① How to implement precedence rules and associativity in java language ? Give an example.

Ans:

Java Operators precedence and Associativity :

Java Operators have two properties i.e., Precedence and Associativity Precedence is the priority order of an operator, if there are two or more operators in an expression then the operator of higher priority will be executed first. then the operator of highest priority will later high. for example, in expression 1+2*5, multiplication (*) operator will be processed first and then addition (+). Its because multiplication has higher priority (or) precedence than addition.

Alternatively, we can say that when an operand is shared by two operands (2 in above example is shared by + and *) then higher priority operator picks the shared operand for processing. From above example we can understand the role.

When all operators in an expression have same priority, Associativity acts. It tells the direction of execution of operators that can either be left to right or right to left. for example, in expression a=b=c=8, the assignment operator is executed from right to left that is 'c' will be assigned by 'b'. We can parenthesize as (a=(b=(c=8))

We can also change the priority of a Java operator by enclosing the lower order priority operator in parenthesis but not the associativity. for example, In (1+2) * 3, addition will be done first because paranthesses has higher priority than multiplication operator.

19BQ1A05F7

Precedence	Operator	Description	Associativity
1	[ ] ( ) .	array index method call member access	Left to Right
2	+ + - - + - ~ !	pre/post fix increment pre/post fix decrement unary plus, minus bitwise NOT logical NOT	Right to Left
3	(type cast) new	type cast object creation.	Right to Left
4	* / %	multiplication division modulus (remainder)	Left to Right
5	+ - +	addition, subtraction string concatenation	left to Right
6	< < > >>>> > >>	left shift signed right shift unsigned or zero-fill right shift	Left to Right
7	< <= > >= instanceof	less than less than or equal to greater than greater than or equal to reference test	left to Right
8	= = ! =	equal to not equal to	left to Right
9	&	bitwise AND	left to Right
10	^	bitwise XOR	left to Right
11	|	bitwise OR	left to Right
12	& &	logical AND	left to Right
13	||	logical OR	left to Right
14	?:	Conditional (ternary)	Right to Left
15	= %= <<= += &= >>= -= ^= >>>= *= |=	assignment and short hand assignment operators	Right to Left

(2) CODE:

Ans:

```
import java.util.Scanner;
public class BankAccount{
    double act_num;
    String name;
    String act_type;
    int bal;
    void setData (double a ,String b, String c, int d){
        act_num = a;
        name = b;
        act_type = c;
        bal = d;
    }
    void Deposit (int a){
        System.out.println ("Balance before deposit is " +bal);
        bal = bal+a;
        System.out.println ("Balance after deposit is " + bal);
    }
    void Withdraw(int a){
        System.out.println ("Balance before withdrawal is "+bal);
        bal = bal-a;
        if (bal<0){
            System.out.println ("cannot withdraw");
            bal=bal-a;
        }
        else
            System.out.println ("Balance after withdrawal is "+bal);
    }
```

```
    void Display(){
        System.out.println("Name : "+name);
        System.out.println("Balance : "+bal);
    }
    public static void main(String[] args){
        BankAccount ba = new BankAccount();
        Scanner s = new Scanner(System.in);
        System.out.println("Enter act-num, name, type, bal");
        ba.setData(s.nextInt(),s.next(),s.next(),s.nextInt());
        System.out.println("enter the amount to deposit");
        ba.Deposit(s.nextInt());
        System.out.println("enter the amount to withdraw");
        ba.Withdraw(s.nextInt());
        ba.Display();
    }
}
```

Output:

```
Enter act-num, name, act-type, bal
191057 , yeshhh , Savings , 25,000
Enter the amount to deposit
4000
Balance before Deposit is 25000
Balance after Deposit is 29000
Enter the amount to withdraw
9000
Balance before withdraw is 29000
Balance after withdraw is
20000
Name : yeshhh
Balance : 20000
```

Do you need to use static keyword for the above bank account program ? Explain.

The meaning behind the static keyword is whenever we declare our member as static the static member belongs to the class instance instead of some specific instance of a class. So we no need to use the static keyword for the above bank account program.

③ CODE

```
import java.util.*;
public class ElectricBill {
        int units;
        String n;
        double bill;
        Scanner ob = new Scanner (System.in);
        void accept() {
                System.out.println ("Enter Name of the customer");
                n = ob.next();
                System.out.println ("Enter Number of units consumed");
                units = ob.nextInt();
        }
        void calculate(){
                if (units <= 100)
                        bill = units * 2;
                else
                if (units > 100 && units <= 300)
                        bill = 100 * 2 + (units - 300) * 5;
```

```
        else
            bill = 100*2 + 200*3 + (units -300)*5;
        if (units >300)
            bill = bill +2.5 /100 * bill;
    }
    void print (){
        System.out.println ("Bill Amount : "+bill);
    }
    public static void main (String args[]){
        ElectricBill obj = new ElectricBill();
            obj.accept();
            obj.calculate();
                obj.print();
    }
}
```

Output-1

```
Name of the customer :
yeshnh
No. of units consumed:
150
Bill Amount : 350.0
```

Output-2

```
Name of the customer:
NVKYN
No. of units consumed
375
Bill Amount : 1204.375
```

④ Design a class to overload a function check() as follows:

Ans: CODE!

```
class overload{
    public static void check(String s, char ch){
        int c=0;
        for (int i=0; i<s.length(); i++){
            if (s.charAt(i)==ch)
                c++;
        }
        System.out.println("number of "+ch+" present is="+c);
    }
    public static void check(String s1){
        s1=s1.toLowerCase();
        for (int i=0; i<s1.length(); i++){
            char ch = s1.charAt(i);
            if (ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u')
                System.out.print(ch+" ");
        }
    }
    public static void main(String args[]){
        check("success",'s');
        check("computer");
    }
}
```

19B01A05F7

Output:

number of e present is = 3

o u e