

Binary Number System, Java Operators & Taking User Input

In This Lecture

1. Binary Number System
2. Operators in Java
3. Taking User Input

Binary Number System

Convert Decimal To Binary

10

0 1

→ 0101001 →

2	49	1
2	24	0
2	12	0
2	6	0
2	3	1
	1	

49 → (110001)₂

5 → 101

6 → 110

2	26	0
2	13	1
2	6	0
2	3	1
	1	

(26)₁₀ → (11010)₂

Binary Number System

Convert Binary To Decimal

$$(110010)_2 = (50)_{10}$$

$$\begin{array}{cccccc} & 4 & 3 & 2 & 1 & 0 \\ (1 & 0 & 1 & 0 & 1 &)_2 & = & (21)_{10} \\ \hookrightarrow & 16 & + & 4 & + & 1 \end{array}$$

`int age = 21;`

5	4	3	2	1	0 ←
1	1	0	0	1	0
*	*	*	*	*	*
2^5	2^4	2^3	2^2	2^1	2^0
$32 + 16 + 0 + 0 + 2 + 0$					
$= 50$					

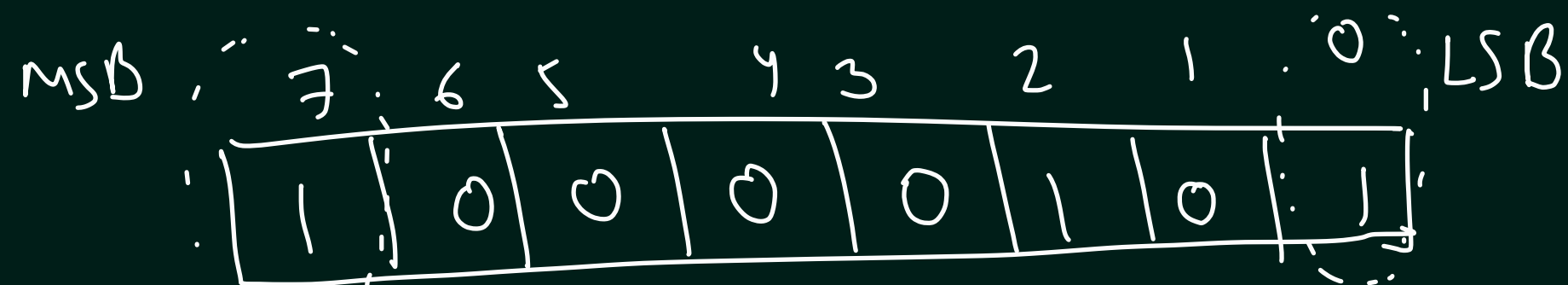
Binary Number System

Convert Binary To Decimal

```
int age = 128 ;
byte newAge = (byte) age;
```

↳ -128

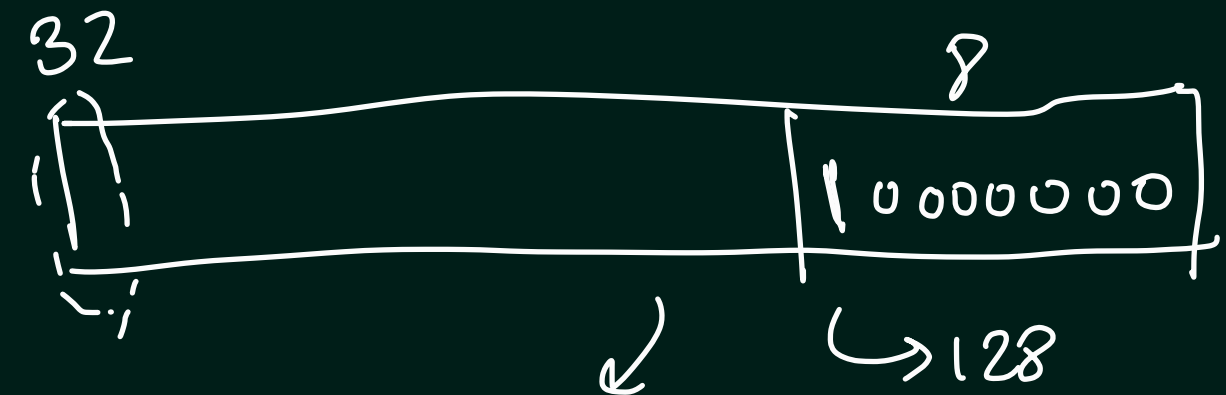
1 Byte → 8 bits



↳ 1 → -ve
↳ 0 → +ve

01111111
↓
+127

↓
-ve
10000000
↓
-128



1	0	0	0	0	0	0	0	
0	1	1	1	1	1	1	1	
							+	1
<hr/>								
1	0	0	0	0	0	0	0	
<hr/>								

