Binary Literal :
----
* If an integral literal starts with **0B** (zero capital) OR **0b** (zero small b) then it will become Binary Literal. Base is 2 so, It will accept two digits 0 and 1. Binary Literal is available in java from JDK 1.7V.

Example :

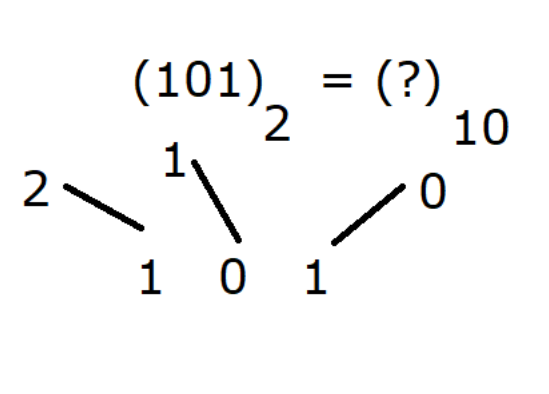
```
int x = 0B101; //Valid
int y = 0b111; //valid
int z = 0B112; //Invalid [Digit 2 is out of the range]
```

Program :

```
public class Test
{
    public static void main(String[] args)
    {
        int x = 0B101;
        System.out.println(x); //5
    }
}
```

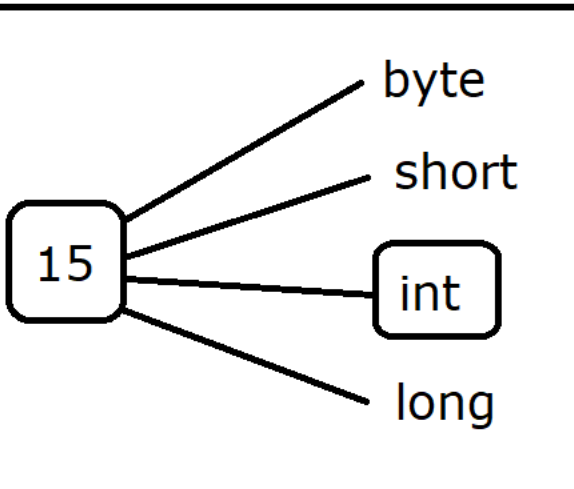
0B101 -> Here 0B describes that it is Binary Literal

$(101)_2 = (?)_{10}$



$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$4 + 0 + 1 = 5$



Internal Memory Representation :

```
byte b = 7;      00000111 [8 bits Representation]

short s = 7;     00000000 00000111 [16 bits Representation]
```

* By default every integral literal is of type **int** only. If we assign an integral literal to byte OR short data type then compiler will verify the range, If the values are within the range then It is possible to assign integral literal i.e int type value to byte OR short data data type.

Range of byte data type :   -128 to 127
Range of short data type : -32768 to 32767

Different Cases :

Case 1 :
----
byte b = 1; //Internally compiler will convert int to byte

Case 2 :
----
byte c = 127; //Internally Compiler will convert int to byte

Case 3 :
----
byte d = 128; //Compilation error because 128 is out of the range of byte

Case 4 :
----
short s = 32767; //Compiler will convert int to short type

Case 5 :
----
short p = 32768; //Compilation error 32768 is out of the range of short

Case 6 :
----
long x = 12; //12 int type is converted into long type, It is known as Implicit type Casting OR **Widening**

```
public class Test
{
    public static void main(String[] args)
    {
        accept(1);
    }

    public static void accept(byte b)
    {
        System.out.println(b);
    }
}
```

Note : Will generate CE

* If we want to represent long literal value then we should write L OR l (Capital L OR small l) as a suffix to integral literal so compiler will represent as a long literal using 64 bits.
* According to industry standard, L is more preferable because l looks lke digit 1.

Example : long mob = 9835641234L;

//Programs :

```
/* By default every integral literal is of type int only*/
public class Test4
{
public static void main(String[] args)
    {
      byte b = 128; //error
        System.out.println(b);

        short s = 32768; //error
        System.out.println(s);
    }
}

//Assigning smaller data type value to bigger data type
public class Test5
{
public static void main(String[] args)
    {
       byte b = 125;
        short s = b;  //Automatic type casting OR Widening
        System.out.println(s);

        short x = 345;
        int y = x;  //Automatic type casting OR Widening
        System.out.println(y);
    }
}
//Converting bigger type to smaller type
public class Test6
{
public static void main(String[] args)
    {
       short s = 127;
        byte b = (byte) s;  //Explicit Type Casting OR Narrowing
        System.out.println(b);
    }
}

public class Test7
 {
 public static void main(String[] args)
 {
        byte x = (byte) 18L;
        System.out.println("x value  = "+x);

        long l = 29L;
        System.out.println("l value  = "+l);

      int y = (int) 1L;
        System.out.println("y value  = "+y);
  }
 }
```

Is java pure object oriented language or not ?
----
* No, Java is not a pure object oriented langauge because It accepts primary data types like byte, short, int, long, float, double and so on.

* Except these 8 primitive data types, everything in java is object.

* Only objects are moving in the network but not the primitive data type.

Example :

```
class Student
{
  int id;
  int age;
  double fees;
}
```

Student raj = new Student();

Student priya = new Student();

* In order to convet these primitive data types into corresponding object, Java software people has introduced Wrapper classes concept.

* From JDK 1.5V. Java has provided the following two concepts :

1) Autoboxing (Converting the primitive data type into Wrapper object)
2) Unboxing (Converting Wrapper object back to primitive type)

Primitive Data type	Wrapper Object
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean