Class - X , L - 9 Loops in Java

Notes and Exercise programs of L-9

In computer programming, **loops are used to repeat a specific block of code until a certain condition is met** (test expression is false). For example,

Imagine we need to print a sentence 50 times on your screen. Well, we can do it by using the print statement 50 times (without using loops). How about you need to print a sentence one million times? You need to use loops. With loops, we can simply write the print statement one time and run it for any number of times.

It's just a simple example showing the importance of loop in computer programming.

There are 3 **types** of loops in Java:

1. for loop(fixed or entry control loop,
2. while loop(user control or entry control), and
3. do-while loop(user control or exit control loop).

(1) for ( ) Loop

The **syntax** of for Loop in Java is:

```
for (initialization; testExpression; update)
 {
 // codes inside for loop's body
 }
```

Working of for loop/Dry run

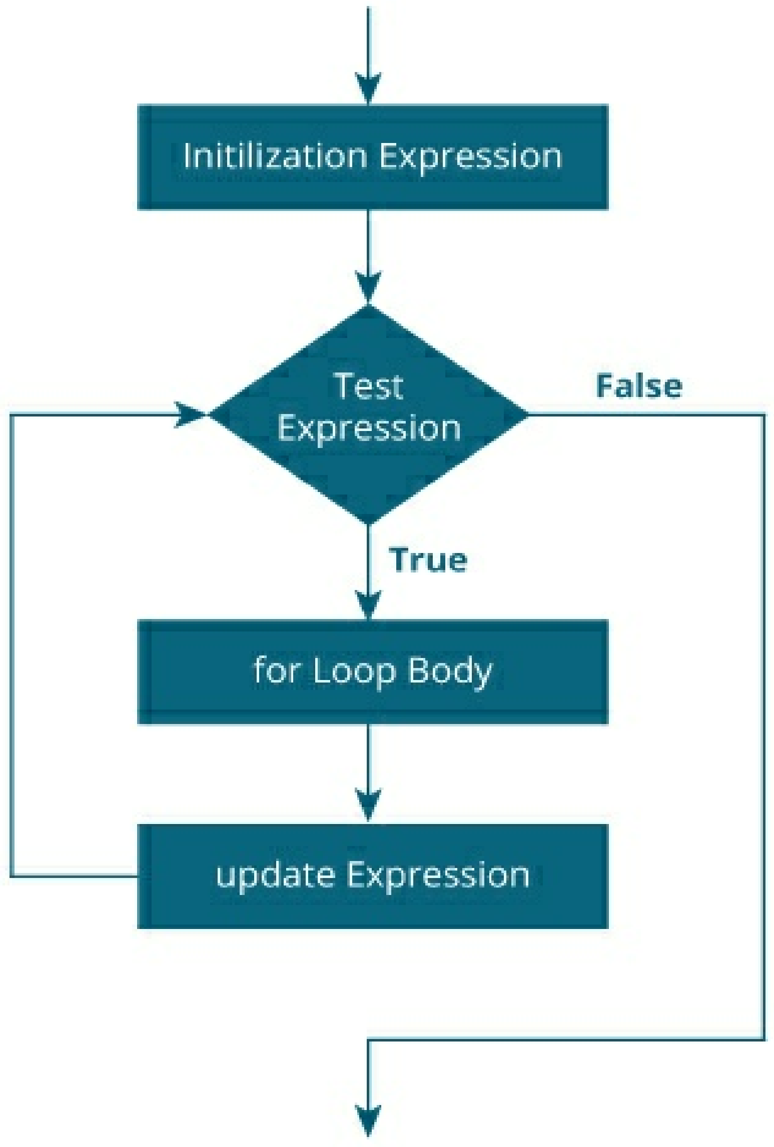
1 - The **initialization** expression is executed only once.

2 - Then, the **test** expression / condition is evaluated.
Here, test expression is a boolean expression.

3 - If the test expression is evaluated to true,
Codes inside the body of for loop is executed.
4- Then the **update** expression is executed.
Again, the test expression is evaluated.
If the test expression is true, codes inside the body of for loop is executed and update expression is executed.
This process goes on until the test expression is evaluated to false.

If the test expression is evaluated to false, for loop terminates.

for Loop Flowchart



Working of for loop

Example 1: for Loop

```
// Program to print a sentence 10 times class Loop
{
public static void main(String[] args)
{
      for (int i = 1; i <= 10; ++i)
      {
           System.out.print("Line " + i+" ");
      }
```

```
 }
}
```

Output:

```
Line 1 Line 2 Line 3 Line 4 Line 5 Line 6 Line 7 Line 8
Line 9 Line 10
```

In the above example, we have

initialization expression: int i = 1.e

test expression: i <=10

update expression: ++i

Here, initially, the value of i is 1. So the test expression evaluates to true for the first time. Hence, the print statement is executed. Now the update expression is evaluated.

Each time the update expression is evaluated, the value of i is increased by 1. Again, the test expression is evaluated. And, the same process is repeated.

So, i is the **control variable** here which is used to control the execution of loop.
This process goes on until i is 11. When i is 11, the test expression (i <= 10) is false and the for loop terminates.

To learn more about test expression and how it is evaluated, visit relational and logical operators.

Example 2: for Loop

```
// Program to find the sum of natural
//numbers from 1 to 1000.
class Number
{
 public static void main(String[] args)
 {
     int sum = 0;
      for (int i = 1; i <= 1000; ++i)
      {
             sum += i; // sum = sum + i
       }
       System.out.println("Sum = " + sum);
    }
 }
```

Output:

```
Sum = 500500
```

Here, we have a variable named sum. Its initial value is 0. Inside for loop, we have initialized a variable named iwith value 1.

