Lecture 21

Looping Statements

for loop

- Sometimes in our program we have to execute some set of instructions many times to determine the result & this process of repeating the same sequence of steps is referred to iteration or looping.
- In looping a sequence of statements are executed until some conditions for termination of the loop are satisfied.
- Loops in java are essential for executing a block of code repeatedly based on a condition. (Real life example :- Buy something from Supermark

Need :- Lets print Jenny's lectures 5 times.

One way is use 5 System.out.println() statement. but if I say print Jenny's lectures 500 times, then ?

↓

Solution is use loops

- Loops help avoid writing the same block of code multiple times. We can automate the task using loops
- Loops allow us to iterate over arrays, list or any data structure enabling easy data manipulation.
- Loops handle tasks where the number of iterations is not fixed or known beforehand. The loop can continue running until a condition is met.

eg:- Reading user input until the user type "exit".

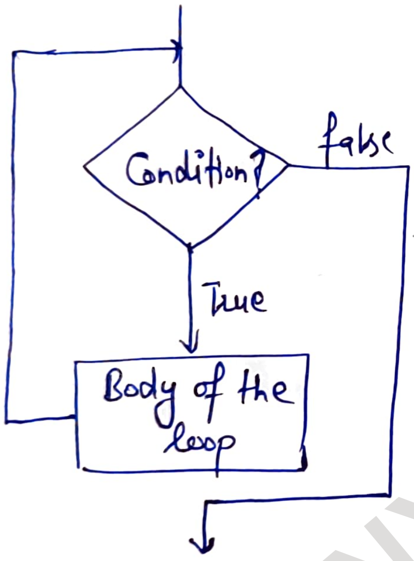
- Loops make your code shorter, more efficient & easier to maintain. Loops reduce the redundancy So minimize the risk of errors.

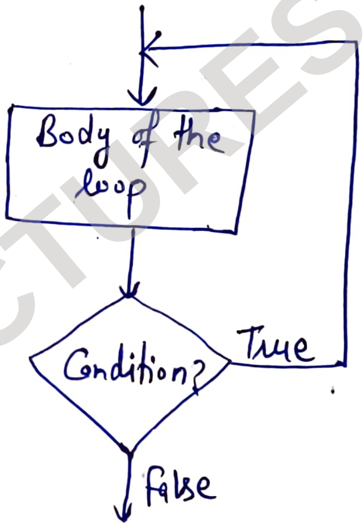
→ In Java loops are classified into 2 types based on when the condition is evaluated

1. Entry controlled loops
2. Exit controlled loops

1. Entry Controlled loops:- The condition / termination condition is evaluated before the loop body is executed. This means that if the condition is false right at the beginning, the loop body will not be executed at all.

e.g:- for loop & while loop





Entry Controlled

Exit Controlled

→ In exit-controlled, the condition is evaluated after the loop body has been executed. It means that the loop body will always execute at least once, even if the condition is false initially

e.g:- do-while loop

for loop:- It is an iteration statement. It will iterate or execute a block of code for some number of times and how many times block of code would be executed depends on some conditions

Syntax:-

```
for( expression 1 ; expression 2 ; expression 3) {
    //body of loop
}
```

* expression 1 is an initialization expression
* expression 2 is a condition / termination condition
* expression 3 is an update expression (increment / Decrement)

or we can also write:-

```
for( initialization ; condition expression ; update expression )
{
    // body of for loop
}
```

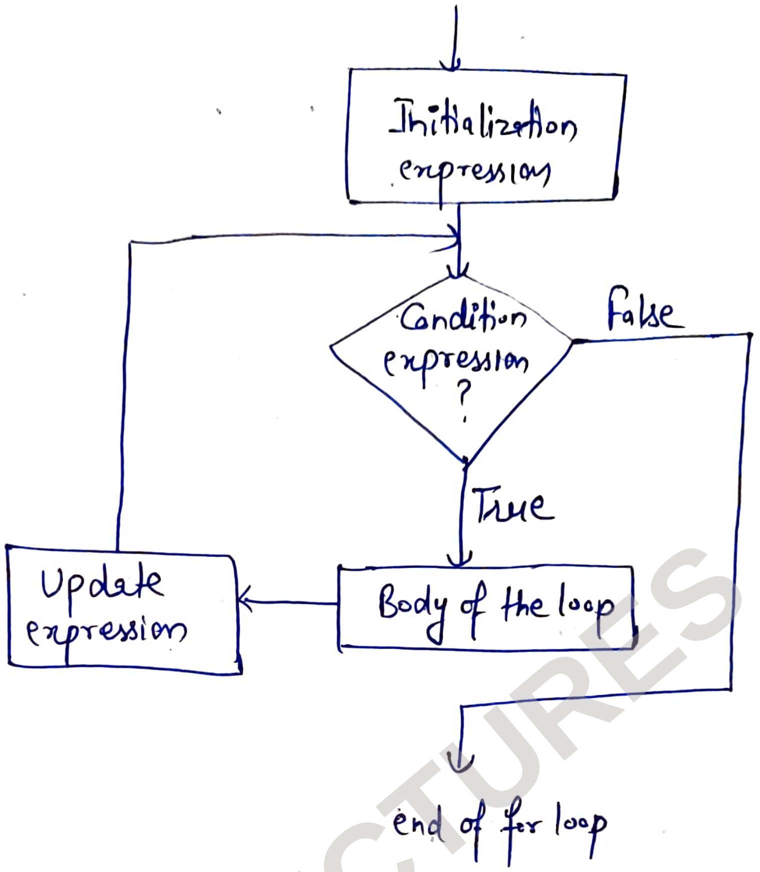
e.g:- ① print numbers from 1 to 5

```
for( int i = 1 ; i <= 5 ; i++)
{ System.out.println(i);
}
```

