COMPUTER (PRACTICAL WORK)
CLASS-IX: Java Arithmetic Operators with Examples

Operators constitute the basic building block to any programming language. Java too provides many types of operators which can be used according to the need to perform various calculation and functions be it logical, arithmetic, relational etc. They are classified based on the functionality they provide. Here are a few types:

1. Arithmetic Operators
2. Unary Operators
3. Assignment Operator
4. Relational Operators
5. Logical Operators
6. Ternary Operator
7. Bitwise Operators
8. Shift Operators

Addition(+): This operator is a binary operator and is used to add two operands.

```
class Addition {
    public static void main(String[] args)
    {
        int num1 = 10, num2 = 20, sum = 0;
        System.out.println("num1 = " + num1);
        System.out.println("num2 = " + num2);
        sum = num1 + num2;
        System.out.println("The sum = " + sum);
    }
}
```

Output:

```
num1 = 10
num2 = 20
The sum = 30
```

Subtraction(-): This operator is a binary operator and is used to subtract two operands.

```
class Subtraction {
    public static void main(String[] args)
    {
        int num1 = 20, num2 = 10, sub = 0;
        System.out.println("num1 = " + num1);
        System.out.println("num2 = " + num2);
        sub = num1 - num2;
        System.out.println("Subtraction = " + sub);
    }
}
```

Output:

```
num1 = 20
num2 = 10
Subtraction = 10
```

Multiplication(*): This operator is a binary operator and is used to multiply two operands.

```
class Multiplication {
    public static void main(String[] args)
    {
        int num1 = 20, num2 = 10, mult = 0;
        System.out.println("num1 = " + num1);
        System.out.println("num2 = " + num2);
        mult = num1 * num2;
        System.out.println("Multiplication = " + mult);
    }
}
```

Output:
num1 = 20
num2 = 10
Multiplication = 200

Division(/): This is a binary operator that is used to divide the first operand(dividend) by the second operand(divisor) and give the quotient as result.

```
class Division {
    public static void main(String[] args)
    {
        int num1 = 20, num2 = 10, div = 0;
        System.out.println("num1 = " + num1);
        System.out.println("num2 = " + num2);
        div = num1 / num2;
        System.out.println("Division = " + div);
    }
}
```

Output:
num1 = 20
num2 = 10
Division = 2

Modulus(%): This is a binary operator that is used to return the remainder when the first operand(dividend) is divided by the second operand(divisor).

```
class Modulus {
    public static void main(String[] args)
    {
        int num1 = 5, num2 = 2, mod = 0;
        System.out.println("num1 = " + num1);
        System.out.println("num2 = " + num2);
        mod = num1 % num2;
        System.out.println("Remainder = " + mod);
    }
}
```

Output:

num1 = 5
num2 = 2

COMPUTER – CLASS-IX

Java Assignment Operator

These operators are used to assign values to a variable. The left side operand of the assignment operator is a variable and the right side operand of the assignment operator is a value.

1. “=”: This is the simplest assignment operator which is used to assign the value on the right to the variable on the left. This is the basic definition of assignment operator and how does it functions.

```
class Assignment {
    public static void main(String[] args)
    {
        int num;
        String name;
        num = 10;
        name = "CJCS";
        System.out.println("num is assigned: " + num);
        System.out.println("name is assigned: " + name);
    }
}
```

Output:
num is assigned: 10
name is assigned: CJCS

2. “+=”: This operator is a compound of ‘+’ and ‘=’ operators. It operates by adding the current value of the variable on left to the value on the right and then assigning the result to the operand on the left.

```
class Assignment {
    public static void main(String[] args)
    {
        int num1 = 10, num2 = 20; " + num1);
        System.out.println("num2 = " + num2);
        num1 += num2;
        System.out.println("num1 = " + num1);
    }
}
```

Output:
num1 = 10
num2 = 20
num1 = 30

3. “-=”: This operator is a compound of ‘-‘ and ‘=’ operators. It operates by subtracting the value of the variable on right from the current value of the variable on the left and then assigning the result to the operand on the left.

```
class Assignment {
    public static void main(String[] args)
    {
        int num1 = 10, num2 = 20;
```

```
        System.out.println("num1 = " + num1);
        System.out.println("num2 = " + num2);
        num1 -= num2;
        System.out.println("num1 = " + num1);
    }
}
```

