FACE

Looping

FACE

Looping

- A **loop** statement allows us to execute a statement or group of statements multiple times.

Looping/Iteration statements

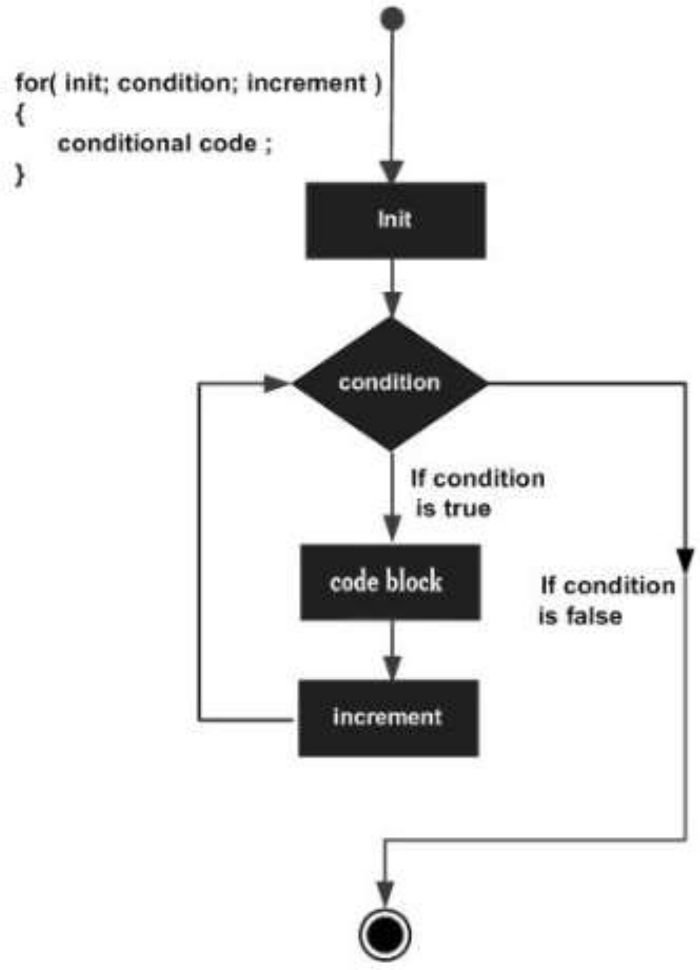
- for
- while
- do while
- Enhanced for

for

- It is used to iterate a part of the program several times.
- If the number of iteration is fixed, it is recommended to use for loop.

Syntax:

for (*initialization condition*; *testing condition*; *increment/decrement*)
{
        statement(s) ;
}



```java
class forLoopDemo
{
    public static void main(String args[])
    {
        for (int x = 2; x <= 4; x++)
        System.out.println("Value of x:" + x);
    }
}
```

output

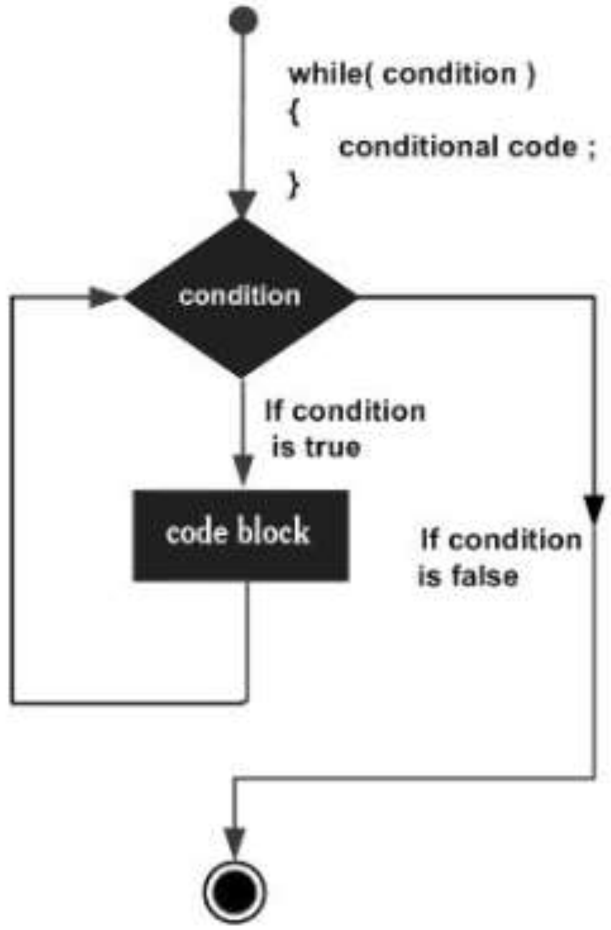
```
Value of x:2
Value of x:3
Value of x:4
```

while

- It is used to iterate a part of the program several times.
- If the number of iteration is not fixed, it is recommended to use while loop.

