

Sukkur institute of business administration University

**Department of computer science**

Programming Fundamentals (BSCS –II)

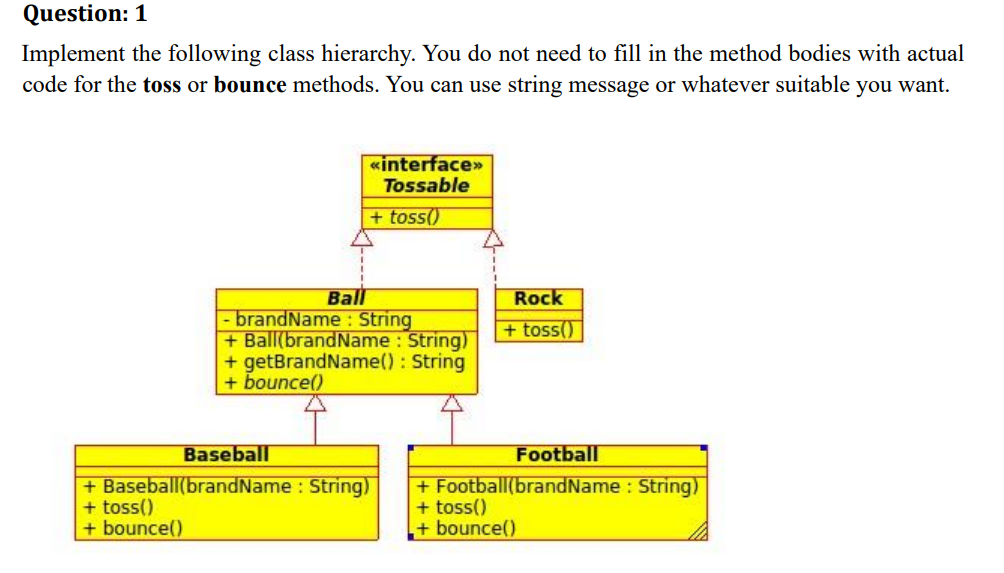
Spring 2024

Assignment – (11)

**Submitted to Ma’am Nimra Mughal**

**Submitted by Fahad Hussain**

**Section E”**

****

Solution:

interface Tossable {

    void toss();

}

abstract class Ball implements Tossable {

    String brandname;

    Ball(String brandname) {

        this.brandname = brandname;

    }

    String getBrandName() {

        return brandname;

    }

    abstract void bounce();

    public void toss() {

        System.out.println("Ball's method abstract toss from interface");

    }

}

class BaseBall extends Ball {

    BaseBall(String brandname) {

        super(brandname);

    }

    public void toss() {

        System.out.println("BaseBall's toss method ");

    }

    void bounce() {

        System.out.println("BaseBall's bounce method");

    }

}

class Football extends Ball {

    Football(String brandname) {

        super(brandname);

    }

    public void toss() {

        System.out.println("FooBall's toss method ");

    }

    void bounce() {

        System.out.println("FooBall's bounce method");

    }

}

class Rock implements Tossable {

    public void toss() {

        System.out.println("Rock's toss method");

    }

}

public class Question1 {

    public static void main(String[] args) {

        Tossable a2 = new BaseBall("Taiwan");

        Tossable a3 = new Football("MyFootball");

        Tossable a4 = new Rock();

        a2.toss();

        a3.toss();

        a4.toss();

        // a2.bounce(); we cant call becuase it is not in Tossable interface

        Ball a = new BaseBall("Taiwan");

        Ball a1 = new Football("MyFootball");

        a.bounce();

        a.toString();

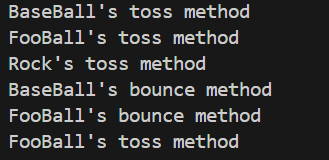
        a1.bounce();

        a1.toss();

    }

}

Output:



Question: 2 Declare the Rectangle, SportCar, Manager classes as classes that implement the Printable interface and run the given application. The output of this application should be the details of each one of the objects that were instantiated.

Solution:

interface Printable{

    void print();

}

class Rectangle implements Printable{

    int hight, width;

    Rectangle(int hight, int width){

        this.hight= hight;

        this.width= width;

    }

    public void print(){

        System.out.println("Hight: "+hight+" Width: "+width);

    }

}

class SportCar implements Printable{

    String brandname;

    int price;

    SportCar(String brandname, int price){

        this.brandname = brandname;

        this.price = price;

    }

    public void print(){

        System.out.println("Brandname: "+brandname+" price: "+price);

    }

}

class Manager implements Printable{

    String name;

    int age;

    Manager(String name, int age){

        this.name = name;

        this.age = age;

    }

    public void print(){

        System.out.println("Manager Name: "+name+" age: "+age);

    }

}

public class Question2{

    public static void main(String[] args){

        Printable[] vec = {

            new Rectangle(130, 30),

            new SportCar("Toyota", 1500000),

            new Manager("Mukesh", 22),

            new Rectangle(100, 40),

            new SportCar("BMW", 1500000),

            new Manager("Maam Nimra", 29)