In each iteration of for loop,

the sum variable is assigned value: sum + i

the value of i is increased by 1

The loop continues until the value of i is greater then 1000. For better visualization,

<u>**Dry run**</u>

1st iteration: i = 1 and sum = 0+1 = 1 2nd iteration: i = 2 and sum = 1+2 = 3 3rd iteration: i = 3 and sum = 3+3 = 6 4th iteration: i = 4 and sum = 6+4 = 10 ... . 999th iteration: i = 999 and sum = 498501 + 999 = 499500 1000th iteration: i = 1000 and sum = 499500 + 1000 = 500500

infinite for Loop

We should be always careful while working with loops. It is because **if we mistakenly set test expression in such a way that it is never false, the for loop will run forever.**

This is called infinite for loop. For example,

```
// Infinite for Loop
 class Infinite
{
public static void main(String[] args)
{
    int sum = 0;
    for (int i = 1; i <= 10; - -i)
   {
         System.out.println("Hello");

    }
}
}
```

Here, the test expression (i <= 10) is
never false and hello is printed infinite number to times (at least in theory).

Note: The initialization, update and test expression used in for statement is optional. Here's another example of the **infinite** for loop.

```
for (   ;   ;   )
{     }
```

(2) while ( ) Loop

The **syntax** of while loop in Java is:

```
while (testExpression)
{
  //codes inside the body of while loop
 }
```

How while loop works?

In the above syntax, the **test**expression inside parenthesis is a boolean expression. If the test expression is evaluated to true,

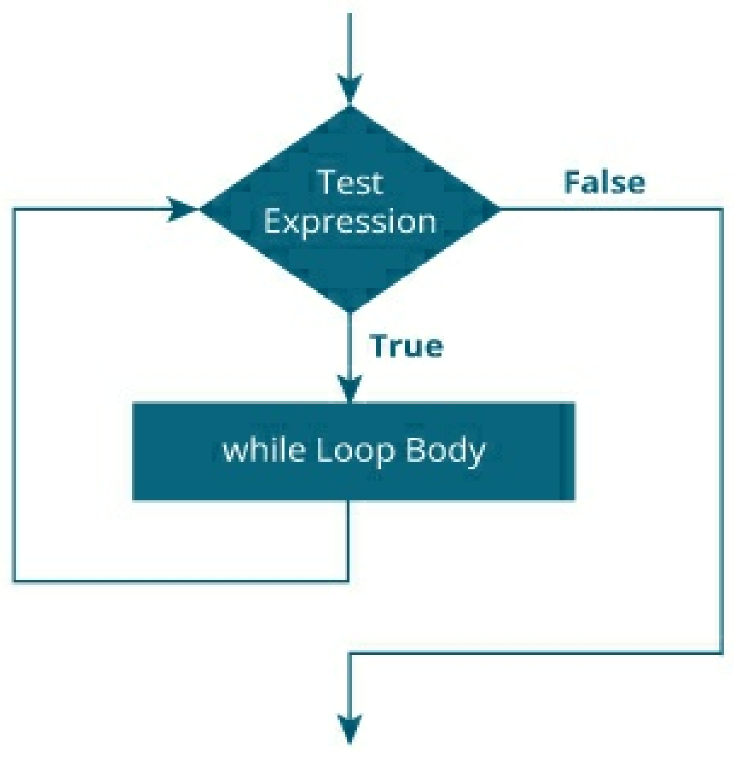
statements inside the while loop are executed.

then, the test expression is evaluated again.

This process goes on until the test expression is evaluated to false. If the test expression is evaluated to false,

the while loop is terminated.

Flowchart of while Loop



Working of while Loop

Example 1: while Loop

```
// Program to print line 10 times
class Loop
{
public static void main( )
 {
 int i = 1;
while (i <= 10)
{
System.out.print("Line " + i+" ");
++i;
}
 }
}
```

Output:

```
Line 1 Line 2 Line 3 Line 4 Line 5 Line 6 Line 7 Line 8
Line 9 Line 10
```

In the above example, we have a test expression (i <= 10). It checks whether the value of i is less than or equal to 10.

Here initially, the value of i is 1. So the test expression evaluates to true for the first time. Hence, the print statement inside while loop is executed.

Inside while loop notice the statement

```
++i;
```

This statement increases the value of i by 1 in each iteration. After 10 iterations, the value of i will be 11. Then, the test expression (i <= 10) evaluates to false and whileloop terminates.

Example 2: Java while Loop

```
// Program to find the sum of natural
 //numbers from 1 to 100.
class AssignmentOperator
{
public static void main( )
 {
int sum = 0, i = 100;
 while (i != 0)
 {
 sum += i;
 - -i;
}
 System.out.println("Sum = " + sum);
 }
}
```

Output:

```
Sum = 5050
```

Here, we have two variables named sum and i whose initial values are 0 and 100 respectively.

In each iteration of the while loop,

the sum variable is assigned value: sum + i

the value of i is decreased by 1

The loop continues until the value of i is equal to 0. For better visualization,

1st iteration: i = 100, sum = 0+100 = 100, and --i = 99
2nd iteration: i = 99, sum = 100+99 = 199, and --i =
98 3rd iteration: i = 98, sum = 199+98 = 297, and --i =
97 ... . 99th iteration: i = 2, sum = 5047+2 =
5049, and --i = 1 100th iteration: i = 1, sum = 5049+1 =
5050, and --i = 0

(3) do...while ( ) Loop

The do...while loop is similar to while loop with one key difference. **The body of do...while loop is executed for once before the test expression is checked**.

Here is the syntax of the do...whileloop.

```
do
 {
```

```
// codes inside body of do while loop
 } while (testExpression);
```

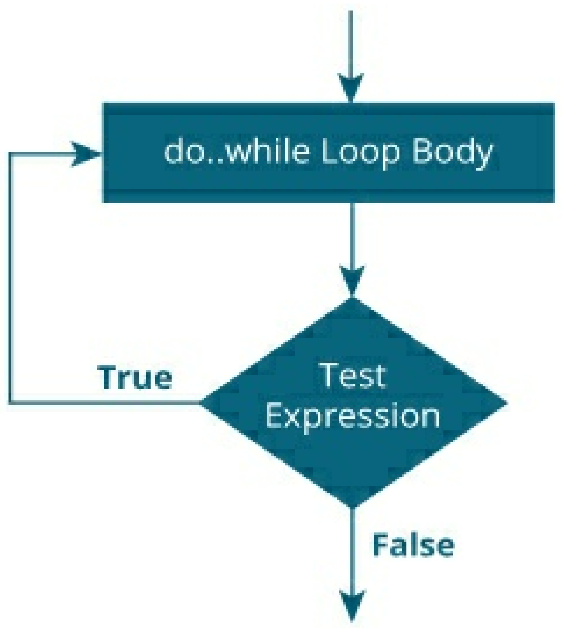
How do...while loop works?