Output:
num1 = 10
num2 = 20
num1 = -10

4. “*=”: This operator is a compound of ‘*’ and ‘=’ operators. It operates by multiplying the current value of the variable on left to the value on the right and then assigning the result to the operand on the left.

```
class Assignment {
    public static void main(String[] args)
    {
        int num1 = 10, num2 = 20;
        System.out.println("num1 = " + num1);
        System.out.println("num2 = " + num2);
        num1 *= num2;
        System.out.println("num1 = " + num1);
    }
}
```

Output:
num1 = 10
num2 = 20
num1 = 200

5. “/=”: This operator is a compound of ‘/’ and ‘=’ operators. It operates by dividing the current value of the variable on left by the value on the right and then assigning the quotient to the operand on the left.

```
class Assignment {
    public static void main(String[] args)
    {
        int num1 = 20, num2 = 10;
        System.out.println("num1 = " + num1);
        System.out.println("num2 = " + num2);
        num1 /= num2;
        System.out.println("num1 = " + num1);
    }
}
```

Output:
num1 = 20
num2 = 10
num1 = 2

6. “%=”: This operator is a compound of ‘%’ and ‘=’ operators. It operates by dividing the current value of the variable on left by the value on the right and then assigning the remainder to the operand on the left.

```
class Assignment {
    public static void main(String[] args)
    {

        int num1 = 5, num2 = 3;

        System.out.println("num1 = " + num1);
        System.out.println("num2 = " + num2);
        num1 %= num2;
        System.out.println("num1 = " + num1);
    }
}
```

Output:

```
num1 = 5
num2 = 3
num1 = 2
```

COMPUTER – CLASS-IX
Relational Operators with Examples

1. 'Equal to' operator (==): This operator is used to check whether the two given operands are equal or not. The operator returns true if the operand at the left-hand side is equal to the right-hand side, else false.

```
class Relational {
   public static void main(String[] args)
   {
      int var1 = 5, var2 = 10, var3 = 5;
      System.out.println("Var1 = " + var1);
      System.out.println("Var2 = " + var2);
      System.out.println("Var3 = " + var3);
      System.out.println("var1 == var2: "+ (var1 == var2));
      System.out.println("var1 == var3: "+ (var1 == var3));
   }
}
```

Output:
Var1 = 5
Var2 = 10
Var3 = 5
var1 == var2: false
var1 == var3: true

2. 'Not equal to' Operator(!=): This operator is used to check whether the two given operands are equal or not. It functions opposite to that of the equal-to operator. It returns true if the operand at the left-hand side is not equal to the right-hand side, else false.

```
class Relational {
   public static void main(String[] args)
   {
      int var1 = 5, var2 = 10, var3 = 5;

      System.out.println("Var1 = " + var1);
      System.out.println("Var2 = " + var2);
      System.out.println("Var3 = " + var3);

      System.out.println("var1 == var2: "+ (var1 != var2));

      System.out.println("var1 == var3: "+ (var1 != var3));
   }
}
```

Output:
Var1 = 5
Var2 = 10
Var3 = 5
var1 == var2: true
var1 == var3: false

3. ‘Greater than’ operator(>): This checks whether the first operand is greater than the second operand or not. The operator returns true when the operand at the left-hand side is greater than the right-hand side.

```
class Relational {
   public static void main(String[] args)
   {
      int var1 = 30, var2 = 20, var3 = 5;

      System.out.println("Var1 = " + var1);
      System.out.println("Var2 = " + var2);
      System.out.println("Var3 = " + var3);

      System.out.println("var1 > var2: "+ (var1 > var2));

      System.out.println("var3 > var1: "+ (var3 >= var1));
   }
}
```

Output:
Var1 = 30
Var2 = 20
Var3 = 5
var1 > var2: true
var3 > var1: false

4. ‘Less than’ Operator(<): This checks whether the first operand is less than the second operand or not. The operator returns true when the operand at the left-hand side is less than the right-hand side. It functions opposite to that of the greater than operator.

```
class Relational {
   public static void main(String[] args)
   {
      int var1 = 10, var2 = 20, var3 = 5;

      System.out.println("Var1 = " + var1);
      System.out.println("Var2 = " + var2);
      System.out.println("Var3 = " + var3);

      System.out.println("var1 < var2: "+ (var1 < var2));

      System.out.println("var2 < var3: "+ (var2 < var3));
   }
}
```

Output:
Var1 = 10
Var2 = 20
Var3 = 5
var1 < var2: true
var2 < var3: false

5. 'Greater than or equal to' operator(>=): This checks whether the first operand is greater than or equal to the second operand or not. The operator returns true when the operand at the left-hand side is greater than or equal to the right-hand side.

```
class Relational {
   public static void main(String[] args)
   {
      int var1 = 20, var2 = 20, var3 = 10;

      System.out.println("Var1 = " + var1);
      System.out.println("Var2 = " + var2);
      System.out.println("Var3 = " + var3);

      System.out.println("var1 >= var2: "+ (var1 >= var2));

      System.out.println("var2 >= var3: "+ (var3 >= var1));
   }
}
```

