Athematic Operators:

```
public class Arithmetic_Operators {

      public static void main(String[] args) {

            /*
              by using arthamatic operators in java we can do math
operations like
              addition, substraction, multiplication, division,modulo.
             */

            //  Addition "+"
            //  Subtraction  "-"
            //  Multiplication "*"
            //  Division "/"
            //  Modulo "%"

            int i=10;
            int j=20;

            int addtition=i+j; // adding the two values
            System.out.println("Addition of two values:"+addtition);

            int substraction=i-j; // substracting two values
            System.out.println("Subtracting the two
values:"+substraction);

            int multiplication=i*j; // Multiplying the two values
            System.out.println("Multiplication of two
values:"+multiplication);

            int division=i/j; // division will give the quotient when we
divide by value
            System.out.println("Division of two values:"+division);

            int modulo=i%j;   // modulo will give the remender when we
divided by value
            System.out.println("modulo of two values:"+modulo);



            // Operator precedence determines the order in which the
operators in an expression are evaluated

            // * / % this three are a higher precedence in java
            // + - this are a lower precedence in java

            int pre = 12 - 4 * 2; // as per precedence first 4*2 will
exgecute =8 then 8 substract with 12.
            System.out.println("precedence value before:"+pre);

            pre = (12 - 4) * 2; /*
```

```
                                        here we use () that is
grouping values that is we are telling compiler to
                                        first excegute values in
brackets then multiple with next value.
                                       */
            System.out.println("precedence value After:"+pre);

            // Result of data types

            // byte+int=int -->this will give  int value
            // short+int =int  -->this will give  int value
            // int +long = long --> this will give long value
            // long+float=float --> this will give the float value
            // float+double=double --> this will give double value
            // char+ int =int --> this will give int value
            // double+int

            byte b=1;
            short s=12;
            int i1=1234;
            long l=123456789;
            float f=123.3f;
            double d=123.45;


            System.out.println("Adding byte and short:"+b+s); // here we
are adding byte and short so it will give int value
            System.out.println("Adding short and int:"+s+i1); // here we
are adding short and int so it will give int value
            System.out.println("Adding int and long:"+i1+l); // here we
are adding int and long so it will give long value
            System.out.println("Adding long and float:"+i1+f); // here we
are adding long and float so it will give float value
            System.out.println("Adding float and double:"+f+d); // here we
are adding float and double so it will give double value
            System.out.println("Adding char and int:"+('a'+10)); // here
we are adding char and int so it will give int value
            /*
              when we are adding char with int char is converted into a
int values based on
              the Ascii A-Z =65-90 a-z=97-122
             */
            // as per ascii the valueof  "a" is a 97 when we add with 10
it will give 107
      }

}
```

Output:

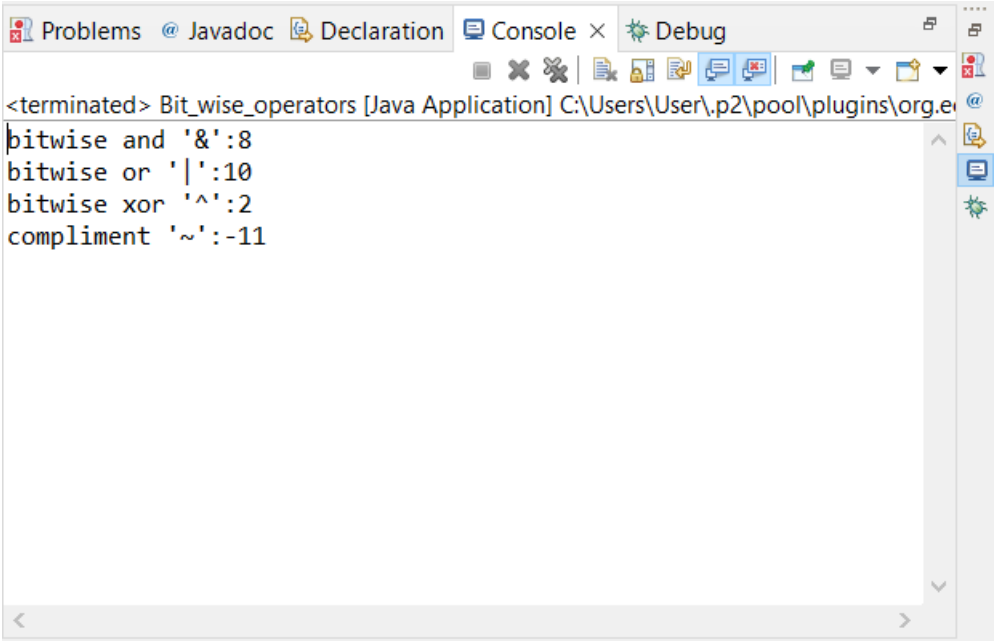
```
Problems  @ Javadoc  Declaration  Console ×  Debug
<terminated> Arithmetic_Operators [Java Application] C:\Users\User\.p2\pool\plugins\org
Addition of two values:30
Subtracting the two values:-10
Multiplication of two values:200
Division of two values:0
modulo of two values:10
precedence value before:4
precedence value After:16
Adding byte and short:112
Adding short and int:121234
Adding int and long:1234123456789
Adding long and float:1234123.3
Adding float and double:123.3123.45
Adding char and int:107
```

