Practical File

Lab Name…………*Web Technology LAB……………………. *Lab Code…………………………

.

Estd. 2000

Name……………………SAMARPIT DUA……………………………………………………………………………..

Adm.No……2019B111068…………. Univ. Roll No……1900321290050………

Course ………………B. TECH…………………..……Branch……………CEIT…………………………………..

Sem………………….V…………………… Section………………………………………………..

ABES
Engineering College
(College Code-032)

NAAC Accredited, NBA Accredited Branches (CSE, ECE, EN & IT)

ISO 9001:2015
TÜVRheinland
CERTIFIED
www.tuv.com
ID 9105047739

19th Km. Stone, NH-09, Ghaziabad - 201009 (UP), India
Phone : 0120-7135112, 9999889341 Fax : 0120-7135115 Website : www.abes.ac.in, Email: info@abes.ac.in

1 . PRACTICAL STATEMENT OF PRACTICAL: **Write a program to implement Multiple classes in Java.**

2 . OBJECTIVE OF PRACTICAL : **To write a program to implement Multiple classes in Java.**

3 . THEORY:

A java file can have multiple classes. But, there is one restriction, that we can have as many classes in one file but only one public class is allowed. If we try to declare two classes as public in the same file, the code will not compile.

The reason being is we need to name the file with the name of the class which is declared as public and we cannot have the same file with different names, and also that public class should be containing the main method as the compiler will check for the main method first.

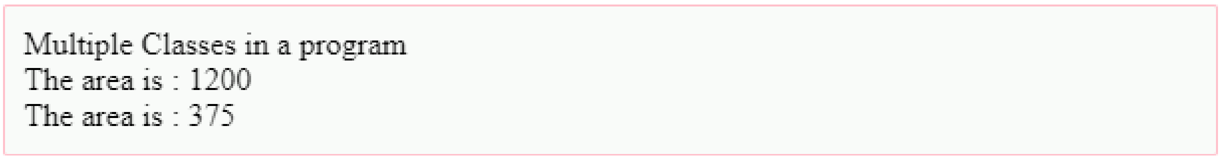
If we want to access the methods, instances of the other classes we can just create their respective objects/instances in the public class and simply access them.

IMPLEMENTATION :

```
class Room

{

    int length, breadth;

    void getdata(int x, int y)

    {

        length = x;

        breadth = y;

     }

    void area()
    {
       int area = length* breadth;
       System.out.println("The area is : "+ area);
    }
}
public class Simple
{
    public static void main(String[] a)
        {
            System.out.println("Multiple Classes in a program");
                 Room BedRoom = new Room();
            BedRoom.getdata(100,12);
            BedRoom.area();
            Room Kitchen = new Room();
            Kitchen.getdata(15,25);
            Kitchen.area();


        }
}
```

1. Result /Output :

Run by: java Simple



```
Multiple Classes in a program
The area is : 1200
The area is : 375
```

1. OBJECTIVE OF PRACTICAL:

Write a program to implement Constructor Overloading in java.

2. THEORY:

In addition to overloading methods, we can also overload constructors in java. Overloaded constructor is called based upon the parameters specified when new is executed.

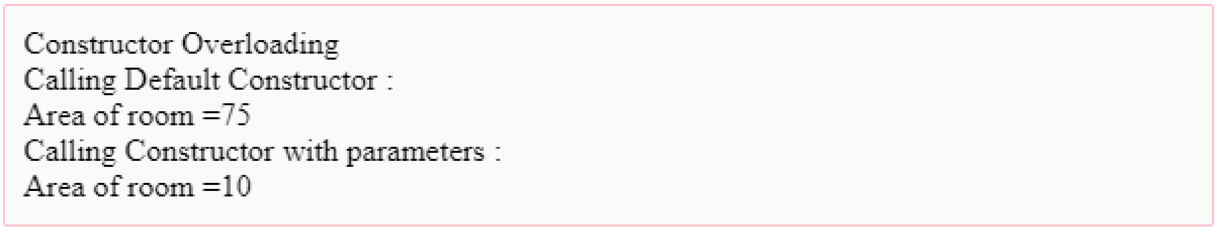
Sometimes there is a need of initializing an object in different ways. This can be done using constructor Overloading.