Syntax:

```
while(condition)
{
        statement(s) ;
}
```



```java
class whileLoopDemo
{
    public static void main(String args[])
    {
        int x = 1;
        while (x <= 4)
        {
          System.out.println("Value of x:" + x);
          x++;
        }
    }
}
```

output

```
Value of x:1
Value of x:2
Value of x:3
Value of x:4
```

do while

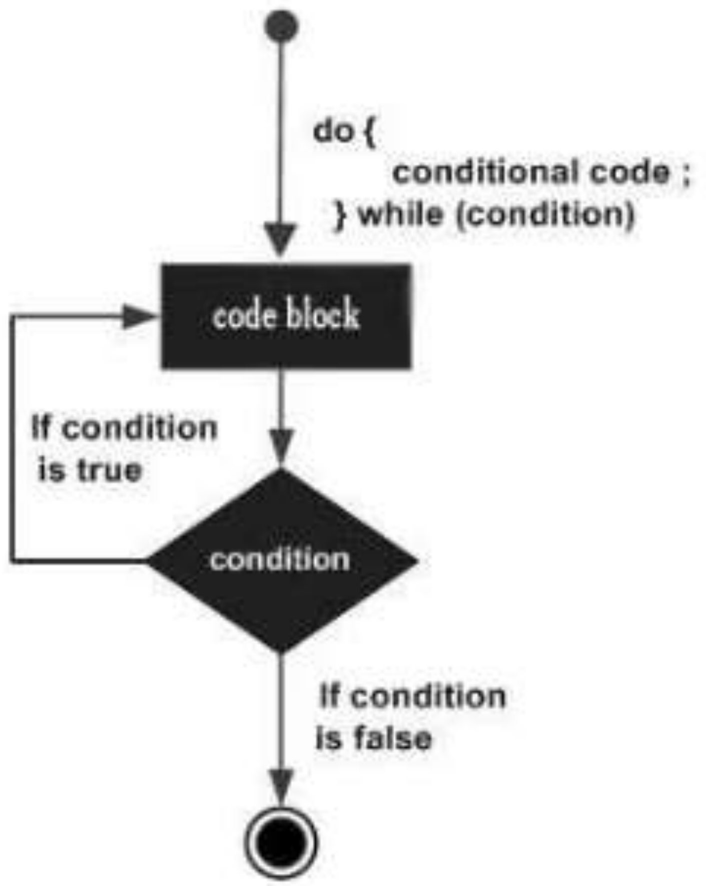
- It is used to iterate a part of the program several times.
- Use do while if the number of iteration is not fixed and you must have to execute the loop at least once.

Syntax:

```
do
{

        statement(s) ;

} while(condition);
```



```java
class dowhileloopDemo
{
    public static void main(String args[])
    {
        int x = 21;
        do
        {
          System.out.println("Value of x:" + x);
          x++;
        }while (x < 20);
    }
}
```

output

```
Value of x: 21
```

What is the difference between **while** and **do while** ?

Enhanced for

- Enhanced for loop provides a simpler way to iterate through the elements of a **collection** or **array**.

Syntax:

for (*T element:Collection obj/array*)
{
statement(s) ;
}

```java
public class enhancedforloop
{
 public static void main(String args[])
 {
 String array[] = {"Ron", "Harry","Hermoine"};

      for (String x:array)
        {
            System.out.println(x);
        }

     }
}
```

output

```
Ron
Harry
Hermoine
```

When to use **Enhanced for** loop ?

FACE

Infinite loop

- One of the most common mistakes while implementing any sort of looping is that it may not ever exit, that is the loop runs for infinite time.

```java
public class LooppitfallsDemo
{
    public static void main(String[] args)
    {
        int x = 5;
        while (x == 5)
        {
            System.out.println("In the loop");
        }
    }
}
```

output

```
In the loop
In the loop
In the loop
In the loop
In the loop
In the loop
In the loop
In the loop
In the loop
In the loop
In the loop
    ..
    ..
    ..
```

Jump statements

- break
- continue
- return

break

- Used to break loop or switch statement.
- When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

```java
public class BreakExample {
public static void main(String[] args) {
    for(int i=1;i<=10;i++){
        if(i==5){
            break;
        }
        System.out.println(i);
    }
}
}
```

output

```
1
2
3
4
```

continue

- Used to continue the loop, it continues the current flow of the program and skips the remaining code at the specified condition.
- The continue statement is used in loop control structure when you need to jump to the next iteration of the loop immediately.
- It can be used with for loop or while loop.

FACE

```java
public class ContinueExample {
public static void main(String[] args) {
    for(int i=1;i<=10;i++){
        if(i==5){
                continue;
         }
        System.out.println(i);
    }
}
}
```

output

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

return

- Return is used to **exit** from a method, with or without a value.

```java
// Predict the output
class Test {
public static void main(String[] args)
    {
        int j = 0;
        do
            for (int i = 0; i++ < 1 ; )
                System.out.println(i);
        while (j++ < 2);
    }
}
```

OUTPUT

1. 111
2. 222
3. 333
4. error

```
// Predict the output
class Test {
        static String s = "";
public static void main(String[] args)
        {
        P:
                for (int i = 2; i < 7; i++) {
                        if (i == 3)
                                continue;
                        if (i == 5)
                                break P;
                        s = s + i;
                }
                System.out.println(s);
        }
}
```

OUTPUT

1. 32
2. 23
3. 24
4. 42

```java
// Predict the output
class Test {
public static void main(String[] args)
    {
        for (int i = 0; i < 10; i++)
            int x = 10;
    }
}
```

OUTPUT

1. No Output
2. Compile time error
3. Runtime error
4. Runtime Exception

```java
// Predict the output
class Test {
public static void main(String[] args)
    {
        int i = 0;
        for (System.out.println("HI"); i < 1; i++)
            System.out.println("HELLO");
    }
}
```

OUTPUT

1. HI HELLO
2. No Output
3. Compile time error
4. HELLO

```
// Predict the output
class Test {
public static void main(String[] args)
    {
        for (int i = 0;; i++)
            System.out.println("HELLO");
    }
}
```

OUTPUT

1. Compile time error
2. HELLO
3. HELLO(Infinitely)
4. Run-time Exception

THANK YOU

FACE