

# Bounding Box Area Comparison of Regular Shapes

**Author:** Ulukbek Tursunov – EY594J

ELTE Faculty of Informatics

Programming technology - Assignment 1, Task 10

## 1. Description of the Exercise

The goal of this exercise is to:

- Load several regular shapes (circle, regular triangle, square, and regular hexagon) from a text file.
- Represent these shapes using a common superclass (`Shape`), and create specific subclasses for each shape type.
- Compute the bounding box area of each shape.
- Determine which shape has the largest bounding box area.

The **bounding box** of a shape is the smallest rectangle, aligned with the coordinate axes, that completely contains the shape.

Each shape is described by its **center coordinates** ( $x, y$ ) and **size parameter** (side length for polygons, radius for circles). The program reads this information from a text file, validates the data, and prints which shape has the largest bounding box.

## 2. Program Overview

The solution follows object-oriented design principles. The classes are structured as follows:

Class	Description
Main	The entry point of the program. Handles file input, validation, shape creation, and result calculation.
Shape	Abstract superclass for all shape types. Contains shared properties and declares the <code>calculateBoxArea()</code> abstract method.
Circle	Subclass representing a circle. Calculates its bounding box area as $4r^2$ .
Square	Subclass representing a square. Bounding box area = $a^2$ .
Triangle	Subclass representing an equilateral triangle. Bounding box area = $\frac{\sqrt{3}}{2}a^2$ .
Hexagon	Subclass representing a regular hexagon. Bounding box area = $2\sqrt{3}a^2$ .

`InvalidInputException` Custom exception class for handling invalid or malformed input data.

---

### 3. Input File Format

The input file must follow this structure:

```
<number_of_shapes>
<ShapeType> <centerX> <centerY> <sideLength_or_radius>
```

Where:

- `ShapeType`: C = Circle, S = Square, T = Triangle, H = Hexagon
- `centerX`, `centerY`: Coordinates of the shape's center
- `sideLength_or_radius`: Radius for circles or side length for polygons

**Example:**

```
5
C -2 -4 5
T 0 0 4
S 0 0 4
H 0 0 3
C 9 8 6
```

### 4. Example Execution

**Input:**

```
5
C -2 -4 5
T 0 0 4
S 0 0 4
H 0 0 3
C 9 8 6
```

**Step-by-Step Calculation:**

Shape	Parameters	Formula	Bounding Box Area
-------	------------	---------	-------------------

Circle (r=5)	radius = 5	$4r^2$	100.00
Triangle (a=4)	side = 4	$\frac{\sqrt{3}}{2}a^2$	13.86
Square (a=4)	side = 4	$a^2$	16.00
Hexagon (a=3)	side = 3	$2\sqrt{3}a^2$	31.18
Circle (r=6)	radius = 6	$4r^2$	144.00

---

## Output:

```
Shape with the largest bounding box: Circle
Bounding box area: 144.0000
```

## 5. Error Handling

The program validates input data thoroughly:

- Ensures the file exists and is not empty.
- Confirms that the first line contains a positive integer (number of shapes).
- Verifies each shape line includes valid parameters.
- Checks that side lengths and radii are positive.
- Detects and reports extra or missing data.
- Throws descriptive errors using the custom `InvalidInputException`.

### Example error messages:

```
Input error: File shapes.txt not found!
Input error: File is empty!
Input error: Side length or radius must be positive at line 3
Input error: File contains extra data after the expected 5 shapes.
```

## 6. Object-Oriented Concepts Applied

**Inheritance:** All shape types inherit from a common superclass `Shape`, sharing common data and behavior.

**Polymorphism:** The `calculateBoxArea()` method is overridden in each subclass, allowing uniform handling of all shapes in a single collection (`List<Shape>`).

**Abstraction:** The abstract class `Shape` defines the contract for all shape types while leaving implementation details to subclasses.

**Encapsulation:** Shape attributes (`centerX`, `centerY`, `side`) are protected, accessed, and modified only through getter and setter methods.

**Exception Handling:** The custom `InvalidInputException` class provides clean error management and user-friendly messages.

## 7. Key Formulas

Shape	Bounding Box Area Formula	Description
Circle	$4r^2$	Square bounding box with side = $2r$
Square	$a^2$	Bounding box equals the square itself
Triangle	$\frac{\sqrt{3}}{2}a^2$	Derived from the triangle's height and base
Hexagon	$2\sqrt{3}a^2$	Derived from hexagon's height and width

## 8. Results and Verification

When tested with the provided `shapes.txt` file:

- The program correctly reads all shapes.
- The bounding box areas are calculated accurately.
- The program outputs the shape with the largest bounding box: **Circle (radius = 6)**.

### Final Output:

```
Shape with the largest bounding box: Circle
Bounding box area: 144.0000
```

## 9. Conclusion

The program fulfills all requirements:

- Uses object-oriented programming principles effectively.
- Handles input validation and error reporting gracefully.
- Computes bounding box areas accurately for all supported shapes.
- Produces clear and correct output.

This implementation is modular, easily extendable (additional shapes could be added by subclassing `Shape`), and adheres to proper design and documentation standards.

# UML Class Diagram

The following UML class diagram illustrates the object-oriented structure of the program. It shows the inheritance hierarchy where all specific shape classes (`Circle`, `Square`, `Triangle`, and `Hexagon`) derive from the common abstract superclass `Shape`. Each subclass implements its own version of the `calculateBoxArea()` method to determine the bounding box area based on its geometric properties. The diagram also highlights the `Main` class, which serves as the program's entry point, and the `InvalidInputException` class, which is responsible for handling input validation errors.

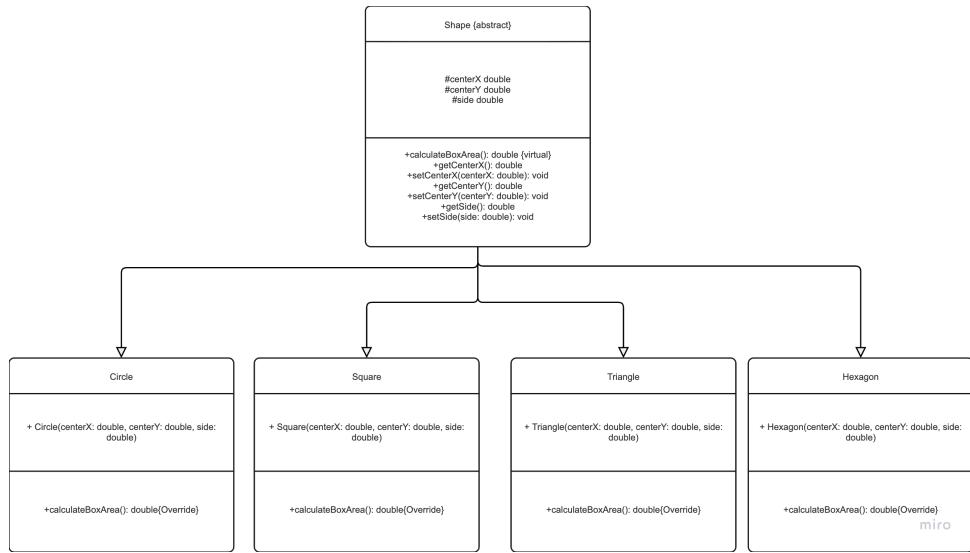


Figure 1: UML Class Diagram of the Shape Hierarchy