Vugar Nasraddinli SWCE8Y Programming Technology  
Assignment 1 – Task 7

**Task Description**

Fill a collection with several regular shapes (circle, regular triangle, square, regular hexagon). 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.

Determine the smallest bounding box, which contains all the shapes, and its sides parallel with an x or y axis.

**UML Diagram**

A close-up of a diagram

Description automatically generated

**Description of the classes and methods**

* **Shape**   
  Description: The Shape class is as an abstract superclass for *Circle*, *Triangle*, *Hexagon* and *Square*classes.   
    
  Properties:  
  *x (double)*: The x-coordinate of the shape's center.  
  *y (double):* The y-coordinate of the shape's center.  
    
  Methods:  
  *getMinX(),* *getMinY(),* *getMaxX(),* *getMaxY():* Abstract methods that must be implemented by subclasses to calculate the minimum and maximum coordinates of the shape.
* **InvalidShapeDataException**Description: This class is a custom exception used to handle errors related to inappropriate user input, such as invalid coordinates or invalid shape types.
* **ShapeFactory**The *ShapeFactory* class is a utility class responsible for creating instances of various shape types based on user input. It includes methods for validation and creation of shapes.  
  Methods:   
  *createShape*(String type, double x, double y, double size): Creates and returns a shape based on the input type, center coordinates, and size. It throws a custom error in invalid data.
* **BoundingBoxCalculator**Description:   
  This class is a utility class responsible for calculating the smallest bounding box. It includes methods to find the minimum and maximum coordinates among the shapes.  
  Methods:  
  *calculateBoundingBox(List<Shape> shapes):* Calculates the smallest bounding box that contains all the shapes in the input list.  
  *findMinX, findMaxX, findMinY, findMaxY:* Those methods finds the needed coordinates among the shapes in the input list. Accepts the list of shapes as parameter.

**Test cases**

* **testCreateShapes**Test focused on functionality of *ShapeFactory*. Here *Circle* and *Square* classes (which are inherited from *Shape* class) are created as an example. We are testing the validity of *createShape* method by the assertions below:Verifying that the created circle is an instance of Circle.   
  Verifying that the created square is an instance of Square.  
  Checking the x coordinate of the circle is equal to 1.0  
  Checking that the y coordinate of the square is equal to 2.0
* **testCreateInvalidShape**This test case checks the behavior when attempting to create an invalid shape (in this case, "Rectangle") using the *ShapeFactory* helper class. It expects the creation to throw an *InvalidShapeDataException*.
* **testCreateNegativeSize**This test case checks the behavior when attempting to create a shape with a value less or equal than 0 size using the *ShapeFactory*. It expects the creation to throw an *InvalidShapeDataException*.
* **testCalculateBox**This test case ensures that the *calculateBoundingBox* method correctly calculates the normal values taken by user input. Testing assertions are given below:   
  Creates instances of various shapes.  
  Creates an *ArrayList* of these shapes.  
  Calculates the minimum and maximum x and y coordinates using *BoundingBoxCalculator* methods.  
  Checks that the coordinates are correct
* **testCalculateBoxWithMinimalValue**

This test case ensures that the *calculateBoundingBox* method correctly calculates the minimum and maximum x and y coordinates for the given shapes in collection. Testing assertions are similar to *testCalculateBox* test case, but here we are giving shape sizes as minimal double values.