Loops Flow Control →

Loops or Iterations are part of Control Statements.

Loops are used to repeatedly execute a block of statement till condition is true or satisfied.

Types of loops in Java →

(1) Entry Controlled loop →
In this type of loop, condition is checked before execution of loop start or take place.
type of Entry Controlled loop →
(i) for loop

```
for (Initialization; Condition;
        Increment Decrement)
{
// Block of Statements
}
```

(ii) While loop.

```
Initialization;
while (Condition) {
    // Statements
    // Statements and code
    Increment / Decrement;
}
```

Exit Controlled Loop =>

In this Loop is executed once even if condition evaluates to be false.

do while Loop =>

In this loop, the loop executes once even if condition evaluates to be false and then normal loop execution takes place.

```
Initialization;
do {
  // Statements of Code
  Increment / Decrement;
} (Condition);
```

Print Hello World 100 times =>

```java
class PrintHW100 {
    public static void main(String[] args) {
        int i = 1;
        while (i <= 100) {
            System.out.println("Hello World! " + i);
            i++;
        }
    }
}
```

Print Numbers from 1 to 10

Output →
1 2 3 4 5 6 7 8 9 10

```java
class Print1to10 {
    public static void main(String[] args) {
        int i = 1;
        while(i <= 10){
            System.out.print(i+" ");
            i++;
        }
    }
}
```

Print Number from 1 to N →

```java
import java.util.Scanner;

class Print1toN {
    public static void main(String[] args) {
        System.out.println("Enter the Range = ");
        Scanner input = new Scanner(System.in);
        int N = input.nextInt();
        System.out.println("Printing 1 to "+N);
        int i = 1;
        while (i <= N) {
            System.out.println(i);
            i++;
        }
        input.close();
    }
}
```

Program to Calculate Sum of N Natural No in Java →

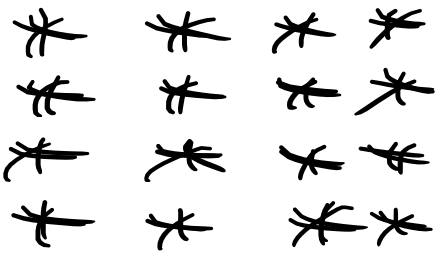
```java
import java.util.Scanner;

class PrintSumofNNaturalNo {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the Range to calculate the Sum of N Natural Numbers = ");
        int Range = input.nextInt();
        int sum = 0, i = 1;
        while(i <= Range){
            sum += i;
            i++;
        }
        System.out.println("Sum of First N Natural Numbers is = "+sum);
        input.close();
    }
}
```

for Loop →
Entry Controlled loop.
Best loops to fast writing of loop

```
for(Initialization ; Condition ;
        Increment/Decrement ){
// Block of for
}
```

Program to Print →



```java
class Pattern {
    public static void main(String[] args) {
        for(int i = 1 ; i <= 4 ; i++){
            for (int j = 1; j<=4; j++){
                System.out.print("* ");
            }
            System.out.println("");
        }
    }
}
```

Print Reverse of a Number →

```
while (num != 0) {
    rev = rev * 10 + num % 10;
    num = num / 10;
}
```

```java
import java.util.Scanner;

class ReverseofNumber {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the value of Number = ");
        int Number = input.nextInt();
        int rev = 0;
        while (Number != 0) {
            rev = rev * 10 + Number % 10;
            Number = Number / 10;
        }
        System.out.println("Reverse of Number is = "+rev);
        input.close();
    }
}
```

- ★ To get last Digit of No → `No%10`
- ★ To remove last Digit of No → `No/10`

Do while loop →

```
do {
    //do something
} while (condition);
```

Similar to while loop but it executes once even if condition evaluates to be false.

`break;` → It is used to break the execution of loop based on condition.

`break;` → Break Loop Execution

Program → Keep entering numbers till user enters a multiple of 10.

```java
// Keep entering numbers till user enters a multiple of 10.

import java.util.Scanner;

class Multipleof10 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the Value = ");

        while(true){
            int N = input.nextInt();
            if(N % 10 == 0){
                System.out.println("Number "+N+" is a Multiple of 10.");
                break;
            }
            System.out.println("Number you entered is = "+N);
        }

        input.close();
    }
}
```

`break;` Jump Control Statement is used to switch control at end of loop leads to termination of loop.

Continue — Jump Control → to skip an Iteration based on condition

★ Continue jump statement is used to shift the control of flow at starting of the loop or block

```java
//Display all numbers entered by user except multipes of 10

import java.util.Scanner;

class DisplayExceptMultipleof10 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        while (true) {
            System.out.print("Enter the Number = ");
            int N = input.nextInt();

            if (N % 10 == 0) {
                continue;
            }

            System.out.println("Number is = "+N);
        }
    }
}
```

Jump statements in the C programming language allow programmers to alter the normal flow of execution in their code. These statements enable efficient control flow by providing options to break out of loops, skip iterations, transfer control to specific points, and terminate functions.

Check if a No is Prime or Not →
Prime has only two factors 1 and itself.