Byte ←

Binary Addition

$$\begin{array}{r}
 1 \\
 00018 \\
 + 23 \\
 \hline
 41 \\
 \hline
 \end{array}$$

$$\begin{array}{r}
 1 \\
 0101 \rightarrow 5 \\
 1001 \rightarrow 9 \\
 \hline
 1110 \rightarrow 14 \\
 \hline
 \end{array}$$

10

$$\begin{array}{r}
 111 \\
 01110 \rightarrow 14 \\
 01011 \rightarrow 11 \\
 \hline
 11001 \rightarrow 25 \\
 \hline
 \end{array}$$

$$\begin{array}{r}
 11 \\
 1111 \\
 + 1110 \\
 \hline
 11101 \\
 \hline
 \end{array}$$

Binary Subtraction

$$9 - 4 = 5$$

$$9 + (-4) = 5$$

$$(-4)_{10} \rightarrow (\quad)_2$$

000000100

(1) 1111011

(2)
$$\begin{array}{r} + 1 \\ \hline 11111100 \end{array}$$

$$\begin{array}{r} 1001 \\ - 100 \\ \hline \end{array}$$

$$\begin{array}{r} 0001001 \leftarrow \\ + 1111000 \rightarrow (-4) \\ \hline 0000101 \end{array}$$

2's Complement

① swap the bits

② add 1

Binary Subtraction

$$13 - 6 = 7$$

1101

110 → 000110

↓ ①

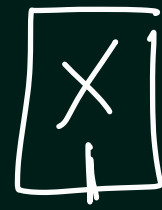
111001

↓ ②

1001

+ 1

1111010



11101

111101010

00010111

→ 7

Types of Operators in Java

1. Arithmetic Operators
2. Assignment Operators
3. Relational Operators
4. Logical Operators
5. Unary Operators
6. Bitwise Operators



1. Arithmetic Operators

Operator	Operation
<code>+</code>	Addition
<code>-</code>	Subtraction
<code>*</code>	Multiplication
<code>/</code>	Division
<code>%</code>	Modulo Operation (Remainder after division)

$$\begin{array}{r} 1 \\ 7 \overline{) 12} \\ \underline{7} \\ 5 \end{array}$$

$$12 \% 7 = 5$$

$$12 / 7 = 1$$

2. Assignment Operators





Operator	Example	Equivalent to
<code>=</code> ↙	↪ <code>a = b;</code>	<code>a = b;</code>
<code>+=</code>	→ <code>a += b;</code>	<code>a = a + b;</code>
<code>-=</code>	→ <code>a -= b;</code>	<code>a = a - b;</code>
<code>*=</code>	→ <code>a *= b;</code>	<code>a = a * b;</code>
<code>/=</code>	→ <code>a /= b;</code>	<code>a = a / b;</code>
<code>%=</code>	→ <code>a %= b;</code>	<code>a = a % b;</code>

3. Relational Operators

→ always return boolean value
→ true/false

Operator	Description	Example
<u>==</u>	Is Equal To	3 == 5 returns false
!=	Not Equal To	3 != 5 returns true
>	Greater Than	3 > 5 returns false
<	Less Than	3 < 5 returns true
>=	Greater Than or Equal To	3 >= 5 returns false
<=	Less Than or Equal To	3 <= 5 returns true

4. Logical Operators

Operator	Example	Meaning
 <code>&&</code> (Logical AND)	<code>expression1 && expression2</code>	<code>true</code> only if both <code>expression1</code> and <code>expression2</code> are <code>true</code> 
 <code> </code> (Logical OR)	<code>expression1 expression2</code>	<code>true</code> if either <code>expression1</code> or <code>expression2</code> is <code>true</code>
 <code>!</code> (Logical NOT)	<code>!expression</code>	<code>true</code> if <code>expression</code> is <code>false</code> and vice versa

a	b	y = a AND b	y = a OR b
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

AND OR

a	~a
0	1
1	0

5. Bitwise Operators

Operator	Description
<code>~</code>	Bitwise Complement
<code><<</code>	Left Shift
<code>>></code>	Right Shift
<code>>>></code>	Unsigned Right Shift
<code>&</code>	Bitwise AND
<code>^</code>	Bitwise exclusive OR

5 → 101 >>
 <<
 &
 1

Bitwise operation
↳ Bit manipulation

Other Operators

Increment/ Decrement Operators

++

--

$a++;$ \rightarrow $a = a + 1$

$a--;$

$\hookrightarrow a = a - 1;$

Ternary Operators

\hookrightarrow

?	:

 \leftarrow if else

Taking User Input using Scanner

In order to use the object of Scanner, we need to import java.util.Scanner package.

↑
Class

```
Scanner sc = new Scanner(System.in);
```

↳ `sc.nextInt()`

↳ `sc.nextFloat()`

↳ `sc.nextLine()`

↳ `sc.next()`

Various Input Types using Scanner

We can use `nextLong()`, `nextFloat()`, `nextDouble()`, and `next()` methods to get long, float, double, and string input respectively from the user.

Note: It is recommended to close the scanner object once the input is taken using the `close()` method