Shape in Shapes

Written and

Implemented By

MD JAODUN MUNTASIR (BVLLR5)

Course: Programming Technology

Faculty of Informatics

Eötvös Loránd University

Date: 11 OCT 2023

Contents

Task

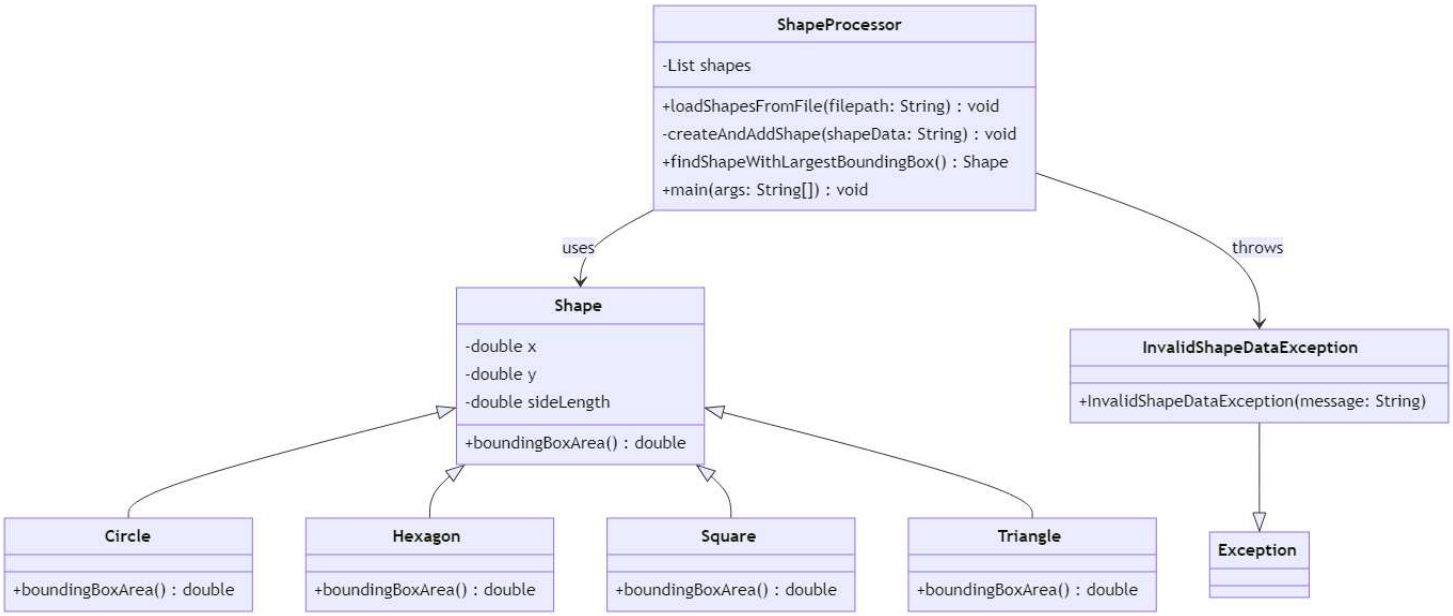
Fill a collection with several regular shapes (circle, regular triangle, square, regular hexagon). **Which shape has the greatest bounding box area?**

A bounding box of a shape covers the shape completely, and its sides are parallel with the x or y axis. Each shape can be represented by its center and side length (or radius), if we assume that one side of the polygons are parallel with the x axis, and its nodes lie on or above this side. Load and create the shapes from a text file. The first line of the file contains the number of the shapes, and each following line contains a shape. The first character will identify the type of the shape, which is followed by the center coordinate and the side length or radius. Manage the shapes uniformly, so derive them from the same super class.

Description of the task

In this task, the primary objective is to manage and analyze a collection of shapes, specifically geometric figures like circles, squares, triangles, and hexagons. The shapes are processed based on their geometric characteristics, stored uniformly in a collection, and analyzed to identify the shape with the largest bounding box area. Exception handling ensures graceful error management, particularly for input data validation.

