POORNIMA

Experiment - 1

(a) Write a program to print all the data type in java by using variables

```
public class Exp1_a {
    public static void main (String[] args) {
        int i;
        float f;
        double d;
        String s;
        char c;

        i = 10;
        f = 10.3f;
        d = 18.367894;
        c = 'a';
        s = "Divyanshi Singh";

        System.out.println ("int:" +i);
        System.out.println ("float:" +f);
        System.out.println ("double:" +d);
        System.out.println ("String:" +s);
        System.out.println ("char:" + c);
    }
}
```

(b) Write a program to find out the given year is a leap year or not

```
import java.util.*;
public class Expt_b {
    public static void main (String[] args) {
        System.out.println("Enter the year : ");
        Scanner scan = new Scanner(System.in);
        int year = scan.nextInt();

        if (year % 400 == 0 || year % 4 == 0 && year % 100 != 0) {
            System.out.println(year + "is a leap year");
        }

        else {
            System.out.println(year + " not a leap year");
        }
    }
}
```

(c) Write a program to find that a given number is armstrong or not

```
import java.util.*;
public class Expt_c {
    public static void main (String[] args) {
        System.out.println("Enter the number : ");
        int original, arm = 0, r;
```

```
Scanner scan = new Scanner(System.in);
int x = scan.nextInt();
original = x;
while (x > 0){
    r = x % 10;
    arm = (r*r*r) + arm;
    x = x/10;
}
if(original == arm){
    System.out.println(original + " is an armstrong number");
}
else{
    System.out.println(original + " not an armstrong number");
}
}}
```

(d) Write a program to add two 1-D arrays.

```
import java.util.*;
public class Exp1_d{
public static void main (String[] args){
    int n, i;
    Scanner scan = new Scanner(System.in);
    System.out.println("Enter the length of both arrays");
    n = scan.nextInt();
    int a[] = new int[10];
    int b[] = new int[10];
    System.out.println("Enter the elements in array 'a':");
```

```
for ( i=0 ; i<n ; i++) {
        a[i] = scan.nextInt();
}

System.out.println("Enter the elements in array 'b':");
for ( i=0; i<n ; i++) {
        b[i] = scan.nextInt();
}

System.out.println("Sum of two arrays :");
for ( i=0 ; i<n ; i++) {
        int s[] = new int[10];
        s[i] = a[i] + b[i];
        System.out.println(s[i]);
    }
} }
```

Write a program to find the multiplication of two-dimensional array

```
import java.util.*;
public class Exp1 {
    public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("enter the rows of array 1");
        int a = sc.nextInt();
        System.out.println("enter the columns of array 1");
        int b = sc.nextInt();
```

POORNIMA

```
int arr1[][] = new int[a][b];
system.out.println("enter the elements of array 1");
  for(int i=0 ; i<a ; i++){
      for(int j=0 ; j<b ; j++){
          arr1[i][j] = sc.nextInt();
      }
  }
System.out.println("Enter the number of rows in array1");
int p= sc.nextInt();
System.out.println("Enter the number of columns in array1");
int q = sc.nextInt();
int arr2[][] = new int[p][q];
system.out.println("Enter the elements of array 2");
  for(int i=0 ; i<p ; i++){
      for(int j=0 ; j<q ; j++){
          arr2[i][j] = sc.nextInt();
      }
  }
system.out.println("Multiplication of two array");
for(int i=0 ; i<b ; i++){
    for(int j=0 ; j<p ; j++){
      int arrm[][] = new int[b][qp];
      arrm[i][j] = 0;
      for(int k=0 ; k<3 ; k++){
          arrm[i][j] += arr1[i][k] * arr2[k][j];
      }
      system.out.print(arrm[i][j] + "  ");
    }
}
```

12/3/24

output →

int : 10
float : 10.3
double : 18.367894
string : Divyanshi Singh
char : a

Output →

Enter the year:
2000
2000 is a leap year

Enter the year:
100
100 is not a leap year

Output →

Enter the number :
153
153 is an armstrong number

Enter the number :
1331
1331 is not an armstrong number

output →

```
Enter the length of both arrays:
4
Enter the elements in array 'a':
10
20
30
40
Enter the elements in array 'b':
20
30
40
50
Sum of two arrays:
30
50
70
90
```

Output →

Enter the rows of array1
2
Enter the column of array1
2
Enter the elements of array1
1
2
2
1
Enter the rows of array2
2
Enter the ~~rows~~ column of array2
2
Enter the elements of array2
1
1
1
1
Multiplication of 2 arrays
3
2
2
3

POORNIMA

Experiment NO 2

In what scenarios would you use single inheritance VS multiple inheritance in object oriented design and why?

Single Inheritance refers to the concept of a class inheriting from only one superclass while. Multiple Inheritance allows a class to inherit from multiple superclasses

Single Inheritance is used in a scenario where you have a clear hierarchy and want to keep the design simple & avoid complications like diamond problem. For example, 'Car' is a sub class that inherits from the 'vehicle' class

```
class vehicle {
    void wheel () {
        System.out.println ("This is a vehicle");
    }
}
class car extends vehicle {
    void wheel () {
        System.out.println ("have four wheels");
    }
}
class main {
    public static void main (String args[]) {
        car c = new car ();
    }
```

Page No.....

