Problem: Triangle challenge

Write a program that will determine the type of a triangle. It should take the lengths of the triangle's three sides as input, and return whether the triangle is equilateral, isosceles or scalene.

We are looking for solutions that showcase problem solving skills and structural considerations that can be applied to larger and potentially more complex problem domains. Pay special attention to tests, readability of code and error cases.

The way you reflect upon your decisions is important to us, why we ask you to include a brief discussion of your design decisions and implementation choices. The resulting code and discussion is vital for us and will be used as a way for us to validate your engineering skills.

Solution

The Core Classes

In order to reach a general, reusable and easy to maintain design, we can see this problem as part of a bigger picture by assuming a hierarchy of Shapes. This way everything is somehow a descendant of a common object called “Shape”. Then we can define Polygon and after that Triangle in this hierarchy. We can easily add other shapes in this structure if we need to. An abstract view of the design would be like this (In the below picture, the shaded classes are implemented):

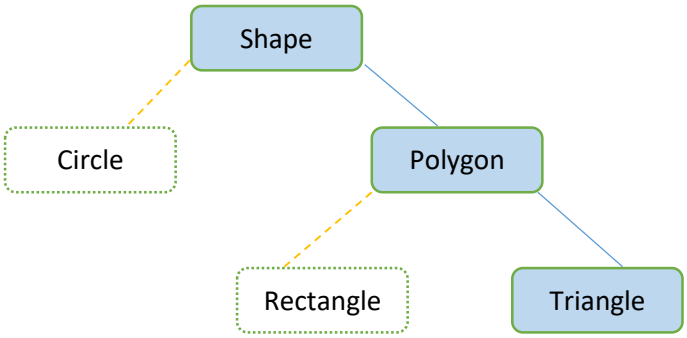

Figure 1-Overall structure for shapes hierarchy

After crafting and adding the details the class diagram for the core objects is as follows:

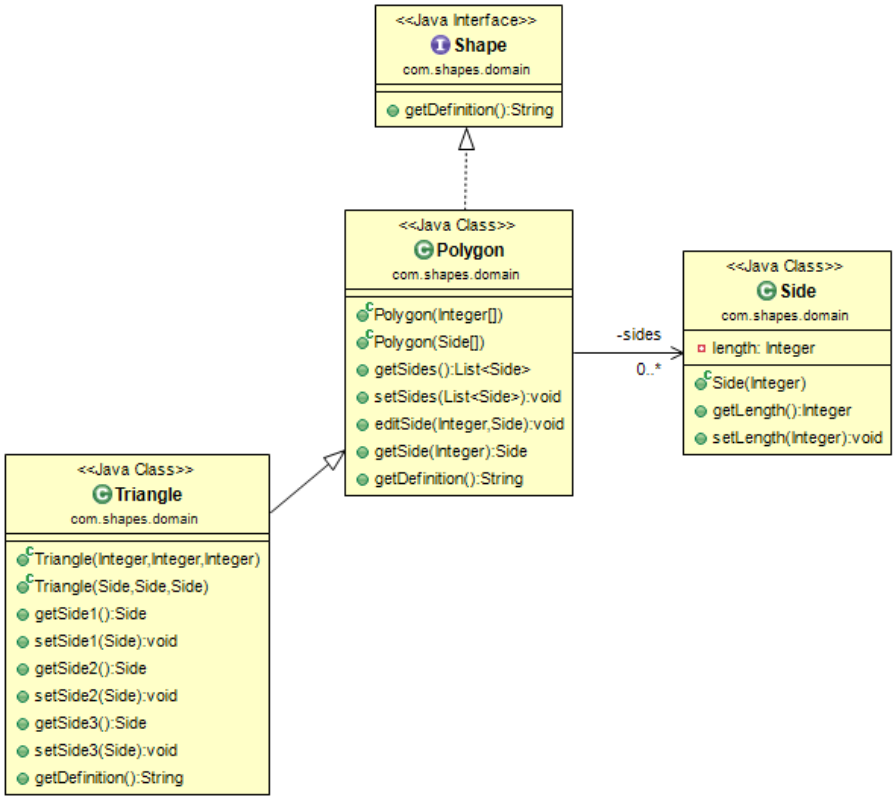

Figure 2-Class diagram of core objects

The following points are noteworthy:

- “Shape” is defined as an interface to share common methods with all types of shapes and make them implement their own logic for them. For example, a method for getting a string definition.
- “Side” only keeps the length of a side as its only attribute. It is defined as a separated class since we may want to add more details to it in the future, such as location (X,Y). If we assumed it as plain Integer, it would require us a lot of change and effort to make the aforementioned changes in the code.
- “Polygon” is defined as a set of sides and therefore has a one-to-many relation with the “Side” class. This gives us the flexibility of (1) constructing any type of polygon, (2) using this structure in further inheritance hierarchy (Triangle, Rectangle, Pentagon, etc.)
- For simplicity a “Triangle” is determined be providing its three sides. It internally uses the Polygon’s structure for Sides. However, this is hidden from the user of this object for clarity and ease of use.

