Lecture – 2 (Java Basic Syntax)

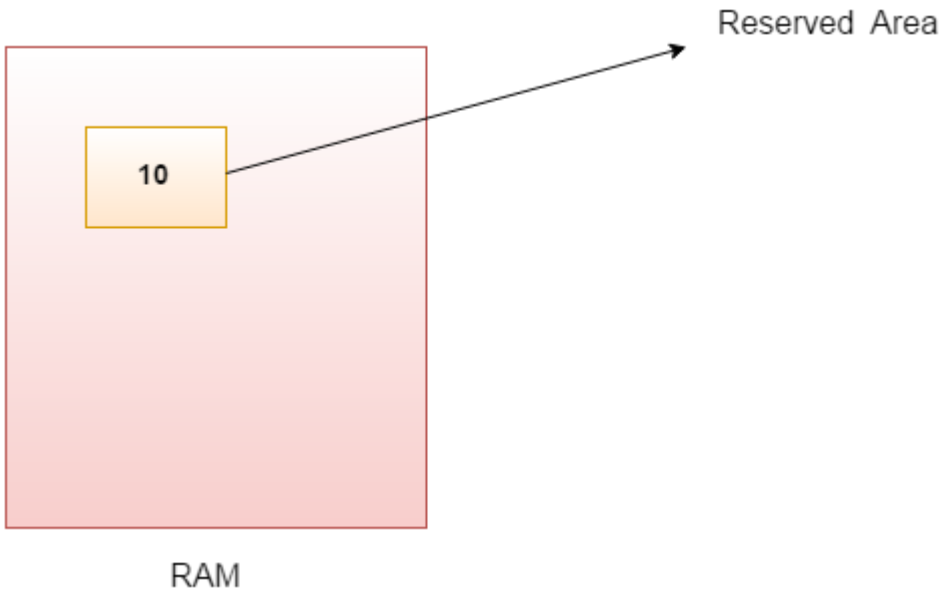
Variables and Data Types in Java

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

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

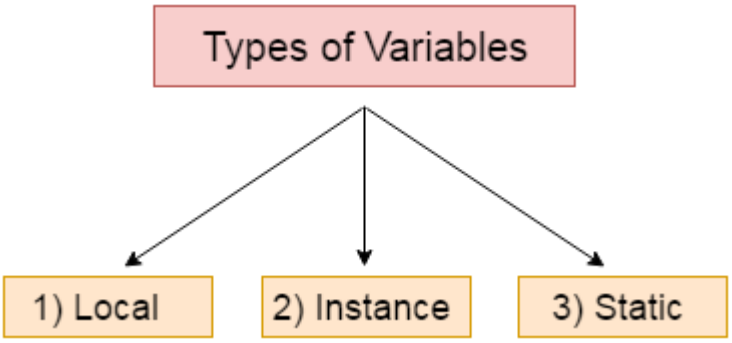


int data=50;//Here data is variable

Types of Variable

There are three types of variables in java:

- local variable
- instance variable
- static variable



1) Local Variable

A variable which is declared inside the method is called local variable.

2) Instance Variable

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

3) Static variable

A variable that is declared as static is called static variable. It cannot be local.

We will have detailed learning of these variables in next chapters.