The body of do...while loop is executed once (before checking the test expression). Only then, the test expression is checked.

If the test expression is evaluated to true, codes inside the body of the loop are executed, and the test expression is evaluated again. This process goes on until the test expression is evaluated to false.

When the test expression is false, the do..while loop terminates.

Flowchart of do...while Loop



Working of do...while Loop

Example 3: do...while Loop

The program below calculates the sum of numbers entered by the user until user enters 0.

To take input from the user, we have used the Scanner object. To learn more about Scanner, visit Java Scanner.

```
import java.util.Scanner;
class Sum
{
public static void main(String[] args)
{
   double number, sum = 0.0;
// creates an object of Scanner class Scanner input =
new Scanner(System.in); do
 {
// takes input from the user System.out.print("Enter a
number: "); number = input.nextDouble();
sum += number;
} while (number != 0.0); // test expression
System.out.println("Sum = " + sum);
}
}
```

Output:

```
Enter a number: 2.5 Enter a number: 23.3 Enter a
number: -4.2 Enter a number: 3.4 Enter a number: 0
Sum = 25.0
```

Infinite while Loop

We should be always careful while working with loops. It is because if we mistakenly set the test expression in such a way that it is never false, the while and do...while loop will run forever.

This is called infinite while and do...while loop. For example,

```
// Infinite while loop
while (true==true)
{
// body of while loop
}
```

Let's take another example,

```
// Infinite while loop
 int i = 100;
 while (i == 100)
{
System.out.print("Hey!");
}
```

Jump statements

While working with loops, it is sometimes desirable to skip some statements inside the loop or terminate the loop immediately without checking the test expression.

In such cases, break and continue statements are used.

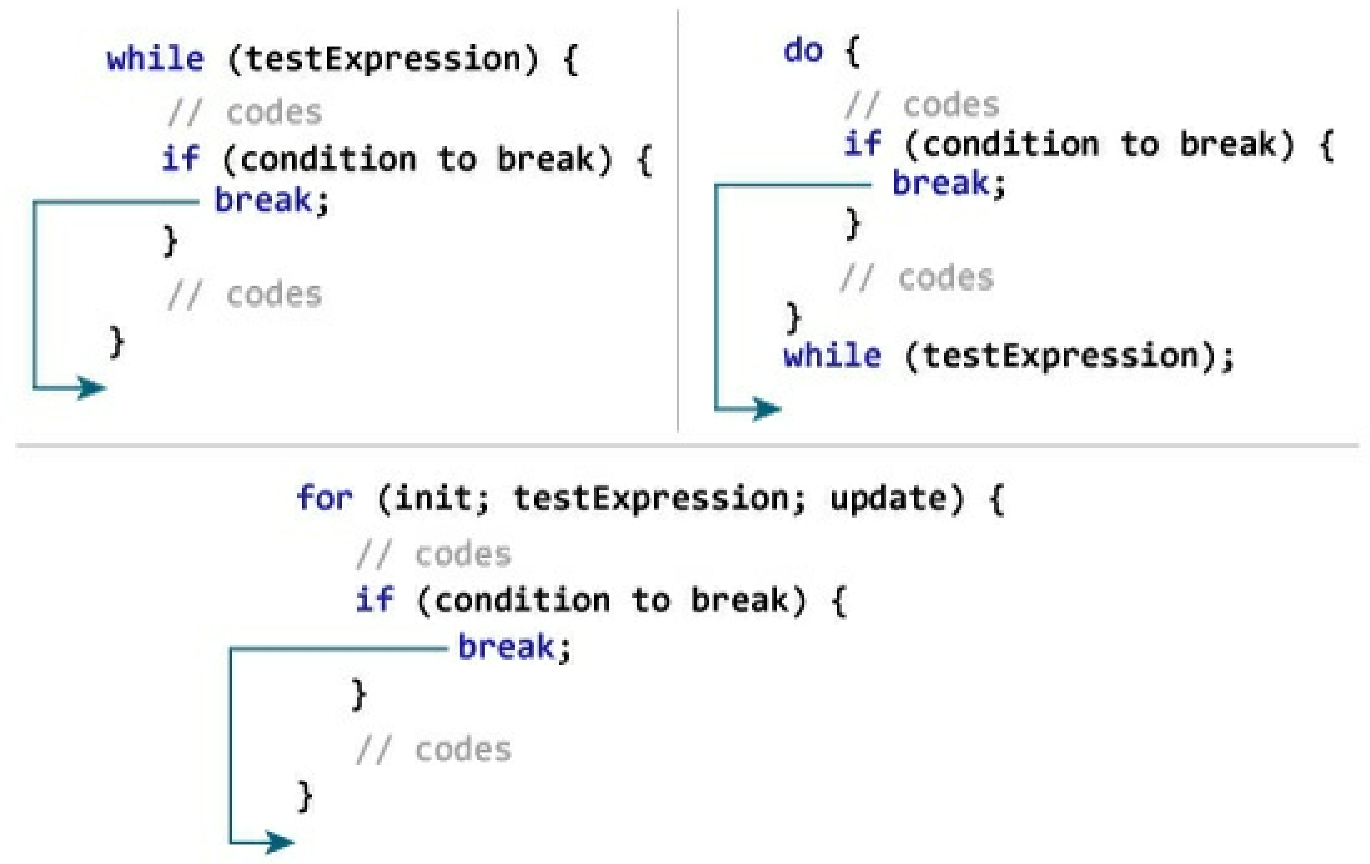
The **break** statement in Java terminates the loop immediately, and the control of the program moves to the next statement following the loop.

