JAVA ASSIGNMENT-1

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

Ans: Operator precedence determines the order in which the operators in an expression are evaluated.

Generally, Java operators have two properties those are precedence and associativity. Precedence is the priority order of an operator. If there are more than two or two operators in an expression then the operator of highest priority will be executed first. If all the operators in an expression are of same priority then associativity plays a major role, Associativity tells the direction of execution of operators that can be either left to right or right to left.

Ex:

1: $a = 5 + 12 * 4$

Here multiplication is of higher precedence so $5+(12*4)$ multiplication will be evaluated first than addition.

$\therefore a = 5+(12*4)$

$= 5 + 48$

$\boxed{a = 53}$

2. $x = y = z = 9$.

Here, the assignment operator is executed from right to left i.e, first $z=9$, then 'y' will be assigned by 'z', and finally 'x' will be assigned by 'y'.

i.e, $x = (x = (y = (z = 9)))$

The table below lists the precedence of operators in Java; higher it appears in the table, the higher its precedence.

Precedence	Operator	Description	Associativity
1.	[]	array index	Left to Right
	()	method call	
	.	member access	
2.	++	pre or postfix increment	Right to left
	--	pre or postfix decrement	
	+ -	unary plus, unary minus,	
	~	bitwise NOT	
	!	logical NOT	
3.	(type cast)	type cast	Object creation Right to left
	new		
4.	*	multiplication	Left to Right
	/	division	
	%	modulus (remainder)	
5.	+, -	addition, subtraction	Left to Right
	+	string concatenation	
6.	<<	left shift	Left to right
	>>	signed right shift	
	>>>	unsigned or zerofill rightshift	
7.	<	less than	Left to right
	<=	less than or equal to	
	>	greater than	
	>=	greater than or equal to	
	instance of	reference test	

MBQIA05P2

Precedence	Operators	Description	Associativity
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 hang assignment operators	Right to left

2. Design a class that represents a bank account and construct the methods to i) Assign initial values
   ii) Deposit an amount
   iii) Withdraw amount after checking balance.
   iv) Display the name and balance.

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

Ans-

```
import java.io.*;
import java.util.*;
public class Demo2{
    public static void main(String args[]){
        BankAccount b = new BankAccount("Divya",100000);
        b.Initialvalues();
        b.Deposite();
        b.Withdraw();
        b.Display();
    }
}
class BankAccount{
    private String name;
    private int balance;
    static int accNO=01234567;
    public BankAccount(){
    }
    public BankAccount(String name,int balance){
        this.name=name;
        this.balance=balance;
    }
    public void setName(String name){
        this.name=name;
    }
    public String getName(){
        return name;
    }
```

```
    public void setBalance(int balance){
        this.balance = balance;
    }
    public int getBalance(){
        return balance;
    }
    public void Initialvalues(){
        String name = "Divya";
        int balance = 100000;
    }
    public void Deposite(){
        int deposite = 25000;
        balance = balance + deposite;
    }
    public void Withdraw(){
        int withdraw = 50000;
        if (balance > withdraw)
            balance = balance - withdraw;
    }
    public void Display(){
        System.out.println("Name of person :" + name);
        System.out.println("Balance :" + balance);
    }
}
```

Output: javac Demo2.java
java Demo2
Name of person : Divya
Balance : 75000

The values of variable may be different in two different objects. Generally static keyword is used to share data between the two objects or more.

But especially static is used infront of constant values because constant values are same throughout the program.

we need to use static keyword for the above bankaccount program because Account number is constant throughout the program. we can't change it any where. so we use static in Bank Account program.

3. Define a class ElectricBill with the following specifications:

class: ElectricBill

Instance Variable / data member:

String n - to store the name of the customer

int units - to store the number of units consumed

double bill - to store the amount to be paid

Member methods:

Void accept() - to accept the name of the customer and number of units consumed.

Void calculate() - to calculate the bill as per the following tariff:

Number of units - Rate per unit

First 100 units - Rs.2.00

Next 200 units - Rs.3.00

Above 300 units - Rs.5.00

A surcharge of 2.5% charged if the number of units consumed is above 300 units.

Void Print() - To print the details as follows:

Name of the customer ----

Number of units consumed ---

Bill amount ----

write a main method to create an object of the class and call the above member methods.

Pgm:

```
import java.io.*;
import java.util.*;
public class Demo3 {
    public static void main(String args[]) {
        ElectricBill e = new ElectricBill();
        e.accept();
        e.calculate();
        e.print();
    }
}
class ElectricBill {
    int units;
    double bill = 0.0;
    String n;

    Void accept() {
        Scanner sc = new Scanner(System.in);
        n = sc.next();
        units = sc.nextInt();
    }
```

```
    void calculate() {
        for (int i=1; i<=units; i++) {
            if (i<= 100)
                bill += 2;
            else if (i>100 && i<= 300)
                bill += 3;
            else
                bill += 5;
        }
        if (units > 300)
            bill = bill + bill * 2.5/100;
    }
    void print() {
        System.out.println("Name of the customer: "+ n);
        System.out.println("Number of units consumed: "+
                                                units);
        System.out.println("Bill amount : " + bill);
    }
}
```

Output:

```
javac Demo3.java
java Demo3
XYZ
150
Name of the customer : XYZ
Number of units consumed : 150
Bill amount : 350.
```

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

i) void check(String str, char ch) - to find and print the freauency of a character in a string.

Example:

Input — output

Str = "success" number of s present is = 3

ch = "s"

ii) void check (String S1) - to display only the vowels from string S1, after converting it to lower case.

Example:

Input:

S1 = "computer"

Output: o u e.

Prog:

```
import java.io.*;
import java.util.*;
public class Demo4 {
    public static void main (String args[]) {
        Overloading o = new Overloading();
        o.Check ("success", 's');
        o.Check ("Computer");
    }
}
```

```
class Overloading {
    public String ser,s1;
    public char ch;
    public void Check(String ser, char ch){
        int count=0;
        for(int i=0;i<ser.length();i++){
            if(ser.charAt(i)==ch){
                count=count+1;
            }
        }
        System.out.println("In"+ser+"Number of"+
                           ch+"present is="+count);
    }
    public voic Check(String s1){
        s1.toLowerCase();
        for(int i=0;i<s1.length();i++){
            if(s1.charAt(i)=='a'){
                System.out.println('a'+" ");
            }
            else if(s1.charAt(i)=='e'){
                System.out.println('e'+" ");
            }
            else if(s1.charAt(i)=='i'){
                System.out.println('i'+" ");
            }
            else if(s1.charAt(i)=='o'){
                System.out.println('o'+" ");
            }
            else if(s1.charAt(i)=='u'){
                System.out.println('u'+" ");
            }
        }
    }
```

Output: javac demo4.java
java demo4
In success Number of s present i's = 3
o u e