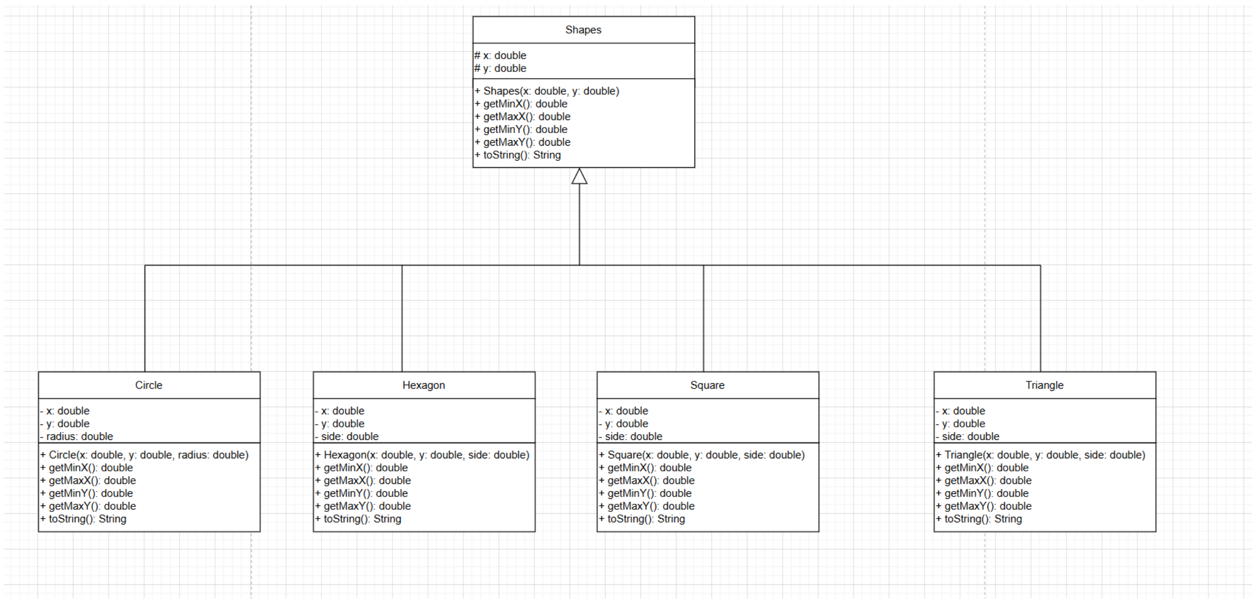
Description of task

7. Fill a collection with several regular shapes (circle, regular triangle, square, regular hexagon). Determine the smallest bounding box, which contains all the shapes, and its sides parallel with an 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 x axis, and its nodes lies 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 contain 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.

UML Class Diagram



Methods description

Shapes.java

- getMinX() → Gives the left border of the shape.
- getMaxX() → Gives the right border of the shape.
- getMinY() → Gives the bottom border of the shape.
- getMaxY() → Gives the top border of the shape.
- getX() → Gives the X coordinate of the shape's center.
- getY() → Gives the Y coordinate of the shape's center.
- toString() → Shows the type of shape and its center.

Circle.java

- getMinX() / getMaxX() / getMinY() / getMaxY() → Gives the edges of the circle.
- toString() → Shows circle's center and radius.

Square.java

- getMinX() / getMaxX() / getMinY() / getMaxY() → Gives the edges of the square.
- toString() → Shows square's center and side length.

Triangle.java

- getMinX() / getMaxX() / getMinY() / getMaxY() → Gives the edges of the triangle.
- toString() → Shows triangle's center and side length.

Hexagon.java

- getMinX() / getMaxX() / getMinY() / getMaxY() → Gives the edges of the hexagon.
- toString() → Shows hexagon's center and side length.

ShapeMain.java

- main() → Loads shapes from a file and calculates the smallest bounding box containing all shapes.

Test cases

Input file: shapes1

Input:

```
5

C 0 0 5

T 2 2 3

S -4 -4 6

H 5 5 2

C -3 3 1
```

Output:

```
Circle at (0.0, 0.0) radius = 5.0

Triangle at (2.0, 2.0) side=3.0

Square at (-4.0, -4.0) side=6.0

Hexagon at (5.0, 5.0) side = 2.0

Circle at (-3.0, 3.0) radius = 1.0


Bounding Box:

Bottom-left  = (-7.0, -7.0)

Top-left     = (-7.0, 6.7)

Bottom-right = (7.0, -7.0)
```

```
Top-right    = (7.0, 6.7)

Width  = 14.0

Height = 13.7
```

Input file: shapes2

Input:

```
3

C 0 0 2

S 1 1 4

T -1 -1 3
```

Output:

```
Circle at (0.0, 0.0) radius = 2.0

Square at (1.0, 1.0) side=4.0

Triangle at (-1.0, -1.0) side=3.0


Bounding Box:

Bottom-left  = (-2.5, -2.0)

Top-left     = (-2.5, 3.0)

Bottom-right = (3.0, -2.0)

Top-right    = (3.0, 3.0)

Width  = 5.5

Height = 5.0
```

input file: shapes3

Input:

```
6

H 0 0 2

C 2 2 1
```

```
S -3 4 5

T 1 -2 3

H -1 -1 1

C 0 5 2
```

Output:

```
Hexagon at (0.0, 0.0) side = 2.0

Circle at (2.0, 2.0) radius = 1.0

Square at (-3.0, 4.0) side=5.0

Triangle at (1.0, -2.0) side=3.0

Hexagon at (-1.0, -1.0) side = 1.0

Circle at (0.0, 5.0) radius = 2.0


Bounding Box:

Bottom-left  = (-5.5, -2.9)

Top-left     = (-5.5, 7.0)

Bottom-right = (3.0, -2.9)

Top-right    = (3.0, 7.0)

Width  = 8.5

Height = 9.9
```

Input file: shapes4

Input:

```
4

S 0 0 3

T 2 2 4

C -2 -3 -2

H 1 -1 3
```

Output:

```
Exception in thread "main" java.lang.IllegalArgumentException: Radius must be positive.
	at shape.Circle.<init>(Circle.java:17)
	at shape.ShapeMain.main(ShapeMain.java:25)
```

Input file: shapes5

Input:

```
5
C 0 0 5
H 5 5 2
T -2 -2 6
S 3 -1 4
C -3 3 1
```

Output:

```
Circle at (0.0, 0.0) radius = 5.0
Hexagon at (5.0, 5.0) side = 2.0
Triangle at (-2.0, -2.0) side=6.0
Square at (3.0, -1.0) side=4.0
Circle at (-3.0, 3.0) radius = 1.0


Bounding Box:
Bottom-left  = (-5.0, -5.0)
Top-left     = (-5.0, 6.7)
Bottom-right = (7.0, -5.0)
Top-right    = (7.0, 6.7)
Width  = 12.0
Height = 11.7
```

Input file: shapes6

Input:

Empty

Output:

```
Exception in thread "main" java.util.NoSuchElementException: No line found
	at java.base/java.util.Scanner.nextLine(Scanner.java:1660)
	at shape.ShapeMain.main(ShapeMain.java:15)
```

White-box Testing

In this project, I applied white-box testing to the **individual shape classes** (Circle, Square, Triangle, Hexagon).

Black-box Testing

In this project, I used black-box testing to validate the **overall bounding box** calculation when multiple shapes are combined.

I provided different sets of shapes (circles, squares, triangles, hexagons) at various positions and checked that the program returned the correct minimum and maximum x/y values.