Looping 1

Basic Programming Teaching Team 2022

Objectives

After studying this material, students should be able to:

- Understand the loop 1 algorithm (for, while, and do-while)
- Provide simple examples of repetition
- Solve case study problems of looping 1 using a flowchart

Definition of Loop

- Loop statement is a command to **repeat** one or more statements **several times**.
- Loop statements are used so that we don't need to write one / a set of statements over and over. That way, you can reduce typing errors
- In Java, there are 3 types of loop commands that are commonly used, namely:
  - for() command
  - while() command
  - do-while() command

FOR Loop

FOR Loop

- Generally used in repetitions where the number of iterations is certain or has been previously known.
- Syntax for

```
for ([exp1]; [exp2]; [exp3]) statement;
```

or:

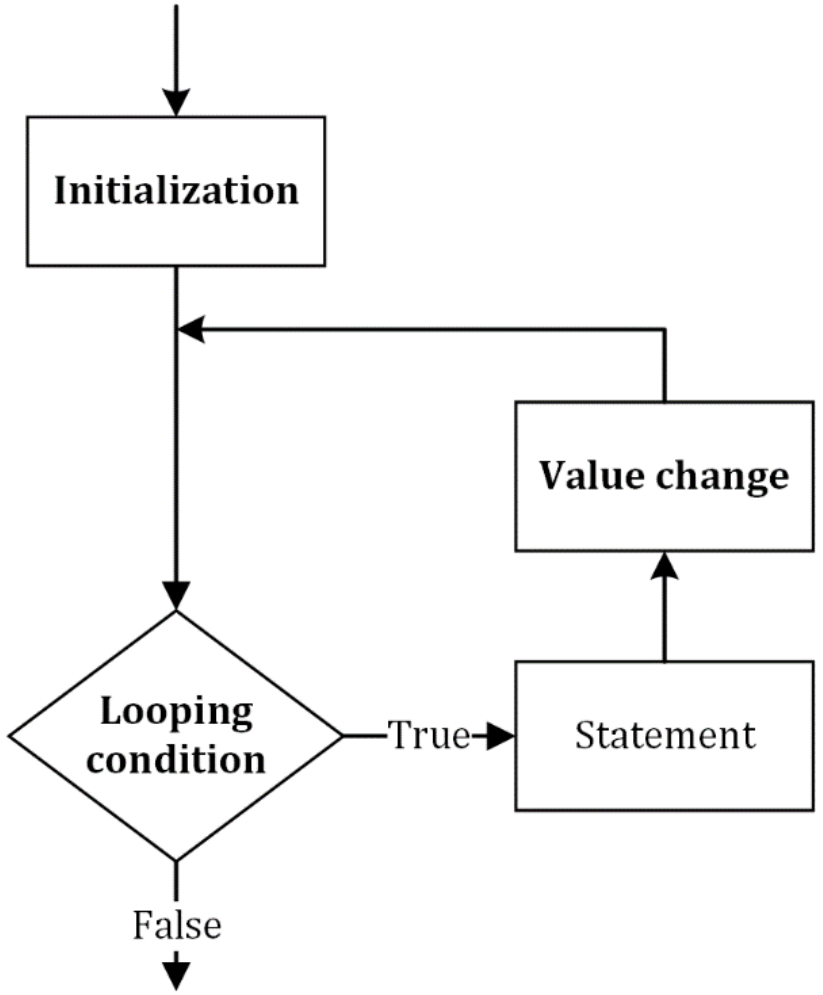
```
for ([exp1]; [exp2]; [exp3]){
    statement1;
    statement2;
    ……
}
```

- **exp1**: expression for initialization
- **exp2**: conditional expression (loop limit)
- **exp3**: increment or decrement expression
- exp1, exp2, and exp3 are optional (may / may not exist)

FOR Loop

- **Value initialization** is where we **assign the initial value** to the counter variable (the variable used to calculate the number of iterations).
- **Recurrence conditions** are conditions that must be met in order for the repetition to continue.
- **Value changes** are changes that will be made in each round to ensure that the loop will not continue.