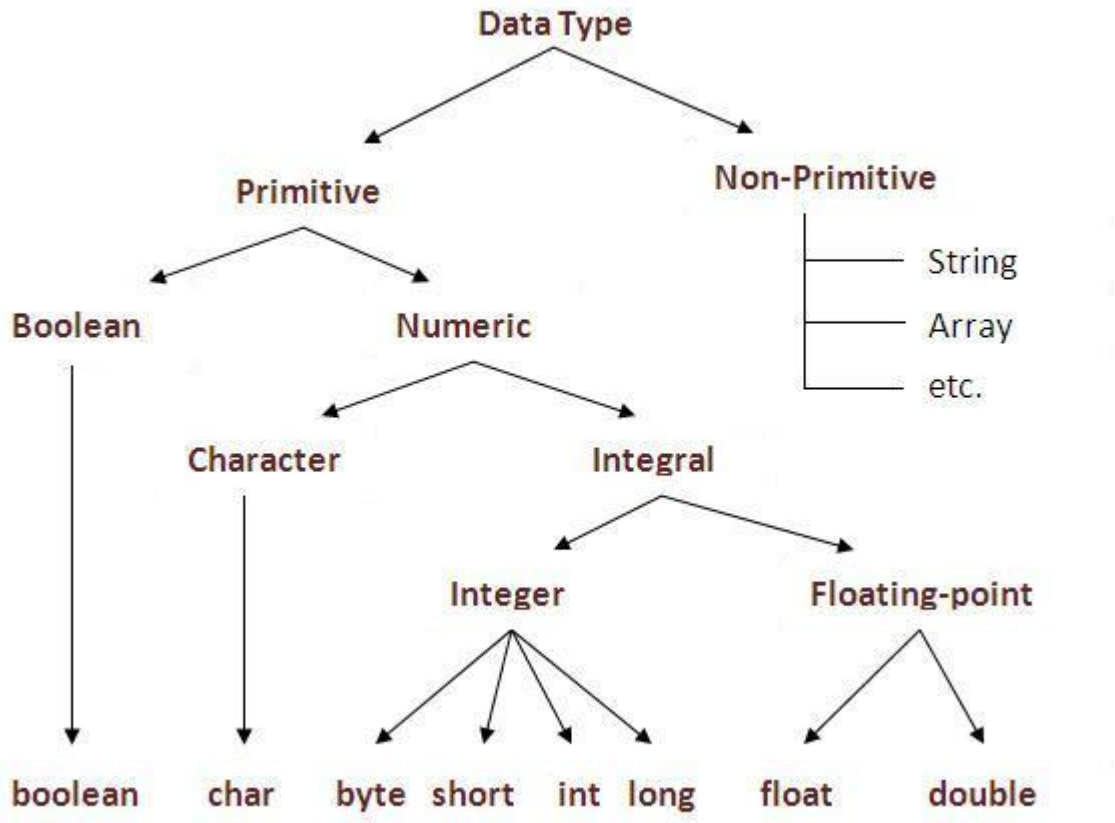
Example to understand the types of variables in java

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

Data Types in Java

Data types represent the different values to be stored in the variable. In java, there are two types of data types:

- Primitive data types
- Non-primitive data types



Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte

float	0.0f	4 byte
double	0.0d	8 byte

Why char uses 2 byte in java and what is \u0000 ?

It is because java uses Unicode system than ASCII code system. The \u0000 is the lowest range of Unicode system. To get detail explanation about Unicode visit next page.

Java Variable Example: Add Two Numbers

```
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

```
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)

```
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

```
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

```
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.

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 --

```
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 --

```
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 !

```
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 Operator Example

```
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

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

Output:
21

Java Shift Operator Example: Left Shift

```
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 Shift Operator Example: Right Shift

```
class 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 >>>

```
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 nagative 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 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.

```
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 &

```
    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 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.

```
    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 Example

```
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:

```
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 Example

```
    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

```
    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

```
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:

```
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

Java Comments

The java comments are statements that are not executed by the compiler and interpreter. The comments can be used to provide information or explanation about the variable, method, class or any statement. It can also be used to hide program code for specific time.

Java Single Line Comment

The single line comment is used to comment only one line.

Syntax:

```
//This is single line comment
```

Example:

```
public class CommentExample1 {
public static void main(String[] args) {
   int i=10;//Here, i is a variable
   System.out.println(i);
}
}
```

Output:

10

Java Multi Line Comment

The multi-line comment is used to comment multiple lines of code.

Syntax:

```
/*
This
is
multi line
comment
```

```
*/
```

Example:

```
public class CommentExample2 {
public static void main(String[] args) {
/* Let's declare and
 print variable in java. */
   int i=10;
   System.out.println(i);
}
}
```

Output:

10