```
Operators
 1. new operator[oops]
 2. [] operator[Arrays]
 3. unary operator
 4. Arithmetic operator
 5. relational operator
 5. bitwise operator
 6. shortcircuit operator
 7. ternary operator
 8. Assignment operator


Java Operator precedence
++++++++++++++++++++++++
1. Unary operator : [],x++,x--,~,!,new
2. Arithmetic operator : *,/,%,+,-
3. shift operators : >>,>>>,<<,...
4. Comparison operator : <,<=,>,>=
5. Equality operator : ==,!=
6. Bitwise operator  : &,^,|
7. Logical/Shortcircuit: &&,||
8. Ternary operator : ? :
9. Assignment operator : =,+=,-=,*=,/=,%=,...

Note: Associativity for Ternary,increment and Assignment is from [Right to Left]
where as for other operators it is  [Left to Right]

Solve the expression
++++++++++++++++++++
1.
x = 1+2*3/4*5+6[Left to Right]
  = 1+6/4*5+6
  = 1+1*5+6
  = 1+5+6
  = 6+6
  = 12

2.
x = 5*3/4+5-6/2
  = 15/4+5-6/2
  = 3+5-6/2
  = 3+5-3
  = 8-3
  = 5

3.
int i = 1;
i+=++i + i++ + ++i + i++;
System.out.println(i);//13

i = 1,2,3,4,5,13
i = i + ++i + i++ + ++i + i++
  = 1 + 2   + 2   + 4   + 4
  = 13

4.
   int x= 10;
   x= ++x;
   System.out.println(x);//11
```

```
5. int x= 10;
    x = x++;
    System.out.println(x);//10

6. int x= 10;
   int y = x++;
   System.out.println(x);//11
   System.out.println(y);//10
```

++++++++++++++++++
Object Orientation
++++++++++++++++++

Orientation
 a. Direction
 b. Perspective
 c. Allignment
 d. path

Object Orientation
 => it is the perspective of looking at this world as "Collection of Objects".
 => Every object in the realworld comes with 2 parts
      a. HAS-Part(properites/fields/attributes)
      b. DOES-Part(beviour/actions/methods)
 => Every object belongs to a particular type,though that type doesn't exists in reality.
 => The objects belonging to the particular type exists and that type in java is called as "class".

Note:
 => class is a blueprint using which we create an object.
 => object is an "instance" of a class/It is also called as "physical" representation of an class.
 => new is an operator which is used to create an object/instance for a class in java.
 => we don't have delete operator in java to destroy the objects, where destroying the objects is taken care by "JVM(GarbageCollector)".
 => using dot(.) opertor in java we call the behaviours/methods of the class.
 => For a class(blueprint), any no of objects(instances) can be created.

eg#1.
```
//class related keywords[import,extends,implements,class]
class Dog
{
      //HAS-PART[varaibles]
       String name;
       String breed;
       int    cost;

      //DOES-PART[method]
      void bark(){
            System.out.println("Dog is barking");
      }

      void eat(){
            System.out.println("Dog is eating");
      }
```

```
	void sleep(){
		System.out.println("Dog is sleeping");
	}
}
public class Test
{
	public static void main(String[] args)
	{
			//Creating an instance of Dog
			Dog d1=new Dog();

			//using reference we are performing actions[methods]
			d1.eat();
			d1.bark();
			d1.sleep();

			System.out.println("***********************");

			//Creating an instance of Dog
			Dog d2=new Dog();
			//using reference we are performing actions[methods]
			d2.eat();
			d2.bark();
			d2.sleep();
	}
}
```

```
D:\OctBatchMicroservices>javac Test.java

D:\OctBatchMicroservices>java Test
Dog is eating
Dog is barking
Dog is sleeping
***********************
Dog is eating
Dog is barking
Dog is sleeping
```

eg#2.

```
//class related keywords[class,package,extends,implements]
class Fan
{
	//HAS-PART[variables]
	String color;
	int noOfWings;
	int price;

	//DOES-PART[methods]
	void rotate(){
		System.out.println("Fan is rotating...");
	}
	void blowAir(){
		System.out.println("Fan is blowing...");
	}
	void stop(){
		System.out.println("Fan is stoping...");
	}
}
```

```
public class Test
{
      public static void main(String[] args)
      {
              //Creating an instance of Fan
             Fan f1 = new Fan();

             //Calling the methods of fan class
             f1.rotate();
             f1.blowAir();
             f1.stop();
      }
}
D:\OctBatchMicroservices>javac Test.java

D:\OctBatchMicroservices>java Test
Fan is rotating...
Fan is blowing...
Fan is stoping...
```

Conventions about writing
  a. classname   ====> Pascal convention [First letter should be in uppercase]
                   eg: String,StringBuilder,StringBuffer
  b. variablename ===> camelCase convention[First letter lowercase,joining word first letter Uppercase]
                   eg: noOfWings,fatherName,genderOfCandidate
  c. methodname   ===> camelCase convention[First letter lowercase,joining word first letter Uppercase]
                   eg: blowAir(),toString(),.....

