Programming Technology

Assignment 1

Task 6

Nazifa Nazrul Rodoshi

Neptun code: W8CZNE

Contents

Task.....3

Description of the task.....3

UML Diagram.....3

Description of the methods.....4

Testing.....5

White box test cases.....5

Black box test cases.....5

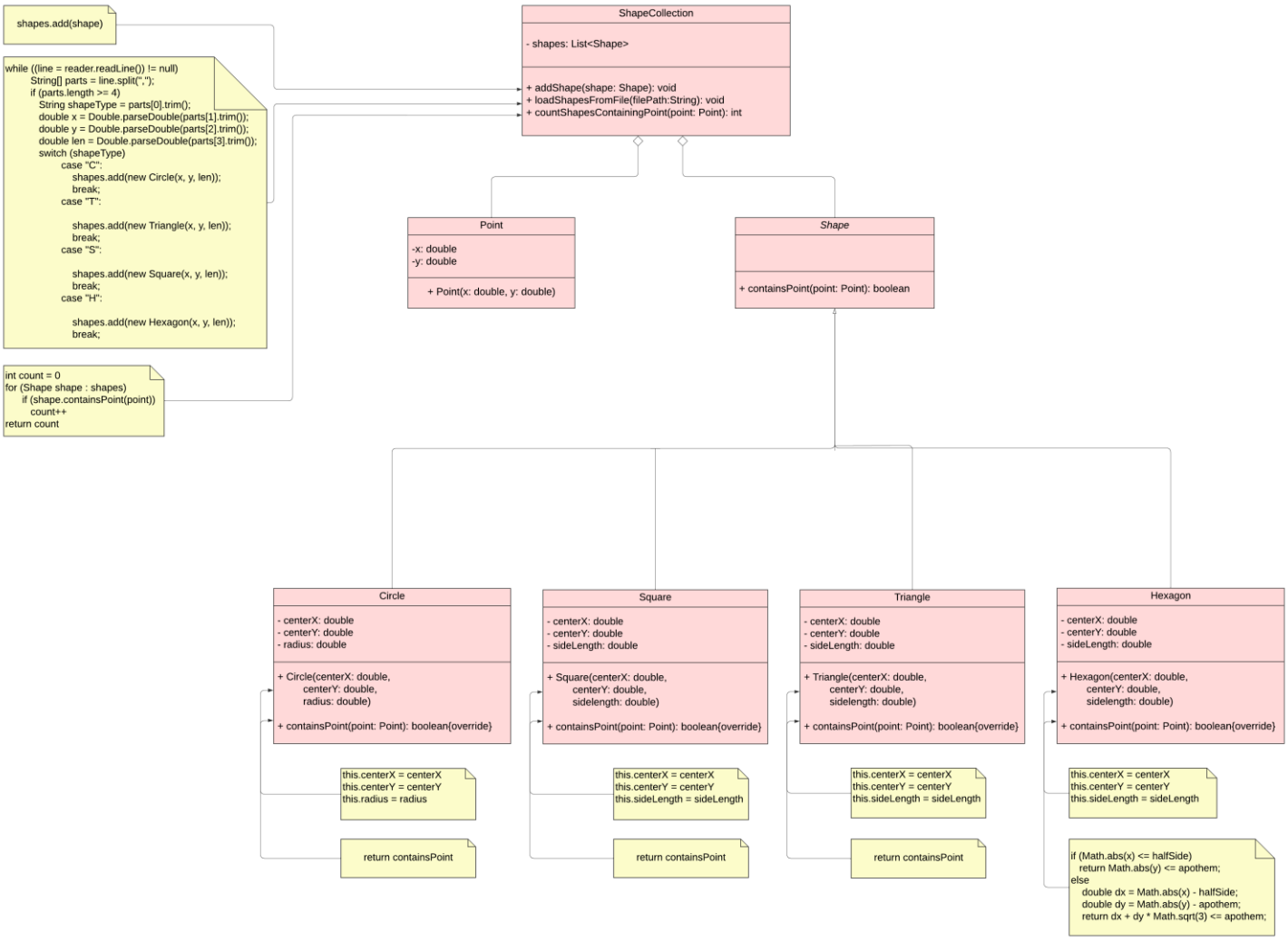
Task

Choose a point on the plane and fill a collection with several regular shapes (circle, regular triangle, square, regular hexagon). How many shapes contain the given point? 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 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

One parent class is made, Shape. This will be used for inheritance later, when calling its subclasses, i.e., Circle, Square, Triangle and Hexagon. The input file is used to read the type of the shape, which is followed by the center coordinate and the side length or radius. All the reading is done in the loadShapesFromFile method. Exception handling throughout the code is used.

UML Diagram



Description of the methods

shapeCollection

- addShape(Shape shape): adds shape to the list.
- loadShapesFromFile(String filePath): loads shape from the text file where 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.
- countShapesContainingPoint(Point point): Counts the number of shapes containing the point and then returns the count of shapes containing the point.

Point

- Point(double x, double y): Constructor. Sets the x and y coordinates of a point.

Shape (superclass)

- containsPoint(point: Point): Abstract method. It will be inherited in the Subclasses (Circle, Square, Triangle, Hexagon)

Circle (subclass)

- Circle(double centerX, double centerY, double radius): Constructor. Sets center coordinates and the radius.
- containsPoint(Point point): finds distance between point and the center of the circle. Returns true or false if distance is less than or more than the radius respectively.

Square (subclass)

- Square(double centerX, double centerY, double sideLength): Constructor. Sets center coordinates and the side length.
- containsPoint(Point point): finds half of side length. Then using that, finds min and max values of x and y. then compares those values to x and y coordinates of the point. If x and y coordinates of the given point falls within the range [minX, maxX] and [minY, maxY], it means that the point is inside or on the boundary of the square, so the method returns true. Otherwise, if any of these conditions is not met, it means the point is outside the square, and the method returns false.

Triangle (subclass)

- Triangle(double centerX, double centerY, double sideLength): Constructor. Sets center coordinates and the side length.
- containsPoint(Point point): calculates the area of the given equilateral triangle and then checks whether a given point is inside or on the boundary of the triangle by calculating the areas of sub-triangles formed by the point and the triangle's vertices and comparing them to the area of the entire triangle. If the areas match within a small tolerance, the point is considered to be inside the triangle and returns True. Otherwise, False.

Hexagon (subclass)

- Hexagon(double centerX, double centerY, double sideLength): Constructor. Sets center coordinates and the side length.
- containsPoint(Point point): determining if the point is within the central rectangle of the hexagon and then checking if it is within one of the two triangular portions. If either condition is met, the method returns true; otherwise, it returns false

Test Cases

White Box test cases:

Test case	Input	Expected output
Point inside triangle	new Point(0, 0);	Passed.
Point inside square	new Point(1, 1);	Passed.
Point inside circle	new Point(2, 2);	Passed.
Point inside hexagon	new Point(1, 1);	Passed.

Black Box test cases:

Test case	Input	Expected Output
Working test	shapes1.txt	Correct count of shapes
load shapes from a non-existent file	-	File not found
Invalid shape data	shapes3.txt	Error parsing a numeric value
file with no shapes	shapes4.txt	Empty file
shape collection is null	shapes5.txt	Correct count of shapes

- The default working file: Should return an answer assuming the inputs are correct.
- Throws FileNotFoundException: If the file does not exist.
- Throws NumberFormatException: If text file contains invalid shape data.
- Returns empty file error if file contains no shapes.
- Working file: Should return an answer assuming the inputs are correct.