(Week-8) Lab 4 and 5 - Practice Programs.

PAGE No.
DATE 21/12/20

1) WAP which has abstract class solid with inheriting from solid to find Surface Area and Volume.

```
// import java.io.*;
// import java.util.*;
abstract class Solid
 {
 double d1, d2, d3;
Solid()
 {
 }
abstract void surface_area();
abstract void volume();
}
class Cylinder extends Solid
{
Cylinder(double a, double b)
 {
 d1 = a;
 d2 = b;
 }
void surface_area()
{
System.out.println("Surface Area is: " +
(2*3.14*d1*d1) + (2*3.14*d1*d2));
}
void volume()
 {
System.out.println("Volume is: " + (3.14*d1*d1*d2));
 }
}
```

```
class Cone extends Solid
{
Cone (double a, double b, double c)
  {
   d1=a;
   d2=b;
   d3=c;
  }
 void surface_area()
  {
System.out.println ("Vol. Surface Area is" +
((3.14*d1*d1)+(3.14*d1*d2));
}
void volume():
{
 System.out.println("Volume is" +((3.14*
 d1*d1*d3)/3.0));
}
}
class Sphere extends Solid
{
Sphere(double a)
 {
 d1=a;
  }
void Surface_area()
{
System.out.println("Surface Area is" +
                    (4*3.14*d1*d1));
}
```

PAGE No.
DATE / /

```
void volume()
{
 System.out.println("volume is:"+((4*3.14*d*d*d)/3));
}
class Solid_test
 {
public static void main(String args[])
 {
 Cylinder c=new Cylinder(1,2);
 Cone c1=new Cone(1,2,3);
 Sphere s1=new Sphere(2);
 c.surface_area();
 c.volume();
 c1.surface_area();
 c1.volume();
 s1.surface_area();
 s1.volume();
 }
}
```

Extra - Program - 2 (week-5)

1)
```
import java.io.*;
import java.util.*;
class Person
{
String name;
Person (String name)
{
this.name = name
System.out.println("* * * * * * * * * ");
System.out.println ("Name :"+name);
}
}
class Employee extends Person
{
String designation;
Employee (String name, String designation)
{
super(name);
this.designation = designation;
System.out.println ("Designation:"+designation);
}}
class Student extends Person
{
string designation;
Student (String name, String designation)
{
super(name);
this.designation = designation;
System.out.println("Designation:"+designation);
}}
```

PAGE No.
DATE / /

```
class teaching extends Employee
{
  String emptype;
teaching (String name, String designation, String
                                        emptype)
  {
  super(name, designation);
  this.emptype = emptype;
  System.out.println("Emptype:" + emptype);
}}
class non-teaching extends Employee
 {
  String emptype;
non teaching (String name, String designation,
  String emptype)
{
super (name, designation);
this.emptype = emptype;
System.out.println ("Emptype:" +emptype);
}}
class ug extends Student.
{ String course;
  ug (String name, String designation, String course)
{
super (name, designation);
this.course = course)
System.out.println("Coure:" +course);
}}
class pg extends Student
   {
    String course)
pg(String name, String designation, String course)
```

```
{
super(name, designation);
this.course = course;
System.out.println("Course:"+course);
}
}
class test_hierachy
{
public static void main(String args[])
{
Scanner sc=new Scanner(System.in);
teaching t1=new teaching("Sharat", "Employee
"teaching");
non_teaching t2=new non_teaching("Ravi", "Empl
oyee", "non-teaching");
ug u1=new ug("Vamsi", "Studet", "ug");
pg p1=new pg("Pankaj", "Studet", "pg");
}
}
```