Project 1 - Polymorphism/Inheritance OOP

Ryan A Burkhardt

UMGC

CMSC 335

Professor Ghosh

28May2024

Polymorphism/Inheritance OOP – Approach

For my approach to this project, I started by reading all required documentations in the Project 1 pdf. I then went into [Calculator.net](https://www.calculator.net/area-calculator.html) to pull all required calculations for area and volume. After having all the calculations present, I went top down and started creating my UML diagram. After creating the UML diagram I went into structuring the classes and finding out the hierarchy from shape to dimensional shape to the actual objects, square, triangle etc. After creating one class I realized that mostly all the same dimensional classes were reusable and proceeded to copy and paste the classes while changing the incoming variables and appropriate calculations. After the classes were created, I moved to the testing and evaluation of the code. This is where I hit some errors on how I was using the super classes in creating my functions such as setArea and setVolume. After realizing my error in the super class I just put my formula in and the setArea and setVolume that calculated each at the time they were passed in. This works and is efficient to give proper feedback to the user but could be improved.

**Polymorphism/Inheritance OOP – Assumptions**

The assumptions I have made in this assignment is that the user has basic knowledge of programming, will input accurate values such as non-negative integers, triangle class is a right triangle. With respect to the user having programming knowledge, I presume that they can run a javac command and import all appropriate classes as I am just providing the .java files. It is also assumed with their knowledge that they can run terminal-based applications and input appropriate data as instructed on the terminal. With respect to input data, it is assumed that they know the background of the calculation they are trying to perform. For instance, it is assumed that they know their measurements for each object they will be attempting to pull an area or volume for. If the user has a square, it is assumed that they know they only need one side to calculate the area and not a length and a width. This follows suit for each class created to calculate the area and volume. With the triangle class it is assumed that the user id inputting a right triangle for the area calculation which simply calculates ½(base\*height).

Polymorphism/Inheritance OOP – Not Implemented

In the non-implemented section, I would have liked to have one area where all calculations were present. This could have been a separate class where I defined a simple calculation class where I could have implemented all calculations to use for are and volume. I also would like to add more error handling to the program itself. For example, negative integers and zeroes can currently be inputted into the calculations giving you errors or invalid calculations. This is something that in a full-scale development would have to be addressed and implemented into the code. Another non-implemented method would be for a more user-friendly terminal app. This could have been done by creating a list and having them select from that list like they do but then offering more feedback such as I did in the Torus class where if they do enter incorrect baseline information it tells them what they did wrong and prompts them to correct it. Lastly, I think I would have liked to input the code into the actual class so that the user only sees the class name in the main code. As I did not know exactly what to do for the assignment I chose to do all variable calculations in the main method. This instead would have been a much simpler code such as case 1: circle() and the circle class would have had the code inside for the menu such as enter base enter height and the calculations were done there. This is something I do in python but have not yet started doing in java.

**Polymorphism/Inheritance OOP – User Guide**

The user guide is quite simple for the terminal-based application taking in the assumptions I have listed above. From an IDE you would simply run the main class, and everything would work as is. After you would follow the prompts on the terminal and input a numerical selection from the displayed menu. Next you should again follow the instructions and input the data it is asking for on the terminal until you have concluded the calculations you would like to have made. Lastly to close the application you would select 10 from the main menu or hit n after your calculations are made.

Polymorphism/Inheritance OOP – Lessons Learned

Lessons I learned in this project were that I simply could not create one class and then just implement other classes in the code. For instance, my original plan was to have a shape class then inside that java file have a two-dimensional shape and three-dimensional shape and then also inside that file create all the required objects to calculate the area and volume. I learned that putting them inside the separate java files is when you are creating the objects and using supers and overrides in the separate java classes is when you are creating that implemented override.

**Polymorphism/Inheritance OOP – Possible improvements**

For possible improvements I would start with implementing all error handling inside the program. This is essential and should be improved on if this were to be implemented in the real world. My second improvement would be to use a GUI based application such as JavaFX with text boxes and a calculate button. I feel like this would give the user a much more interactive experience with the application and make it much easier to use.

Polymorphism/Inheritance OOP – UML

A screenshot of a computer

Description automatically generated

Polymorphism/Inheritance OOP – Source Code

/\*\*Shape Class

 \* 18May2024

 \* Ryan Burkhardt

 \* This class will be used to make a shape that passes in a simple int of number of dimensions

 \*/

public class Shape {

    public int numberOfDimensions;

    public Shape(int numberOfDimensions) {

        this.numberOfDimensions = numberOfDimensions;

    }

}

/\*\*Two Dimensional Object Class

 \* 18May2024

 \* Ryan Burkhardt

 \* This class will extend the shape class and add area to be calculated in the

 \* constructor.

