**SNS COLLEGE OF TECHNOLOGY**

**(AN AUTONOMOUS INSTITUTION)**

COIMBATORE – 35

**DEPARTMENT OF COMPUTER SIENCE AND ENGINEERING**

## UNIT I- INTRODUCTION TO OOP

Object Oriented Programming concepts
Evolution of java
Java Architecture
Data Types
**Variables and Operators**
Environment setup
Command Line Arguments, Comments

### Java Variables

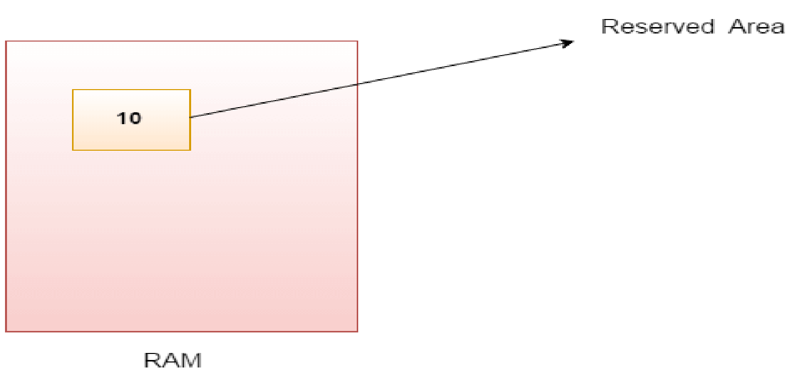
A variable is a container which holds the value while the Java program is executed. A variable is assigned with a data type.

Variable is a name of memory location. There are three types of variables in java: local, instance and static.

There are two types of data types in Java: primitive and non-primitive.

### Variable

A variable is the name of a reserved area allocated in memory. In other words, it is a name of the memory location. It is a combination of "vary + able" which means its value can be changed.

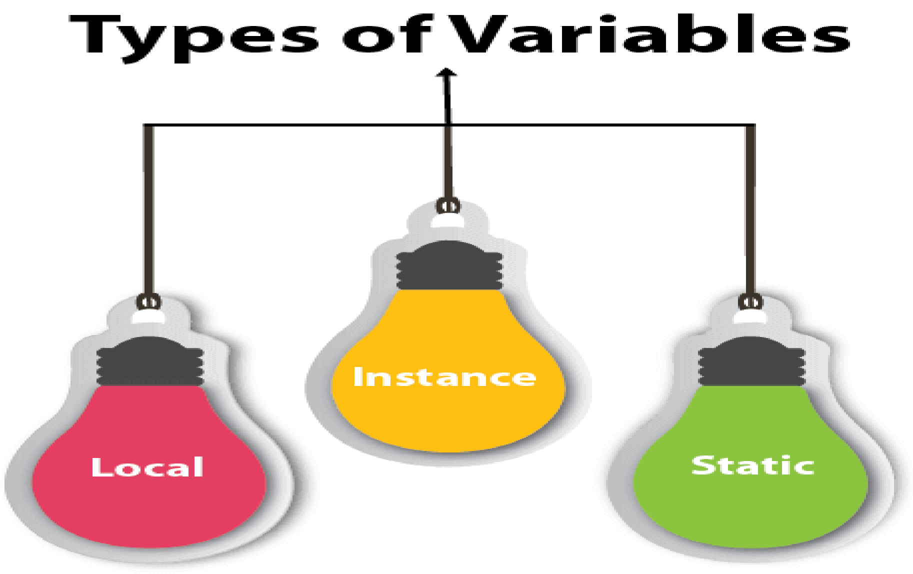


```
1.          int data=50;//Here data is variable
```

## Types of Variables

There are three types of variables in Java:

- local variable
- instance variable
- static variable



### 1) Local Variable

A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with "static" keyword.

### 2) Instance Variable

A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as static.

It is called an instance variable because its value is instance-specific and is not shared among instances.

### 3) Static variable

A variable that is declared as static is called a static variable. It cannot be local. You can create a single copy of the static variable and share it among all the instances of the class. Memory allocation for static variables happens only once when the class is loaded in the memory.

Example to understand the types of variables in java

```
public class A
{
   static int m=100;//static variable
   void method()
   {
      int n=90;//local variable
   }
   public static void main(String args[])
   {
      int data=50;//instance variable
   }
}//end of class
```

Java Variable Example: Add Two Numbers

```
public class Simple
{
public static void main(String[] args)
{
int a=10;
int b=10;
int c=a+b;
System.out.println(c);
}
}
```

**Output:**

```
20
```

## Java Variable Example: Widening

```java
public class Simple
{
public static void main(String[] args)
{
int a=10;
float f=a;
System.out.println(a);
System.out.println(f);
}
}
```