Flowchart of FOR Loop



FOR Loop

- The for() command is usually used to perform a known number of loops.
- Example: “I like programming” 10 times

Output:

```
I like programming
I like programming
I like programming
I like programming
I like programming
I like programming
I like programming
I like programming
I like programming
I like programming
```

The loop will be done as long as i <= 10

```
int i;
for (i = 1; i <= 10; i++) System.out.println("I like programming");
```

Before looping, the variable i is assigned a value of 1

At each turn, the variable i will be added by 1

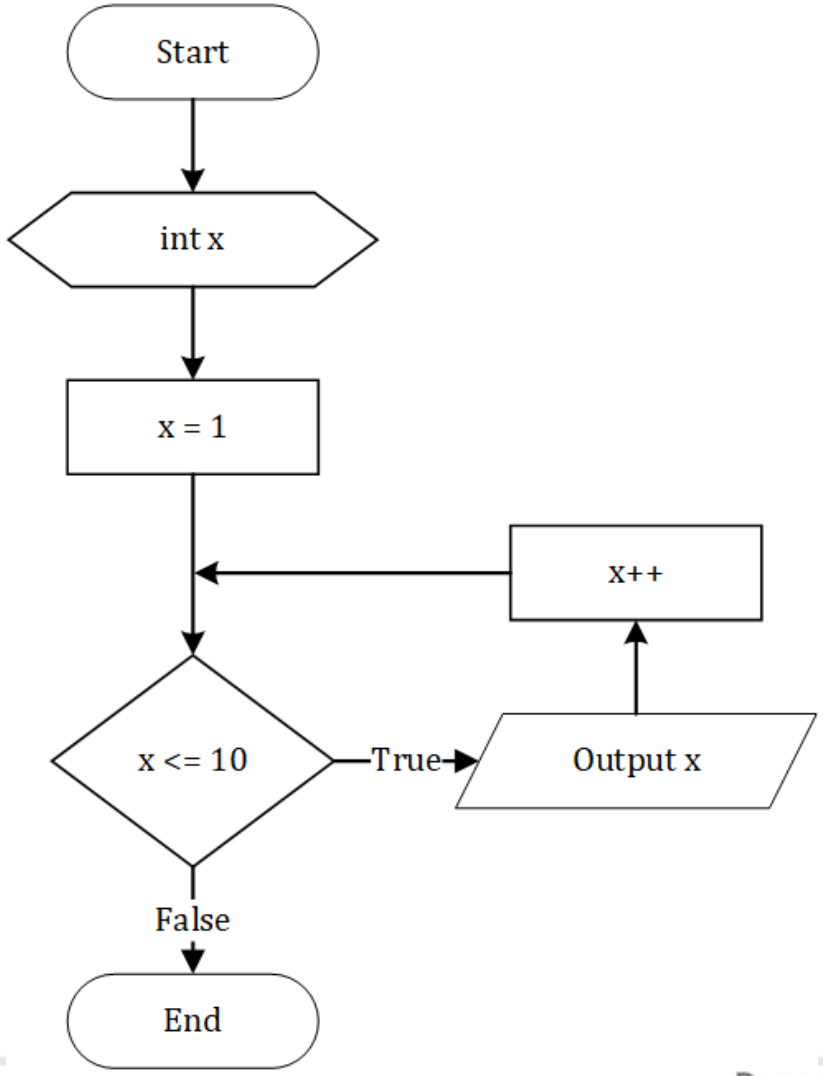
Example of FOR Loop

Create a program to print to the screen numbers 1 through 10.

```
int x;
for (x = 1; x <= 10; x++)
    System.out.printf("%d\n",x);
```

Output:

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



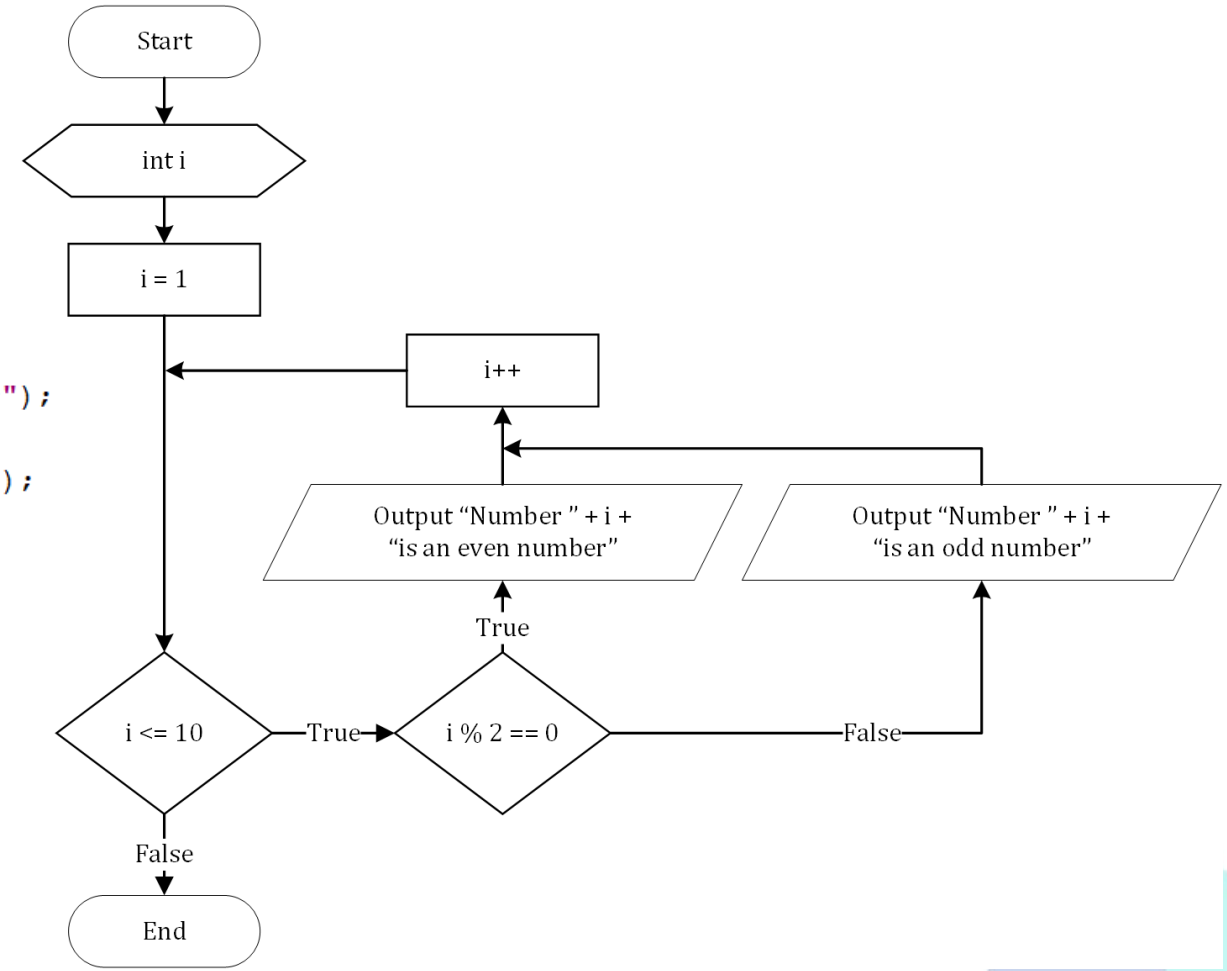
Example of FOR Loop

Create a program to print descriptions of even and odd numbers from 1 to 10.

```
int i;
for (i = 1; i <= 10; i++) {
    if (i % 2 == 0) {
        System.out.println("Number " + i + " is an even number");
    } else {
        System.out.println("Number " + i + " is an odd number");
    }
}
```

Output:

```
Number 1 is an odd number
Number 2 is an even number
Number 3 is an odd number
Number 4 is an even number
Number 5 is an odd number
Number 6 is an even number
Number 7 is an odd number
Number 8 is an even number
Number 9 is an odd number
Number 10 is an even number
```



FOR Variations

- **exp1** and **exp3** may contain multiple expressions separated by commas.
- Example:

```
int i, j;
for (i = 1, j = 30; i < j; i++, j--) {
    System.out.printf("%04d -- %04d\n", i, j);
}
```

