Loops one shot

Loops Introduction:

If we want to perform a similar operation repeatedly then we will use loop there.

Example : If we want to listen a same song repeatedly then we will loop it .

If we want to print a name say "NAZEER" some 1000 times we cannot copy and paste this 1000 times right so, loops have many applications in our day-to-day life.

Types of loops :- 1) for loop 2) while loop 3) do while loop.

While Loop:

while loop, this will be useful mostly when we know the termination condition .

Example :- If I say drive the bike till fuel tank in not emptied , here I am giving a termination condition that you should stop when fuel tank is empty so , here we will use while loop.

Syntax:

```
//loop iterator initialization
while(condition) {
    //statements to execute in while loop


    //loop iterator increment or decrement part
}
```

#Note: It is mandatory that at a point the condition should fail else it will be an infinite loop , which keeps on running until the memory gets filled.

As a good programmer it is our duty to avoid these infinite loops in our program.

Program: Write code in java to print numbers from 1 to n using while loop.

```java
// Write code in java to print numbers from 1 to n using while loop.
// sample input: 6
// sample output: 1 2 3 4 5 6
// Exaplanation: 1 2 3 4 5 6 are the numbers from 1 to 6.

import java.util.*;

public class loops_003_print_one_to_n_numbers {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in); // creating object to scanner class to take inputs.
        System.out.println("Enter value of n ");
        int n = sc.nextInt(); // taking input from the user.
        int i = 1; // loop iterator initialization starts with 1 as we are printing from 1 to n.
        while (i <= n) { // condition is iterate till i is less then or equal to n
            System.out.print(i + " "); // at each iteration output the value of iterator i.e i
            i++; // incrementing the iterator.
        }
        sc.close(); // closing the scanner object to prevent data leak.
    }
}
```

Sample output:

```
Enter value of n
6
1 2 3 4 5 6
```

Code available here

Program: Write code in java to print sum of first n natural numbers.

```java
// Write code in java to print sum of first n natural numbers.
// sample input: 6
// sample output: 21
// Exaplanation: 1+2+3+4+5+6 = 21

import java.util.*;

public class loops_004_sumOf_n_naturalNumbers {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in); // creating object to
scanner class to take inputs.
        System.out.println("Enter value of n ");
        int n = sc.nextInt(); // taking input from the user.
        int sum = 0, i = 1; // loop iterator initialization starts
with 1 as we are finding sum from 1 to n.
                            // Sum is initialized to 0 as we are
adding values to it.
        while (i <= n) { // condition is iterate till i is less
then or equal to n
            sum += i; // adding the value of i to sum
            i++; // incrementing the iterator.
        }
        System.out.println("Sum of first " + n + " natural numbers
is : " + sum); // printing the sum of first n natural

                // numbers.
        sc.close(); // closing the scanner object to prevent data
leak.
    }
}
```

Sample output:

```
Enter value of n
6
Sum of first 6 natural numbers is : 21
```

Code available here

For Loop:

for loop , the most used loop in programming . This loop is almost similar to while loop but here we will be knowing how many iterations we need to perform before termination

Syntax:

```
for(initialization; condition ; updation) {
    // statements to execute inside loop
}
```

Example:

```
for(int i=1; i<10; i++) {
    System.out.println(i);
}
```

In the above example , we are initializing a variable i with value 1 , then we are checking if i is less than 10 , if it is true then we are printing the value of i and then incrementing the value of i by 1 . This process will continue until the condition becomes false . In this case , the loop will run for 10 times and the output will be 1 to 9.

We can also write the above example as below , notice that we are not initializing the variable i in the for loop , we are initializing it before the loop. We can also write the updation part inside the loop.This is just similar to while loop.

```
int i=1; // initialization
for(; i< 10; ){
    System.out.println(i);
```

```
        i++; // updation
    }
```

Program: Write code in java to reverse a number using for loop.

```java
// Write a program in java to reverse a number using for loop.
// sample input: 1234
// sample output: 4321
// Exaplanation: 1234 is reversed to 4321.

import java.util.*;

public class loops_006_reverse_a_number {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in); // creating object to scanner class to take inputs.
        System.out.println("Enter a number ");
        int n = sc.nextInt(); // taking input from the user.
        int rev = 0; // initializing rev to 0 to store the reverse of the number.
        while (n > 0) { // condition is iterate till n is greater than 0
            rev = rev * 10 + n % 10; // adding the last digit of n to rev
            n = n / 10; // removing the last digit of n
        }
        System.out.println("Reverse of the number is : " + rev); // printing the reverse of the number.
        sc.close(); // closing the scanner object to prevent data leak.
    }
}
```

Sample output:

```
Enter a number
1234
Reverse of the number is : 4321
```

Code available here

do while loop is used rarely in any program. It is used when we want to execute the loop at least once. The syntax of do while loop is given below:

Syntax:

```
do
{
    //statements to be executed
    //increment/decrement;
} while(condition);
```

Example:

```
    int i = 1;
    do{
        System.out.println(i);
        i++;
    } while(i<=10);
```

Output: 1 2 3 4 5 6 7 8 9 10

In the above example, the loop will be executed at least once because the condition is checked at the end of the loop.

break and continue:

Break and continue statements are used to alter the flow of a normal loop. break is used to exit a loop, while continue is used to skip the current iteration of the loop and continue with the next iteration.