UML Diagram


ShapeProcessor
-List shapes
+loadShapesFromFile(filepath: String) : void
-createAndAddShape(shapeData: String) : void
+findShapeWithLargestBoundingBox() : Shape
+main(args: String[]) : void
uses
throws
Shape
-double x
-double y
-double sideLength
+boundingBoxArea() : double
InvalidShapeDataException
+InvalidShapeDataException(message: String)
Circle
+boundingBoxArea() : double
Hexagon
+boundingBoxArea() : double
Square
+boundingBoxArea() : double
Triangle
+boundingBoxArea() : double
Exception

Description of the methods

1) Shape (Abstract Superclass)
    a) Shape(double x, double y, double sideLength)
        i) Description: Constructor that initializes the shape's center coordinates and side length/radius.
    b) Parameters:
        i) x: X-coordinate of the center.
        ii) y: Y-coordinate of the center.
    c) sideLength: Length of a side or radius.
    d) boundingBoxArea(): double
        i) Description: Abstract method intended to provide a template for calculating the bounding box area in derived classes.
2) Circle (Subclass of Shape)
    a) Circle(double x, double y, double radius)
        i) Description: Constructor that initializes the circle's properties.
    b) Parameters:
        i) x, y: Center coordinates.
    c) radius: Radius of the circle.
    d) boundingBoxArea(): double
        i) Description: Calculates and returns the area of the bounding box for the circle.
        ii) Returns: The area of the bounding box.
3) Hexagon (Subclass of Shape)
    a) Methods parallel to Circle but with specific calculations related to hexagons in boundingBoxArea().
4) Square & Triangle (Subclasses of Shape)
    a) Similar method descriptions to Circle and Hexagon, with relevant geometric calculations in boundingBoxArea().
5) InvalidShapeDataException (Custom Exception Class)
    a) InvalidShapeDataException(String message)

i) Description: Constructor that initializes the exception with a custom message.

b) Parameter:

i) message: A string detailing the error message associated with the exception.

6) ShapeProcessor (Utility Class)

a) loadShapesFromFile(String filepath): void

i) Description: Loads shape data from a specified file and creates respective shape objects.

ii) Parameter:

1) filepath: Path to the input file.

iii) Throws:

1) IOException: If an I/O error occurs.

iv) InvalidShapeDataException: If shape data is invalid.

b) createAndAddShape(String shapeData): void

i) Description: Parses, validates, and creates a shape object based on input data, then adds it to the collection.

ii) Parameter:

1) shapeData: A string containing data to create a shape.

iii) Throws:

1) InvalidShapeDataException: If shape data is invalid.

c) findShapeWithLargestBoundingBox(): Shape

i) Description: Identifies and returns the shape with the largest bounding box area from the collection.

ii) Returns: The shape object with the largest bounding box area.

d) main(String[] args): void

i) Description: Entry point of the program. Loads shapes, performs analyses, and handles potential exceptions.

ii) Parameter:

1) args: Command-line arguments.

Testing

White Box Test Cases

Type of test	Input	Expected Output	Original Output
Sample Input with Different Results	shapes.txt	Shape with the largest bounding box: Circle Bounding box area: 100.0	Shape with the largest bounding box: Circle Bounding box area: 100.0
Sample Input with Different Results	shapes2.txt	Shape with the largest bounding box: Square Bounding box area: 100.0	Shape with the largest bounding box: Square Bounding box area: 100.0

Black Box Test Cases

Type of test	Input	Expected Output	Original Output
Negative Radius	shapes3.txt	Error: Invalid shape data. Invalid radius/side length: -3.0	Error: Invalid shape data. Invalid radius/side length: -3.0
Invalid Shape Type	shapes4.txt	Error: Invalid shape data. Unknown shape type: X	Error: Invalid shape data. Unknown shape type: X
Insufficient Data	shapes5.txt	Error: Invalid shape data. Invalid data format: C 1 1	Error: Invalid shape data. Invalid data format: C 1 1
Non-numeric Data	shapes6.txt	Error: An unexpected error occurred. For input string: "one"	Error: An unexpected error occurred. For input string: "one"
Too Many Shapes Declared	shapes7.txt	Error: Invalid shape data. The number of shapes declared (3) does not match the actual number of shapes provided (4).	Error: Invalid shape data. The number of shapes declared (3) does not match the actual number of shapes provided (4).