Operators

① Arithamatic operators

Our Brain does Mathematicals, Analyticals, logicals, polytical and other operations.

Similarly computes can do some specials operations programming need this operations to perform some work or get result

Types of operators

1) Arithametic Operators (Mathamaticals operations)

+	Addition	(performs Addition)
-	Subtraction	(" difference)
*	Multiplication	(" product operation)
/	Division	(" division)
%	Modulus	(Finds the Remander

2) Relational operators (output is Boolean true or false)

= = Equal to operators (check two values equal or not)

! = Not equal to operators (check two values are not same or not)

> Greater than symbol (compares "a" value is greater than "b" value or not.

< less than symbol (compares "a" value is lesser than "b" value or not.

> = Greater than or equal to (performs the task of both equal to & greater operators.

< = less than or equal to (performs the task of both equal to & less than operator.

3> logical operators (playing with Boolean values)

a) AND
b) OR
c) NOT

Result becomes true (1) only when x & y are 1 & 1 or else false

@ AND

x	y	Result
0	0	0
0	1	0
1	0	0
1	1	1

(b) OR

x	y	Result
0	0	0
0	1	1
1	0	1
1	1	1

The Result is 0 false when Both x & y are false 0

Else Result is always true if any one of the x OR y is true Result is True

@ NOT → Gives the opposite output for the Result

~1 → 0 ~True → False

~0 → 1 ~False → True

@ Bitwise Operators (works ON the Bit of the Entity)

Bitwise AND
Bitwise OR
Bitwise XOR
Bitwise NOT
left shift
Right shift

operations happens at each bit of the number of Entity.

Bitwise AND OR performs similar task as logical and & logical OR But on each bit it performs the task

XOR

x	y	Result
0	0	0
0	1	1
1	0	1
1	1	0

Result is true when x, y are different if x, y, are same bits Result is false

left Swift << Shifts the Bit to the left side

Right swift >> Shifts the Bit to the Right side

~~left swift~~ increase the value 2^x times

~~Right swift decrease~~ the value $\frac{1}{2^x}$ times

---

a=5 b=2 AND operation

5 → 1 0 1

2 → 0 1 0 &

0 0 0

5 & 2 = 0

a=5 b=2 or operation

5 → 1 0 1

2 → 0 1 0 OR or operation

1 1 1 5 | 2 = 7

a=5 b=2 xor operation

5 → 1 0 1

2 → 0 1 0 XOR a ^ b = 7

7 → 1 1 1 5 ^ 2 = 7

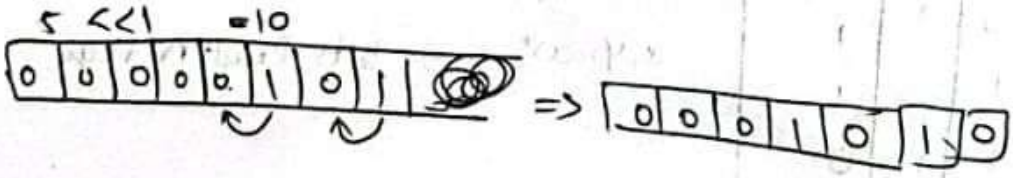
a=5

~5 = 101 NOT operator

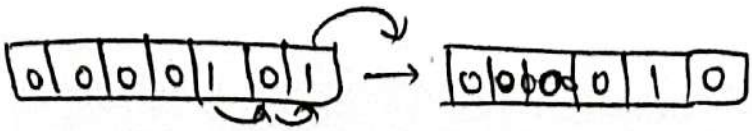
~101 ⇒ 010 = -6

a=5

5 <<1 =10



a=5 a>>1 = 2



a<<1 → a x 2^1	a>>1 $a/2^1$
a<<2 → a x 2^2	a>>2 $a/2^2$
a<<n → a x 2^n	a>>n $a/2^n$

⑤ Assignment operators

assignment operators assigns the value

total_score = final_score

value stored in total_score

+ = Number = Number + Some value

- = Number = Number - Some value

* = Number = Number * Some value

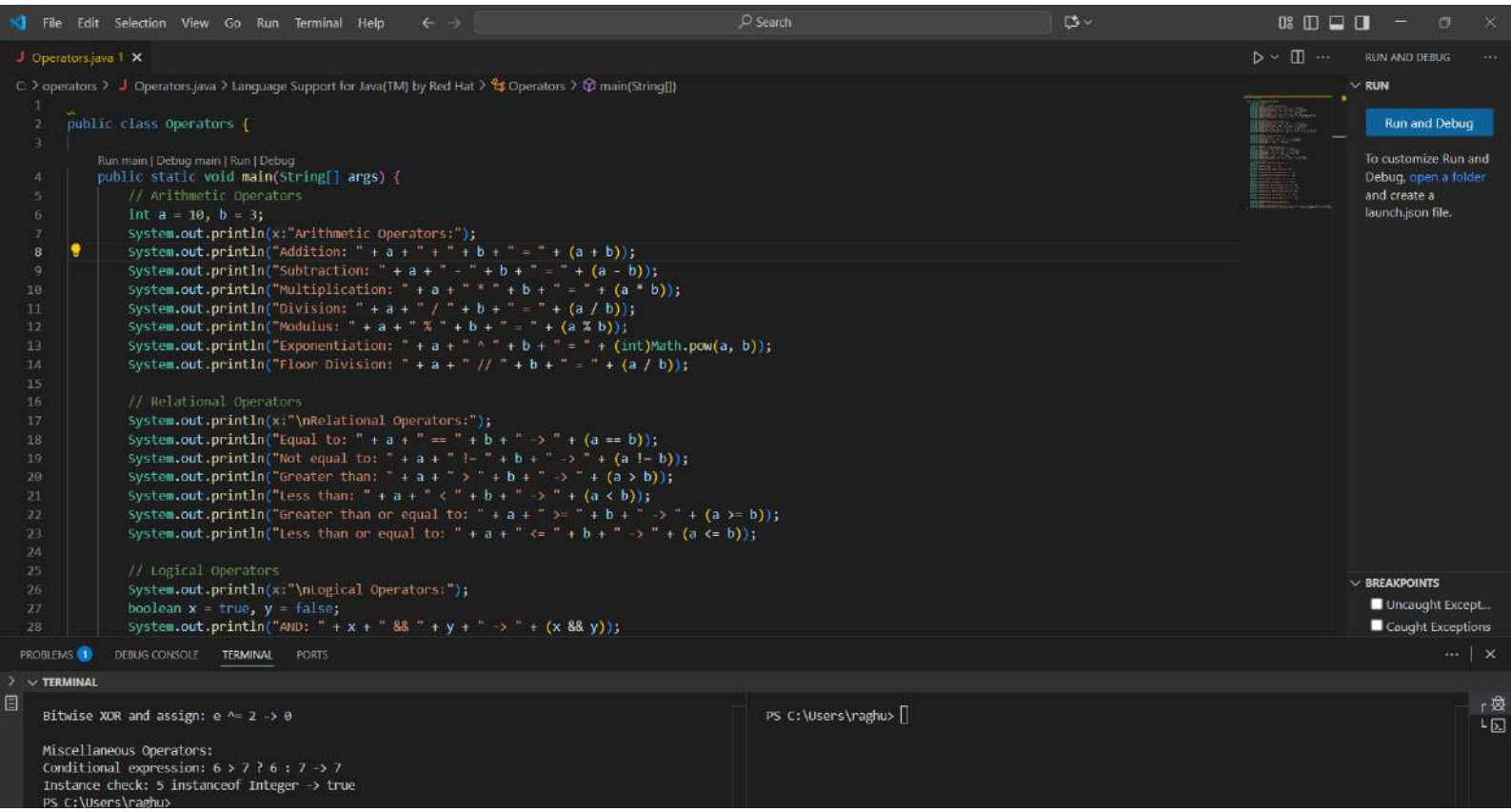
% = Number = Number % Some value

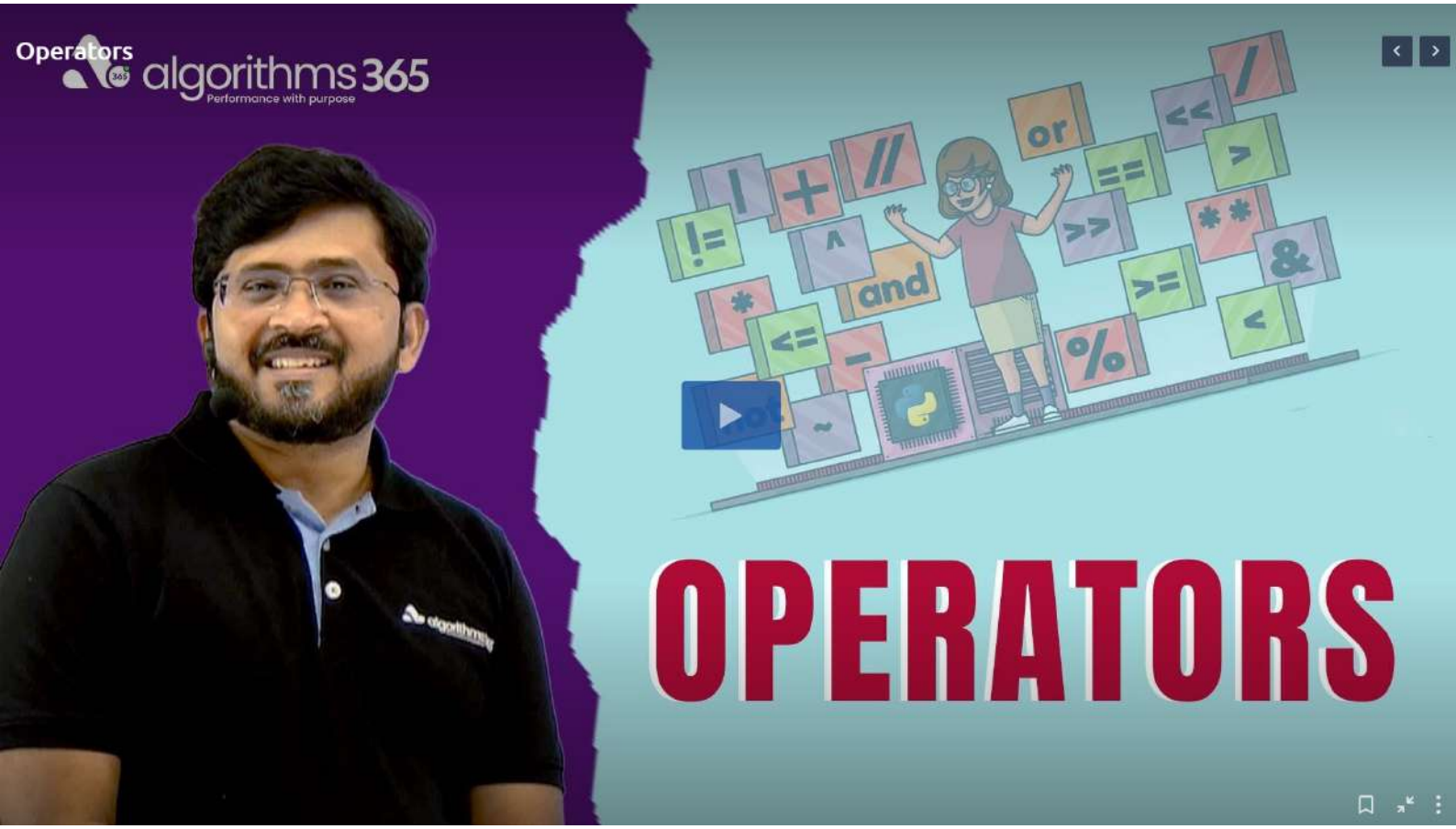
/ = Number = Number / Some value

Category	Operator	C	Java	Python
Arithmetic	Addition	+	+	+
	Subtraction	-	-	-
	Multiplication	*	*	*
	Division	/	/	/
	Modulus	%	%	%
	Increment	++	++	+= 1
	Decrement	--	--	-= 1
Relational	Equal to	==	==	==
	Not equal to	!=	!=	!=
	Greater than	>	>	>
	Less than	<	<	<
	Greater than or equal to	>=	>=	>=
	Less than or equal to	<=	<=	<=
Logical	AND	&&	&&	and
	OR	`		`
	NOT	!	!	not
Bitwise	AND	&	&	&
	OR	`	`	`
	XOR	^	^	^

	NOT	~	~	~
	Left shift	<<	<<	<<
	Right shift	>>	>>	>>
Assignment	Assign	=	=	=
	Add and assign	+=	+=	+=
	Subtract and assign	-=	-=	-=
	Multiply and assign	*=	*=	*=
	Divide and assign	/=	/=	/=
	Modulus and assign	%=	%=	%=
	Bitwise AND and assign	&=	&=	&=
	Bitwise OR and assign	`	=`	`
	Bitwise XOR and assign	^=	^=	^=
	Left shift and assign	<<=	<<=	<<=
	Right shift and assign	>>=	>>=	>>=
Miscellaneous	Conditional	? :	? :	if else
	Comma	,	,	,
	Size of	sizeof	N/A	N/A
	Cast	(type)	(type)	N/A

	Instance of	N/A	instanceof	isinstance


File Edit Selection View Go Run Terminal Help
Search
Operators.java 1
C: > operators > Operators.java > Language Support for Java(TM) by Red Hat > Operators > main(String[])
public class Operators {
Run main | Debug main | Run | Debug
public static void main(String[] args) {
// Arithmetic Operators
int a = 10, b = 3;
System.out.println(x:"Arithmetic Operators:");
System.out.println("Addition: " + a + " + " + b + " = " + (a + b));
System.out.println("Subtraction: " + a + " - " + b + " = " + (a - b));
System.out.println("Multiplication: " + a + " * " + b + " = " + (a * b));
System.out.println("Division: " + a + " / " + b + " = " + (a / b));
System.out.println("Modulus: " + a + " % " + b + " = " + (a % b));
System.out.println("Exponentiation: " + a + " ^ " + b + " = " + (int)Math.pow(a, b));
System.out.println("Floor Division: " + a + " // " + b + " = " + (a / b));
// Relational Operators
System.out.println(x:"\nRelational Operators:");
System.out.println("Equal to: " + a + " == " + b + " -> " + (a == b));
System.out.println("Not equal to: " + a + " != " + b + " -> " + (a != b));
System.out.println("Greater than: " + a + " > " + b + " -> " + (a > b));
System.out.println("Less than: " + a + " < " + b + " -> " + (a < b));
System.out.println("Greater than or equal to: " + a + " >= " + b + " -> " + (a >= b));
System.out.println("Less than or equal to: " + a + " <= " + b + " -> " + (a <= b));
// Logical Operators
System.out.println(x:"\nLogical Operators:");
boolean x = true, y = false;
System.out.println("AND: " + x + " && " + y + " -> " + (x && y));
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
BREAKPOINTS
Uncaught Except...
Caught Exceptions
PROBLEMS 1
DEBUG CONSOLE
TERMINAL
PORTS
TERMINAL
Bitwise XOR and assign: e ^= 2 -> 0
Miscellaneous Operators:
Conditional expression: 6 > 7 ? 6 : 7 -> 7
Instance check: 5 instanceof Integer -> true
PS C:\Users\raghu>
PS C:\Users\raghu>


Operators
algorithms 365
Performance with purpose
OPERATORS