Output:
Var1 = 20
Var2 = 20
Var3 = 10
var1 >= var2: true
var2 >= var3: false

6. 'Less than or equal to' Operator(<): This checks whether the first operand is less than or equal to the second operand or not. The operator returns true when the operand at the left-hand side is less than or equal to the right-hand side.

```
class Relational {
   public static void main(String[] args)
   {
      int var1 = 10, var2 = 10, var3 = 9;

      System.out.println("Var1 = " + var1);
      System.out.println("Var2 = " + var2);
      System.out.println("Var3 = " + var3);

      System.out.println("var1 <= var2: "+ (var1 <= var2));

      System.out.println("var2 <= var3: "+ (var2 <= var3));
   }
}
```

Output:
Var1 = 10
Var2 = 10
Var3 = 9
var1 <= var2: true
var2 <= var3: false

COMPUTER CLASS-IX
Logical Operators

Write into activity copy

1. 'Logical AND' Operator(&&): This operator returns true when both the conditions under consideration are satisfied or are true. If even one of the two yields false, the operator results false. For example, cond1 && cond2 returns true when both cond1 and cond2 are true (i.e. non-zero).

```
class Logical {
   public static void main(String[] args)
   {
      int a = 10, b = 20, c = 20, d = 0;

      System.out.println("Var1 = " + a);
      System.out.println("Var2 = " + b);
      System.out.println("Var3 = " + c);

      if ((a < b) && (b == c)) {
         d = a + b + c;
         System.out.println("The sum is: " + d);
      }
      else
         System.out.println("False conditions");
   }
}
```

Output:
Var1 = 10
Var2 = 20
Var3 = 20
The sum is: 50

2. 'Logical OR' Operator(||): This operator returns true when one of the two conditions under consideration are satisfied or are true. If even one of the two yields true, the operator results true. To make the result false, both the constraints need to return false.

```
class Logical {
   public static void main(String[] args)
   {
      int a = 10, b = 1, c = 10, d = 30;
      System.out.println("Var1 = " + a);
      System.out.println("Var2 = " + b);
      System.out.println("Var3 = " + c);
      System.out.println("Var4 = " + d);
      if (a > b || c == d)
         System.out.println("One or both"+ " the conditions are true");
      else
```

```
        System.out.println("Both the"+ " conditions are false");
    }
}
```
Output:
Var1 = 10
Var2 = 1
Var3 = 10
Var4 = 30
One or both the conditions are true

3. 'Logical NOT' Operator(!): Unlike the previous two, this is a unary operator and returns true when the condition under consideration is not satisfied or is a false condition. Basically, if the condition is false, the operation returns true and when the condition is true, the operation returns false.

```
class Logical {
    public static void main(String[] args)
    {
        int a = 10, b = 1;

        System.out.println("Var1 = " + a);
        System.out.println("Var2 = " + b);

        System.out.println("!(a < b) = " + !(a < b));
        System.out.println("!(a > b) = " + !(a > b));
    }
}
```
Output:
Var1 = 10
Var2 = 1
!(a < b) = true
!(a > b) = false

##

Bitwise operators in Java

Bitwise OR (|) –
This operator is binary operator, denoted by '|'. It returns bit by bit OR of input values, i.e,
if either of the bits is 1, it gives 1, else it gives 0.
For example,
a = 5 = 0101 (In Binary)
b = 7 = 0111 (In Binary)

Bitwise OR Operation of 5 and 7
```
  0101
| 0111
 _______
  0111  = 7 (In decimal)
```

Bitwise AND (&) –
This operator is binary operator, denoted by '&'. It returns bit by bit AND of input values, i.e,

if both bits are 1, it gives 1, else it gives 0.
For example,

a = 5 = 0101 (In Binary)
b = 7 = 0111 (In Binary)

Bitwise AND Operation of 5 and 7
 0101
& 0111

 _____
 0101 = 5 (In decimal)

Bitwise XOR (^) –
This operator is binary operator, denoted by '^'. It returns bit by bit XOR of input values,
i.e, if corresponding bits are different, it gives 1, else it gives 0.
For example,
a = 5 = 0101 (In Binary)
b = 7 = 0111 (In Binary)

Bitwise XOR Operation of 5 and 7
 0101
^ 0111

 _____
 0010 = 2 (In decimal)

Bitwise Complement (~) –
This operator is unary operator, denoted by '~'. It returns the one's compliment representation of the input value,
i.e, with all bits inversed, means it makes every 0 to 1, and every 1 to 0.
For example,
a = 5 = 0101 (In Binary)

Bitwise Compliment Operation of 5

~ 0101

 _____
 1010 = 10 (In decimal)
Note – Compiler will give 2's complement of that number, i.e., 2's compliment of 10 will be -6.

Examples

```
public class operators {
   public static void main(String[] args)
   {

      int a = 5;
      int b = 7;
      System.out.println("a&b = " + (a & b));
      System.out.println("a|b = " + (a | b));


      System.out.println("a^b = " + (a ^ b));


      System.out.println("~a = " + ~a);


      a &= b;
      System.out.println("a= " + a);
   }
}
```

Output :

```
a&b = 5
a|b = 7
a^b = 2
~a = -6
a= 5
```

DO PRACTICE FOR ALL OPERATOR PROGRAM, TEST WILL BE VERY SOON

COMPUTER – CLASS-IX
PRACTICE PROGRAMS (DO PRACTICE INTO ROUGH COPY)

1. Arithmetic operators are: +, -, *, /, %

```
public class ArithmeticOperatorDemo {
  public static void main(String args[]) {
    int num1 = 100;
    int num2 = 20;

    System.out.println("num1 + num2: " + (num1 + num2) );
    System.out.println("num1 - num2: " + (num1 - num2) );
    System.out.println("num1 * num2: " + (num1 * num2) );
    System.out.println("num1 / num2: " + (num1 / num2) );
    System.out.println("num1 % num2: " + (num1 % num2) );
  }
}
```

Output:

```
num1 + num2: 120
num1 - num2: 80
num1 * num2: 2000
num1 / num2: 5
num1 % num2: 0
```

2. Assignments operators in java are: =, +=, -=, *=, /=, %=

```
public class AssignmentOperatorDemo {
  public static void main(String args[]) {
    int num1 = 10;
    int num2 = 20;

    num2 = num1;
    System.out.println("= Output: "+num2);

    num2 += num1;
    System.out.println("+= Output: "+num2);
```

```
    num2 -= num1;
    System.out.println("-= Output: "+num2);

    num2 *= num1;
    System.out.println("*= Output: "+num2);

    num2 /= num1;
    System.out.println("/= Output: "+num2);

    num2 %= num1;
    System.out.println("%= Output: "+num2);
  }
}
```

Output:

```
= Output: 10
+= Output: 20
-= Output: 10
*= Output: 100
/= Output: 10
%= Output: 0
```

3. Auto-increment and Auto-decrement Operators ++ and --

num++ is equivalent to num=num+1;

num–- is equivalent to num=num-1;

Example of Auto-increment and Auto-decrement Operators

```
public class AutoOperatorDemo {
  public static void main(String args[]){
    int num1=100;
    int num2=200;
    num1++;
    num2--;
    System.out.println("num1++ is: "+num1);
    System.out.println("num2-- is: "+num2);
  }
}
```

Output:

```
num1++ is: 101
num2-- is: 199
```

4. Logical operators in java are: &&, ||, !

```
public class LogicalOperatorDemo {
  public static void main(String args[]) {
    boolean b1 = true;
    boolean b2 = false;

    System.out.println("b1 && b2: " + (b1&&b2));
    System.out.println("b1 || b2: " + (b1||b2));
    System.out.println("!(b1 && b2): " + !(b1&&b2));
  }
}
```

Output:

```
b1 && b2: false
b1 || b2: true
!(b1 && b2): true
```

5) Comparison(Relational) operators ==, !=, >, <, >=, <=

```
public class RelationalOperatorDemo {
  public static void main(String args[]) {
    int num1 = 10;
    int num2 = 50;
    if (num1==num2) {
        System.out.println("num1 and num2 are equal");
    }
    else{
        System.out.println("num1 and num2 are not equal");
    }

    if( num1 != num2 ){
        System.out.println("num1 and num2 are not equal");
    }
    else{
        System.out.println("num1 and num2 are equal");
    }

    if( num1 > num2 ){
        System.out.println("num1 is greater than num2");
    }
```

```
    else{
        System.out.println("num1 is not greater than num2");
    }

    if( num1 >= num2 ){
        System.out.println("num1 is greater than or equal to num2");
    }
    else{
        System.out.println("num1 is less than num2");
    }

    if( num1 < num2 ){
        System.out.println("num1 is less than num2");
    }
    else{
        System.out.println("num1 is not less than num2");
    }

    if( num1 <= num2){
        System.out.println("num1 is less than or equal to num2");
    }
    else{
        System.out.println("num1 is greater than num2");
    }
  }
}
```

