# Approach

My approach to this project was to be organized and complete each piece once at a time. I started by first writing on a piece of paper how I would organize each shape into their respective branch. I did this creating a sketch UML of how I wanted it to be. I started with the base shape class, then decided to do two interfaces for Two- and Three-dimensional shapes. After that I refined each specific shape class by overriding the methods in the interface and tailoring them to what they needed.

# Assumptions

The only assumption I made about this project was that it was very simple. The instructions were straight forward and gave a clear explanation on what was needed. My assumption was correct, and the project was relatively simple to accomplish.

# Not Implemented

The only thing that was not implemented was adding checks to make sure that the inputted information using the menu option in the console. I could have implemented this but the program wouldn’t compile anyway if the proper info wasn’t entered (i.e letters instead of numbers).

# How to set-up and Compile the Program

1. Download the program
2. Unzip the project and place into a folder.
3. Open a new project in preferred java software.
4. Select to start project with the downloaded project that was placed into the folder.

# Lessons Learned

While accomplishing this project I grasped a deeper understanding of the Parent and Child concepts. Parent and child class are fundamental to understanding inheritance. Parent class is usually extended or inherited by another class. Parent class contain attributes and methods that can be shared with child classes. Child class inherits from a parent class, it can use methods and attributes of a parent class but can also have its own additional attributes and methods. The main benefit of using this concept is that helps create code reusability and creates a structure for your code which allows the building of complex systems more efficiently.

# Possible Improvements

My code could have been improved by making the base class shape more useful regarding using more reusable variables. The next improvement I could have made would have been to add more input checks to make sure that correct info was inputted, and the program would continue to compile after that.

UML

Abstract Class Shape {

Attributes:

private int numberOfDimensions;  
private int area;

Methods:

public Shape(int numberOfDimensions)

public int getNumberOfDimensions()

Public Interface ThreeDimensionalShape

double getVolume();

Public Interface TwoDimensionalShape

double getArea();

All classes listed here contain their own version of the getVolume() method

public class Cone

public class Cylinder

public class Sphere

public class Torus

All classes listed here contain their own version of the getArea() method

public class Circle

public class Rectangle

public class Square

public class Triangle

# Source Code

Test.java

public class Test {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int choice;  
 do {  
 System.*out*.println("Welcome to the Java OO Shapes Program");  
 System.*out*.println("Select from the menu below:");  
 System.*out*.println("1. Construct a Circle");  
 System.*out*.println("2. Construct a Rectangle");  
 System.*out*.println("3. Construct a Square");  
 System.*out*.println("4. Construct a Triangle");  
 System.*out*.println("5. Construct a Sphere");  
 System.*out*.println("6. Construct a Cube");  
 System.*out*.println("7. Construct a Cone");  
 System.*out*.println("8. Construct a Cylinder");  
 System.*out*.println("9. Construct a Torus");  
 System.*out*.println("10. Exit the program");  
 System.*out*.print("Enter your choice: ");  
 choice = scanner.nextInt();  
  
 Shape shape = null;  
 switch (choice) {  
 case 1:  
 System.*out*.print("Enter radius: ");  
 double radius = scanner.nextDouble();  
 shape = new Circle(radius);  
 System.*out*.println("Area of Circle: " + ((Circle) shape).getArea());  
 break;  
 case 2:  
 System.*out*.print("Enter length: ");  
 double length = scanner.nextDouble();  
 System.*out*.print("Enter width: ");  
 double width = scanner.nextDouble();  
 shape = new Rectangle(length, width);  
 System.*out*.println("Area of Rectangle: " + ((Rectangle) shape).getArea());  
 break;  
 case 3:  
 System.*out*.print("Enter side: ");  
 double side = scanner.nextDouble();  
 shape = new Square(side);  
 System.*out*.println("Area of Square: " + ((Square) shape).getArea());  
 break;  
 case 4:  
 System.*out*.print("Enter base: ");  
 double base = scanner.nextDouble();  
 System.*out*.print("Enter height: ");  
 double height = scanner.nextDouble();  
 shape = new Triangle(base, height);  
 System.*out*.println("Area of Triangle: " + ((Triangle) shape).getArea());  
 break;  
 case 5:  
 System.*out*.print("Enter radius: ");  
 radius = scanner.nextDouble();  
 shape = new Sphere(radius);  
 System.*out*.println("Volume of Sphere: " + ((Sphere) shape).getVolume());  
 break;  
 case 6:  
 System.*out*.print("Enter side: ");  
 side = scanner.nextDouble();  
 shape = new Cube(side);  
 System.*out*.println("Volume of Cube: " + ((Cube) shape).getVolume());  
 break;  
 case 7:  
 System.*out*.print("Enter radius: ");  
 radius = scanner.nextDouble();  
 System.*out*.print("Enter height: ");  
 height = scanner.nextDouble();  
 shape = new Cone(radius, height);  
 System.*out*.println("Volume of Cone: " + ((Cone) shape).getVolume());  
 break;  
 case 8:  
 System.*out*.print("Enter radius: ");  
 radius = scanner.nextDouble();  
 System.*out*.print("Enter height: ");  
 height = scanner.nextDouble();  
 shape = new Cylinder(radius, height);  
 System.*out*.println("Volume of Cylinder: " + ((Cylinder) shape).getVolume());  
 break;  
 case 9:  
 System.*out*.print("Enter major radius: ");  
 double majorRadius = scanner.nextDouble();  
 System.*out*.print("Enter minor radius: ");  
 double minorRadius = scanner.nextDouble();  
 shape = new Torus(majorRadius, minorRadius);  
 System.*out*.println("Volume of Torus: " + ((Torus) shape).getVolume());  
 break;  
 case 10:  
 System.*out*.println("Thank you for using the Java OO Shapes Program!");  
 DateTimeFormatter dtf = DateTimeFormatter.*ofPattern*("yyyy/MM/dd HH:mm:ss");  
 LocalDateTime now = LocalDateTime.*now*();  
 System.*out*.println("Current date and time: " + dtf.format(now));  
 break;  
 default:  
 System.*out*.println("Invalid choice. Please try again.");  
 }  
 } while (choice != 10);  
 scanner.close();  
 }  
 }

