19B01A05N5
CSE-D

20/09/2020

JAVA-ASSIGNMENT

SET-2

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

Ans:

Java operators have two properties those are precendce 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 and then high for example, in expression 1+2*5, multiplication(*) operator will be processed first and then addition. It's because multiplication has higher priority or precedence than addition.

Associativity tells the direction of execution of operators that can be either 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 means c will be assigned by 8, then b will be assigned by c and finally a will be assigned by b. You can parenthesize this expression as (a=(b=(c=8))).

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

19BQ1A05N5

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

19BQIA05N5

Precedence	Operator	Description	Associativity
9	&	bitwise AND	left → right
10.	^	bitwise XOR	left → right
11.	|	bitwise OR	left → right
12.	&&	logical AND	left → right
13.	||	logical OR.	left → right
14.	?:	conditional (ternary)	right to left
15.	= += -= *= /= %= &= ^= |= <<= >>= >>>=	assignment and short hand assignment operators.	Right → left

19BCLA05NS

(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:

```
Public class Bank {
    Private int balance;
    Private long accountNum;
    Private string name;
    Public Bank (long accountNum, string name) {
        this.name = name;
        this.balance = 0;
        this.accountNum = accountNum;
    }
    Public void addAmount(int amnt) {
        balance += amnt;
        System.out.println(amnt + "rs added");
    }
    Public void withdraw (int amnt) {
        ba
        if (amnt <= balance) {
            balance -= amnt;
            system.out.println(amnt+" rs withdrawn succ-
            essfully);
            system.out.println("Remaining Balance is"+balance
            +" rs");
        }
        else {
            System.out.println("Insufficient Balance");
        }
}
```

```
Public void checkBalance() {
        System.out.Println("Name of the account holder:"
        + name);
    System.out.Println("Account Balance :"+ balance);
}

/* Public static void main(String[] args) {
        Bank acc1=new Bank(12345678, "Raghu");
        acc1.addAmount(20000);
        acc1.withdraw(1500);
        acc1.checkBalance();
} */

}
```

=) ~~Yes~~ No, we don't need to use the static keyword while building the class for Bank. But if we want to use the main method and test our class Bank class we need to use the static keyword.

(3)

Ans.

```
Public class ElectricityBill {
    Private String n;
    Private int units;
    Private double bill;
    Public void accept(String name, int u) {
        n=name;
        units=u;
    }
    Public void calculate() {
        if(units<=100) {
            bill= units * 2.0
        }
        else if(units <=300) {
            bill = units * 3.0;
        }
        else {
            bill = units* 5.0;
            bill += bill * 0.025;
        }
    Public void print() {
        System.out.println("Name of the customer :"+n);
        System.out.println("Number of units consumed :"+units);
        System.out.println("Bill amount :"+bill);
    }
    Public static void main(String[] args) {
        ElectricalityBill cus1 = new ElectricityBill();
        cus1.accept("Vamsi", 320);
        cus1.calculate();
        cus1.print();
    }
}
```

19BQ1A05N5

④
```
Public class Overload {
    Public static void check (String str, char ch) {
        int count =0;
        for(int i=0 ; i< str.length() ; i++) {
            if (str.charAt(i) == ch) {
                count++ ;
            }
        }
        System.out.println(count);
    }

Public static void check(string s1) {
        s1= s1.toLowerCase();
        for(int i=0; i<s1.length(); i++){
            char c=s1.charAt(i);
            if (c=='a' || c=='e' || c=='i' || c=='o' || c=='u') {
                System.out.print(c+" ");
            }
        }
}
Public static void main(String[] args){
        check("hello world",'l');
        check("hello world");
    }
}
```