            };

            for(int i = 0; i<vec.length; i++){

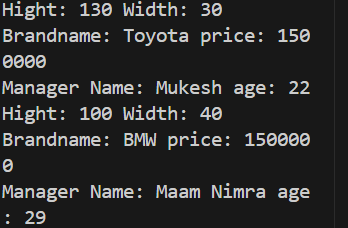
                vec[i].print();

            }

    }

}

Output:



Question: 3 (Multiple Inteheritance) Create multiple inheritance using interfaces. Create a class Person that derived from class Employee and Officer class. Employee class contains details( ) method and Officer class contains basic info( ) method. You can simply type display any text in details and info methods.

Solution:

interface Employee {

    void detail();

}

interface Officer {

    void basicIbnfo();

}

class Person implements Employee, Officer {

    public void detail() {

        System.out.println("Senoir AI Researcher  ");

    }

    public void basicIbnfo() {

        System.out.println("Maam: Nimra Mughal");

    }

}

public class Question3 {

    public static void main(String[] args) {

        Person p = new Person();

        p.basicIbnfo();

        p.detail();

    }

}

Output:

Question: 4 (Abstract Class) We have to calculate the percentage of marks obtained in three subjects (each out of 100) by student A and in four subjects (each out of 100) by student B. Create an abstract class 'Marks' with an abstract method 'getPercentage'. It is inherited by two other classes 'A' and 'B' each having a method with the same name which returns the percentage of the students. The constructor of student A takes the marks in three subjects as its parameters and the marks in four subjects as its parameters for student B. Create an object for each of the two classes and print the percentage of marks for both the students.

Solution:

public class Question4 {

    public static void main(String[] args) {

        A student1 = new A(45, 67, 90);

        B student2 = new B(45, 67, 90, 45);

        System.out.println("s1:"+student1.getpercentage());

        System.out.println("s2:"+student2.getpercentage());

    }

 }

abstract class Marks {

   abstract double getpercentage();

}

class A extends Marks {

  private int s1, s2, s3;

   A(int s1, int s2, int s3) {

       this.s1 = s1;

       this.s2 = s2;

       this.s3 = s3;

   }

   double getpercentage() {

       return  ((s1 + s2 + s3)\*100)/300;

   }

}

class B extends Marks {

 private  int s1, s2, s3, s4;

   B(int s1, int s2, int s3, int s4) {

       this.s1 = s1;

       this.s2 = s2;

       this.s3 = s3;

       this.s4 = s4;

   }

   double getpercentage() {

       return ((s1 + s2 + s3 + s4)\*100)/400;

   }

}

Output:



Question: 5 (Abstract class) We have to calculate the area of a rectangle, a square and a circle. Create an abstract class 'Shape' with three abstract methods namely 'RectangleArea' taking two parameters, 'SquareArea' and 'CircleArea' taking one parameter each. The parameters of 'RectangleArea' are its length and breadth, that of 'SquareArea' is its side and that of 'CircleArea' is its radius. Now create another class 'Area' containing all the three methods 'RectangleArea', 'SquareArea' and 'CircleArea' for printing the area of rectangle, square and circle respectively. Create an object of class 'Area' and call all the three methods.

Solution:

abstract class Shape{

   double PI = 3.142;

   abstract double ReactangleArea(int length, int width);

   abstract double SquareArea(int side);

   abstract double CircleArea(int raduis);

}

class Area extends Shape{

     double ReactangleArea(int length, int width){

        return (length\*width);

     }

    double SquareArea(int side){

        return (side\*side);

    }

    double CircleArea(int raduis){

        return (PI\*raduis\*raduis);

    }

}

    public class Question5 {

    public static void main(String[] args){

        Area area = new Area();

        System.out.println("Cicle area :"+area.CircleArea(5)+" meters");

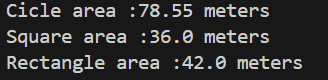
        System.out.println("Square area :"+area.SquareArea(6)+" meters");

        System.out.println("Rectangle area :"+area.ReactangleArea(6, 7)+" meters");

    }

}

Output:



Question: 6 (Abstract classes and interfaces) 1) Write an abstract class Shape – Data members: numSides – Constructor: initialize numSides – Concrete method: get method for numSides – Abstract methods: getArea(), getPerimeter() 2) Write a concrete subclass Rectangle – Data members: width, height 3) Write a concrete subclass RtTriangle – Data members: width, height 4) In another class, write a main method to define a Rectangle and a Triangle.

abstract class Shap{

    int numSlides;

    Shap(int numSlides){

        this.numSlides = numSlides;

    }

    int getNumSlides(){

        return numSlides;

    }

   abstract double getArea();

    abstract int getPerimeter();

}

class Rectangle1 extends Shap{

    int width, hight;

