**Operators**

“An operator is a symbol that allows a programmer to perform certain operation like (arithmetic or logical) on data and variables (operands)”.

|  |  |
| --- | --- |
| **Arithmetic** | **(+, /, %, \*, -, ++, --)** |
| **Assignment** | **(=, +=, -=, \*=, /=, %=)** |
| **Relational** | **(==, <, >, <=, >=, !=)** |
| **Instanceof** | **(instanceof)** |
| **Logical** | **(&, |, !, &&, ||)** |
| **Condition** | **(?:)** |

Depending numbers of operands, operators can be of the following three types:

**Unary operators:** It takes one operand, such as ++x, y--, here x and y are variables and ++ and -- are operators.

**Binary operators:** It takes two operands, such as x+y, x>y, here x and y are variables while + and > are operators.

**Ternary operators:** It takes three operands, z = x>y ?x:y, here x>y, x and y are operands while ?: is an operator.

**Assignment operators**

The equal (=) is known as the assignment operator. The assignment operator is used to store or assign a value to a variable.

Simple assignment operator (=)

Compound assignment operator (+=, -=, \*=, /=, %=)

**Examples: If i=5**

|  |  |  |
| --- | --- | --- |
| **i+=25** |  |  |
| **i-=10** |  |  |
| **i\*=5** |  |  |
| **i/=10** |  |  |
| **i%3** |  |  |

**Relational operator:**

It is use to comparison and it will return boolean value.

Example:

Class Orange{

public static void main(String[] args){

Orange o1 = new Orange();

Orange o2 = new Orange();

}

}

o1==o2 possible // to check if o1 and o2 are equal and have same object reference.

o1!=o2 possible// to check o1 and o2 are not equal and they doesn’t have same object reference.

**instanceof operator**

The instanceof operator is a binary operator that checks whether an object is of particular type (here type can be class, interface or an array). It is used for object or reference variable only.

You can’t use instanceof operator to test across two different class hierarchies.

Example:

Class Demo{

public static void main(String…args){

Demo d1 = new Demo();

System.out.println(d1 instanceof Demo);

d1=null;

System.out.println(d1 instanceof Demo);

}

}

o/p=true

false

**Logical operators**

|  |  |
| --- | --- |
| Category | Operators |
| AND(bit-wise) | **&** |
| OR(bit-wise) | **|** |
| NOT | **!** |
| OR(short-circuit) | **||** |
| AND(short-circuit) | **&&** |

The NOT (!) operator returns the opposite of the current value of a

boolean operand.

The operators || and && evaluates only boolean values.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | A&B | A&&B | A|B | A||B |
| T | **T** | **T** | **T** | **T** | **T** |
| T | **F** | **F** | **F** | **T** | **T** |
| F | **T** | **F** | **F** | **T** | **T** |
| F | **F** | **F** | **F** | **F** | **F** |

**<Q> what is the difference between | and ||, & and &&?**

Ans: (|| and |) If the first condition is true then || does not check

for second condition, whereas | checks the both.

(&& and &) If the first condition is false then && does not check

for second condition, whereas & checks the both**.**

**Conditional operators**

The conditional operator (?:) is a ternary operator that takes three operands. It works similar to if-else statement.

**Syntax:**

operand1?operand2 : operand3

The first operand is a boolean expression, if the expression is true then the value of the second operand is returned otherwise the value of the third operand is returned.

boolean expression?value1:value2

Example:

int i=20, j=25;

boolean test=false;

test = i<j ? true:false;

**Scanner class**

It is a class which help us to take input from keyboard.

Example:

import java.util.Scanner;

Class Demo{

static public void main(String args[]){

**Scanner input = new Scanner(System.in);**

int i;

double d;

System.out.print(“Enter the value of i:”);

i=input.nextInt();

System.out.print(“Enter the value of d:”);

d=input.nextDouble();

System.out.println(“Value of i: “+i);

System.out.println(“Value of d: “+d);

input.nextLine(); // to remove the \n which is left by previous variable.

System.out.print(“Enter name: );

String name = input.nextLine();

} }

**Unary Operator**

Operator which work on single operand. It is an arithmetic operator.

++ Increment Operator. (Post and Pre)

-- Decrement Operator. (Post and Pre)

Post: Once use then increment.

Pre: Before use increment.

**Precedence and associativity**

We need to consider precedence and associativity shall be considered during arithmetic operators