**Day 3**
Thu 19 Aug 2017
Java Tutorial
Instructor: Hameed Mahmoud

# Day 3

## Basics of Java Programming Language

### Objectives

1. Practice
2. Conditions
3. Operators
4. Exercise

# Java If-else Statement

The Java *if statement* is used to test the condition. It checks boolean condition: *true* or *false*. There are various types of if statement in java.

- if statement
- if-else statement
- nested if statement
- if-else-if ladder

# Java IF Statement

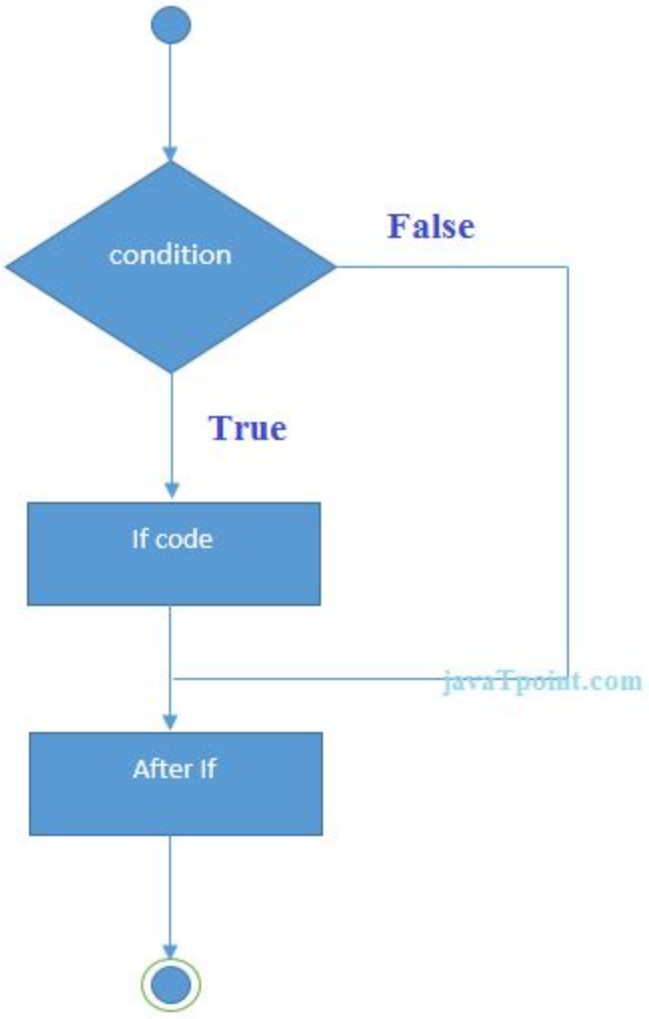
The Java if statement tests the condition. It executes the *if block* if condition is true.

**Syntax:**

```
if(condition){

//code to be executed

}
```



**Example:**

```
public class IfExample {

public static void main(String[] args) {

   int age=20;

   if(age>18){

      System.out.print("Age is greater than 18");

   }

}

}
```

Output:

```
Age is greater than 18
```

# Operators in java

**Operator** in java is a symbol that is used to perform operations. For example: +, -, *, / etc.

There are many types of operators in java which are given below:

- Unary Operator,
- Arithmetic Operator,
- shift Operator,
- Relational Operator,
- Bitwise Operator,
- Logical Operator,
- Ternary Operator and
- Assignment Operator.

# Java Operator Precedence

| Operator Type | Category | Precedence |
|---|---|---|
| Unary | postfix | *expr*++ *expr*-- |
| | prefix | ++*expr* --*expr* +*expr* -*expr* ~ ! |
| Arithmetic | multiplicative | * / % |
| | additive | + - |
| Shift | shift | << >> >>> |
| Relational | comparison | < > <= >= instanceof |
| | equality | == != |
| Bitwise | bitwise AND | & |
| | bitwise exclusive OR | ^ |
| | bitwise inclusive OR | \| |
| Logical | logical AND | && |
| | logical OR | \|\| |
| Ternary | ternary | ? : |

| | | |
|---|---|---|
| Assignment | assignment | = += -= *= /= %= &= ^= \|= <<= >>= >>>= |

## Java Unary Operator Example: ++ and --

```
1. class OperatorExample{
2. public static void main(String args[]){
3. int x=10;
4. System.out.println(x++);//10 (11)
5. System.out.println(++x);//12
6. System.out.println(x--);//12 (11)
7. System.out.println(--x);//10
8. }}
```

Output:

```
10
12
12
10
```

## Java Unary Operator Example 2: ++ and --

```
1. class OperatorExample{
2. public static void main(String args[]){
3. int a=10;
4. int b=10;
5. System.out.println(a++ + ++a);//10+12=22
6. System.out.println(b++ + b++);//10+11=21
7.
8. }}
```

Output:

```
22
21
```

## Java Unary Operator Example: ~ and !

```
1. class OperatorExample{
2. public static void main(String args[]){
3. int a=10;
4. int b=-10;
5. boolean c=true;
6. boolean d=false;
7. System.out.println(~a);//-11 (minus of total positive value which starts from 0)
8. System.out.println(~b);//9 (positive of total minus, positive starts from 0)
9. System.out.println(!c);//false (opposite of boolean value)
10. System.out.println(!d);//true
11. }}
```