Break:

The break statement is used to exit a loop. The break statement is used inside loops. The break statement breaks the loop and continues executing the code after the loop (if any):

Example:

take input from the user until he enters a specific type of number , say 0

then we can use break statement to exit the loop

* It's same break statement used in switch case statements , if any case is true then break statement is used to exit the switch case.

Continue:

The continue statement is used to skip the current iteration of the loop and continue with the next iteration. The continue statement is used inside loops. The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

Example:

take input from the user repeatedly and print all except multiples of 10.

Note: Break statement is the most used statement in loops. Continue is not used that much, because the same thing can be done by using if-else statement.

Program: Write a program in java that asks the user to enter a number repeatedly and print them, until the user enters a multiple of 10.

```java
// write a program in java that asks the user to enter a number repeatedly and print them, until the user enters a multiple of 10.
// sample input: 1 2 3 4 5 6 7 8 9 121 11 12 13 14 15 16 17 18 19 20
// sample output: 20
// Explanation: 20 is the multiple of 10 so the loop breaks and the program ends. Till 19 the loop iterates and prints the numbers as these are not the multiple of 10.

import java.util.*;

public class loops_009_break_example {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in); // creating object to scanner class to take inputs.
        while (true) { // this loop will iterate infinitely as the condition is always true.
            System.out.println("Enter a number: ");
            int n = sc.nextInt(); // taking input from the user.
            if (n % 10 == 0) { // checking if the number is multiple of 10 or not.
                break; // break the loop if it's multiple of 10.
            } else {
                System.out.println(n); // if the number is not a multiple of 10 then print the number.
            }
        }
        sc.close(); // closing the scanner object to prevent data leak.
    }
}
```

Sample output:

```
Enter a number:
1
1
Enter a number:
2
2
Enter a number:
3
3
Enter a number:
10
```

Code available here

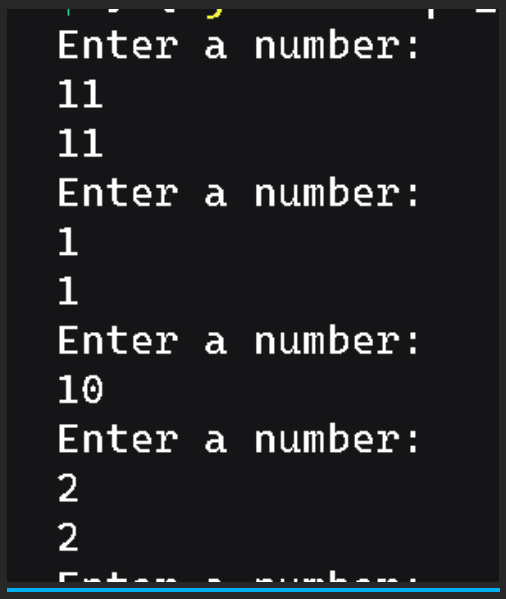
Program: Write code in java to print all numbers except those which are divisible by 10.

```java
// Write a program to print all the numbers except the numbers which are divisible by 10.
// sample input: 1 2 3 4 5 6 7 8 9 121 10 11 12 13 14 15 16 17 18 19 20
// sample output: 1 2 3 4 5 6 7 8 9 121 11 13 14 15 16 17 18 19
// Explanation: 10 , 20 are the multiple of 10 so the loop continues and these are not printed.

import java.util.*;

public class loops_010_continue_example {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in); // creating object to scanner class to take inputs.
        while (true) { // this loop will iterate infinitely as the condition is always true.
            System.out.println("Enter a number: ");
            int n = sc.nextInt(); // taking input from the user.
            if (n % 10 == 0) { // checking if the number is multiple of 10 or not.
                continue; // continue the loop if it's multiple of 10.
            } else {
                System.out.println(n); // if the number is not a multiple of 10 then print the number.
            }
        }
    }
}
```

Sample output:



Code available here

Program: Write code in java to check if a given number is prime or not.

```
// Write a program in java to check if a number is prime or not.
// sample input: 3
// sample output: 3 is a prime number.
// Explanation: 3 is only divisible by 1 and 3 so it's a prime
number.

// Approach:
/*
    Prime number is a number that is greater than 1 and divided by
1 or itself only.
    In other words, prime numbers can't be divided by other
numbers than itself or 1.
    For example 2, 3, 5, 7, 11, 13, 17.... are the prime numbers.
    Note: 0 and 1 are not prime numbers.

    Brute Force Approach:
    A simple solution is to iterate through all numbers starting
from 2 to n-1 and for every number check if it divides n.
    If we find any number that divides, we return false.
    else we return true.
    Time Complexity: O(n)
```

```java
    4 = {1*4 , 2*2 , 4*1}           sqrt(4) = 2;
    6 = {1*6 , 2*3 , 3*2 , 6*1}     sqrt(6) = 2.44 = 2;
    8 = {1*8 , 2*4 , 4*2 , 8*1}     sqrt(8) = 2.82 = 2;
    9 = {1*9 , 3*3 , 9*1}           sqrt(9) = 3;

    Note that the highest divisor of a number n can be sqrt(n),
also after sqrt(n) the pairs are repeating in interchanged order.
n = √n * √n

    Optimized Approach:
    A better approach is to check if a number is divisible by any
number from 2 to square root of that number.
    If we find any number that divides, we return false.
    else we return true.
    Time Complexity: O(sqrt(n))

 */

import java.util.*;

public class loops_011_isPrime {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in); // creating object to
scanner class to take inputs.
        System.out.println("Enter a number: ");
        int n = sc.nextInt(); // taking input from the user.
        boolean isPrime = true; // initializing a boolean variable
to true.
        if (n < 2) { // checking if the number is less than 2 or
not.
            isPrime = false; // if the number is less than 2 then
it's not a prime number.
        }
        for (int i = 2; i <= Math.sqrt(n); i++) { // iterating
from 2 to sqrt(n).
            if (n % i == 0) { // checking if the number is
divisible by any number from 2 to sqrt(n).
                isPrime = false; // if the number is divisible by
any number from 2 to sqrt(n) then it's not a
                                 // prime number.
                break; // break the loop if the number is
divisible by any number from 2 to sqrt(n).
            }
        }
        if (isPrime) { // checking if the number is prime or not.
```

```
            System.out.println(n + " is a prime number."); // if
the number is prime then print the number is prime.
        } else {
            System.out.println(n + " is not a prime number."); //
if the number is not prime then print the number is
                                                                //
not prime.
        }
        sc.close(); // closing the scanner class object to prevent
data leak.
    }
}
```

Sample output:

```
Enter a number:
4
4 is not a prime number.
```

Code available here

Program: Write code in java to print multiplication table of a given number.

```
// Program to print multiplication table of a number entered by a
user in pretty form
// sample input: 2
// sample output: 2 x 1 = 2
//                2 x 2 = 4
//                2 x 3 = 6
//                2 x 4 = 8
//                2 x 5 = 10
//                2 x 6 = 12
//                2 x 7 = 14
//                2 x 8 = 16
//                2 x 9 = 18
//                2 x 10 = 20
// Explanation: 2 is the number entered by the user. The program
will print the multiplication table of 2 till 10.

import java.util.*;

public class loops_012_multiplication_table {
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in); // creating object to
scanner class to take inputs.
        System.out.println("Enter a number to print it's
multiplication table: "); // asking user to enter a number.
        int n = sc.nextInt(); // taking input from the user.
        for (int i = 1; i <= 10; i++) { // iterating the loop from
1 to 10.
            System.out.println(n + " x " + i + " = " + n * i); //
printing the multiplication table.
        }
        sc.close(); // closing the scanner object to prevent data
leak.
    }
}
```

Sample output:

```
Enter a number to print it's multiplication table:
2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

Code available here

Program: Write code in java to print factorial of a given number.

```
// Program to print factorial of a number.
// sample input: 5
// sample output: Factorial of 5 is 120
// Explanation: 5! = 1 x 2 x 3 x 4 x 5 = 120.

// Note: 0! = 1 do not confuse with ! operator. ! is a logical
operator which is used to reverse the logical state of its
```

```
operand. If a condition is true then Logical NOT operator will
make false. In mathematics and computer science, the factorial of
a non-negative integer n, denoted by n!, is the product of all
positive integers less than or equal to n. For example, 5! = 5 x 4
x 3 x 2 x 1 = 120.

/*
    Approach:
    1. Take a variable to store the factorial of a number, let's
say factorial = 1.
    2. Run a loop from 1 to n.
    3. Multiply factorial with i.
    4. Print the factorial.

    Example: for 5!
    i    factorial
    1    1 x 1 = 1
    2    1 x 2 = 2
    3    2 x 3 = 6
    4    6 x 4 = 24
    5    24 x 5 = 120

 */

import java.util.*;

public class loops_013_factorial {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in); // creating object to
scanner class to take inputs.
        System.out.println("Enter a number to print it's
factorial: "); // asking user to enter a number.
        int n = sc.nextInt(); // taking input from the user.
        int factorial = 1; // initializing factorial variable to
1.
        for (int i = 1; i <= n; i++) { // iterating the loop from
1 to n.
            factorial *= i; // multiplying factorial with i.
        }
        System.out.println("Factorial of " + n + " is " +
factorial); // printing the factorial.
        sc.close(); // closing the scanner object to prevent data
leak.
    }
}
```

Sample output:

```
Enter a number to print it's factorial:
5
Factorial of 5 is 120
```

Code available here