 \*/

 //Extend the class

public class TwoDimensionalShape extends Shape {

    public double area;

    //Construct the two dimensional object with area

    public TwoDimensionalShape(int numberOfDimensions, double area) {

        super(2);

        this.area = area;

    }

    //set the area of the object

    public void setArea(double area) {

        this.area = area;

    }

    //get the area of the object

    public double getArea() {

        return Math.round(area\*100)/100.0;

    }

}

/\*\*Three Dimensional Object Class

 \* 18May2024

 \* Ryan Burkhardt

 \* This class will be used to extend the shape class to add three dimension and

 \* the volume to our object it takes in the number of dimensions from the shape

 \* and adds the volume to the three dimensional shape

 \*/

 //Extend our Shape

public class ThreeDimensionalShape extends Shape {

    public double volume;

    //Construct the three dimensional object

    public ThreeDimensionalShape(int numberOfDimensions, double volume) {

        super(3);

        this.volume = volume;

    }

    //Set the volume of the object

    public void setVolume(double volume) {

        this.volume = volume;

    }

    //Get the volume of the object

    public double getVolume() {

        return Math.round(volume\*100)/100.0;

    }

}

/\*\*Circle Class

 \* 18May2024

 \* Ryan Burkhardt

 \* This class will be used to make a Circle object that calculates

 \* its area based off Pir^2 using its parent class two dimensional

 \* shape.

 \*/

public class Circle extends TwoDimensionalShape {

    //Initialize variables

    private double radius;

    //Construct Circle Object and call super for number of dimensions and area

    Circle(int numberOfDimensions, double radius, double area){

        super(numberOfDimensions, (Math.PI \* (radius \* radius)));

        this.radius = radius;

    }

    //getters and setters for Circle class

    public void setRadius(double radius){

        this.radius = radius;

        super.setArea(Math.PI \* (radius \* radius));//update area as you update radius

    }

    public double getRadius(){

        return radius;

    }

}

/\*\*Cone Class

 \* 18May2024

 \* Ryan Burkhardt

 \* This class will be used to make a Cone object that calculates

 \* its area based off the radius and its height using its parent class three dimensional

 \* shape.

 \*/

 //Extend our Three Dimensional Shape

 public class Cone extends ThreeDimensionalShape {

    private double radius;

    private double height;

    //Construct our cone

    Cone(int numberOfDimensions, double radius, double height, double volume){

        super(numberOfDimensions, (1.0/3.0) \* Math.PI \* Math.pow(radius, 2) \* height);;

        this.radius = radius;

    }

    //Set a new radius while updating its volume

    public void setRadius(double radius) {

        this.radius = radius;

        setVolume((1.0/3.0) \* Math.PI \* Math.pow(radius, 2) \* height);

    }

    //Set a new height while updating its volume

    public void setHeight(double height) {

        this.height = height;

        setVolume((1.0/3.0) \* Math.PI \* Math.pow(radius, 2) \* height);

    }

    //Return the radius

    public double getRadius() {

        return radius;

    }

    //Return the height

    public double getHeight() {

        return height;

    }

}

/\*\*Cube Class

 \* 18May2024

 \* Ryan Burkhardt

 \* This class will be used to make a Cube object that calculates

 \* its area based off the side using its parent class three dimensional

 \* shape.

 \*/

 //Extend our Three Dimensional Shape

 public class Cube extends ThreeDimensionalShape {

    private double side;

    //Construct our cube

    Cube(int numberOfDimensions, double side, double volume){

        super(numberOfDimensions, Math.pow(side, 3));;

        this.side = side;

    }

    //Set a new side length while updating its volume

    public void setSide(double side) {

        this.side = side;

        setVolume(Math.pow(side, 3));

    }

    //Return the side length

    public double getSide() {

        return side;

    }

}

/\*\*Cylinder Class

 \* 18May2024

 \* Ryan Burkhardt

 \* This class will be used to make a Cylinder object that calculates

 \* its area based off the radius and its height using its parent class three dimensional

 \* shape.

 \*/

 //Extend our Three Dimensional Shape

 public class Cylinder extends ThreeDimensionalShape {

    private double radius;

    private double height;

    //Construct our cylinder

    Cylinder(int numberOfDimensions, double radius, double height, double volume){

        super(numberOfDimensions, Math.PI \* Math.pow(radius, 2) \* height);;

        this.radius = radius;

    }

    //Set a new radius while updating its volume

    public void setRadius(double radius) {

        this.radius = radius;

        setVolume(Math.PI \* Math.pow(radius, 2) \* height);

    }

    //Set a new height while updating its volume

    public void setHeight(double height) {

        this.height = height;

        setVolume(Math.PI \* Math.pow(radius, 2) \* height);

    }

    //Return the radius

    public double getRadius() {

        return radius;

    }

    //Return the height

    public double getHeight() {

        return height;

    }

}

/\*\*Rectangle Class

 \* 18May2024

 \* Ryan Burkhardt

 \* This class will be used to make a rectangle object that calculates

 \* its area based off length and width using its parent class two dimensional

 \* shape.

 \*/

public class Rectangle extends TwoDimensionalShape {//Extend our class

    //State our variables

    private double length;

    private double width;

    //Construct our object and also set its area inside the object

    public Rectangle(int numberOfDimensions, double length, double width, double area) {

        super(numberOfDimensions, length\*width);

        this.length = length;

        this.width = width;

    }

    //Setters and Getters for the Rectangle class to include setting

    //the area every time the length or width is updated

    public void setLength(double length){

        this.length = length;

        super.setArea(length \* width);

    }

    public void setWidth(double width){

        this.width = width;

        super.setArea(length \* width);

    }

    public double getLength() {

        return length;

    }

    public double getWidth() {

        return width;

    }

}

/\*\*Sphere Class

 \* 18May2024

 \* Ryan Burkhardt

 \* This class will be used to make a Sphere object that calculates

 \* its area based off the radius using its parent class three dimensional

 \* shape.