Bit wise operators:

```
public class Bit_wise_operators {

	public static void main(String[] args) {

		// bit wise operators are used to perform the manipulation of
individual bits of a numbers

		// & -->this is bit wise "and"  it will return true if both
are true otherwise it will return false
		// | --> this is a bitwise "or" it will true if any one is
true
		// ^ --> this is bitwise "xor" it will return true if both are
not equal
		// ~ --> this is compliment(not) it use 2"s compiliment to
return value
		// >> --> this is "rightswift" it will swift bits to rightside
based on the given value
		// << --> this is "leftswift" it will swift bits leftside
based on the given value

		int i=10;
		int j=8;
		// bitwise and
```

```
            int and=i&j;
            System.out.println("bitwise and '&':"+and); // this will print
8 as output.

          // bitwise or

            int or=i|j;
            System.out.println("bitwise or '|':"+or);  // this will print
10 as output.

            // bitwise xor

            int xor=i^j;
            System.out.println("bitwise xor '^':"+xor);   // this will
print 2 as output.

            // compliment(not)

            int compliment=~i;
            System.out.println("compliment '~':"+compliment);  // this
will print -11 as output.


      }

}
```

Out put:

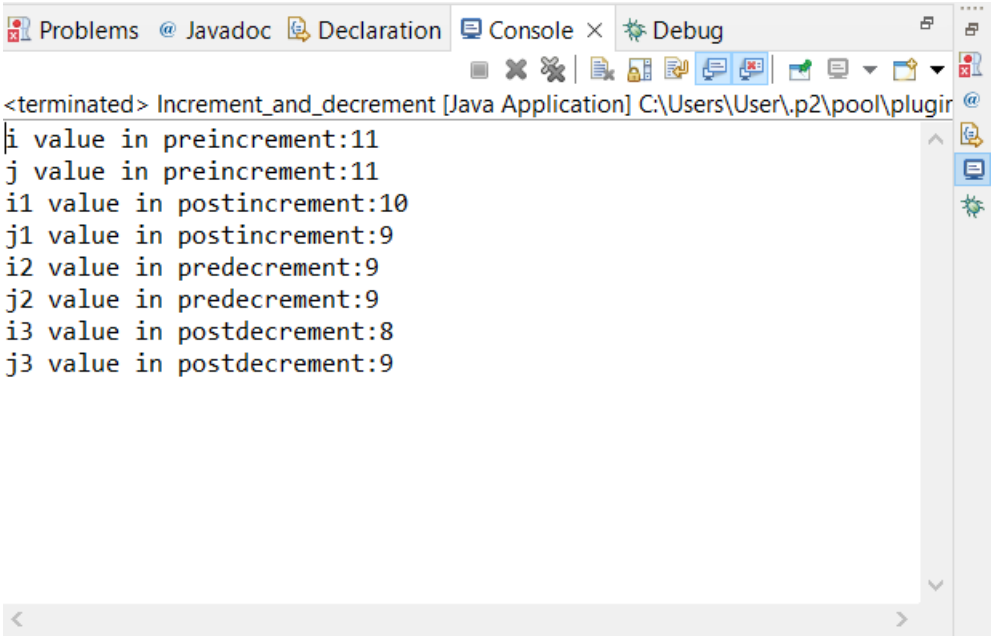


Increment and decrement:

```
public class Increment_and_decrement {

      public static void main(String[] args) {

            // Increment and decrement is a unary operators in java
            // by using that increment and decrement we can increase value
and we can decrease the value

            /*
              increment in java as two types:
              1.pre increment.
              2.post increment.

              pre increment means first we can increment the value then
its assign to
              another variable
              int i=9;
              int j=++i; here first i value increment by one and

