Literals :

* Any constant which we can assign to the variable/field is called Literal.

* In Java, Initially We have 5 types of Literals :

a) Intergral Literal
b) Floating Point Literal
c) Boolean Literal
d) Character Literal
e) String Literal

Note : null is also literal.

Integeral Literal :

* If any numeric literal **does not** contain **decimal OR fraction then It is called Integral Literal.**
Example :
920, 78, 90, 12, 56

* In Integral Literal we have 4 data types which are as follows :
a) byte (8 bits)
b) short (16 bits)
c) int (32 bits)
d) long (64 bits)

An integral literal we can represent in four different forms :
a) Decimal Literal (Base is 10)
b) Octal Literal (Base is 8)
c) Hexadecimal Literal (Base is 16 [0-9 and a - f OR A - F])
d) Binary Literal (Base 2) [Available from JDK 1.7v]

Decimal Literal :

By default our numeric literals are decimal literal. Here base is 10 so, It accepts 10 digits i.e. from 0-9.

Example :
int x = 20;
int y = 223;
int z = 234;

Octal Literal :

If any Integer literal starts with 0 (Zero) then it will become octal literal. Here base is 8 so it will accept 8 digits i.e 0 to 7.

Example :
int a = 018; //Invalid because it contains digit 8 which is out of range
int b = 0777; //Valid
int c = 0123; //Valid

Hexadecimal Literal :

If any integric literal starts with 0X or 0x (Zero capital X Or 0 small x) then it is hexadecimal literal. Here base is 16 so it will accept 16 digits i.e 0 to 9 and A to F OR [a to f]

Example :

int x = 0X12; //Valid
int y = 0xadd; //Valid
int z = 0Xface; //valid
int a = 0Xage; //Inavlid because character 'g' out of range

Binary Literal :

If an Integral literal starts with 0B (Zero capital B) or 0b (Zero small b) then it will become Binary literal.
Binary literal is available from JDK 1.7v.
Here base is 2 so it will accept 2 digits i.e 0 and 1 only.

Example :

int x = 0B101; //valid
int y = 0b111; //Valid
int z = 0B112; //Invalid [2 is out of range]

By default our Integeral literals are Decimal Literal (Base 10), As a Programmer we can represent an integral literal in 4 different forms i.e decimal, octal, hexadecimal and Binary but **In order to get the output we need to convert into decimal number system.**

How to convert Octal to Decimal :

0 1 5 -> Here the number 0 describes that It is an octal Literal

1X8^1 + 5X8^0

8 + 5 = 13

Octal.java

```
void main()
{
    int x = 015;
    IO.println(x);  //Output is 13
}
```

How to convert hexadecimal to decimal :

0Xadd -> Here 0X describes that It is hexadecimal number.

a X 16^2 + d X 16^1 + d X 16^0

10 X 256 + 13 X 16 + 13 X 1

2560 + 208 + 13 = 2781

0 - 9
a = 10
b = 11
c = 12
d = 13
e = 14
f = 15

Hexadecimal.java

```
void main()
{
    int x = 0Xadd;
    IO.println(x);  //2781
}
```

How to convert binary to decimal :

0B101 : Here 0B describes that It is a binary number.

1 X 2^2 + 0 X 2^1 + 1 X 2^0

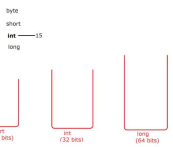
4 + 0 + 1 = 5

Binary.java

```
void main()
{
    int x = 0B101;
    IO.println(x);
}
```

*** By default every integral literal is of type int only so whenever we take an integral literal It must be represented by 32 bits (because It is int type)**

Example :
int x = 7;
00000000000000000000000000000111 [Internal Memory Representation in 32 bits]



byte data type Range -128 to 127

short data type Range -32768 to 32767

* by default every integral literal is of type int only, With this we have following cases :

Case 1 :

byte b = 1; //java compiler will automatically convert int into byte type but within the range only

byte c = 128; //Not compile because 128 is out of the range of byte [128 cannot be represented in 8 bits format]

Case 2 :

short s = 32555; //Java compiler will automatically convert int to short type but within the range only

short s = 32768; //Not compile because 32768 out of the range of short.

Case 3 :

int x = 90; // Here int value (90) we are assigning to int type

Case 4 :

long x = 12; //Java compiler will convert int (12) to long type [Aotomatic Type Casting OR Widening]

Case 5 :

long mob = 9812345678; //Not compiler this number is out of the range from integer number.

Case 6 :

long mob = 9812345678L; //This mobile number is treated as long value

Case 7 :

```
void main()
{
    accept(1); //1 IS int type so, searching int type of parameter
}

void accept(short x) //ERROR
{
    IO.println(x);
}

void accept(byte x) //ERROR
{
    IO.println(x);
}
```