It is almost always used with decision-making statements (Java if() Statement).

Here is the syntax of the break statement in Java:

break;

How break statement works?



```
while (testExpression) {
   // codes
   if (condition to break) {
      break;
   }
   // codes
}
```

```
do {
   // codes
   if (condition to break) {
      break;
   }
   // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
   // codes
   if (condition to break) {
      break;
   }
   // codes
}
```

Working of Java break Statement

Example 1: Java break statement

```
class Test
{
 public static void main(String[] args)
 {
// for loop
 for (int i = 1; i <= 10; ++i)
 {
// if the value of i is 5 the loop terminates
if (i == 5)
 { break; }
System.out.println(i);
 } } }
```

Output:

```
1 2 3 4
```

In the above program, we are using the for loop to print the value of i in each iteration. To know how for loop works, visit the Java for loop. Here, notice the statement,

```
if (i == 5)
{
break;
}
```

This means when the value of i is equal to 5, the loop terminates. Hence we get the output with values less than 5 only.

Example 2: Java break statement

The program below calculates the sum of numbers entered by the user until user enters a negative number.

To take input from the user, we have used the Scanner object. To learn more about Scanner, visit Java Scanner.

```
import java.util.Scanner;
```

```
class UserInputSum
{
public static void main(String[] args)
 {

double number, sum = 0.0;
 // create an object of Scanner
 Scanner input = new Scanner(System.in); while (true)
 {
 System.out.print("Enter a number: ");
// takes double input from user
number = input.nextDouble();
 // if number is negative the loop terminates
if (number < 0.0)
{ break; }
sum += number;
}
System.out.println("Sum = " + sum);
} }
```

Output:

```
Enter a number: 3.2 Enter a number: 5 Enter a number:
2.3 Enter a number: 0 Enter a number: -4.5 Sum = 10.5
```

In the above program, the test expression of
the while loop is always true. Here, notice the line,

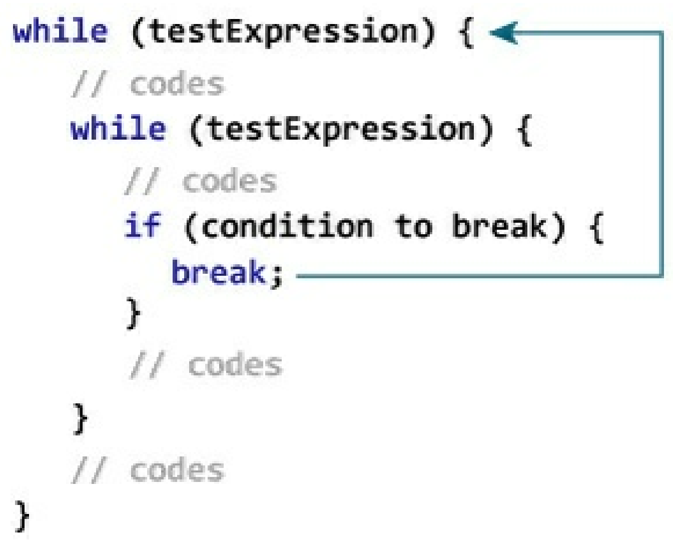
```
if (number < 0.0)
{
```

break;
}

This means when the user input negative numbers, the while loop is terminated.

Java break and Nested Loop

In the case of nested loops, the break statement terminates the innermost loop.



```
while (testExpression) {
   // codes
   while (testExpression) {
      // codes
      if (condition to break) {
         break;
      }
      // codes
   }
   // codes
}
```

Working of break Statement with Nested Loops

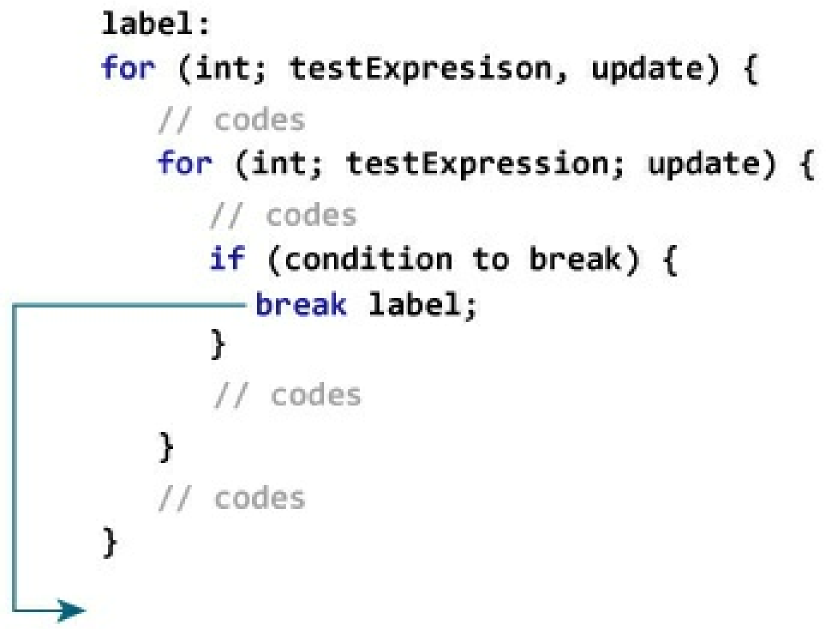
Here, the break statement terminates the innermost whileloop, and control jumps to the outer loop.

Lesson 10

Labeled break Statement

Till now, we have used the unlabeled break statement. **It terminates the innermost loop and switch statement.** However, there is another form of break statement in Java known as the labeled break.

We can use the labeled break statement to terminate the outermost loop as well.



```
label:
for (int; testExpresison, update) {
   // codes
   for (int; testExpression; update) {
      // codes
      if (condition to break) {
         break label;
      }
      // codes
   }
   // codes
}
```

Working of the labeled break statement in Java

As you can see in the above image, we have used the label identifier to specify the outer loop. Now, notice how the break statement is used (break label;).

