**WEEK-7**

**Program-I**

**Aim:**

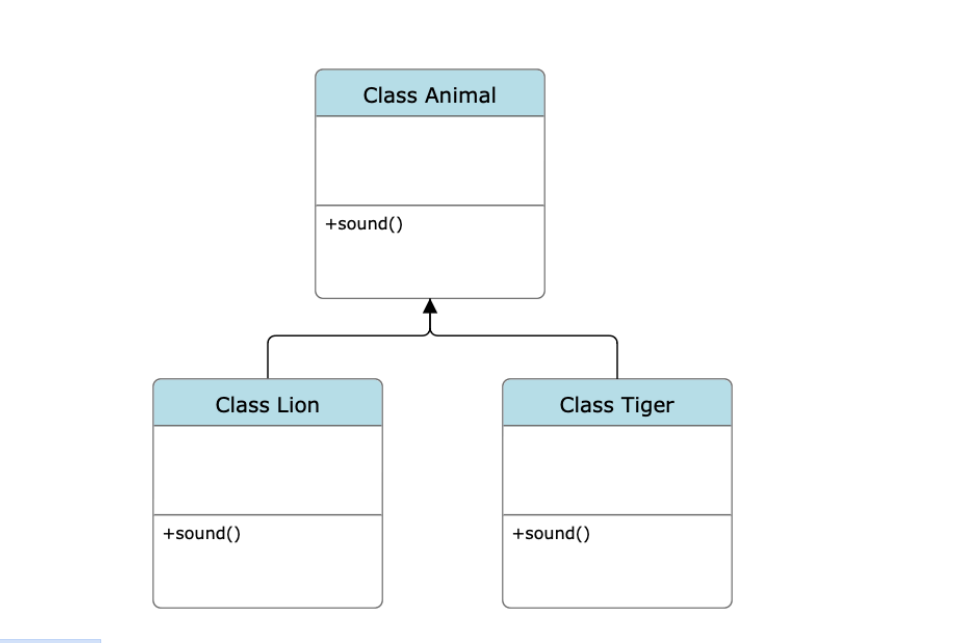
Write a java program to create an abstract class Animal with an

abstract method called sound(). Create subclasses Lion and Tiger

that extend the Animal class and implement the sound() method to

make a specific sound for each animal. 49

**Class Diagram:**



**Program:**

abstract class Animal {

public abstract void sound();

}

class Lion extends Animal {

@Override

public void sound() {

System.out.println("Lion: Roar!");

}

}

class Tiger extends Animal {

@Override

public void sound() {

System.out.println("Tiger: Growl!");

}

}

public class Q1labw7 {

public static void main(String[] args) {

Animal lion = new Lion();

Animal tiger = new Tiger();

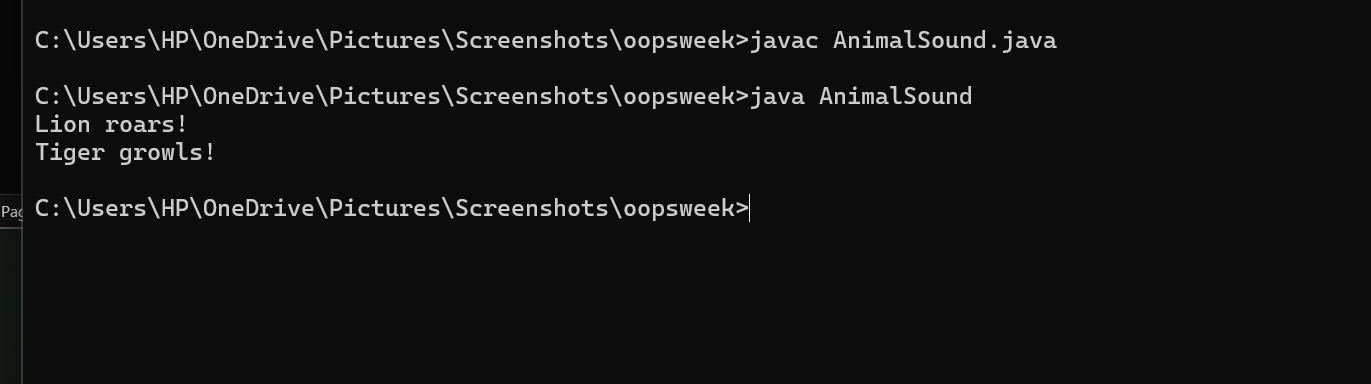
lion.sound();

tiger.sound();

}

}

OUTPUT:

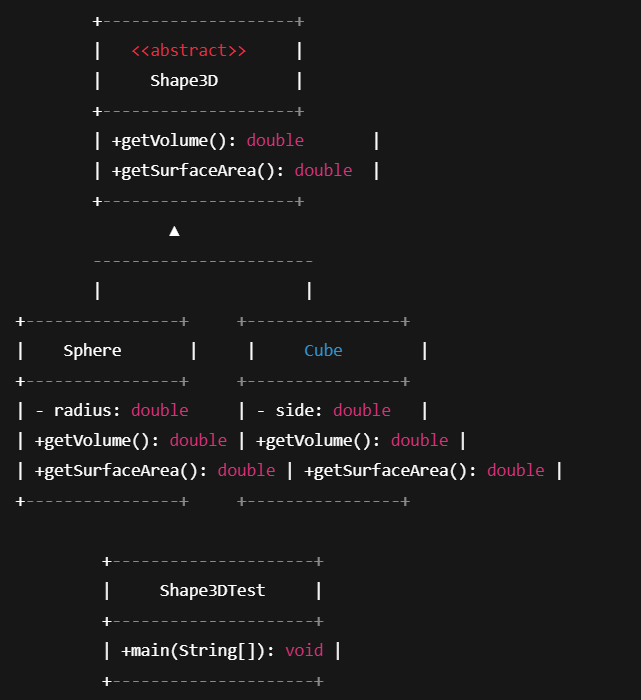


|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error** | **Solution** |
| **1** | void sound(); in a regular class | In non-abstract classes, methods must have bodies unless declared abstract |
| **2** | @overide | Causes compile error — annotation is case-sensitive=@override |

QUESTION-2

**Aim:** Write a Java program with an abstract class shape3D with methods calculateVolume() and calculateSurfaceArea() and subclasses sphere and cube that extend shape3D and implement methods.

**Class diagram:**



**Procedure**

**Step1:** Create a new file in Notepad name that file with the “.java” extension make sure that the class name and file name are the same for errorless code running

**Step2:** Write the code with an abstract class shape3D with methods calculateVolume() and calculateSurfaceArea() and subclasses sphere and cube that extend shape3D and implement methods.

PROGRAM:

abstract class shape3D {

abstract double getVolume();

abstract double getSurfaceArea();

}

class Sphere extends shape3D {

private double radius;

public Sphere(double radius) {

this.radius = radius;

}

@Override

double getVolume() {

return (4.0 / 3.0) \* Math.PI \* Math.pow(radius, 3);

}

@Override

double getSurfaceArea() {

return 4 \* Math.PI \* Math.pow(radius, 2);

}

}

class cube extends shape3D {

private double side;

public cube(double side) {

this.side = side;

}

@Override

double getVolume() {

return Math.pow(side, 3);

}

@Override

double getSurfaceArea() {

return 6 \* Math.pow(side, 2);

}

}

public class shape\_3D {

public static void main(String[] args) {

shape3D sphere = new Sphere(5);

System.out.println("Sphere Volume: " + sphere.getVolume());

System.out.println("Sphere Surface Area: " + sphere.getSurfaceArea());

shape3D cube = new cube(4);

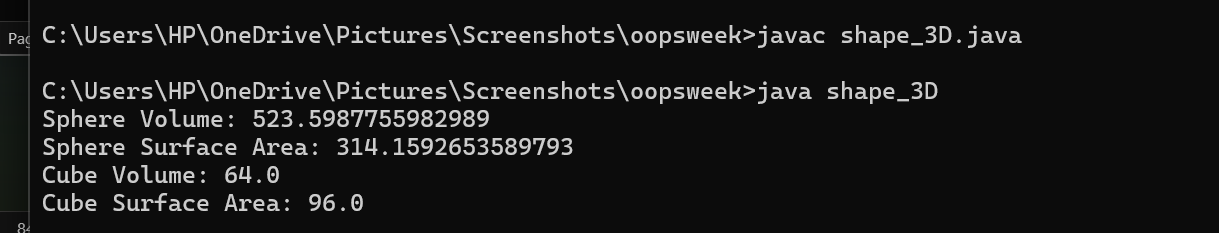
System.out.println("Cube Volume: " + cube.getVolume());

System.out.println("Cube Surface Area: " + cube.getSurfaceArea());

}

}

OUTPUT:



ERRORS:

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error** | **Solution** |
| **1** | Math.pow(radius 3) | Missing comma — should be Math.pow(radius, 3) |
| **2** | getVolume() | Return type must be double as declared in abstract method |