3. IMPLEMENTATION :

```
class Room2
{
        int length,breadth;
        Room2()
        {
                length=15;
                breadth=5;
        }
        Room2(int x,int y)
        {
                length=x;
                breadth=y;
        }
        void area()
        {
                int area=length*breadth;
                System.out.println("Area of room ="+area);
        }
}
public class Simple
{
public static void main(String[] a)
        {

            System.out.println("Constructor Overloading");
                Room2 obj=new Room2();
            System.out.println("Calling Default Constructor : ");
                obj.area();
                Room2 obj2=new Room2(2,5);
           System.out.println("Calling Constructor with parameters : ");
                obj2.area();
        }

}
```

4. Result /Output :

Run by: java Simple



```
Constructor Overloading
Calling Default Constructor :
Area of room =75
Calling Constructor with parameters :
Area of room =10
```

Practical : 3

1. PRACTICAL STATEMENT OF PRACTICAL: Write a program to implement Method Overloading in Java.

2. OBJECTIVE OF PRACTICAL: Program to implement Method Overloading

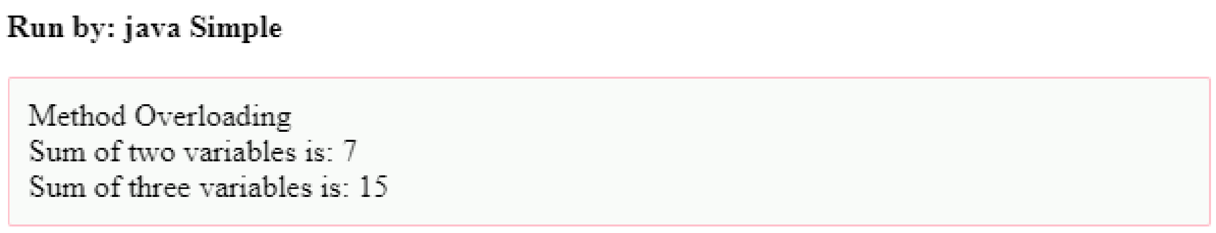
3. THEORY:

Method Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters or both.
Overloading is related to compile-time (or static) polymorphism.

4. IMPLEMENTATION :

```
class calculate
{
void find(int a,int b)
{
    int c=a+b;
    System.out.println("Sum of two variables is: "+c);
}
void find(int a,int b,int y)
{
    int c=a+b+y;
    System.out.println("Sum of three variables is: "+c);
}
}

public class Simple
{
    public static void main(String[] a)
    {
          System.out.println("Method Overloading");
           calculate obj=new calculate();
           obj.find(2,5);
           obj.find(2,5,8);

    }
}
```

5. Result /Output :



Run by: java Simple

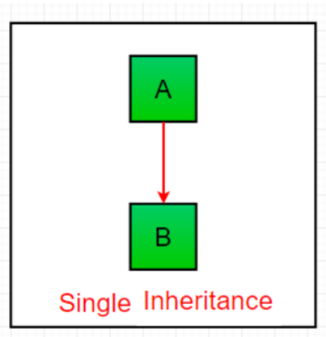
```
Method Overloading
Sum of two variables is: 7
Sum of three variables is: 15
```

Practical : 4

1. **OBJECTIVE OF PRACTICAL: Write a program to implement Single Level Inheritance.**

2. **THEORY:**

Single Inheritance: In single inheritance, subclasses inherit the features of one superclass. In the image below, class A serves as a base class for the derived class B.



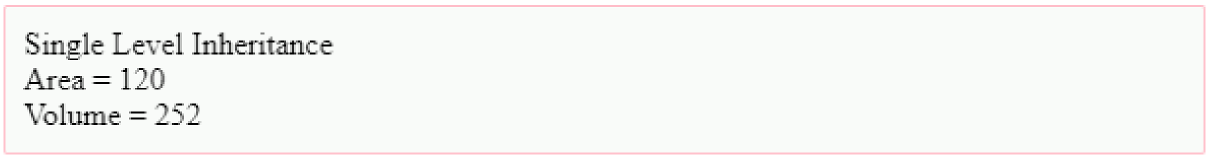
3. **IMPLEMENTATION :**

```
class room
{
    int length,breadth;
    room(int a,int b)
    {
            length=a;
            breadth=b;
    }
    void calculate_area()
    {
            int area=length*breadth;
            System.out.println("Area = "+area);
    }
}
class room_volume extends room
```

```
{
    int height;
    room_volume(int a,int b,int c)
    {
            super(a,b);
            height=c;
    }
    void Volume()
    {
            int volume=length*breadth*height;
            System.out.println("Volume = "+volume);
    }
}



public class Simple
{
     public static void main(String[] a)
    {
            System.out.println("Single Level Inheritance");
            room obj2=new room(10,12);
            obj2.calculate_area();
            room_volume obj=new room_volume(12,3,7);
            obj.Volume();
    }
}
```