Here, the break statement is terminating the labeled statement (i.e. outer loop). Then, the control of the program jumps to the statement after the labeled statement.

Here's another example:

```
while (testExpression)
{ // codes second:
      while (testExpression)
      { // codes
            while(testExpression)
            { // codes break second;
            }
       } // control jumps here
 }
```

In the above example, when the statement break second; is executed, the while loop labeled as second is terminated. And, the control of the program moves to the statement after the second whileloop.

Example 3: labeled break Statement

```
class LabeledBreak
{
public static void main(String[] args)
 {
// the for loop is labeled as first
 first:
for( int i = 1; i < 5; i++)
 {
 // the for loop is labeled as second second:
```

```
 for(int j = 1; j < 3; j ++ )
 {
 System.out.println("i = " + i + "; j = " +j);
// the break statement breaks the first for loop
if ( i == 2)
 break first;
}
}
}
}
```

Output:

```
i = 1; j = 1 i = 1; j = 2 i = 2; j = 1
```

In the above example, the labeled break statement is used to terminate the loop labeled as first. That is,

```
first: for(int i = 1; i < 5; i++) {...}
```

Here, if we change the statement break first; to break second; the program will behave differently. In this case, for loop labeled as second will be terminated. For example,

```
class LabeledBreak
{
 public static void main(String[] args)
{
// the for loop is labeled as first
```

```
 first:
for( int i = 1; i < 5; i++)
 {
// the for loop is labeled as second second:
for(int j = 1; j < 3; j ++ )
{
 System.out.println("i = " + i + "; j = " +j);
// the break statement terminates the loop labeled as
second
if ( i == 2)
break second;
}
 }
 }
 }
```

Output:

```
i = 1; j = 1 i = 1; j = 2 i = 2; j = 1 i = 3; j = 1 i = 3; j = 2 i = 4; j
= 1 i = 4; j = 2
```

Note: The break statement is also used to terminate cases inside the switch statement.

Java continue Statement

The continue statement in Java **skips** the current iteration of a loop (for, while, do...while, etc) and the control of the program moves to the end of the loop.

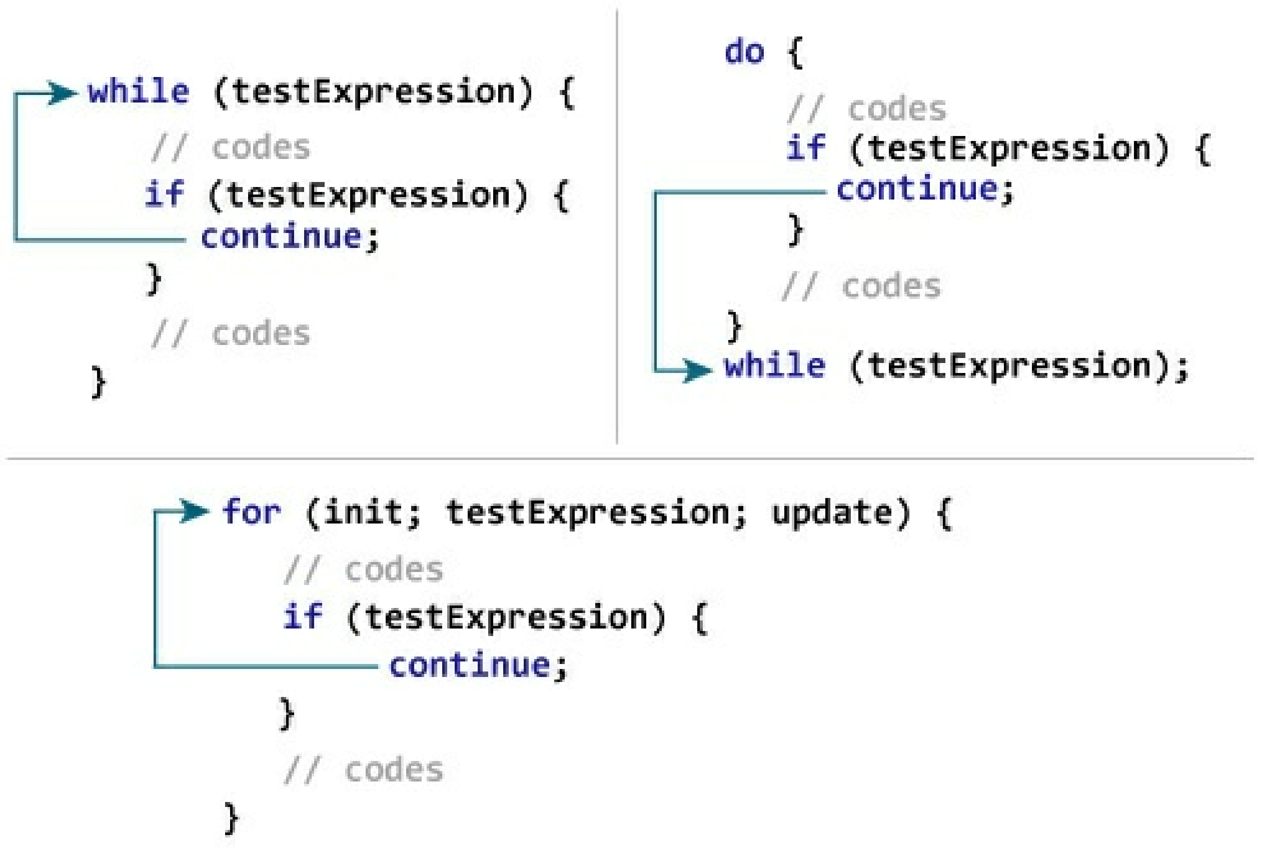
And, the test expression of a loop is evaluated.

In the case of for loop, the update statement is executed before the test expression.

The continue statement is almost always used in decision-making statements (if...else Statement). It's syntax is:

continue;

How continue statement works?



```
while (testExpression) {
    // codes
    if (testExpression) {
      continue;
    }
    // codes
}
```

```
do {
    // codes
    if (testExpression) {
      continue;
    }
    // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (testExpression) {
      continue;
    }
    // codes
}
```

Working of Java continue statement

Example 1: Java continue statement

```
class Test
{
 public static void main(String[] args)
 {
 // for loop
 for (int i = 1; i <= 10; ++i)
 {
// if value of i is between 4 and 9, continue is executed
 if (i > 4 && i < 9)
 {
continue;
}
 System.out.println(i);
}
 }
}
```

Output:

```
1 2 3 4 9 10
```

In the above program, we are using for loop to print the value of i in each iteration. To know how forloop works, visit Java for loop. Here, notice the statement,

```
if (i > 5 && i < 9)
{ continue; }
```

This means when the value of ibecomes more than 4 and less then 9, the print statement inside the loop is skipped. Hence we get the output with values 5, 6, 7, and 8 skipped.

Example 2: Java continue statement

The program below calculates the sum of 5 positive numbers entered by the user. If the user enters negative number or zero, it is skipped from the calculation.

To take input from the user, we have used the Scanner object.

```
import java.util.Scanner;
class AssignmentOperator
{
 public static void main(String[] args)
 {
double number, sum = 0.0;
// create an object of Scanner
Scanner input = new Scanner(System.in); for (int i = 1; i
< 6; ++i)
{
 System.out.print("Enter a number: ");
// takes double type input from the user number =
input.nextDouble();
// if number is negative, the iteration is skipped
if (number <= 0.0)
{ continue; }
sum += number;
 }
System.out.println("Sum = " + sum);
}
}
```

Output:

```
Enter a number: 2.2 Enter a number: 5.6 Enter a number: 0 Enter a number: -2.4 Enter a number: -3 Sum = 7.8
```

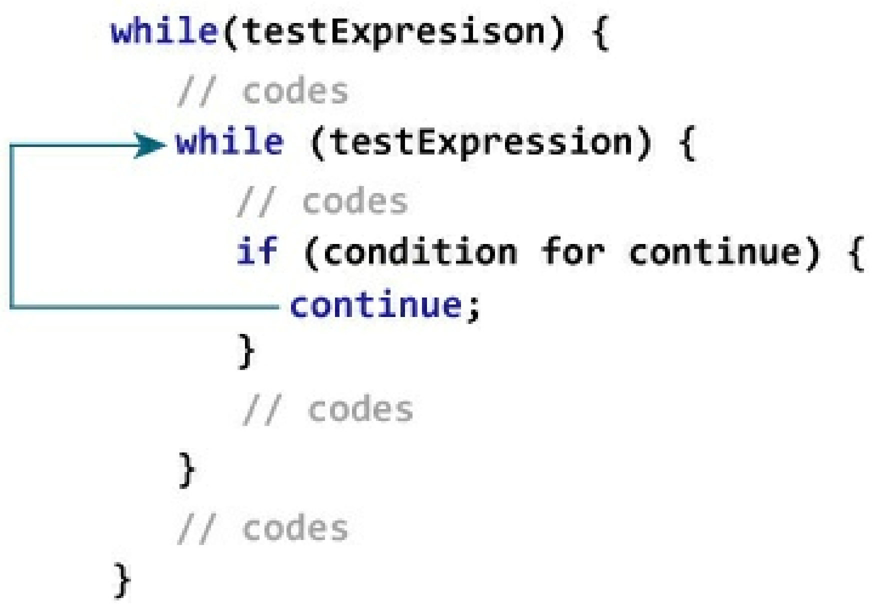
In the above program, notice the line,

```
if (number < 0.0)
{ continue; }
```

This means when the user input negative numbers, the current iteration of the loop is skipped. And the next iteration is started.

Java continue and Nested Loop

In the case of nested loops, the continue skips the current iteration of the innermost loop.



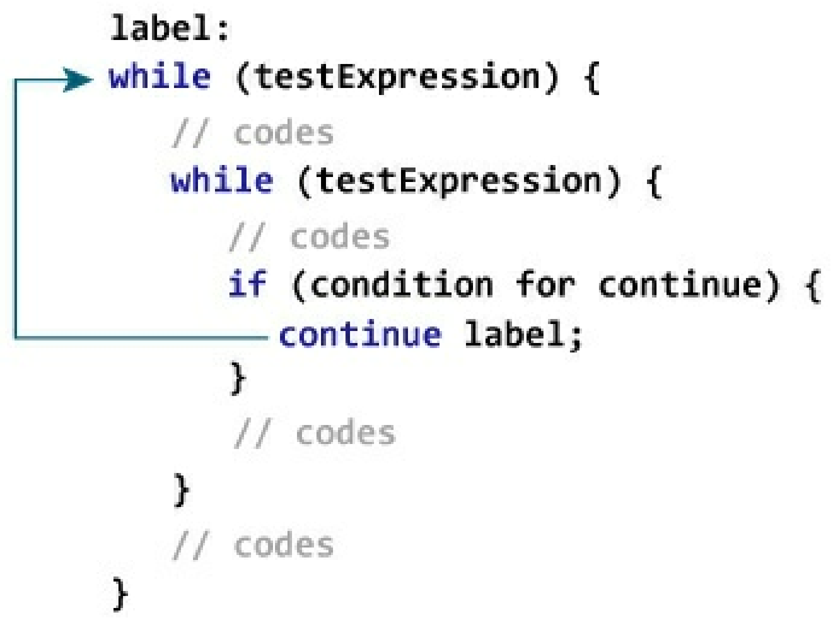
```
while(testExpresison) {
   // codes
   while (testExpression) {
      // codes
      if (condition for continue) {
         continue;
      }
      // codes
   }
   // codes
}
```

Working of the continue statement with Nested Loops

Lesson 10
Labeled continue Statement

Till now, we have used the
unlabeled continue statement. It is used to terminate the innermost loop . However, there is another form of continue statement in Java known as labeled contine.

We can use the labeled continue statement to terminate the outermost loop as well.



```
label:
while (testExpression) {
   // codes
   while (testExpression) {
      // codes
      if (condition for continue) {
        continue label;
      }
      // codes
   }
   // codes
}
```

Working of the labeled continue Statement in Java

As you can see in the above image, we have used the label identifier to specify the outer loop. Now, notice how the continue statement is used (continue label;).

Here, the continue statement is skipping the current iteration of the labeled statement (i.e. outer loop). Then, the control of the program goes to the next iteration of the labeled statement (outer loop)

Example 3: labeled continue Statement

```
class LabeledContinue
{
public static void main(String[] args)
 {
// the outer for loop is labeled as label first:
 for (int i = 1; i < 6; ++i)
{
 for (int j = 1; j < 5; ++j)
{
 if (i == 3 || j == 2) // skips the iteration of label (outer for
loop)
continue first;
System.out.println("i = " + i + "; j = " + j);
}
}
}
 }
```

Output:

```
i = 1; j = 1 i = 2; j = 1 i = 4; j = 1 i = 5; j = 1
```

In the above example, the labeled continue statement is used to skip the current iteration of the loop labeled as first.

```
if (i==3 || j==2)
 continue first;
```

Here, we can see the outermost for loop is labeled
as first,

```
first:
for (int i = 1; i < 6; ++i)
 {..}
```

Hence, the iteration of the outer for loop is skipped if the value of i is 3 or the value of j is 2.

Note: The use of labeled continue is often discouraged as it makes your code hard to understand.

Exercise programs
Write the programs With comments and Variable Description.

Where ever
 InputStresmReader and BufferedReader classes are use use Scanner class and it's methods for input.

Q15

```
class Q15
{
public static void main( )
{
for( int i=150; i<=250;i++)
{

//If the number is divisible by 5 and 6
//don't display it
   if(i%5==0 && i%6==0)
           ;
   else if(i%5==0)
          System.out.println( i);
   else if(i%6==0)
```

```
            System.out.println( i);
}
}
}
```

Q16

```
// Reversing a number by using while loop
import java.util.Scanner;
class Q16
{
   public static void main( )
   {
      Scanner sc= new Scanner( System.in);
       int n=0;
      System.out.println( " Enter a number ");
       n=sc.nextInt( );
       int rev=0,t=n;
       while( n>0)
       {
           int r=n%10;
           rev=(rev×10)+r;
           n=n/10;
      }
System.out.println( "Original no is : "+ t);
System.out.println( "Reverse no is :"+ rev);
}
}
```

Q17

```
// Removing 0 from the entered number.
```

```
import java.util.Scanner;
class Q17
{
   public static void main( )
   {
      Scanner sc= new Scanner( System.in);
       String n="", st="";
      System.out.println( " Enter a number ");
       n=sc next( );

      for( int i=0; i<n.length( );i++)
      {
           char ch=n.charAt(i);
          if(ch!='0')
             st=st+ch;
       }

System.out.println("New number is : "+ st);
}
}
```

Q18

```
//Tribonic  series
import java.util.Scanner;
class Q18
{
   public static void main( )
   {
      Scanner sc= new Scanner( System.in);
       int n=0, a=0,b=1,c=2, s=0;
      System.out.println( " Enter a number ");
```

```
       n=sc next( );
       System.out.print( a+" "+b +" "+c);
      for( int i=4; i<=n;i++)
      {
          s=a+b+c;
           System.out.print( "  "+s);
          a=b;
          b=c;
          c=s;
       }
}
}
```

Q20

```
//Menu driven program
import java.util.Scanner;
class Q20
{
   public static void main( )
   {
      Scanner sc= new Scanner( System.in);
       int ch=0;
      System.out.println(" (1) for Pattern1\n (2 )for
Pattern2");

      System.out.println( " Enter your choice :");
       n=sc nextInt( );
      switch(ch)
      {
       case 1:
       String s=" ICSE";
       for( int i=0; i<s.length( );i++)
      {
          char ch=s.charAt(i);
          for( int sp=0; sp<=i;sp++)
              System.out.print("  ");
          System.out.println(ch);
      }
       break;
       case 2:
       String s=" ICSE";
       for( int i=0; i<s.length( );i++)
      {
          char ch=s.charAt(i);
          for( int sp=s.length()-1; sp>=i;sp--)
              System.out.print("  ");
          System.out.println(ch);
      }
       break;
  default:
System.out.println( " Wrong choice ......");
}
}
}
```

Q22

```
//Pronic number
import java.util.Scanner;
class PronicNumber
{
  public static void main( )
  {
      Scanner sc= new Scanner( System.in);
       int n=0;
      System.out.println( " Enter a no.");
       n=sc nextInt( );
      for( int i=1; i<=n/2;i++)
      {
        if(i × (  i+1)==n)
        {
         System.out.println((i × (  i+1)+"= "+ i +" × "+(i+1));
           break;
         }
      }
if(i>n/2)
System.out.println(n+" is not a Pronic no ");
}
}
```

Q(24)

<u>**Use Scanner class for accepting the number.**</u>

Answer:

```
class test
{
public static void main (int n)
{
int S = 0, R;
int c = n;
while (n! = 0)
{
     R = n % 10;
     S = S + R;
```

```
      n = n/ 10;
}
if (c % S == 0)
System.out.println ("it is Niven number");
else
System.out.println ("It is Not a Niven No.");
}
}
```