Output

This is a vehicle
Have four wheels

}

Multiple Inheritance is used in a scenario where you need to inherit behaviour from multiple unrelated sources and when design can handle & resolve the complexities that come with it. For example, 'son' can inherit both 'father is' and 'mother'

```
class father {
    void fname();
}
class mother {
    void mname();
}
class son extends father, mother {
    void fname() {
        System.out.println("Father is A");
    }
    void mname() {
        System.out.println("Mother is B");
    }
}
class main() {
    public static void main (String[] args) {
        son s = new son();
        s.fname();
        s.mname();
    }
}
```

output:

Father is A
Mother is B

POORNIMA

Ques ② Provide an example of polymorphism in actions where a base class method is overridden by a sub-class method to achieve diffuent behaviour.

```
class animal {
    void makesound() {
        System.out.println("Animal make a sound");
    }
}
class dog extends animal {
    void makesound() {
        system.out.println("Dog barks");
    }
}
class cat extends animal {
    void makesound() {
        System.out.println("Cat Meows");
    }
}
class main {
    public static void main(String[] args) {
        dog DOG = new dog();
        Cat CAT = new Cat();
        DOG.makesound();
        CAT.makesound();
    }
}
```

output

Animal make sounds
Dog barks
Cat Meows

POORNIMA

Ques ③ Describe the concept of an encapsulation & how it achieve data hiding & abstraction in OOPs.

Encapsulation is one of the four fundamental principals of OOP. It refers to building of data and methods that operate on the data into single unit or class. Here is how data hiding and abstraction is achieved.

(i) Data hiding:

- hide the internal state of an object from outside world
- for example, consider a 'person' class with private 'name' & 'age'. These can be accessed within a method which is defined in same class.

```
class person {
    private string name;
    private int age;
    void get (string name, int age) {
        this.name = name;
        this.age = age;
    }
    void put () {
        system.out.println ("Name :-" + name + "Age:" +
                            age);
    }
}
class main () {
    public static void main (String [] args) {
        person pr = new person ();
        pr.get ("Abstract", 21);
        pr.put ();
    }
}
```

Output :

name : Dwija
age : 21

(ii) Abstraction:

⇒ It hides the implementation detail of class & exposing only essential feature or behaviour through its public methods.

for example:

```
abstract class animal {
    public abstract void sound();
    void sleep() {
        System.out.println("zzz");
    }
}
class pig extends animal {
    public void sound() {
        System.out.println("The pig says: wwww");
    }
}
class main() {
    public static void main(String[] args) {
        Pig pig = new Pig();
        pig.sound();
        pig.sleep();
    }
}
```

output :

zzz

The pig says: wee wee

POORNIMA

Ques ④ How would you use abstraction to model a complex system such as banking application while hiding unnecessary detail from user

```
import java.util.Scanner;
    static int token = 1;
    int acc_no;
    String name;
    int age;
    abstract void get_details();
    abstract void show_details();
}
class Saving_account extends details {
    Scanner sc = new Scanner(System.in);
    Saving_account() {
    acc_no = ++token; }
    double balance = 0;
    void get_details() {
        System.out.println("Enter Name, age & balance");
        name = sc.nextInt();
        age = sc.nextInt();
        balance = sc.nextDouble();
    }
System.out.println("Your account number:" + acc_no);
    void show_details() {
        System.out.println("Name:" + name + "\n age:"
            + age + "\n balance:" + balance);
    }
}
```

Output :

Enter name, age & balance

Divya
20
62000000

Your account no: 9046 4321 1000 26
name: Divya
age = 20
balance = 62000000

```
class balance {
    public static void main (String [] args ) {
        saving account acc = new saving_account ();
        acc.get_details ();
        acc.show_details ();
    }
}
```

Balance
02/04/24

POORNIMA

Experiment - 3

Write a program to print the names of students by creating a student class if no name is passed while creating an object of student class and the name should be "unknown" otherwise the name should be equals to the string value passed while creating object of student class.

```
class Student {
    String name;
    Student() {
        name = "unknown";
    }
Student (String name) {
    this.name = name;
}
void display() {
System.out.println("Name is" + name);
}}
class Lab1 {
    public static void main (String[] args) {
        Student S1 = new Student();
        Student S2 = new Student("Divyanshi");
            S1.display();
            S2.display();
}
}
```

Output

Name is unknown
Name is Divyanshi.

POORNIMA

Experiment 4

Write a program to create an interface flyable with a method called fly_obj() create three classes that implement the flyable interface. Implement the fly_obj method for each of three classes

```
interface flyable {
    void fly_obj();
}
class spacecraft implements flyable {
    public void fly_obj() {
System.out.println("space craft is flying");
}}

class airplane implements flyable {
        public void fly_obj() {
    System.out.println("Airplane is flying");
}}
class helicopter implements flyable {
    public void fly_obj() {
        System.out.println("Helicopter is flying");
}}
public class flying {
    public static void main(String[] args) {
spacecraft s = new spacecraft();
airplane a = new airplane();
helicopter h = new helicopter();
s.fly_obj();   a.fly_obj();
h.fly_obj();
}
```

Page No.....

Output

Spacecraft is flying
Airplane is flying
Helicopter is flying

POORNIMA COLLEGE OF ENGINEERING , JAIPU

Evaluation Report

Name of the Laboratory.....Java Lab.....

Code...4CS4-25.

To be filled by the student					To be filled by the faculty				
S.No.	Name of the experiment	Pg. No.	Date of Allotment	Date of Performance	Attendance (2)	Record* (3)	Performance** (5)	Total (10)	Signa Faculty
1.	Experiment1 : Basic concepts	2-7	5.3.24	5.3.24	2	3	5	10	
2.	Experiment2: Inheritance, OOPs	8-12	12.3.24	12.3.24	2	3	4	09	
3.	Experiment3:								
4.	Experiment4: Interface								
5.									
6.									
7.									
8.									
9.									
10.									
11.									
12.									
13.									
14.									
15.									
16.									
17.									
18.									
19.									
20.									

* Preparation & Lab record

** Overall Quality of Performance, Knowledge about application of experiment, Technical details of equipments, Process & Theory involved in practical & Viva-voce

Max. Marks :..... Marks Obtained ..... Average.....