- Output:

```
0001 -- 0030
0002 -- 0029
0003 -- 0028
0004 -- 0027
0005 -- 0026
0006 -- 0025
0007 -- 0024
0008 -- 0023
0009 -- 0022
0010 -- 0021
0011 -- 0020
0012 -- 0019
0013 -- 0018
0014 -- 0017
0015 -- 0016
```

FOR Variations

- **exp2** can be assigned a boolean variable
- Example:

```
Scanner input = new Scanner(System.in);
int number, i;
boolean stop = false;
for (i = 0; !stop; i++) {
    System.out.print("Enter a number: ");
    number = input.nextInt();
    System.out.println("The number you enter is " + number);
    if (number == 0) {
        stop = true;
    }
}
System.out.println("Done");
```

- Output:

```
Enter a number: 16
The number you enter is 16
Enter a number: 2020
The number you enter is 2020
Enter a number: 5
The number you enter is 5
Enter a number: 0
The number you enter is 0
Done
```

FOR Variations

- **exp1** and **exp3** can be left blank as needed
- Example:

```
Scanner input = new Scanner(System.in);
int number;
boolean stop = false;
for (; !stop; ) {
    System.out.print("Enter a number: ");
    number = input.nextInt();
    System.out.println("The number you enter is " + number);
    if (number == 0) {
        stop = true;
    }
}
System.out.println("Done");
```

- Output:

```
Enter a number: 11
The number you enter is 11
Enter a number: 8
The number you enter is 8
Enter a number: 107
The number you enter is 107
Enter a number: 0
The number you enter is 0
Done
```

WHILE Loop

WHILE Loop

- Syntax:

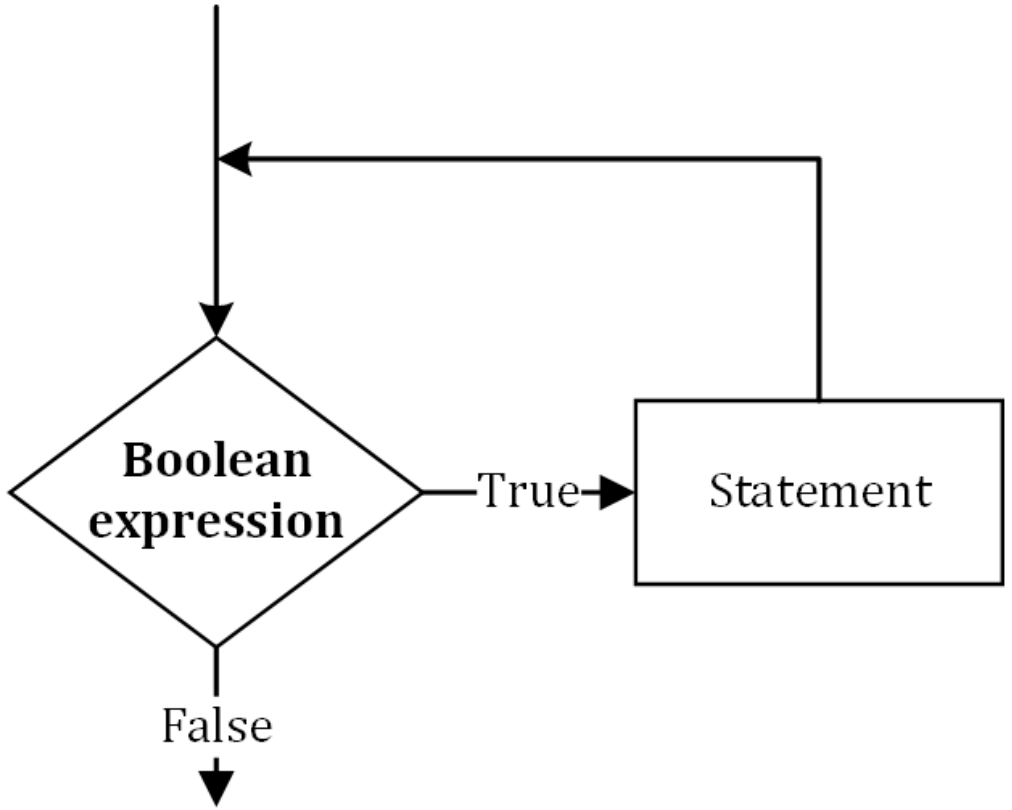
```
while (looping condition)
   statement; //command to be repeated
```