    Rectangle1(int numSlides, int width,int hight){

        super(numSlides);

        this.width= width;

        this.hight = hight;

    }

    double getArea(){

        return (width\*hight);

    }

    int getPerimeter(){

        return 2\*(width+hight);

    }

}

class Triangle1 extends Shap{

    int width, hight,hypo;

    Triangle1(int numSlides, int width,int hight,int hyo){

        super(numSlides);

        this.width= width;

        this.hight = hight;

        this.hypo = hyo;

    }

    double getArea(){

        return 0.5\*(width\*hight);

    }

    int getPerimeter(){

        return (width+hight+hypo);

    }

}

public class Question6 {

    public static void main(String[] arg) {

        Shap  shape = new Rectangle1(4,5,6);

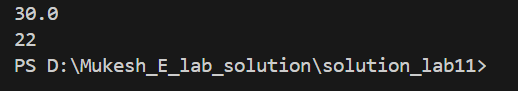
            System.out.println(shape.getArea());

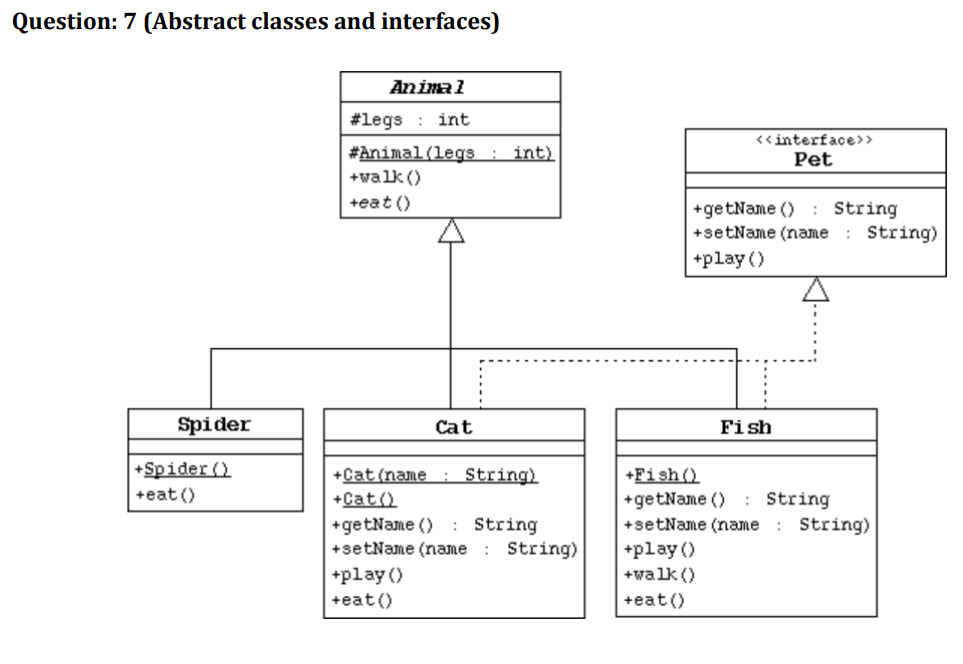
            System.out.println(shape.getPerimeter());

    }

}

Output:





Solution:

abstract class Animal {

    int legs;

    Animal(int legs) {

        this.legs = legs;

    }

    abstract void walk();

    abstract void eat();

}

interface Pet {

    String getName();

    void setName(String name);

    void play();

}

class Spider extends Animal {

    Spider(int legs) {

        super(legs);

    }

    void walk() {

        System.out.println("Hello, I'm a spider.");

    }

    void eat() {

        System.out.println("I eat insects.");

    }

    public String toString() {

        return "legs: " + legs;

    }

}

class Cat extends Animal implements Pet {

    String name;

    Cat(String name, int legs) {

        super(legs);

        this.name = name;

    }

    void walk() {

        System.out.println("Hello, I'm a cat.");

    }

    void eat() {

        System.out.println("I eat milk and fish.");

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    @Override

    public void play() {

        System.out.println("I play with myself.");

    }

    public String toString() {

        return "legs: " + legs + ", name: " + name;

    }

}

class Fish extends Animal implements Pet {

    String name;

    Fish() {

        super(0);

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    void walk() {

        System.out.println("Hello, I'm a fish, I don't walk.");

    }

    void eat() {

        System.out.println("I eat algae.");

    }

    public void play() {

        System.out.println("I play in the water.");

    }

    public String toString() {

        return "legs: " + legs + ", name: " + name;

    }

}

public class Question7 {

    public static void main(String[] args) {

        Pet animal1 = new Cat("cat", 4);

        Animal animal2 = new Spider(8); // Spiders typically have 8 legs

        Pet animal3 = new Fish();

        animal1.setName("Cat");

        animal3.setName("Fish");

        animal1.play();

        System.out.println(animal1);

        animal2.eat();

        animal2.walk();

        System.out.println(animal2);

        animal3.play();

        System.out.println(animal3);

    }

}

Output:

