

LAB-9

By-Arun Lal

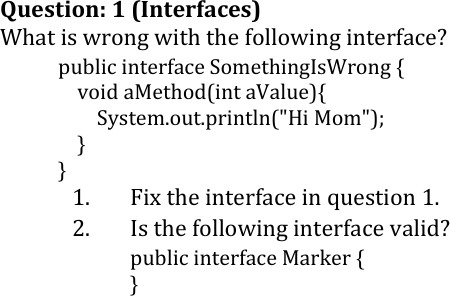


Sec-E

BSCS(II)

CMS-ID=023-24-0120

Exercise of Lab: 9

****

## Problem:

In Java **interfaces cannot have instance method implementations** unless they are marked as:

* default
* or static (from Java 8 onwards) But in the given interface, aMethod is:
* Not marked default
* Has a body (which is **not allowed** in abstract methods)

## ❌ So this line is invalid

void aMethod(int aValue){ // ❌ Invalid in interface

✅ 1. Fix the Interface

## Option A: If you want a default implementation:

public interface SomethingIsWrong { default void aMethod(int aValue){

System.out.println("Hi Mom");

}

}

## Option B: If you want a pure abstract method (like traditional interfaces before Java 8):

public interface SomethingIsWrong {

void aMethod(int aValue); // No body

}

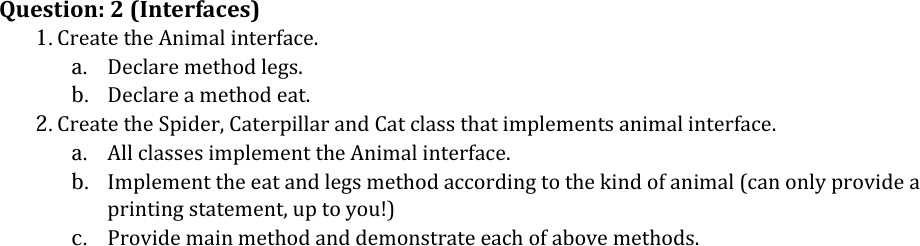
public interface Marker {

}

✅ Yes, this is valid.

It's an example of a marker interface—an interface with no methods.

Java uses marker interfaces for tagging classes (e.g., Serializable, Cloneable).

****

**Source code**

interface Animal{ void legs(int n);

void eat(String e);

}

class Spider implements Animal{ public void legs(int n){

System.out.println("Spider have "+n+" legs");

}

public void eat(String e){ System.out.println("Spider eats "+e);

}

}

class Caterpillar implements Animal{ public void legs(int n){

System.out.println("Caterpiller have "+n+" legs");

}

public void eat(String e){ System.out.println("Caterpiller eats "+e);

}

}

class Cat implements Animal{ public void legs(int n){

System.out.println("cat have "+n+" legs");

}

public void eat(String e){ System.out.println("Cat eat "+e);

}

}

public class TestAnimal{

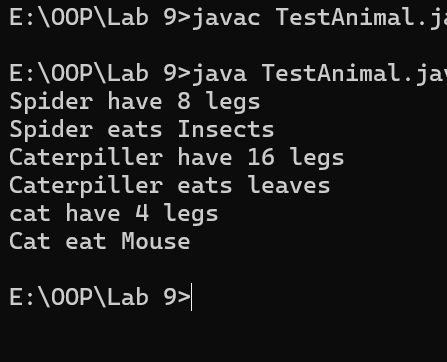
public static void main(String[] args) { Spider sp = new Spider(); Caterpillar cp = new Caterpillar(); Cat ca = new Cat();

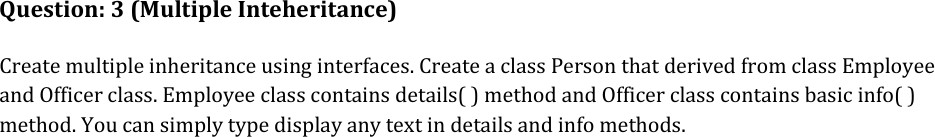
sp.legs(8); sp.eat("Insects"); cp.legs(16); cp.eat("leaves"); ca.legs(4); ca.eat("Mouse");

}

}

**Output**

****



**Source code**

interface Employee{ void details( );

}

interface Officer{ void basicInfo();

}

class Person implements Employee,Officer{ public void details() {

System.out.println("Employee Details: Name: Nazeer, ID: 12345");

}

public void basicInfo() {

System.out.println("Officer Info: Department: IT, Position: Assistant");