or:

```
while (looping condition){
   statement1;
   statement2;
   …
}
```

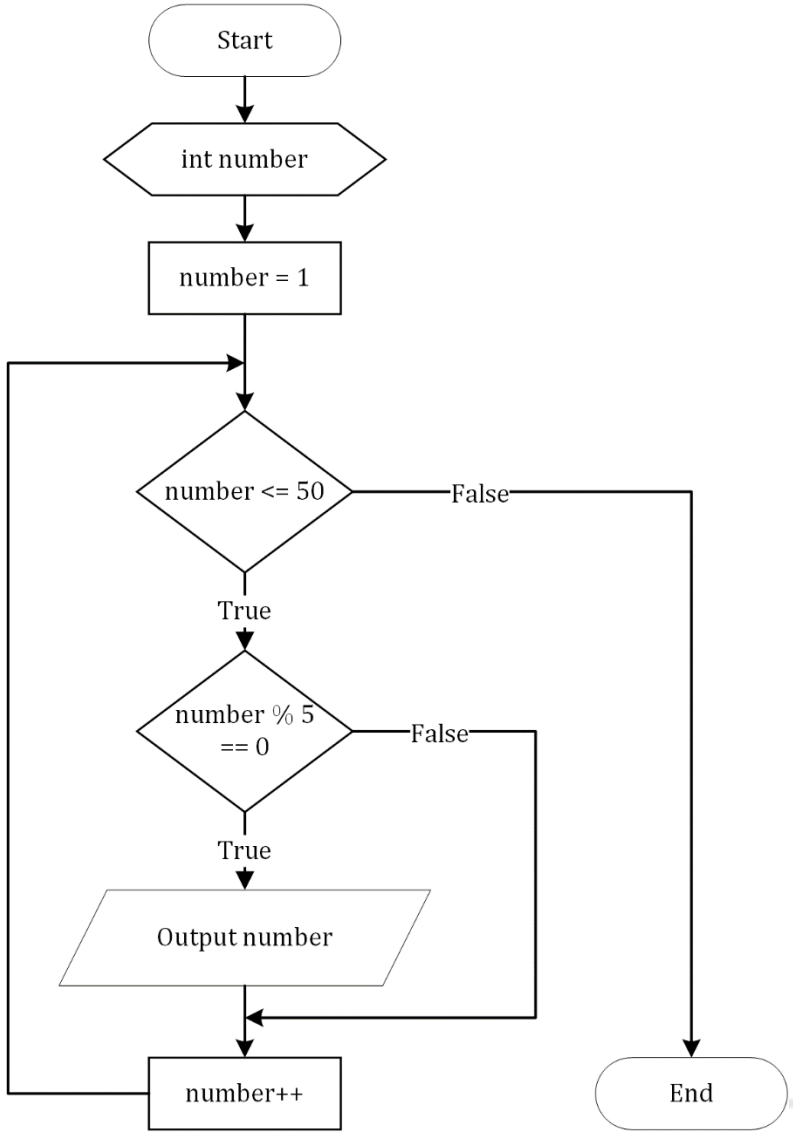
- **Looping condition** is a condition that must be met in order for the repetition to continue
- The while loop will continue **as long as the loop condition is TRUE**

Flowchart of WHILE Loop



Example of WHILE Loop

Make a flowchart to print all numbers multiples of 5 ranging from 5 to 50



FOR vs WHILE

- for and while equivalents

```
exp1;
while ( exp2 ) {
    statement1;
    statement2;
        ....
    exp3
}
```

equal

```
for (exp1; exp2; exp3 ) {
    statement1;
    statement2;
    ....
}
```

- Example:

```
int x = 1;
 while (x<=10) {
        ---
        ---
        x++;
 }
```

similar to

```
int x;
for( x = 1 ;  x <= 10 ;  x++ )
        ---
        ---
}
```

DO-WHILE Loop

DO-WHILE Loop

- In principle, the do-while() command is the same as the while() command.
- The do-while() command will loop the statement as long as the loop conditions are met.
- The **do-while()** command **executes the statement first**, then checks the conditions. The **while()** command **checks the conditions first**.
- Therefore, the do-while() command will execute the statement once, even if the loop conditions are not met.

DO-WHILE Loop

- Syntax:

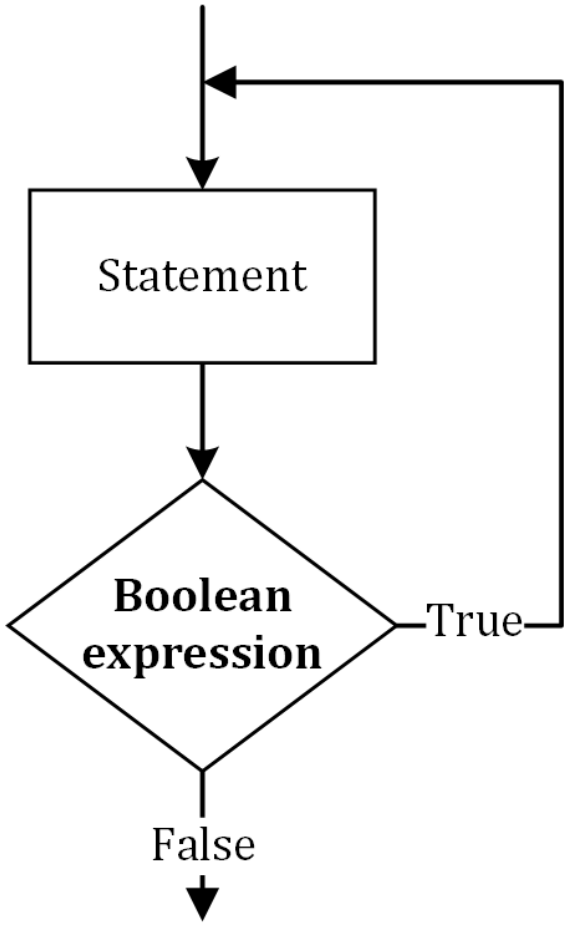
```
do
    statement;
while (boolean expression);
```

or:

```
do {
    statement1;
    statement2;
    …
} while(boolean expression);
```

- As long as the boolean expression is true, the statement is executed again.
- Boolean expression checks are performed after executing the statement

Flowchart of DO-WHILE Loop

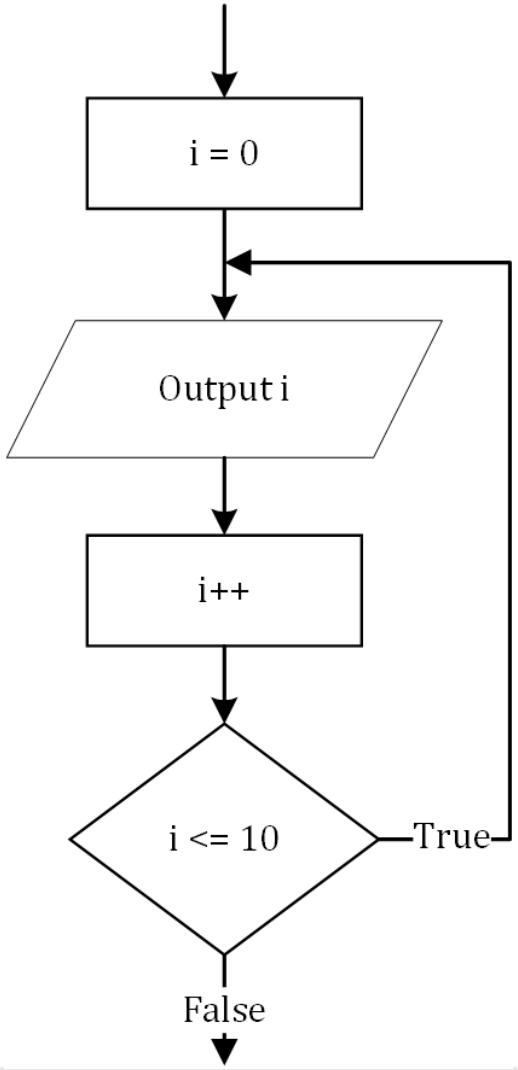


Example of DO-WHILE Loop

```
int i = 0;
do {
    System.out.println(i);
    i++;
} while (i <= 10);
System.out.println("Looping is complete");
```

Output

```
0
1
2
3
4
5
6
7
8
9
10
Looping is complete
```




```
int i = 11;
do {
    System.out.println(i);
    i++;
} while (i <= 10);
System.out.println("Looping is complete");
```

Output

```
11
Looping is complete
```

WHILE vs DO-WHILE

- In the **while()** loop, the statement or statement block may never be executed if the value of the boolen expression is false, because the loop operation begins by executing the boolean expression first.
- In the **do-while()** loop, the statement or statement block must be done at least once, because the boolean expression is only checked at the end of the loop block.

How to Stop the Loop

How to Stop the Loop

Several ways to stop repetition can be done by adding:

1. **Sentinel** or delimiter with a special code
2. **Questions**, for example: "Will the repetition continue?"

Using Sentinel

Example of sentinel in do-while using a value of 0 in the variable length and variable width.

```
int length, width, area;
Scanner input = new Scanner(System.in);
do {
    System.out.print("Length [0=exit]: ");
    length = input.nextInt();
    System.out.print("Width [0=exit]: ");
    width = input.nextInt();
    area = length * width;
    System.out.printf("Area = %d x %d = %d\n\n", length, width, area);
} while ((length != 0) && (width != 0));
```

```
Length [0=exit]: 7
Width [0=exit]: 9
Area = 7 x 9 = 63

Length [0=exit]: 4
Width [0=exit]: 0
Area = 4 x 0 = 0
```

Using Question

Example of a do-while question

```
int length, width, area;
String choice;
Scanner input = new Scanner(System.in);
do {
    System.out.print("Length: ");
    length = input.nextInt();
    System.out.print("Width: ");
    width = input.nextInt();
    area = length * width;
    System.out.printf("Area = %d x %d = %d\n\n", length, width, area);
    System.out.println("Do you want to recount? <Y/N>");
    choice = input.next();
} while ((choice.charAt(0) == 'Y') || (choice.charAt(0) == 'y'));
```

```
Length: 15
Width: 9
Area = 15 x 9 = 135

Do you want to recount? <Y/N>
y
Length: 17
Width: 11
Area = 17 x 11 = 187

Do you want to recount? <Y/N>
n
```

Statement Break and Continue

break

- Used to exit loops (for, while and do-while)
- Used to exit switches

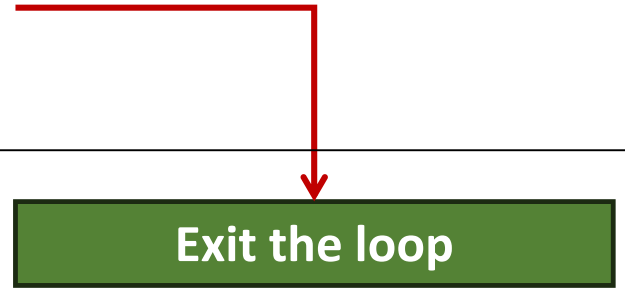
continue

- Used to skip the rest of the instructions in the loop, and loop execution continues to the next stage

Examples of Using Break

An example of using a loop break statement that causes the program to exit the loop

```
int x = 1;
while (x<=10) {
        System.out.printf( "%d\n", x );
        x++;
        if (x>5) break;
}
```



Example of Using Continue

```
int x;
for(x=1; x<=10; x++) {
        if (x == 5) continue;
        System.out.print(x);
}
```

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

Exercise

Create a flowchart from the following problems using for, while, and do-while!

1. Displays the odd number from 11 to 188
2. Displays the sum of the series of numbers 1 to 30
3. Calculates the power of X^Y with X and Y from the user input
4. Displays a series of numbers 2 4 8 16 32… 256