**Output:**

```
10
10.0
```

## Java Variable Example: Narrowing (Typecasting)

```java
public class Simple
{
public static void main(String[] args)
{
float f=10.5f;
//int a=f;//Compile time error
int a=(int)f;
System.out.println(f);
System.out.println(a);
}
}
```

**Output:**

```
10.5
10
```

## Java Variable Example: Overflow

```java
class Simple
{
public static void main(String[] args)
{
//Overflow
int a=130;
byte b=(byte)a;
System.out.println(a);
System.out.println(b);
}
}
```

**Output:**

```
130
-126
```

## Java Variable Example: Adding Lower Type

```java
class Simple
{
public static void main(String[] args)
{
byte a=10;
byte b=10;
//byte c=a+b;//Compile Time Error: because a+b=20 will be int
byte c=(byte)(a+b);
System.out.println(c);
}
}
```

**Output:**

```
20
```

## 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.

## ava 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

The Java unary operators require only one operand. Unary operators are used to perform various operations i.e.:

- incrementing/decrementing a value by one
- negating an expression
- inverting the value of a boolean

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

```java
public class OperatorExample{
public static void main(String args[]){
int x=10;
System.out.println(x++);//10 (11)
System.out.println(++x);//12
System.out.println(x--);//12 (11)
System.out.println(--x);//10
}}
```

**Output:**

```
10
12
12
10
```

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

```java
public class OperatorExample{
public static void main(String args[]){
int a=10;
int b=10;
System.out.println(a++ + ++a);//10+12=22
System.out.println(b++ + b++);//10+11=21
}}
```

**Output:**

```
22
21
```

## Java Unary Operator Example: ~ and !

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

**Output:**

```
-11
9
false
true
```

## Java Arithmetic Operators

Java arithmetic operators are used to perform addition, subtraction, multiplication, and division. They act as basic mathematical operations.

## Java Arithmetic Operator Example

```
public class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
System.out.println(a+b);//15
System.out.println(a-b);//5
System.out.println(a*b);//50
System.out.println(a/b);//2
System.out.println(a%b);//0
}}
```

**Output:**

```
15
5
50
2
0
```

## Java Arithmetic Operator Example: Expression

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

**Output:**

```
21
```

## Java Left Shift Operator

The Java left shift operator << is used to shift all of the bits in a value to the left side of a specified number of times.

## Java Left Shift Operator Example

```
public class OperatorExample{
public static void main(String args[]){
System.out.println(10<<2);//10*2^2=10*4=40
System.out.println(10<<3);//10*2^3=10*8=80
System.out.println(20<<2);//20*2^2=20*4=80
System.out.println(15<<4);//15*2^4=15*16=240
}}
```

**Output:**

```
40
80
80
240
```

## Java Right Shift Operator

The Java right shift operator >> is used to move the value of the left operand to right by the number of bits specified by the right operand.

## Java Right Shift Operator Example

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

```
System.out.println(10>>2);//10/2^2=10/4=2
System.out.println(20>>2);//20/2^2=20/4=5
System.out.println(20>>3);//20/2^3=20/8=2
}}
```

**Output:**

```
2
5
2
```

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

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

**Output:**

```
5
5
-5
1073741819
```

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

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

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

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

**Output:**

```
false
false
```

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

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

**Output:**

```
false
10
false
11
```

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

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

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

```
public class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
int c=20;
System.out.println(a>b||a<c);//true || true = true
System.out.println(a>b|a<c);//true | true = true
//|| vs |
System.out.println(a>b||a++<c);//true || true = true
System.out.println(a);//10 because second condition is not checked
System.out.println(a>b|a++<c);//true | true = true
System.out.println(a);//11 because second condition is checked
}}
```

**Output:**

```
true
```

```
true
true
10
true
11
```

## Java Ternary Operator

Java Ternary operator is used as one line replacement for if-then-else statement and used a lot in Java programming. It is the only conditional operator which takes three operands.

## Java Ternary Operator Example

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

**Output:**

```
2
```

Another Example:

```
public class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
int min=(a<b)?a:b;
System.out.println(min);
}}
```

**Output:**

```
5
```

## Java Assignment Operator

Java assignment operator is one of the most common operators. It is used to assign the value on its right to the operand on its left.

## Java Assignment Operator Example

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

```
int a=10;
int b=20;
a+=4;//a=a+4 (a=10+4)
b-=4;//b=b-4 (b=20-4)
System.out.println(a);
System.out.println(b);
}}
```

**Output:**

```
14
16
```

## Java Assignment Operator Example

```
public class OperatorExample{
public static void main(String[] args){
int a=10;
a+=3;//10+3
System.out.println(a);
a-=4;//13-4
System.out.println(a);
a*=2;//9*2
System.out.println(a);
a/=2;//18/2
System.out.println(a);
}}
```

**Output:**

```
13
9
18
9
```

## Java Assignment Operator Example: Adding short

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

**Output:**

```
Compile time error
```

After type cast:

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

**Output:**

```
20
```