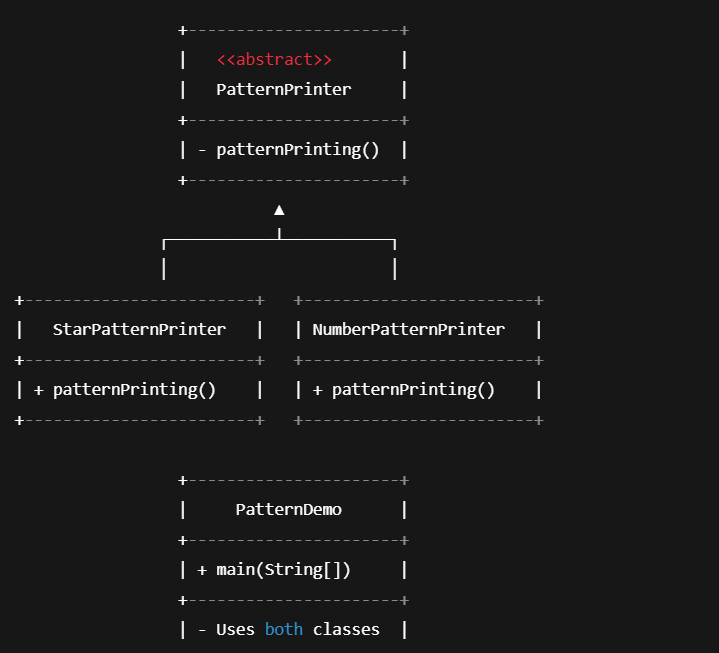
QUESTION:3

**Aim:** Write a Java program using an abstract class patternPrinter to define a method called patternPrinting().

Star pattern of right angled triangle

Number pattern of right angled triangle

**Class diagram:**



**Procedure**

**Step1:** Create a new file in Notepad name that file with the “.java” extension make sure that the class name and file name are the same for errorless code running

**Step2:** Write a Java program using an abstract class patternPrinter to define a method called patternPrinting().

Star pattern of right angled triangle

Number pattern of right angled triangle

PROGRAM:

abstract class PatternPrinter {

abstract void patternPrinting(int rows);

}

class StarPatternPrinter extends PatternPrinter {

@Override

void patternPrinting(int rows) {

System.out.println("Star Pattern:");

for (int i = 1; i <= rows; i++) {

for (int j = 1; j <= i; j++) {

System.out.print("\* ");

}

System.out.println();

}

}

}

class NumberPatternPrinter extends PatternPrinter {

@Override

void patternPrinting(int rows) {

System.out.println("Number Pattern:");

for (int i = 1; i <= rows; i++) {

for (int j = 1; j <= i; j++) {

System.out.print(j + " ");

}

System.out.println();

}

}

}

public class PatternDemo {

public static void main(String[] args) {

int totalRows = 5;

PatternPrinter starPrinter = new StarPatternPrinter();

starPrinter.patternPrinting(totalRows);

System.out.println();

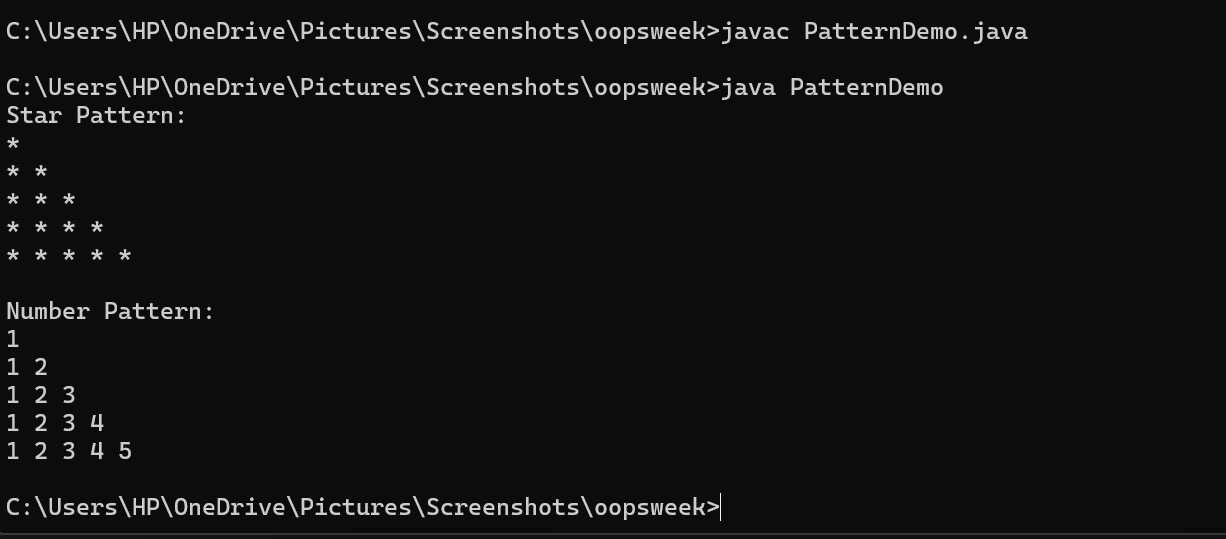
PatternPrinter numberPrinter = new NumberPatternPrinter();

numberPrinter.patternPrinting(totalRows);

}

}

OUTPUT:



ERROS:

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error** | **Solution** |
| **1** | abstract void patternPrinting(int rows) | Missing semicolon ; at the end of method declaration |
| **2** | reached end of file while parsing | } missing after creating sub class |