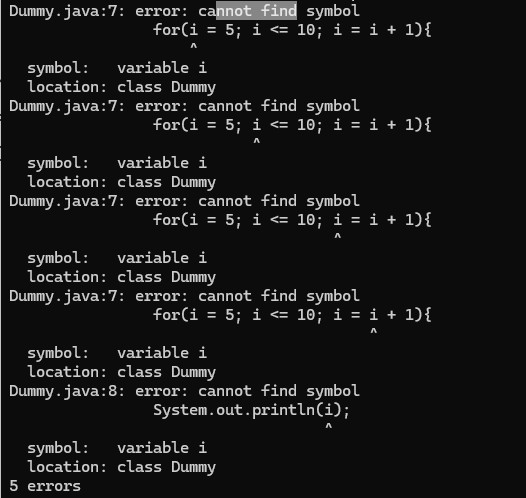
System.out.println(i);

}

}

}

Error: Compile-time error



Problem-2:

========

class Dummy{

public static void main(String[] args)

{

for(double price = 1.0; price <= 5.0; price++)

{

System.out.println(price \* 2);

}

}

}

Problem-3:

=======

class Dummy{

public static void main(String[] args)

{

for(char ch = 'a'; ch <= 'f'; ch++)

{

System.out.print(ch+"\t");

}

System.out.println();

}

}

Problem-4:

=======

class Dummy{

public static void main(String[] args)

{

for(boolean i = true; i <= 4; i = i + 1)

{

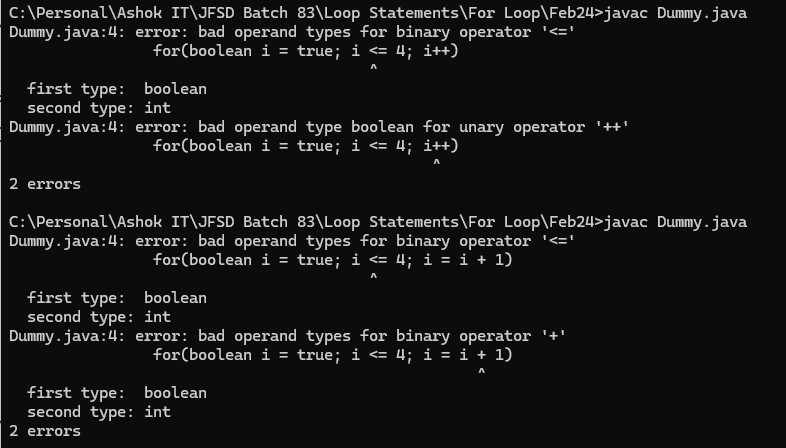
System.out.println(i);

}

}

}

Output: Compile-Time Error



Assignment:

========

1) WAP in java to print all composite numbers from the given range.

2) WAP in java to print the sum of all composite numbers from the given range.

**Day-05**

**25-02-2025**

**==============**

**Nested for loop:**

**===========**

Syntax:

for(initialization; condition; update){ // outer for loop

// outer loop body

for(initialization; condition; update){ // inner for loop

// inner loop body

}

}

-> The control can execute the initialized value of the outer for loop

That value can be checked with condition of the outer for loop.

If the condition "true", then the control can enter into the outer for loop body.

-> When the control in the outer for loop body,

the initialized value of inner for loop can execute

and that value can check with condition of inner for loop.

If the condition "true", then the control can execute the inner loop body statements.

And then the control can update the initialized value of inner for loop.

Then check with the condition of inner for loop again.

If it is "true", then again the inner for loop body statements can execute and then update the initialized value. This can be repeat until the inner for loop condition reaches to "false".

Once the condition of inner for loop is "false", then the update of the outer for loop can execute.

-> The updated value of the outer for loop can check with the condition of the outer for loop,

if it is "true" then: outer for loop body can execute (inner for loop as same above) and again execute the update of the outer for loop. This can be repeat until the condition of outer for loop reaches to "false".

class NestedForLoop{

public static void main(String[] args)

{

for(int i = 1; i <= 3; i = i + 1)

{

for(int j = 1;j <= 3;j = j + 1)

{

System.out.println(j);

}

}

}

}

// HOW MANY NUMBER OF ITERATIONS ARE POSSIBLE IN THE BELOW CODE:

Ans: 6

class NestedForLoop{

public static void main(String[] args)

{

for(int i = 1; i <= 3; i = i + 1)

{

for(int j = 1;j <= 2;j = j + 1)

{

System.out.println("Hi");

}

}

}

}

// HOW MANY NUMBER OF ITERATIONS ARE POSSIBLE IN THE BELOW CODE:

Ans: 18

class NestedForLoop{

public static void main(String[] args)

{

for(int i = 1; i <= 3; i++)

{

for(int j = 1; j <=3; j++)

{

for(int k = 1; k <= 2; k++)

{

System.out.println("Java");

}

}

}

}

}

Assignments:

============

1) Calculate the total number of iterations

class NestedForLoop{

public static void main(String[] args)

{

for(int i = 1; i <= 3; i++)

{

for(int j = 1; j < 3; j++)

{

for(int k = 1; k <= 2; k++)

{

System.out.println("Java");

}

}

}

}

}

2) Calculate number of iterations

class NestedForLoop{

public static void main(String[] args)

{

for(int i = 1; i <= 3; i++)

{

for(int j = 1; j <= i; j++)

{

for(int k = 1; k <= 2; k++)

{

System.out.println("Java");

}

}

}

}

}

3) Calculate number of iterations

class NestedForLoop{

public static void main(String[] args)

{

for(int i = 1; i <= 3; i++)

{

for(int j = 1; j <= i; j++)

{

for(int k = 1; k <= j; k++)

{

System.out.println("Java");

}

}

}

}

}

// WAP TO PRINT ALL PRIME NUMBERS BETWEEN THE GIVEN RANGE.

/\*

boolean flag = true;

for(int i = 1001; i <= 1100; i++){

for(int j = 2; j <= Math.sqrt(i); j++)

{

if(i % j == 0){

flag = false;

break;

}

}

}

\*/

import java.util.Scanner;

class PrimeNumbersFromRange{

public static void primeNumbers(int x, int y)

{

boolean flag = true;

for(int i = x; i <= y; i++)

{

for(int j = 2; j <= Math.sqrt(i);j++)

{

if(i % j == 0)

{

flag = false;

break;

}

}

if(flag == true){

System.out.print(i+"\t");

}

}

}

public static void main(String[] args)

{

Scanner s = new Scanner(System.in);

System.out.println("Enter start value:");

int x = s.nextInt();

System.out.println("Enter last value:");

int y = s.nextInt();

primeNumbers(x,y);

}

}