JAVA PROGRAMMING ASSIGNMENT - 1

ROLLNO: 19BQ1A05F9

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

Ans:

Java has well-defined rules for specifying the order in which the operators in an expression are evaluated when the expression has several operators. For example, multiplication and division have a higher precedence than addition and subtraction. Precedence rules can be overridden by explicit parentheses.

Precedence Order:

When two operators share an operand, the operator with the higher precedence goes first. For example, 1+2*3 is treated as 1+(2*3), whereas 1*2+3 is treated as the (1*2)+3 since multiplication has a higher precedence than addition.

Associativity:

When an expression has two operators with the same precedence, the expression is evaluated according to its associativity. For example x=y=z=17 is treated as x=(y=(z=17)), leaving all three variables with the value 17, since the '=' operator has right-to-left associativity (and an assignment statement evaluates to the value on the right hand side). On the other hand, 72/2/3 is treated as (72/2)/3 since the / operator has left-to-right associativity. Some operators are not associative: for example, the expressions (x<=y<=z) and x++-- are invalid.

Precedence and Associativity of Java Operators:

The table below shows all java operators from highest to lowest precedence, along with their associativity. Most programmers do not memorize them all, and even those that do still use parentheses for clarity.

Level	Operator	Description	Associativity
1	=, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=	Assignment	right to left
2	?:	Ternary	right to left
3	||	logical OR	left to right
4	&&	logical AND	left to right
5	|	bitwise OR	left to right
6	^	bitwise XOR	left to right
7	&	bitwise AND	left to right
8	!=, ==	Equality	left to right
9	instance of, >, >=, <, <=	relational	not associative
10	<<, >>, <<<	shift	left to right
11	+, -, +	additive String concatenation	left to right

12.	*, /, %	multiplicative	left to right
13.	(), new	cast, Object creation	right to left
14.	++, --, +, -, !, ~.	Unary preincrement, Unary predecrement, unary plus, unary minus, unary logical NOT, unary bitwise NOT.	right to left
15.	++, --	unary post increment, unary post decrement	not associative.
16.	[], ., ()	access array element, access object member, Parentheses.	left to right.

→ There is no explicit operator precedence table in the Java Language Specification. Different tables on the web and in textbooks diagree in some minor ways.

Order of evaluation of subexpressions:

When using the conditional 'and' and 'OR' operators (&& and ||), java does not evaluate the second operand unless it is necessary to resolve the result. This allows statements like if (s!=null && s.length() < 10) to work reliably. programmers rarely use the ~~non-length~~ non-short circuting versions (& and |) with the boolean expressions.

Examples:

```
class Precedence{
    public static void main(String args[]){
        int a=10,b=5,c=1, result;
        result = a-++c-++b;
        System.out.println(result);
    }
}
```

Output: 2

Exp: result= a-++c-++b;

it equivalent to

result = a-(++c)-(++b);

= 10 - (2) - (6);

= 2

---

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.Scanner;
public class BankAccount{
    public string name;
    public double totalbalance;
    public double depositamt;
    public double withdrawamt;
```

