3.2. Operators

IT1406 - Introduction to Programming

Level I - Semester 1

3.2. Operators

- An operator is a function that has a special symbolic name and is invoked by using that symbol with an expression.

- Java has several kinds of operators:
  - Arithmetic Operators
    - These are used to perform standard arithmetic operations performed on numbers
  - Assignment Operator
    - This is used to assigns a variable with a value
  - Logical (Boolean) Operators
    - These are used to perform standard logical operations on Boolean values
  - Bitwise Operators
    - These perform operations on the individual bits of integers
  - Conditional Operator
    - This acts as a shorthand for if-then-else

3.2. Operators
Arithmetic Operators

Symbol	Meaning	Example	
+	Addition	3+4	=7
-	Subtraction	15-7	=8
*	Multiplication	7*8	=56
/	Division	15/3	=5
%	Modulus	59%7	=3

3.2. Operators
Arithmetic Operators

- In Java + operator has two roles
  - Arithmetic +

```
int a=1,b=3;
c = a+b; //Output is 4
```

  - Concatenation +

```
String a=“1”;
String b=“2”;
String c=a+b;//Output is 12
```

3.2. Operators
Assignment Operator

- Assignment Operator (=)

- For example:

```
int a;
a = 5;
```

- Right side of Assignment operator is evaluated first and then this value is assigned to the Left side of the Assignment operator

3.2. Operators
Assignment Operator

- Assignment Operator (=)

- For example:

```
int a;
a = 5;
int x,y,z,p;
x=y=z=p=a
```

- Right side of Assignment operator is evaluated first and then this value is assigned to the Left side of the Assignment operator

What would be the values of x,y,z,p?

5 value will be in each and every variable

3.2. Operators
Relational Operator

Symbol	Meaning	Example	
==	Equal to	2==3	= false
!=	not Equal to	2!=3	= true
>	Greater than	2>3	= false
<	Less than	2<3	= true
>=	Greater than or Equal	2>=3	= false
<=	Less than or Equal	2<=3	= true

3.2. Operators
Relational Operator

- One can create conditions using relational or comparison operators in Java

  Eg. a>b

  a<=t

  c==b

- In Java one cannot create conditions as shown below

  a>b<c

3.2. Operators
Logical (Boolean) Operators

Symbol	Meaning	Example	
&&	Short circuit Logical AND	true && false	= false
||	Short circuit Logical OR	true || false	= true
!	Logical NOT	!true	= false
&	Logical AND	true & false	=false
|	Logical OR	true | false	=true

Logical operators help to join conditions like a>b which output a boolean value true of false

3.2. Operators
Logical (Boolean) Operators

Condition1 output	**&**	**condition2 output**	**result**
(eg: a>b)	&	(eg: a!=b)	
true		true	true
true		false	false
false		true	false
false		false	false

(when logical AND is used both the conditions are checked and only if the both conditions true then the result become true as depicted in the above truth table)

3.2. Operators
Logical (Boolean) Operators

Condition1 output (eg: a>b)	&& &&	condition2 output (eg: a!=b)	result
true		true	true
true		false	false
false		~~true~~	false
false		~~false~~	false

(when short circuit logical AND is used both the conditions are checked and only if the both conditions true then the result become true as depicted in the above truth table)

If the first condition is false then second condition will not be checked and result become false directly making the process short circuit.

3.2. Operators
Logical (Boolean) Operators

Condition1 output (eg: a>b)	condition2 output (eg: a!=b)	result
true	true	true
true	false	true
false	true	true
false	false	false

(when logical OR is used both the conditions are checked and only if one condition is true from the both conditions then the result become true as depicted in the above truth table)

3.2. Operators
Logical (Boolean) Operators

Condition1 output (eg: a>b)	||	condition2 output (eg: a!=b)	result
true		~~true~~	true
true		~~false~~	false
false		true	false
false		false	false

(when short circuit logical OR(||) is used both the conditions are checked and if either one of the conditions is true from the both conditions then the result become true as depicted in the above truth table)

If the first condition is true then second condition will not be checked and result become true directly making the process short circuit

3.2. Operators
Logical (Boolean) Operators

Condition1 output (eg: a>b)		result
true	! (Logical NOT)	false

(when logical NOT(!) is applied the current value is converted to its opposite nature)

3.2. Operators
Assignment Shortcuts

- Often in programming, we have to apply some operator to some variable and then assign the resultant value back to the same variable. For example,

```
a = a + 5;
```

- Note, here we have two operations, one addition (a + 5) and one assignment (a = …).
- Java provides shortcut operators that combine the two operations.

```
a += 5;
```

- Similarly we have, -=, *=, /=, %=, &=, |= etc.

3.2. Operators
Increment (++) and Decrement (--)

- Very often we need to perform additions or subtractions by one (increments or decrements), that is expressions of the form a = a + 1 or a = a - 1. These can be done by the Increment Operator and Decrement Operator ++ and --.

```
a++;
a--;
```

3.2. Operators
Increment (++) and Decrement (--)

- When assign incremented or decremented values then the placement of the ++ or -- operator has a great impact in the expression

```
int a=0,b=1;
a=++b;// here a become 2 and b also become 2

int x=7,y=3;
x=y--;// here x become 3 after assignment and then y value get decremented by one and becomes  2
```

3.2. Operators
Bitwise Operator

Symbol	Meaning	Example	
&	Bitwise AND	0 & 1	= 0
|	Bitwise OR	0 | 1	= 1
^	Bitwise XOR	1^0	= 1
<<	Left Shift	1<<4	= 16
>>	Right Shift	256>>4	= 64
>>>	Zero filled right shift		
~	Bitwise Compliment		

3.2. Operators
Bitwise Operator

- Bitwise operators operate on numbers
- Bitwise operators treat each bit of a given value and do the required operations
- Java is a signed programming language

3.2. Operators
Bitwise Operator

Binary Base = 2

	Column 8	Column 7	Column 6	Column 5	Column 4	Column 3	Column 2	Column 1
Base^{exp}	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Weight	128	64	32	16	8	4	2	1

$2^0 = 1$

$2^1 = 2$

$2^2 = 2 * 2 = 4$

$2^3 = 2 * 2 * 2 = 8$

$2^4 = 2 * 2 * 2 * 2 = 16$

$2^5 = 2 * 2 * 2 * 2 * 2 = 32$

$2^6 = 2 * 2 * 2 * 2 * 2 * 2 = 64$

$2^7 = 2 * 2 * 2 * 2 * 2 * 2 * 2 = 128$

3.2. Operators
Bitwise Operator

- How to convert Decimal 7 to its binary representation?

3.2. Operators
Bitwise Operator

- Usage of Bitwise AND(&) is shown using the following truth table

Value1	&	Value2	result
1		1	1
1		0	0
0		1	0
0		0	0

3.2. Operators
Bitwise Operator

- Usage of Bitwise OR(|) is shown using the following truth table

Value1	|	Value2	result
1		1	1
1		0	1
0		1	1
0		0	0

3.2. Operators
Bitwise Operator

- Usage of Bitwise XOR(^) is shown using the following truth table

Value1	^	Value2	result
1		1	0
1		0	1
0		1	1
0		0	0

3.2. Operators
Bitwise Operator

- Usage of Bitwise left shift(<<) is shown here:

Value2 00111111

value2<<2

01111100

Signed will be remained

3.2. Operators
Bitwise Operator

- Usage of Bitwise right shift(>>) is shown here:

Value2   00111111

Value2>>2

00001111

Signed will be remained

3.2. Operators
Bitwise Operator

- Usage of Bitwise right shift(>>) is shown here:

Value2 00111111

Value2>>>2

00001111

Signed will NOT be remained

3.2. Operators
Bitwise Operator

- Usage of Bitwise complement(~) is shown here:

  Value2   00111111

~Value2// 1s will be 0s and verse visa

11000000

3.2. Operators
Conditional Operator

- **Conditional Operator (?:)/ternary if then else operator**

- **This operator has the form:**
  - **(condition)?(value if true):(value if false)**
  - **The condition is evaluated and if it is true value if true is returned, otherwise value if false is returned.**

- **For example, if a and b were integers, the following would return the maximum of a and b.**
  - **(a>b)?a:b**

- **This is a short hand for "if-then-else"**

3.2. Operators
Operator Precedence

- **Consider the following expression**
  6+2*3

- This consists of two operations, one addition and one multiplication.

- The value of the expression depends on which of these operations is performed first.
  - If addition is performed first we get:
    6+2*3 = (6+2)*3 = 8*3 = 24
  - If multiplication is performed first we get:
    6+2*3 = 6+(2*3) = 6+6 = 12

- To avoid ambiguity and confusion, java defines a clear order in which operators are evaluated. This is known as operator precedence.

- According to operator precedence, the multiplication operator (*) has higher precedence than the addition operator (+). Hence, the correct evaluation of **6+2*3** is **12.**

3.2. Operators
Operator Precedence

- The complete operator precedence is as follows.
- Operators at the top have higher precedence.
- Operators at the same level have equal precedence

Highest	. [] ( )	
	++ -- ! ~	Increment Operators
	new	
	* / %	Arithmetic Operators
	+ -	
	<< >> >>>	Bitwise Shift Operators
	< > <= >=	Comparison Operators
	== !=	
	&	Bitwise Operators
	^	
	|	
	&&	Logical Operators
	||	
	?:	Conditional Operator
	= += - = etc.	Assignment Operators
Lowest	&= |= <<= etc.	