Output:
num1 and num2 are not equal
num1 and num2 are not equal
num1 is not greater than num2
num1 is less than num2
num1 is less than num2
num1 is less than or equal to num2

6) Bitwise Operators There are six bitwise Operators: &, |, ^, ~, <<, >>
Example of Bitwise Operators

```
public class BitwiseOperatorDemo {
  public static void main(String args[]) {
```

```
    int num1 = 11;  /* 11 = 00001011 */
    int num2 = 22;  /* 22 = 00010110 */
    int result = 0;

    result = num1 & num2;
    System.out.println("num1 & num2: "+result);

    result = num1 | num2;
    System.out.println("num1 | num2: "+result);

    result = num1 ^ num2;
    System.out.println("num1 ^ num2: "+result);

    result = ~num1;
    System.out.println("~num1: "+result);

    result = num1 << 2;
    System.out.println("num1 << 2: "+result); result = num1 >> 2;
    System.out.println("num1 >> 2: "+result);
  }
}
```

Output:

```
num1 & num2: 2
num1 | num2: 31
num1 ^ num2: 29
~num1: -12
num1 << 2: 44 num1 >> 2: 2
```

7) Ternary Operator

variable num1 = (expression) ? value if true : value if false

```
public class TernaryOperatorDemo {

  public static void main(String args[]) {
      int num1, num2;
      num1 = 25;

         num2 = (num1 == 10) ? 100: 200;
```

```
        System.out.println( "num2: "+num2);


        num2 = (num1 == 25) ? 100: 200;
        System.out.println( "num2: "+num2);
  }
}
```

Output:

```
num2: 200
num2: 100
```

COMPUTER CLASS-IX
JAVA PROGRAMS

Note: Write down into activity copy.

1. Java Program To Calculate Area Of Circle

```
class AreaOfCircle
{
  public static void main(String args[])
   {

    Scanner s= new Scanner(System.in);

      System.out.println("Enter the radius:");
      double r= s.nextDouble();
      double  area=(22*r*r)/7 ;
      System.out.println("Area of Circle is: " + area);
  }
}
```

2. Java Program To Calculate Area Of Triangle

```
class AreaOfTriangle
{
  public static void main(String args[])
   {

    Scanner s= new Scanner(System.in);

      System.out.println("Enter the width of the Triangle:");
      double b= s.nextDouble();

      System.out.println("Enter the height of the Triangle:");
       double h= s.nextDouble();

            //Area = (width*height)/2
    double area=(b*h)/2;
    System.out.println("Area of Triangle is: " + area);
  }
}
```

3. Java Program To Find Area Of Rectangle

```
class AreaOfRectangle
```

```
{
  public static void main(String args[])
   {

    Scanner s= new Scanner(System.in);

       System.out.println("Enter the length:");
       double l= s.nextDouble();
      System.out.println("Enter the breadth:");
       double b= s.nextDouble();


        double  area=l*b;
    System.out.println("Area of Rectangle is: " + area);
  }
}
```

4. Java Program To Find Area Of Isosceles Triangle.

```
class AreaOfIsoscelesTriangle
{
  public static void main(String args[])
   {

       Scanner s= new Scanner(System.in);

       System.out.println("Enter the length of same sided");

       double a= s.nextDouble();

       System.out.println("Enter the side2 of the Triangle");

       double b= s.nextDouble();

       double  area=(b/4)*Math.sqrt((4*a*a)-(b*b));

       System.out.println("Area of Isosceles Triangle is: " + area);
  }
}
```

5. Java Program To Find Area of Parallelogram

```
class AreaOfParallelogram
{
  public static void main(String args[])
```

```
  {

    Scanner s= new Scanner(System.in);

       System.out.println("Enter the height:");
       double d1= s.nextDouble();
      System.out.println("Enter the breadth:");
       double d2= s.nextDouble()

        double  area=(d1*d2) ;
    System.out.println("Area of Parallelogram is: " + area);
  }
}
```

6. Java Program To Calculate Area Of Rhombus

```
class AreaOfRhombus
{
  public static void main(String args[])
   {

    Scanner s= new Scanner(System.in);

       System.out.println("Enter the diagonal 1:");
       double d1= s.nextDouble();
      System.out.println("Enter the diagonal 2:");
       double d2= s.nextDouble();

        double  area=(d1*d2)/2;
    System.out.println("Area of Rhombus is: " + area);
  }
}
```