② print Jenny's lectures 100 times:-

```
for (int i = 1 ; i <= 100 ; i++)
{
   System.out.println("Jenny's Lectures");
}
```

NOTE:- Initialization expression would be executed only once & remaining 2 expression & body of loop would be executed multiple times depending on the condition given.



IMP Points:-

1. Initialization part is optional.

eg:-
```
int i = 1;
for ( ; i<=5; i++) {
    System.out.println(i);
}
```

2. Increment/ Decrement is also optional.

eg:-
```
int i= 1;
for ( ; i<=5; )
{ System.out.println(i);
    i++;
}
```

3. Condition expression is not optional. If the test condition is not present, the for loop becomes an infinite loop & these type of loops can be stopped or broke using break statement.

e.g:-
```
int i = 1;
for( ; ;) {
    System.out.println(i);
    i++;
}
```
output:- it will be an infinite loop

To stop above loop use break inside loop body.

(4) Two semicolons are must. If you miss any semi-colon then it will give compilation error

(5) In initialization part, we can initialize more than one variable.

e.g:-
```
for(int i=1, j=0; ;) {     →//allowed
    - -
}

for(int i=1, int j=0; ;)   →// Not allowed.
```

(6) Initialization expression can be written like:-

(i)
```
int i = 1;
for(i++; i<5; i++) {
    System.out.println(i);
}
```
output:-
```
2
3
4
```

(ii)
```
int i = 1;
for(System.out.println(i); i<5; i++) {
    System.out.println(i);
}
```

(iii)
```
int i;
for(i=2, System.out.println(i); i<5; i++) {
    System.out.println(i);
}
```

But we generally don't use these kind of expressions in initialization part because code would be less readable & Confusing. So generally in initialization part we declare a variable or initialize a variable or assign value to the variable which has already been declared.

7. Condition expression must evaluate to ~~true~~ boolean value (true or false)
8. In update expression we can have more than one expression.

e.g:-

```
int i,j;
~~for(i=1, j<=5;~~
for(i=1, j=5; i<=j; i++, j--){
    System.out.println(i);
}
```

Output:-

```
1
2
3
```

Quiz Time! -

1.

```
int i, j;
for(i=1, j=5; i<=j; i++, j--); {
    System.out.println(i);
}
```

output:- ?

i	j
$\boxed{1}$	$\boxed{5}$

i<=j	i++	j--
1<=5	2	4
2<=4	3	3
3<=3	$\boxed{4}$	2
4<=2		

2.

```
int i,j;
for(i=1, j=5; i<=j; System.out.println(i), i++, j--);
```

output ?

NOTE:- Body of the loop is optional only if you put semicolon (;) after for loop.

9 In condition expression we use logical operators as well.

e.g:-

1
```
for( int i= 1, j=5; i<=5 && j<=10; i++, j++) {
        System.out.println("i= " +i + "j= " +j);
}
```

Output:-

```
i=1   j=5
i=2   j=6
i=3   j=7
i=4   j=8
i=5   j=9
```

2
```
int sum=0;
for( int i=1; i< 20 && sum<100 ; i++)
   {   sum = sum+i;
       system.out.println("i= " +i+ " sum = " +sum);

   }
```

10
```
int i=1;
for (i= 2 ; i<=10; i +=2) {
     System.out.println(i);
}
```

in update expression we can increment it by any number or we can also write i *= 2, i -= 2, i /= 2 etc.

Quiz Time:-

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

2
```
for ( int i= 10; i>=1; i-=2*2) {
    system.out.println ( i);
}
```

What would be output of these two ?