4. Result /Output :

Run by: java Simple

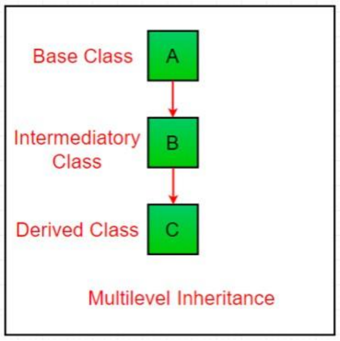


```
Single Level Inheritance
Area = 120
Volume = 252
```

1. OBJECTIVE OF PRACTICAL: Write a program to implement Multi level Inheritance.

2. Theory :

Multilevel Inheritance: In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class. In the below image, class A serves as a base class for the derived class B, which in turn serves as a base class for the derived class C. In Java, a class cannot directly access the grandparent's members.



3. IMPLEMENTATION :

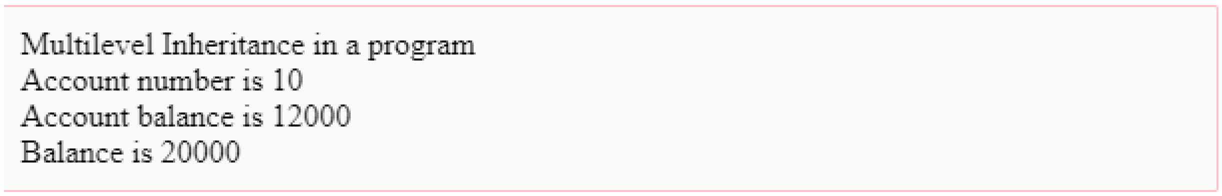
```
class Account1
{
   int acc_no;
   Account1 (int x)
   {
      acc_no = 10;
   }
   void display ()
   {
      System.out.println ("Account number is " + acc_no);
   }
}
```

```
class Sav_account1 extends Account1
{
   int amount;
   Sav_account1 (int x, int y)
   {
      super (x);
      amount = y;
   }

   void display ()
    {
         super.display ();
         System.out.println ("Account balance is " + amount);
    }
}

class bal extends Sav_account1
{
   int bal;
   bal (int x, int y, int z)
    {
      super (x, y);
      bal = z;
    }
   void display ()
    {
      super.display ();
      System.out.println ("Balance is " + bal);
   }
}


public class Simple
{
```

```
public static void main (String[]a)
   {
            System.out.println ("Multilevel Inheritance in a program");
           bal obj = new bal (10, 12000, 20000);
            obj.display ();
    }
}
```

4. Result /Output :

Run by: java Simple



```
Multilevel Inheritance in a program
Account number is 10
Account balance is 12000
Balance is 20000
```

1. PRACTICAL STATEMENT OF PRACTICAL: Write a program to implement Interface in Java.

2. OBJECTIVE OF PRACTICAL : To write a program to implement Interface in Java.

3. THEORY:

Interface: It is used to implement Multiple inheritance in java. It is very similar to class only difference is that in interface we only declare the method, definition of method is given at that place where we implement it.

Syntax:

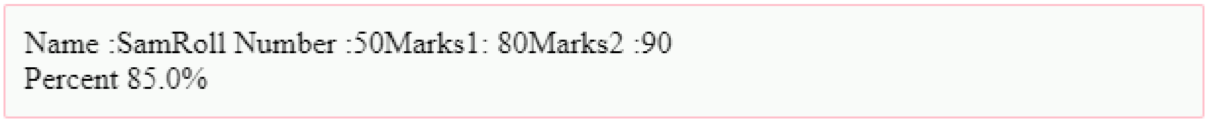
```
Interface interfacename
{
        // declaration of method
}
```

4. IMPLEMENTATION :

```
interface Exam
{
        void percent();
}
class Student
{
        String name;
        int rno,marks1,marks2;
        Student(String n,int rn,int m1,int m2)
        {
                name=n;
                rno=rn;
                marks1=m1;
                marks2=m2;
        }
        void display()
        {
                System.out.println("Name :"+name+"Roll Number
:"+rno+"Marks1: "+marks1+"Marks2 :"+marks2);
        }
}
        class Result extends Student implements Exam
        {
                Result(String n,int r,int m1,int m2)
                {
                        super(n,r,m1,m2);
                }
                public void percent()
                {
                        int total=marks1+marks2;
                        float percent=(total*100)/200;
                        System.out.println("Percent "+percent+"%");
                }
        }
public class Simple
{
    public static void main(String[] a)
        {

                Result x=new Result("Sam",50,80,90);
                x.display();
                x.percent();
        }
}
```

5. Result /Output :

Run by: java Simple



```
Name :SamRoll Number :50Marks1: 80Marks2 :90
Percent 85.0%
```