}

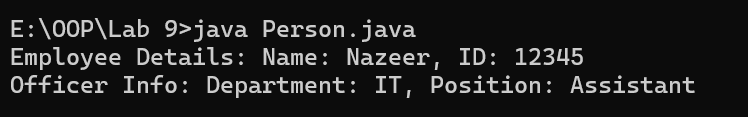
public static void main(String[] args) { Employee emp = new Person(); Officer off = new Person(); emp.details();

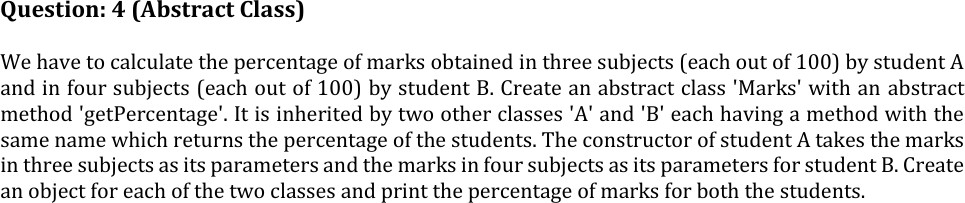
off.basicInfo();

}

}

**Output**

****

****

**Source code**

abstract class Marks{

abstract double getPercentage();

}

class A extends Marks{ double English; double OOP;

double Dbms;

A(double English, double OOP, double Dbms){ this.English = English;

this.OOP = OOP; this.Dbms = Dbms;

}

public double getPercentage(){

return (English+OOP+Dbms)/300\*100;

}

}

class B extends Marks { double English; double OOP;

double Dbms; double Physics;

B(double English, double OOP, double Dbms, double physics) { this.English = English;

this.OOP = OOP; this.Dbms = Dbms; this.Physics = physics;

}

public double getPercentage(){

return (English+OOP+Dbms+Physics)/400\*100;

}

}

public class TestMarks{

public static void main(String[] args) { Marks a = new A(78,88,90);

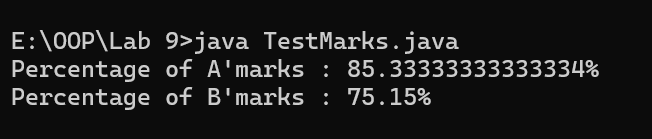
Marks b = new B(88,90.6,67,55);

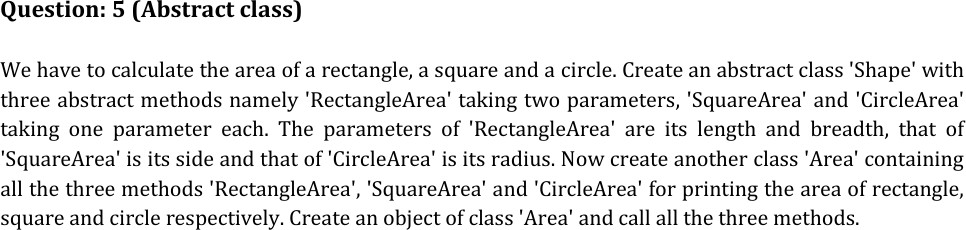
System.out.println("Percentage of A'marks : "+a.getPercentage()+"%"); System.out.println("Percentage of B'marks : "+b.getPercentage()+"%");

}}

}

**Output**

****



**Source code**

abstract class Shape{

abstract double RectangleArea(int l,int w); abstract double SquareArea(int s);

abstract double CircleArea(int r);

}

class Area extends Shape{ @java.lang.Override

double RectangleArea(int l, int w) { return l\*w;

}

@java.lang.Override

double CircleArea(int r) { return Math.PI\*r\*r;

}

@java.lang.Override

double SquareArea(int s) { return s\*s;

}

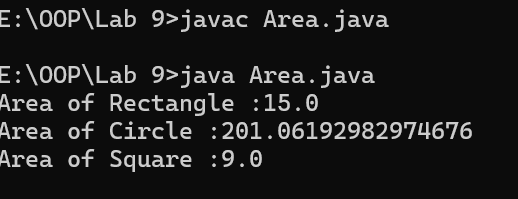
public static void main(String[] args) { Area area = new Area();

System.out.println("Area of Rectangle :"+area.RectangleArea(5,3)); System.out.println("Area of Circle :"+area.CircleArea(8)); System.out.println("Area of Square :"+area.SquareArea(3));

}

}

**Output**

****