JAVA PROGRAMMING

ASSIGNMENT - 1

Name : R. Bindu Sneha
No : 19BQ1A05J0
Branch : CSE - C
Year : 2nd year.
Date : 20.09.2020.
Set : II

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 highest priority will be executed first. then higher, 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 8, then 'b' will be assigned by c, and finally 'a' 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 parentheses but not the associativity. For example, In (1+2)*3, addition will be done first because parantheses has higher priority than multiplication operator.

Precedence	Operator	Description	Associativity
1	[ ] ( ) .	array index method call member access	Left to Right
2	++ -- +- ~ !	pre / postfix increment pre / postfix 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-fillrightshift	Left to Right
7	< <= > >= instance of	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. 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: CODE:

```
import java.util.Scanner;
public class BankAccount {
      int ac_no;
      String name;
      String type;
      int bal;
      void SetData (int a, String b, String c, int d) {
            ac_no = a;
            name = b;
            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 withdrawl is"+bal);
            bal = bal - a;
            if (bal < 0) {
                  System.out.println ("Cannot Withdraw");
            bal = bal = a;
        }
        else
            System.out.println("Balance after withdrawl 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 sc = new Scanner(System.in);
        System.out.println("Enter ac no,name,type,bal ");
        ba.SetData (scnextInt(), sc.next(),sc.next(),sc.nextInt());
        ~~ba. Deposit (sc.nextInt());~~
        System.out.println (" Enter the amount to deposit");
        ba.Deposit (sc.nextInt());
        System.out.println ("Enter the amount to withdraw");
        ba.Withdraw (sc.nextInt());
        ba.Display();
    }
}
```

OUTPUT:

```
Enter ac-no, name, type, bal
12345, Sneha, Savings, 10000
Enter the amount to deposit
2000
Balance before Deposit is 10000
Balance after Deposit is 12000
Enter the amount to Withdraw
3000
Balance before withdraw is
12000
Balance after withdraw is
9000
Name : Sneha
Balance : 9000
```

We don't need the static keyword for the above BankAcco-
unt Program.

Reason : — Definition :

Static means {never bounded.

If we define any class, method or a variable as static, the memory allocated for that- is not bounded by any boundary of that particular class, method or variable.

So, in main statement, we use "Static" keyword.

But here, In this case, the memory should not be allocated in the way that it is not be bounded by any boundary. Instance, it should not be occured in our present program.

So, we ~~sho~~ didn't use a ~~main~~ "static" keyword in our program

3 Define a class Electric Bill with the following specifications:
class : ElectricBill
Instance Variable / Data Member :
String n - to store the name of the costumer.
int units - to store the number of units consumed.
double bill - to store the amount to 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.

Ans CODE :

```
import java.util.*;
public class Electric Bill {
     String n;
     int units;
     double bill;
     Scanner sc = new Scanner(System.in);
```

Scanned by CamScanner

```
    void accept(){
    System.out.println("Name of the customer : ");
    n=sc.next();
    System.out.println("No.of units consumed : ");
    units = sc.nextInt()
    }
    void calculate(){
        if(units<=100)
            bill=units*2;
        else
        if (units>100 && units<=300)
            bill=100*2 + (units-100)*3;
        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 :
Sneha
No.of units consumed :
250
Bill amount : 650.0
```

OUTPUT-2

```
Name of the customer :
Puppy
No.of units consumed :
365
Bill amount : 1153.125
```

4. Design a class to overload a function check() as follows:
(i) void check (String str, char ch) – to find and print the frequency 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.

CODE:

```
class overload {
        public static void check (String str, char ch){
                int c=0;
                for(int i=0; i< ~~str~~.length(); i++){
                        if (str.charAt(i)==ch)
                                c++;
      }     }
      ~~Sy~~  System.out.println ("number of s 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");
        }
}
```

OUTPUT:

```
number of s present is = 3
o u e
```

—— * * * ——