Circle.java

public class Circle extends Shape implements TwoDimensionalShape {  
 private double radius;  
  
 public Circle(double radius){  
 super(2);  
 this.radius = radius;  
 }  
 @Override  
 public double getArea() {  
 return Math.*PI* \* radius \* radius;  
 }  
}

Cone.java

public class Cone extends Shape implements ThreeDimensionalShape {  
 private double radius;  
 private double height;  
  
 public Cone(double radius, double height) {  
 super(3);  
 this.radius = radius;  
 this.height = height;  
 }  
  
 @Override  
 public double getVolume() {  
 return (1.0 / 3.0) \* Math.*PI* \* Math.*pow*(radius, 2) \* height;  
 }  
}

Cube.java

public class Cube extends Shape implements ThreeDimensionalShape {  
 private double side;  
 public Cube(double side) {  
 super(3);  
 this.side = side;  
 }  
  
 @Override  
 public double getVolume() {  
 return Math.*pow*(side, 3);  
 }  
}

Cylinder.java

public class Cylinder extends Shape implements ThreeDimensionalShape {  
 private double radius;  
 private double height;  
  
 public Cylinder(double radius, double height) {  
 super(3);  
 this.radius = radius;  
 this.height = height;  
 }  
  
 @Override  
 public double getVolume() {  
 return Math.*PI* \* Math.*pow*(radius, 2) \* height;  
 }  
}

Rectangle.java

public class Rectangle extends Shape implements TwoDimensionalShape {  
 private double length;  
 private double width;  
  
 public Rectangle(double length, double width) {  
 super(2);  
 this.length = length;  
 this.width = width;  
 }  
  
 @Override  
 public double getArea() {  
 return length \* width;  
 }  
}

Shape.java

abstract class Shape {  
 private int numberOfDimensions;  
 private int area;  
  
 public Shape(int numberOfDimensions) {  
 this.numberOfDimensions = numberOfDimensions;  
 }  
  
 public int getNumberOfDimensions() {  
 return numberOfDimensions;  
 }  
}

Sphere.java

public class Sphere extends Shape implements ThreeDimensionalShape {  
 private double radius;  
  
 public Sphere(double radius) {  
 super(3);  
 this.radius = radius;  
 }  
  
 @Override  
 public double getVolume() {  
 return (4.0 / 3.0) \* Math.*PI* \* Math.*pow*(radius, 3);  
 }  
}

Square.java

public class Square extends Shape implements TwoDimensionalShape {  
 private double side;  
  
 public Square(double side) {  
 super(2);  
 this.side = side;  
 }  
  
 @Override  
 public double getArea() {  
 return side \* side;  
 }  
}

ThreeDimensionalShape.java

public interface ThreeDimensionalShape {  
 double getVolume();  
}

Torus.java

public class Torus extends Shape implements ThreeDimensionalShape {  
 private double majorRadius;  
 private double minorRadius;  
  
 public Torus(double majorRadius, double minorRadius) {  
 super(3);  
 this.majorRadius = majorRadius;  
 this.minorRadius = minorRadius;  
 }  
  
 @Override  
 public double getVolume() {  
 return (Math.*PI* \* Math.*pow*(minorRadius, 2)) \* (2 \* Math.*PI* \* majorRadius);  
 }  
}

Triangle.java

public class Triangle extends Shape implements TwoDimensionalShape {  
 private double base;  
 private double height;  
  
 public Triangle(double base, double height) {  
 super(2);  
 this.base = base;  
 this.height = height;  
 }  
  
 @Override  
 public double getArea() {  
 return 0.5 \* base \* height;  
 }  
}

TwoDimensionalShape.java

public interface TwoDimensionalShape {  
 double getArea();  
}

# Test Plan

To test this program, I will be performing checks with console menu. The main goal of this test plan is to make sure that each option compiles correctly and can utilize the correct formula associated with the class. The reason why I will be only testing this way is because if the logic isn’t right with the parent classes the program will not compile because of the organizational errors.

# Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| Test # | Description | Screenshot | Pass/Fail |
| 1 | Testing choice 1 for Circle | A black background with white text and numbers  Description automatically generated | Pass |
| 2 | Testing choice 2 for Rectangle | A black screen with white text  Description automatically generated | Pass |
| 3 | Testing choice 3 for Square | A screenshot of a phone  Description automatically generated | Pass |
| 4 | Testing choice 4 for Triangle | A screenshot of a black screen  Description automatically generated | Pass |
| 5 | Testing choice 5 for Sphere | A black background with white numbers  Description automatically generated | Pass |
| 6 | Testing choice 6 for Cube | A black background with white text  Description automatically generated | Pass |
| 7 | Testing choice  7 for Cone | A screenshot of a black screen  Description automatically generated | Pass |
| 8 | Testing choice 8 for Cylinder | A screenshot of a phone  Description automatically generated | Pass |
| 9 | Testing choice 9 for Torus | A screenshot of a black background with white text  Description automatically generated | Pass |
| 10 | Testing choice 10 for the exit menu prompt | A black background with white text  Description automatically generated | Pass |