Prime → 2, 3, 5, 7, 11

Non Prime are called Composite Number

```java
//Check if a number is Prime or Not!

import java.util.Scanner;

class PrimeorNot {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Program to check if a Number is Prime or Not!");
        System.out.print("Enter the Value of Number = ");
        boolean isPrime = true;
        int N = input.nextInt();
        //Prime Logic ->
        for(int i = 2 ; i < N ; i++){
            if(N % i == 0){
                isPrime = false;
                break;
            }
        }
        // Printing Prime Logic ->
        if (isPrime) {
            System.out.println("Number is a Prime Number = "+N);
        }else{
            System.out.println("Number is Not a Prime Number = "+N);
        }

        input.close();
    }
}
```

Prime → 2 to N − 1

N % i == 0

Optimized way → Prime

Number expressed $N = a * b$

$N = \sqrt{N} * \sqrt{N}$ True for every N

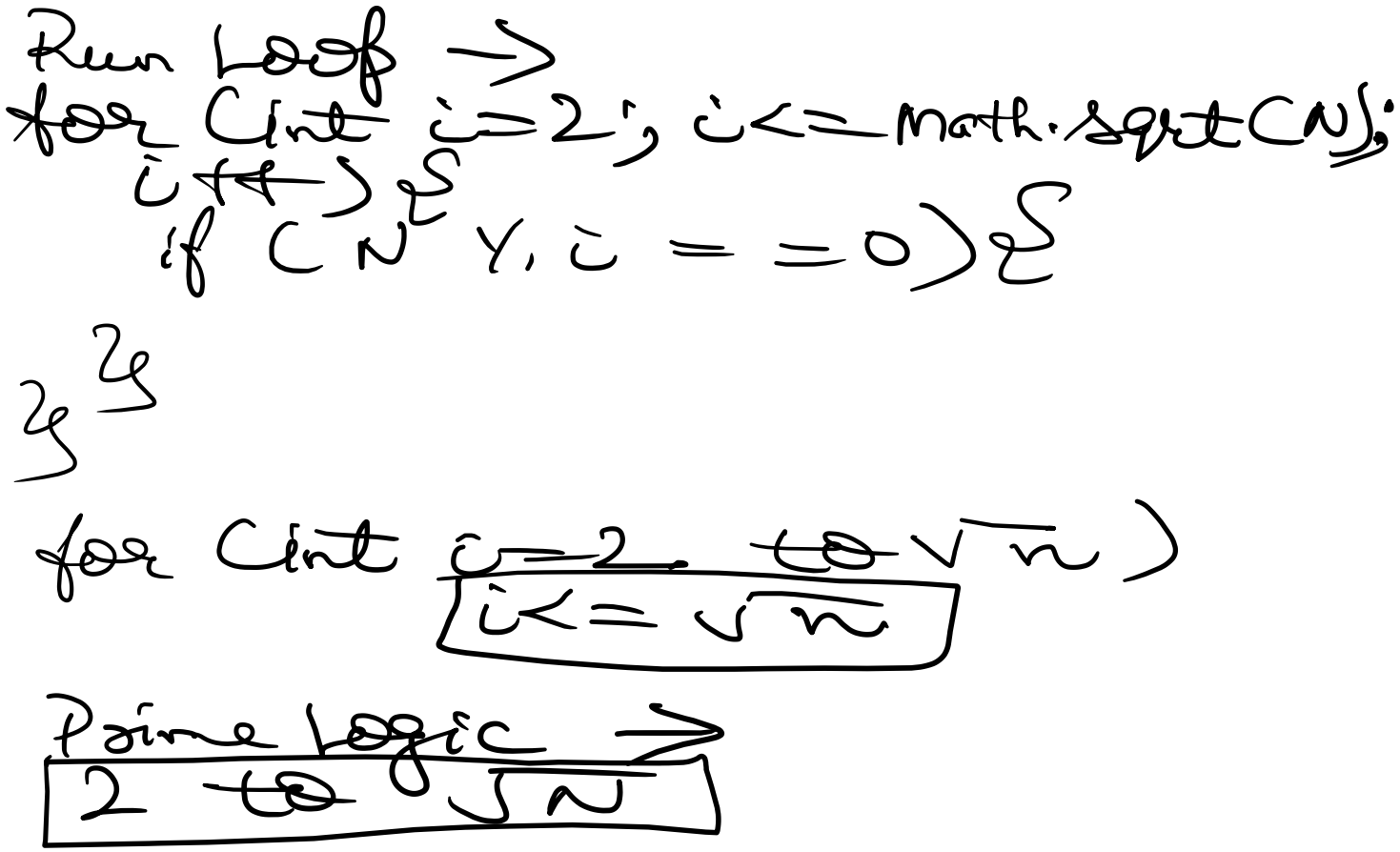
```
Input N
boolean isPrime = true;
if (N<2) {
    Sout ("Not Prime");
}
for(int i=2; i<N; i++) {
    if (N % i == 0) {
        isPrime = false;
    }
}
if (isPrime) { Sout ("Prime"); }
else { Sout ("not prime"); }
```

Prime No → Optimization →

Incase of 12 →

1 * 12
2 * 6
3 * 4 } Unique

4 * 3
6 * 2
12 * 1 } Repeat




```java
//Java Program to use Optimized Approach to solve Prime Number Problem

import java.util.Scanner;

class OptimizedPrime {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the Value of Number = ");
        int N = input.nextInt();
        boolean isPrime = true;
        // Edge Cases
        if(N < 2){
            isPrime = false;
        }
        // N > 2 Cases
        for(int i = 2; i <= Math.sqrt(N) + 1; i++){
            if(N % i == 0){
                isPrime = false;
                break; // No need to continue checking
            }
        }

        // Printing the Output ->
        if (isPrime) {
            System.out.println("Number is a Prime Number = "+N);
        }else{
            System.out.println("Number is Not a Prime Number = "+N);
        }

        input.close();
```