LAB PROGRAMS ON CLASSES 11-17

/*1. Write a Java program to create an abstract class named Shape that contains two integers and an empty method named print Area (). Provide three classes named Rectangle, Triangle, and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method print Area () that prints the area of the given shape.*/

```
import java.util.*;
abstract class Shape {
int length, breadth, radius;
Scanner input = new Scanner(System.in);
abstract void printArea();
}
class Rectangle extends Shape {
void printArea() {
System.out.println("*** Finding the Area of Rectangle ***");
System.out.print("Enter length and breadth: ");
length = input.nextInt();
breadth = input.nextInt();
System.out.println("The area of Rectangle is: " + length * breadth);
}
}
class Triangle extends Shape {
void printArea() {
System.out.println("\n*** Finding the Area of Triangle ***");
System.out.print("Enter Base And Height: ");
length = input.nextInt();
breadth = input.nextInt();
System.out.println("The area of Triangle is: " + (length * breadth) / 2);
}
}
class Cricle extends Shape {
```

```
void printArea() {
System.out.println("\n*** Finding the Area of Cricle ***");
System.out.print("Enter Radius: ");
radius = input.nextInt();
System.out.println("The area of Cricle is: " + 3.14f * radius * radius);
}
}
public class AbstractClassExample {
public static void main(String[] args) {
Rectangle rec = new Rectangle();
rec.printArea();
Triangle tri = new Triangle();
tri.printArea();
Cricle cri = new Cricle();
cri.printArea();
}
}
```

2. Write a java program to illustrate the concept of Single level and multi-level inheritance

```
class One{
String one_str = "class One";
void displayOne(){
System.out.println("parent " + one_str);
}
}
class Two extends One{
String two_str = "class Two";
void displayTwo(){
super.displayOne();
System.out.printf("child %s inherited from %s displaying single level
inheritance\n",two_str,one_str);
}
}
class Three extends Two{
String three_str = "class Three";
void displayThree(){
super.displayTwo();
System.out.printf("child %s inherited from %s and %s displaying
multilevelinheritancen\n",three_str,two_str,one_str);
}
}
public class InheritanceDemo {
```

```
public static void main(String[] args) {
Three threeobj = new Three();
threeobj.displayThree();
}
}
```

Output:

parent class One

child class Two inherited from class One displayingsingle level inheritance

child class Three inherited from class Two andclass One displaying multilevel inheritance

3. Write a java program to illustrate the concept of Multiple Inheritance

```
interface parent1 { // interface 1
  public void method();
```

```
}
interface parent2 { // interface 2
   public void method(String s);
}

public class MultipleInheritance implements parent1, parent2 {
   // using the method here with different definition
   public void method() {
      System.out.println("CSE DEPARTMENT");
   }
   public void method(String s) {
      System.out.println("SEC: " + s);
   }
   public static void main(String []args){
      // object creation
      MultipleInheritance obj = new MultipleInheritance();
      // method calling
      obj.method();
      obj.method("A");
   }
}
```

4. Write a JAVA program that implements Runtime polymorphism

```
class Bank{
float getRateOfInterest(){return 0;}
}
class SBI extends Bank{
float getRateOfInterest(){return 8.4f;}
}
class ICICI extends Bank{
float getRateOfInterest(){return 7.3f;}
}
class AXIS extends Bank{
float getRateOfInterest(){return 9.7f;}
}
class TestPolymorphism{
public static void main(String args[]){
Bank b;
b=new SBI();
System.out.println("SBI Rate of Interest: "+b.getRateOfInterest());
b=new ICICI();
System.out.println("ICICI Rate of Interest: "+b.getRateOfInterest());
b=new AXIS();
System.out.println("AXIS Rate of Interest: "+b.getRateOfInterest());
}
}
```

Output:

SBI Rate of Interest: 8.4

ICICI Rate of Interest: 7.3

AXIS Rate of Interest: 9.7

5. Write a java program to illustrate the concept of class with method overloading.

Solution:

```
class Sum {

int sum(int x, int y)

{ return (x + y); }

int sum(int x, int y, int z)

{ return (x + y + z); }
```

```
double sum(double x, double y)
{ return (x + y); }
}
public class MethodOverloadingDemo {
public static void main(String args[])
{
Sum s = new Sum();
System.out.println(s.sum(10, 20));
System.out.println(s.sum(10, 20, 30));
System.out.println(s.sum(10.5, 20.5));
}
}
```

Output:

30

60

31.0

6. Write Java program(s) on use of inheritance, preventing inheritance using final, abstract classes.

```
class Parent
{
public void p1()
{
System.out.println("Parent method");
}
}
```

```
public class Child extends Parent {
public void c1()
{
System.out.println("Child method");
}
public static void main(String[] args)
{
Child cobj = new Child(); cobj.c1(); //method of Child class
cobj.p1(); //method of Parent class
}
}
```

Sample Input/output:

Child method Parent method

preventing inheritance using final:

```
// create a final class
final class FinalClass {
public void display() {
System.out.println("This is a final method.");
}
}


// try to extend the final class
class Main extends FinalClass {
public void display() {
System.out.println("The final method is overridden.");
}
public static void main(String[] args) {
Main obj = new Main(); obj.display();
```

```
}
}
```

Sample Input/output:

Compile Time Error

Abstract Classes:

```
abstract class A{
abstract void callme();
}
classB extends A{
voidcallme(){
System.out.println("this is callme.");
}
public static void main(String[] args){
B b = new B(); b.callme();
}
}
```

Sample Input/output:

this is callme

7. Write Java program(s) on dynamic binding, differentiating method overloading and overriding.

Dynamic Binding:

```
class A{
void samp(){
System.out.println("hai...");
}
}
class D extends A{
voidsamp(){
System.out.println("hello...");
}
public static void main(String args[]){
A a=new D();
a.samp();
}
}
```

Sample Input/Output:

Hello

Overloading Methods:

```
class Overload{
void demo (int a)
{
```

```
System.out.println ("a: " + a);
}
void demo (int a, int b){
System.out.println ("a and b: " + a + "," + b);
}
double demo(double a) {
System.out.println("double a: " + a); return a*a;
}
}
classMethodOverloading{
public static void main (String args [])
{
Overload Obj = new Overload(); double result;
Obj .demo(10);
Obj .demo(10, 20); result = Obj .demo(5.5);
System.out.println("Output is : " + result);
}
}
```

Sample Input/output:

a: 10

a and b: 10,20 double a: 5.5

Output is : 30.25

Method Overriding:

```
class BaseClass{
public void methodToOverride() //Base class method
{
System.out.println ("I'm the method of BaseClass");
}
}
class DerivedClass extends BaseClass{
public void methodToOverride() //Derived Class method
{
System.out.println ("I'm the method of DerivedClass");
}
}
class TestMethod{
public static void main (String args []) {
// BaseClass reference and object BaseClass obj1 = new BaseClass();
// BaseClass reference but DerivedClass object BaseClass obj2 = new DerivedClass();
// Calls the method from BaseClass class obj1.methodToOverride();
//Calls the method from DerivedClass class obj2.methodToOverride();
}
}
```

Sample Input/output:

I'm the method of BaseClass I'm the method of DerivedClass