Methods
+++++++
 To perform some activity in any programming language, we group set of statements and write a method.
            refer:: image file

 Arguments : When we make a call to the method by passing inputs, such inputs are called "Arguments".
 Parameters : When we write a method to collect inputs, such inputs are called "Parameters".

eg#1.
```
//class related keywords
class Calculator
{
      //DOES-PART[Method]
      //a,b => parameters
      int addTwoNumbers(int a,int b)
      {
            int c  = a+b;
            return c;//returning the value to the caller
      }
}
public class Test
{
      public static void main(String[] args)
      {
             //Create an instance of Calculator class
```

```
             Calculator calc = new Calculator();

            //calling a method using reference
            int x= 100;
            int y= 200;
            int z=calc.addTwoNumbers(x,y);//arguments
            System.out.println("The value of z is "+z);
      }
}
```

```
Output
D:\OctBatchMicroservices>javac Test.java

D:\OctBatchMicroservices>java Test
The value of z is 300
```

eg#2.

```
class Calculator
{
      //DOES-PART[Method]
      //a,b => parameters
      int performArithmeticOperation(int a,int b)
      {
            int add = a+b;
            int sub = a-b;
            int div = a/b;
            int mul = a*b;
            return add,sub,mul,div; //CE
      }
}
public class Test
{
      public static void main(String[] args)
      {
             //Create an instance of Calculator class
             Calculator calc = new Calculator();

            //calling a method using reference
            int x= 100;
            int y= 200;
            calc.performArithmeticOperation(x,y);//arguments
            System.out.println("The value is ");
      }
}
```

```
Output
D:\OctBatchMicroservices>javac Test.java
Test.java:12: error: ';' expected
                return add,sub,mul,div;
```

eg#3.

```
//class related keywords
class Calculator
{
      //DOES-PART[Method]
      //a,b => parameters
      void addTwoNumbers(int a,int b)
      {
```

```
            int add = a+b;
            System.out.println("The sum is :: "+add);
      }
}
public class Test
{
      public static void main(String[] args)
      {
             //Create an instance of Calculator class
             Calculator calc = new Calculator();

            //calling a method using reference
            int x= 100;
            int y= 200;
            calc.addTwoNumbers(x,y);//arguments

      }
}
```

Output

```
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
The sum is :: 30
```

Snippets
++++++++
Q>

```
boolean b1 = true;
boolean b2 = false;
boolean b3 = true;
if ((b1 & b2) | (b2 & b3) & b3)
    System.out.print("alpha ");
if ((b1 = false) | (b1 & b3) | (b1 | b2))
    System.out.print("beta ");
```

What is the result?
A. beta
B. alpha
C. alpha beta
D. Compilation fails.
E. No output is produced.[Answer]
F. An exception is thrown at runtime.

Q>
Given:

```
1. class Maybe {
2.    public static void main(String[] args) {
3.          boolean b1 = true;
4.          boolean b2 = false;//true
5.          System.out.print(!false ^ false); //true ^ false => true
6.          System.out.print(" " + (!b1 & (b2 = true))); //(false & true)=>false
7.          System.out.println(" " + (b2 ^ b1));//true ^ true => false
8.    }
9. }
```

Which are true?
A. Line 5 produces true.
B. Line 5 produces false.
C. Line 6 produces true.
D. Line 6 produces false.

E. Line 7 produces true.
F. Line 7 produces false.
Answer: A,D,F

Q>
```
class Sixties {
      public static void main(String[] args) {
            int x = 5;
            int y = 7;
            System.out.print(((y * 2) % x));// 14 % 5 => 4
            System.out.print(" " + (y % x));// 7 % 5 = 2
      }
}
```
What is the result?
A. 1 1
B. 1 2
C. 2 1
D. 2 2
E. 4 1
F. 4 2
G. Compilation fails.
H. An exception is thrown at runtime.

Answer: F

Q>
```
class Foozit {
      public static void main(String[] args) {
            Integer x = 0;
            Integer y = 0;
            for(Short z = 0; z < 5; z++)
                  if((++x > 2) || (++y > 2))
                        x++;
            System.out.println(x + " " + y);
      }
}
```
What is the result?
A. 5 1
B. 5 2
C. 5 3
D. 8 1
E. 8 2 [Answer]
F. 8 3
G. 10 2
H. 10 3

x = 0,1,2,3,4,5,6,7,8
y = 0,1,2

z = 0, 0<5(true)
z = 1, 1<5(true)
z = 2, 2<5(true)
z = 3, 3<5(true)
z = 4, 4<5(true)
z = 5, 5<5(false)