Q(26)

```
import java.io.*;

class number
{
     public static void main()throws IOException
     {
          InputStreamReader IR=new InputStreamReader(System.in);
          BufferedReader br=new BufferedReader (IR);
          int n, ch;
          System.out.println("Enter the Number less than 100 :");
          n=Integer.parseInt(br.readLine());
          System.out.println("Press 1 to find whether the number is Prime of not")
          System.out.println("Press 2 to find whether the number is
                                                            Automorphic of not");
          System.out.println("Enter your choice : ");
          ch=Integer.parseInt(br.readLine());
          switch(ch)
     {
     case 1 :  int x=0;
               for(int i=2;i<=n;i++)
               {
                    if(n%i==0)
                    {
                         x=1;
                         break;
                    }
               }
               if(x==0)
               {
                    System.out.println("The number is Prime");
               }
               else
               {
                    System.out.println("The number is NOT Prime");
               }
               break;

case 2 :
               int s=n*n;
               int y=0;
               while(n!=0)
               {
                    if(n%10!=s%10)
                    {
                         y=1;
                    }
                    n=n/10;
                    s=s/10;
               }
```

```
            if(y==0)
            {
                System.out.println("The number is Automorphic"
            }
            else

           {
               System.out.println("The number is not Automorphic"
           }
           break;
      default :       System.out.println("Wrong Choice");
     }                //end of switch
   }
}
```

Q(27)

Answer:

```
import java.io.*;
class Text
{
      public static void main()throws IOException
      {
            InputStreamReader IR=new InputStreamReader(System.in);
            BufferedReader br=new BufferedReader(IR);
            System.out.println("Press 1 to check for BUZZ number ");
            System.out.println("Press 2 to check for GCD");
            int ch=Integer.parseInt(br.readLine());
            switch(ch)
            {
                case 1:
                    int n;
                    System.out.println("Enter the Number:");
                    n=Integer.parseInt(br.readLine());
                    if(n%7==0||n%10==7)
                    {
                        System.out.println("BUZZ number");
                    }
                    else
                    {
                        System.out.println("Not a Buzz Number");
                    }
                    break;
                case 2:
                    int a, b, i;
                    System.out.println("Enter the first number :");
                    a=Integer.parseInt(br.readLine());
                    System.out.println("Enter the Second number :");
                    b=Integer.parseInt(br.readLine());
                    for(i=(a<b?a:b); i>=1;i--)
                    {
                        if(a%i==0 && b%i==0)
                        {
                            break;
                        }
                    }
                    System.out.println(i);
                    break;
            }
      }
}
```

OR *(case 2 other way of doing the GCD program)*

```
import java.io.*;
class Text
{
    public static void main()throws IOException
    {
        InputStreamReader IR=new InputStreamReader(System.in);
        BufferedReader br=new BufferedReader(IR);
        System.out.println("Press 1 to check for BUZZ number ");
        System.out.println("Press 2 to check for GCD");
        int ch=Integer.parseInt(br.readLine());
        switch(ch)
        {
            case 1:
                int n;
                System.out.println("Enter the Number;");
                n=Integer.parseInt(br.readLine());
                if(n%7==0||n%10==7)
                {
                    System.out.println("BUZZ number");
                }
                else
                {
                    System.out.println("Not a Buzz Number");
                }
                break;
            case 2:
                int a,b,i;
                System.out.println("Enter the first number :");
                a=Integer.parseInt(br.readLine());
                System.out.println("Enter the Second number :");
                b=Integer.parseInt(br.readLine());
                if(a<b)       //storing bigger number in a and smaller in b
                {
                    int  c=a;
                         a=b;
                         b=c;
                }
                    while(a%b!=0)
                    {
                        int rem=a%b;
                        a=b;
                        b=rem;
                    }
                    System.out.println(b);
                    break;
            }
    }
}
```

Q30
(ii)

```
import java.util.Scanner;
class Series
{
   public static void main( )
   {
      Scanner sc= new Scanner( System.in);
       int n=0,s=0, x=3;
      System.out.println( " Enter the value of n = ");
       n=sc nextInt( );
      for( int i=1; i<=n;i++)
      {
        if(i %2==0)
             s=s-(int)Math.pow(x,i);
        else
             s=s+(int)Math.pow(x,i);
      }
System.out.println("Sum of the series is : "+s);
}
}
```

(vi)

```
import java.util.Scanner;
class Series
{
   public static void main( )
   {
      Scanner sc= new Scanner( System.in);
       int n=0,s=0, x=0;
      System.out.println( " Enter the value of
       n = ");
       n=sc nextInt( );
       System.out.println( " Enter the value
       of x = ");
       x=sc nextInt( );
      for( int i=1; i<=n;i=i+1)
      {
            s=s+(float)(x+i)/(i+2);
      }
System.out.println(" Sum of the series is :"+ s);
}
}
```

(vii)

```
class Series
{
public static void main ( )
{
float s=0f;
int f=1;
for( int i=2; i<=20;i++)
{
     f=f× i;  // calculating the factorial
      s=s+(float)x/f;
}
System.out.println(" Sum of the series is : "+ s);
}
}
```

Q(31)

```
import java.io.*;
class number
{
public static void main()throws IOException
{
InputStreamReader IR=new
InputStreamReader(System.in);
BufferedReader br=new BufferedReader(IR);
System.out.println("Press 1 for Palindrome
Number");
System.out.println("Press 2 for Perfect
Number");
System.out.println("Enter your choice");
int ch=Integer.parseInt(br.readLine());
int num1, num2;
```

```
switch(ch)
{
case 1:
System.out.println("Enter the Number :");
num1=Integer.parseInt(br.readLine());
int rev=0;
int x=num1; //duplicate copy
while (x!=0)
{
rev=rev*10+x%10;
x=x/10;
}
if(rev==num1)
{
System.out.println("The number is
Palindrome");
}
else
{
System.out.println("The Number is Not
Palindrome");
}
break;
```

```
case 2:
System.out.print("Enter the Number :");
num2=Integer.parseInt(br.readLine());
int sum=0;
for(int i=1; i<num2; i++)
{
if(num2%i==0)
{
sum=sum+i;
}
}
if(sum==num2)
{
System.out.println("The Number is Perfect
Number");
}
else
{
System.out.println("Tbe Number is Not a
Perfect Number");
}
break;
```

```
default:
System.out.println("Wrong Choice! Run the
program again");
break;
}
}
}
```

Last modified: 9:31 am