The processing logic

The classes for processing logic look like this:

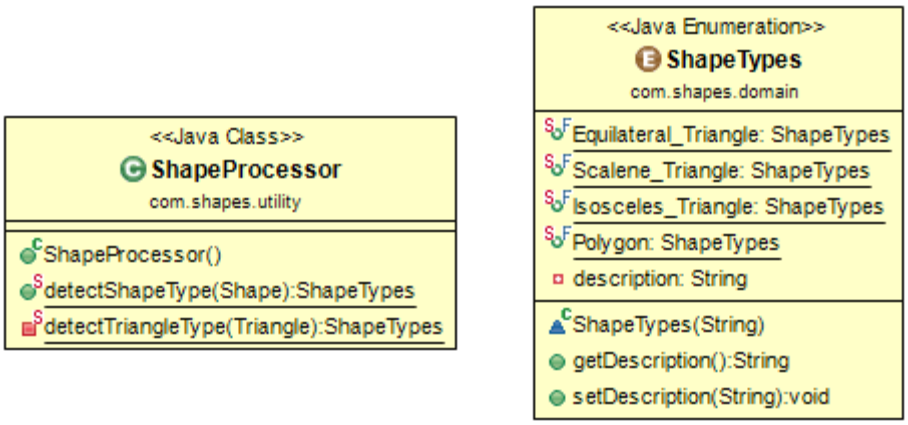

Figure 3-The processing classes

Notes:

- ShapeProcessor is a utility class capable of doing any type of process related to shapes in one place (high cohesion principle). For detecting shape type, it simply accepts any type of shape (by a reference to Shape interface), and then decides about the characteristics of that shape.
- For more tidiness, different shape types are defined in an enum type. It also includes a description for each enum.

The User Interface and Controller layer

The classes regarding these layers are as follows:

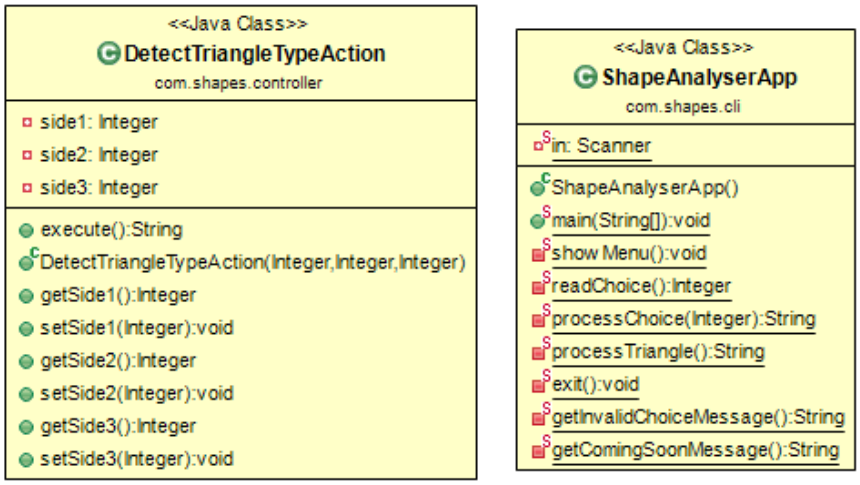

Figure 4-Controller and UI

Notes:

- The user interface is designed as a simple cli application and talks to the core layer through controllers. This way the UI layer can be easily switched with any type of GUI (Swing, web, Rest, etc.), with the least amount of change in the code base.
- DetectTriangleTypeAction is a sample action class and can be imitated by future controllers.

Here is a sample screenshot of running the cli app:

```
C:\Users\Jalal\Desktop>java -jar ShapeAnaylser.jar
_____________________________________________________
Welcome to ShapeAnalyser! Please select a shape :
1. Triangle
2. Polygon
0. exit
1
Please enter side 1
10
Please enter side 2
5
Please enter side 3
-1
The given side length is not valid! Please provide a number greater than zero.
_____________________________________________________
Welcome to ShapeAnalyser! Please select a shape :
1. Triangle
2. Polygon
0. exit
1
Please enter side 1
10
Please enter side 2
10
Please enter side 3
10

Equilateral Triangle
```

Figure 5-Cli application

General Notes

- The code is committed to github (https://github.com/sj-jafari/shape_analyser).
- The project follows maven project structure.
- Tests are written in three different classes according to their context. They follow the naming convention of Maven, so they can be executed automatically by Maven (useful in CI/CD). Here is a sample run:

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.shapes.test.ControllersTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.05 sec
Running com.shapes.test.CoreObjectsTest
Tests run: 16, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running com.shapes.test.ShapeAnalysisTest
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec

Results :

Tests run: 22, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 4.306 s
[INFO] Finished at: 2019-01-17T21:00:09+03:30
[INFO] ------------------------------------------------------------------------
```

Figure 6-Test results

Best Regards, Jalal Jafari