             it will assing to j so, i =10 and j=10

              post increment means first we can assign the value then its
increment the
              value
              int i=9;
              int j=i++; here first i value assign then i value
incremented
              so, i =10 and j=9
             */

            int i=10;
            int j=++i;  // this is a preincrement first its increment i
than it assign to j
            System.out.println("i value in preincrement:"+i);
            System.out.println("j value in preincrement:"+j);

            int i1=9;
            int j1=i1++;  // this is a postincrement first its assign
value to j1 then it increment i1
            System.out.println("i1 value in postincrement:"+i1);
            System.out.println("j1 value in postincrement:"+j1);


            /*
              decrement in java as two types:
              1.pre decrement.
              2.post decrement.

              pre decrement means first we can decrement the value then
its assign to
```

```
                another variable
                int i=9;
                int j=--i; here first i value decrement by one and  it will
assing to j so, i =8 and j=8

                post decrement means first it will assign value then it will
decrease the value
                int i=9;
                int j=i--; here first i value assign then i value decrement
                so, i =9 and j=8
               */
              int i2=10;
              int j2=--i2;  // this is a predecrement first its decrement i2
than it assign to j2
              System.out.println("i2 value in predecrement:"+i2);
              System.out.println("j2 value in predecrement:"+j2);

              int i3=9;
              int j3=i3--;  // this is a postdecrement first its assign
value to j3 then its decrement i3
              System.out.println("i3 value in postdecrement:"+i3);
              System.out.println("j3 value in postdecrement:"+j3);

      }

}
```

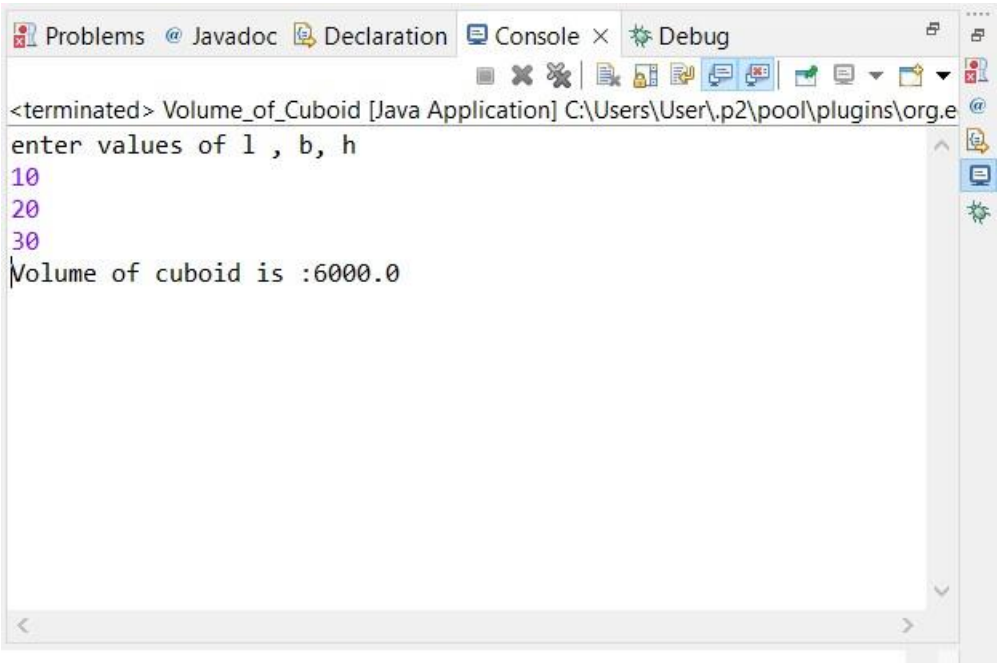
Output:



Volume of cuboid:

```
public class Volume_of_Cuboid {

public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);
        System.out.println("enter values of l , b, h");
        int l=sc.nextInt();
        int b=sc.nextInt();
        int h=sc.nextInt();

        double volume=l*b*h;
        System.out.println("Volume of cuboid is :"+volume);
}

}
```

Output:



Narrowing:

```
public class Narrowing {

      public static void main(String[] args) {

            // narrowing(downcasting) passing a high  datatype into a low
type datatype

            short s=1234;
            int i=12345;
            long l=12345678;
            char c='a';
            float f=1.2f;
            double d=12.3;

/*************************************************************************
*************************************/
         // byte
            byte b=s;    // this will show a error because we cannot store
short value in byte.

            Ex:
/*          b=i;
            b=l;        we cannot do like this
            b=c;
            b=f;
            b=d;   */

            b=(byte)s;   // this will work because we r  downcasting the
short to byte.
            b=(byte)i;
            b=(byte)l;
            b=(byte)c;
            b=(byte)f;
            b=(byte)d;

/*************************************************************************
*************************************/
            // short

      // in short we can store byte values but we not store
int,long,char,float,double, if u want to store for that we have to use
downcasting.

            short s1=i;    // this will show a error because we cannot
store int value in short.

                  Ex:

            /*          s1=l;       we cannot do like this but we can do
                        s1=c;
```

```
                        s1=f;
                        s1=d;   */


                        s1=(short)i;
                        s1=(short)l;   // this will work because we r
downcasting.
                        s1=(short)c;
                        s1=(short)f;
                        s1=(short)d;


/*************************************************************************
*************************************/

            // int

            // in int we can store byte,short,char values but we cannot
store long,float,double. if u want to store for that we have to use
downcasting.

                        int i1=l;    // this will show a error because we
cannot store long value in int.

                        Ex:

                  /*                     we cannot do like this but we
can do

                              i1=f;
                              i1=d;
                                      */


                              i1=(int)l;
                              i1=(int)f;    // this will work because we r
downcasting.
                              i1=(int)d;



/*************************************************************************
*************************************/
            // long

            // in long we can store byte,short,int,char but we cannot
store float,double.we can store by using downcasting.

            long l1=f; // this will show error because we cannot sotre
float value in long.

//          Ex:
            // l1=d;
```

```
      l1=(long)f;
      l1=(long)d;   //this will work because we r downcasting.

/*************************************************************************
*************************************/
            // float

      // in float we can store byte,short,long,char,int but we cannot
store double values ,if we can store by using downcasting.

      float f1=d;  // this will show error because  we cannot store double
value in float

      f1=(float)d;  // this work because we r downcasting.

/*************************************************************************
*************************************/

      // double
      // in double we can store byte,short,int,long,float,char.


      double d1=b;
      double d2=s;
      double d3=i;   // we can store byte,short,int,long,char,float values
      double d4=l;
      double d5=c;
      double d6=f;

/*************************************************************************
*************************************/

      //  in boolean we cannot store any other premitive datatype values

      // it will store only boolean values like true,false.

/*************************************************************************
*************************************/
      // char
      // in char we cannot store any values.

      char c1=(char)i;
      c1=(char)b;
      c1=(char)s;
      c1=(char)l;
      c1=(char)f;
      c1=(char)d;
}

}
```

Widening:

```
public class Widening {

      public static void main(String[] args) {

            // type casting is converting one data type into a another
data type
            /*
              widening(upcasting) is storing a lower datatype value into a
higher type type
              it is also know as implicit conversion.
             */


            short s=1234;
            int i=12345;
            long l=12345678;
            char c='a';
            float f=1.2f;
            double d=12.3;

/*************************************************************************
*************************************/
            // byte
            // byte can compactable only for byte values


            /*
              byte b=s;
              byte b=i;
              byte b=c;
              byte b=l;      this shows a compile time error
             */
            byte b=123;
            byte b1=b;   // byte can store only byte values

/*************************************************************************
*************************************/
            // short
            // short can compactable only with byte,short

            /*
              short s1=i;
              short s1=c;
              short s1=l;    we cannot store
int,long,char,float,double,boolean
              short s1=f;
              short s1=d;
             */
```

```
            short s1=b;
            short s2=s1; // short can store only byte and short

/*************************************************************************
*************************************/
            // int
            // int can compactable only with byte,short,int,char

            /*
              int i1=l;
              int i1=f;    // we can not store long,float,double,boolean
              int i1=d;
             */

            int i1=b;
            int i2=s;
            int i3=c;    // we can store only byte,short,char,int
            int i4=i;
/*************************************************************************
*************************************/
            // long
            // long can compactable with byte,short,int,char

            /*
              long l1=f;
              long l1=d;   // we cannot store float,double values and
boolean in long
             */

            long l1=b;
            long l2=s;
            long l3=i;   // we can store byte,int,short,char
            long l4=c;
/*************************************************************************
*************************************/
            // char
            // in implicity we cannot store any thing in char except char

            /*
              char c1=b;
              char c2=s;
              char c3=i;     // we cannot store
byte,short,int,long,boolean
              char c4=l;
              char c5=f;
              char c6=d;
             */
            char c1=c;  // we can store char
            char c2=123; // we can assign direct int values to char

/*************************************************************************
*************************************/
            // float
            // float can compactable with byte,short,int,long,char
```

```
      //  float f1=d;  we cannot store double values and boolean values

            float f1=b;
            float f2=s;
            float f3=i;    // we can store byte,short,int,long,char values
            float f4=l;
            float f5=c;


/*************************************************************************
*************************************/

            // double
            // double can compactable with byte,short,int,long,char,float

            double d1=b;
            double d2=s;
            double d3=i;   // we can store byte,short,int,long,char,float
values
            double  d4=l;
            double  d5=c;
            double d6=f;
      // we canot store boolean value in double

/*************************************************************************
*************************************/

            // boolean
            // boolean can compactable only with boolean datatypes

/*          boolean  b2=b;
            boolean  b3=s;
            boolean b4=i;
            boolean b5=l;   boolean cannot store any datatype values
            boolean b6=c;
            boolean b7=f;
            boolean b8=d;    */

      boolean bb=true;
      boolean bb1=bb;


      }

}
```

Quadratic expression:

```java
package Task6;

import java.util.Scanner;

public class Qudratic_Expression {

	static void quadratic_expression(){

		Scanner sc=new Scanner(System.in);
		System.out.println("enter ur input values a,b and c");
		int a=sc.nextInt();
		int b=sc.nextInt();
		int c=sc.nextInt();
		double determinant=b*b-4*a*c;
		double root1 = 0,root2 = 0;
		//checking if determinant is greater than 0.
		if(determinant>0)
		{
			root1=(-b+Math.sqrt(determinant))/(2*a);
			root2=(-b+Math.sqrt(determinant))/(2*a);
		System.out.format("root1 =%.2f and  root2=%.2f", root1,root2);
		}
		//checking if determinant is equal to 0
		else if(determinant==0)
		{
		root1=root2=-b/(2*a);
		System.out.format("root1=root2=%.2f",root1);
		}
		else
			//if determinant is less than 0.
		{
			//roots are complex number and distinct.
			double real=-b/(2*a);
			double imaginary=Math.sqrt(-determinant)/(2*a);
			System.out.format("root1=%.2f+%.2fi", real,imaginary);
			System.out.format("\nroot2=%.2f-%.2fi",real,imaginary);
		}

	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		quadratic_expression();
	}

}
```

Output:

Problems @ Javadoc Declaration Console × Debug

<terminated> Qudratic_Expression [Java Application] C:\Users\User\.p2\pool\plugins\org.

```
enter ur input values a,b and c
10
20
30
root1=-1.00+1.41i
root2=-1.00-1.41i
```