 \*/

 //Extend our Three Dimensional Shape

public class Sphere extends ThreeDimensionalShape {

    private double radius;

    //Construct our sphere

    Sphere(int numberOfDimensions, double radius, double volume){

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

        this.radius = radius;

    }

    //Set a new radius while updating its volume

    public void setRadius(double radius) {

        this.radius = radius;

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

    }

    //Return the radius

    public double getRadius() {

        return radius;

    }

}

/\*\*Square Class

 \* 18May2024

 \* Ryan Burkhardt

 \* This class will be used to make a square object that calculates

 \* its area based off side squared using its parent class two dimensional

 \* shape.

 \*/

 public class Square extends TwoDimensionalShape {//Extend our class

    //State our variable

    private double side;

    //Construct our object and also set its area inside the object

    public Square(int numberOfDimensions, double side, double area) {

        super(numberOfDimensions, side \* side);

        this.side = side;

    }

    //Getters and Setters for the side

    public void setSide(double side){

        this.side = side;

        super.setArea(side \* side);

    }

    public double getSide(){

        return side;

    }

}

/\*\*Torus Class

 \* 18May2024

 \* Ryan Burkhardt

 \* This class will be used to make a Torus object that calculates

 \* its volume based off the radius and its starting edge from the

 \* center using its parent class three dimensional

 \* shape.

 \*/

 //Extend our Three Dimensional Shape

 public class Torus extends ThreeDimensionalShape {

    private double minorRadius;

    private double majorRadius;

    //Construct our Torus

    Torus(int numberOfDimensions, double minorRadius, double majorRadius, double volume){

        super(numberOfDimensions, 2 \* Math.pow(Math.PI, 2) \* majorRadius \* Math.pow(minorRadius, 2));;

        this.minorRadius = minorRadius;

    }

    //Set a new minor radius while updating its volume

    public void setMinorRadius(double minorRadius) {

        this.minorRadius = minorRadius;

        setVolume(2 \* Math.pow(Math.PI, 2) \* majorRadius \* Math.pow(minorRadius, 2));

    }

    //Set a new major radius while updating its volume

    public void setMajorRadius(double majorRadius) {

        this.majorRadius = majorRadius;

        setVolume(2 \* Math.pow(Math.PI, 2) \* majorRadius \* Math.pow(minorRadius, 2));

    }

    //Return the minor radius

    public double getMinorRadius() {

        return minorRadius;

    }

    //Return the major radius

    public double getMajorRadius() {

        return majorRadius;

    }

}

/\*\*Triangle Class

 \* 18May2024

 \* Ryan Burkhardt

 \* This class will be used to make a triangle object that takes in the number of

 \* dimensions and area from the two dimensional class and adds the base and height

 \* to calculte the are of the triangle. it is assumed that it is an equilateral triangle.

 \*/

public class Triangle extends TwoDimensionalShape {

    //Initialize our variables

    private double base;

    private double height;

    //Construct our triangle object adding base and height

    Triangle(int numberOfDimensions, double base, double height, double area){

        super(numberOfDimensions, (.5 \* (base \* height)));

        this.base = base;

        this.height = height;

    }

    //Getters and Setters for Triangle class to include updating our area using

    //the parent class.

    public void setBase(double base){

        this.base = base;

        super.setArea((.5 \* (base \* height)));

    }

    public void setHeight(double height) {

        this.height = height;

        super.setArea((.5 \* (base \* height)));

    }

    public double getBase(){

        return base;

    }

    public double getHeight(){

        return height;

    }

}

Polymorphism/Inheritance OOP – Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test # | Purpose of Test | Positive or Negative Test | Input Values | Expected Result | Pass/Fail |
| 1 | Test Exception handing for entering non numerical data as a selection | Negative | “The” | Should prompt the user to re-enter their selection. |  |
| 2 | Test Exception handling for the Torus class by entering a larger minor radius. | Negative | Minor radius = 5  Major radius = 1 | Should provide error feedback and prompt user to reenter the information. |  |
| 3 | Test conditional statements for continuing to use the program by entering a numerical data instead of “y” or “n”. | Negative | 1 | Should prompt the user to make an appropriate selection. |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |
| 8 |  |  |  |  |  |
| 9 |  |  |  |  |  |

References

Last Name, A. B. (Year). Article Title. *Journal Title*, Pages #-#. URL.

Last Name, C. D. (Year). *Book Title (Edition).* Publisher Name. URL.

Last Name, D. E., Last Name, F. G., Last Name, H. I. (Year). *Report Title* (report number). Publisher. URL.

Last Name, J. K. (Year, Month Day). *Article Title/Headline*. Periodical. URL.

Organization Name. (Year, Month Day). *Webpage Title*. URL.