```
public void values(){
        Scanner sc=new Scanner(System.in)
        System.out.println("Enter the name of customer");
        name = sc.nextline();
        System.out.println(" Thank you for coming to our
                                        Bank");
}
public void deposit(){
        System.out.println(" Enter deposit amt");
        depositamt= sc.nextInt();
        totalbalance += depositamt;
}
public void withdraw(){
        System.out.println(" Enter withdraw amt");
        withdrawamt = sc.nextInt();
        if(totalbalance ≥ withdrawamt)
                totalbalance -= withdrawamt
                System.out.println(" Balance --- "+ totalbalance);
        else
                System.out.println(" Insufficient Balance---!!");
}
public ~~void~~ static void main(){
        BankAccount b1= new BankAccount();
        b1. values();
        b1. deposit();
        b1. withdraw();
        b1. display();
}
public void display(){
        System.out. println(" Name of customer ... "+name);
        System. out. println(" Account Balance ... "+totalbalance);
}
}
```

Generally, Static keyword in java is used for the memory management mainly, we can apply static keyword with variables, methods, blocks and in nested classes.

→ The static keyword belongs to the class than an instance of the class.

→ In the above Bank Account program, static can be used but without using static also we can ~~use~~ write the above program.

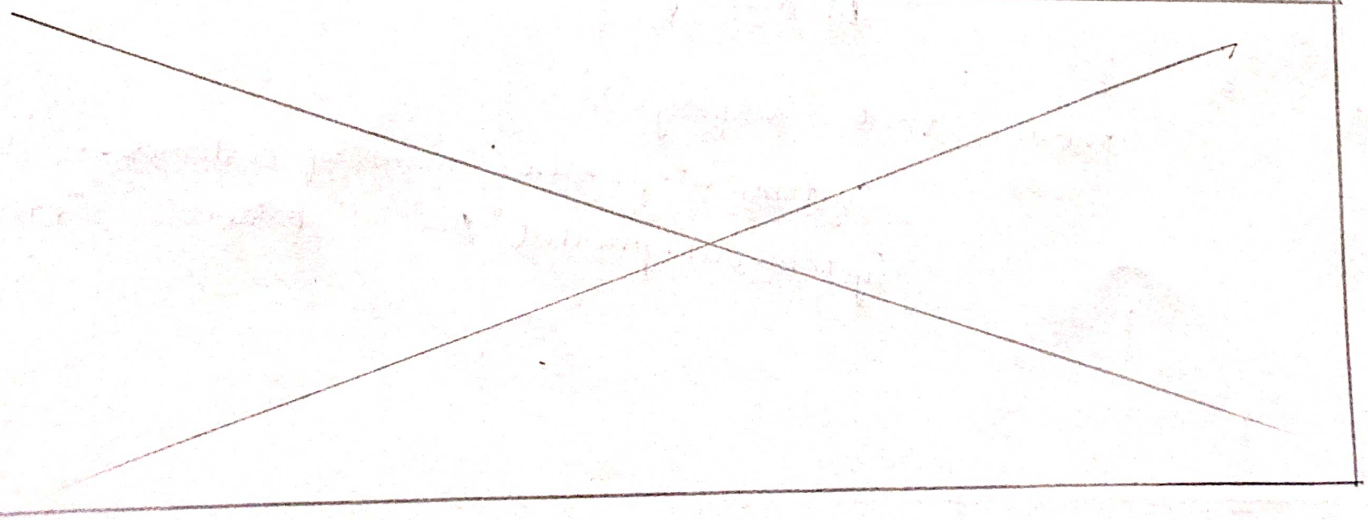
→ There is no variable that is static in the above program.

→ Each variable has to get its own memory in above BankAccount program.

→ Also for every deposit and withdrawl, memory has to be reallocated.

→ So, if we use static, we cannot change the memory every now and then.

→ So, there is no need of usage of static keyword in above program.



(3) Define a class ElectricBill with the following specifications.

class: ElectricBill

Instance Variable/Data Members:

String n - to store the name of customer

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 no 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 class and call the above member methods.

Ans:-

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

```
        System.out.println("Enter name of customer.....");
        n = sc.nextLine();
        System.out.println("No.of units consumed.....");
        units = sc.nextInt();
    }
    public void calculate(){
        if (units <= 100){
            bill = units * 2.00
        }
        else if (units >100 && units <= 300){
            //bill = units *
            // bill = 100 * 2.00 + (u-100) * 3;
            bill = 100 * 2.00 + (units - 100) * 3.00;
        }
        else{
            bill = 100 * 2.00 + 200 * 3.00 + (u-300) * 5.00;
            double surcharge = bill * 2.5/100;
            bill = bill + surcharge;
        }
    }
    public void print(){
        System.out.println("Name of customer..." +n);
        System.out.println("units consumed....." + units);
        System.out.println("Bill amount ....." + bill);
    }
    public static void main(String args[]){
        ElectricBill obj = new ElectricBill();
        obj.accept();
        obj.calculate();
        obj.print();
    }
}
```

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

i, void check(String str1, char ch) - to find and print the frequency of a character in a string.

Ex: 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.

Ex: Input:

S1 = 'computer'

output:

o u e

Ans:

```
Class StringCheck{
public void check(String str, char ch){
        int c=0, code, i, s;
        str = str.toLowerCase();
        int len= str.length();
        for(code=97; code<122; code++){
                c=0;
                for(i=0; i<len; i++){
                        ch= str.charAt(i);
                        s= (int) ch;
                        if(s==code)
                                c=c+1;
                }
                if (c!=0)
                        System.out.println("Frequency of "+ch+
                                        " is " + c);
        }
}
public void check(String S1){
        int i;
        char ch=0; chr=0;
        for(i=0; i<S1.length(); i++){
```

```
ch= s1.charAt(i);
if (Character.isUpperCase(ch))
        chr= Character.toLowerCase (ch);
if ((s1.charAt(i)=='a' || s1.charAt(i)=='e' || s1.charAt(i)
    =='i' || s1.charAt(i)=='o' || s1.charAt(i)=='u'))
        System.out.println(s1.charAt(i));
    }
  }
}
```