Output:

```
-11
9
false
true
```

## Java Arithmetic Operator Example

```
1. class OperatorExample{
2. public static void main(String args[]){
3. int a=10;
4. int b=5;
```

```
5. System.out.println(a+b);//15
6. System.out.println(a-b);//5
7. System.out.println(a*b);//50
8. System.out.println(a/b);//2
9. System.out.println(a%b);//0
10. }}
```

Output:

```
15
5
50
2
0
```

## Java Arithmetic Operator Example: Expression

```
1. class OperatorExample{
2. public static void main(String args[]){
3. System.out.println(10*10/5+3-1*4/2);
4. }}
```

Output:

```
21
```

## Java Shift Operator Example: Left Shift

```
1. class OperatorExample{
2. public static void main(String args[]){
3. System.out.println(10<<2);//10*2^2=10*4=40
4. System.out.println(10<<3);//10*2^3=10*8=80
5. System.out.println(20<<2);//20*2^2=20*4=80
```

```
6. System.out.println(15<<4);//15*2^4=15*16=240
7. }}
```

Output:

```
40
80
80
240
```

## Java Shift Operator Example: Right Shift

```
1. class OperatorExample{
2. public static void main(String args[]){
3. System.out.println(10>>2);//10/2^2=10/4=2
4. System.out.println(20>>2);//20/2^2=20/4=5
5. System.out.println(20>>3);//20/2^3=20/8=2
6. }}
```

Output:

```
2
5
2
```

## Java Shift Operator Example: >> vs >>>

```
1. class OperatorExample{
2. public static void main(String args[]){
3.     //For positive number, >> and >>> works same
4.     System.out.println(20>>2);
5.     System.out.println(20>>>2);
6.     //For nagative number, >>> changes parity bit (MSB) to 0
```

```
7.      System.out.println(-20>>2);
8.      System.out.println(-20>>>2);
9.  }}
```

Output:

```
5
5
-5
1073741819
```

## Java AND Operator Example: Logical && and Bitwise &

The logical && operator doesn't check second condition if first condition is false. It checks second condition only if first one is true.

The bitwise & operator always checks both conditions whether first condition is true or false.

```
1.  class OperatorExample{
2.  public static void main(String args[]){
3.  int a=10;
4.  int b=5;
5.  int c=20;
6.  System.out.println(a<b&&a<c);//false && true = false
7.  System.out.println(a<b&a<c);//false & true = false
8.  }}
```

Output:

```
false
false
```

## Java AND Operator Example: Logical && vs Bitwise &

```
1. class OperatorExample{
2. public static void main(String args[]){
3. int a=10;
4. int b=5;
5. int c=20;
6. System.out.println(a<b&&a++<c);//false && true = false
7. System.out.println(a);//10 because second condition is not checked
8. System.out.println(a<b&a++<c);//false && true = false
9. System.out.println(a);//11 because second condition is checked
10. }}
```

Output:

```
false
10
false
11
```

## Java OR Operator Example: Logical || and Bitwise |

The logical || operator doesn't check second condition if first condition is true. It checks second condition only if first one is false.

The bitwise | operator always checks both conditions whether first condition is true or false.

```
1. class OperatorExample{
2. public static void main(String args[]){
3. int a=10;
4. int b=5;
5. int c=20;
6. System.out.println(a>b||a<c);//true || true = true
7. System.out.println(a>b|a<c);//true | true = true
8. //|| vs |
```

```
9. System.out.println(a>b||a++<c);//true || true = true
10. System.out.println(a);//10 because second condition is not checked
11. System.out.println(a>b|a++<c);//true | true = true
12. System.out.println(a);//11 because second condition is checked
13. }}
```

Output:

```
true
true
true
10
true
11
```

## Java Ternary Operator Example

```
1. class OperatorExample{
2. public static void main(String args[]){
3. int a=2;
4. int b=5;
5. int min=(a<b)?a:b;
6. System.out.println(min);
7. }}
```

Output:

```
2
```

Another Example:

```
1. class OperatorExample{
2. public static void main(String args[]){
```

```
3. int a=10;
4. int b=5;
5. int min=(a<b)?a:b;
6. System.out.println(min);
7. }}
```

Output:

5

## Java Assignment Operator Example

```
1. class OperatorExample{
2. public static void main(String args[]){
3. int a=10;
4. int b=20;
5. a+=4;//a=a+4 (a=10+4)
6. b-=4;//b=b-4 (b=20-4)
7. System.out.println(a);
8. System.out.println(b);
9. }}
```

Output:

14

16

## Java Assignment Operator Example

```
1. class OperatorExample{
2. public static void main(String[] args){
3. int a=10;
4. a+=3;//10+3
```

```
5. System.out.println(a);
6. a-=4;//13-4
7. System.out.println(a);
8. a*=2;//9*2
9. System.out.println(a);
10. a/=2;//18/2
11. System.out.println(a);
12. }}
```

Output:

```
13
9
18
9
```

## Java Assignment Operator Example: Adding short

```
1. class OperatorExample{
2. public static void main(String args[]){
3. short a=10;
4. short b=10;
5. //a+=b;//a=a+b internally so fine
6. a=a+b;//Compile time error because 10+10=20 now int
7. System.out.println(a);
8. }}
```

Output:

```
Compile time error
```

After type cast:

```
1. class OperatorExample{
2. public static void main(String args[]){
3. short a=10;
4. short b=10;
5. a=(short)(a+b);//20 which is int now converted to short
6. System.out.println(a);
7. }}
```

Output:

```
20
```