4.1 Functional Programming using Java

In assignment 1, you implemented a hierarchy of shape classes in Q1. In this assignment, they will be used. You may rewrite or re-use them in the following exercises.

Q 1. Add a "Rhombus" class, similar to **Q5** in assignment III. See attributes and methods in the assignment document. The Rhombus class, too, implements Shape and extends PrintableObject, as in the assignment 1.

Q 2. Using the above classes, write a program that does the following.

- reads a file containing at least 10 shapes (each shape is provided in a single line, in comma separated format); `Arrays.stream` may be used to convert an array into stream.
- sorts and displays the shapes by shape name and area;
- sorts and displays the shapes by perimeter only;
- displays a summary information of averages per shapes;
- displays the average perimeter, average area, and the total number of shapes at the end.

See the following implementation requirements:

Implementation Requirements

- You should strictly use the classes defined in Q1. You may not define additional classes[2].
- To implement the above functions, you must strictly use the stream API. Using loops such as `for`, `while`, etc. are not allowed.
- Use try-with-resources to open the file. The input file must be given by the user.
- You should strictly use functional programming and java API for sorting and displaying the shape objects. Use `Arrays.sort()` or `Collections.sort()`. No additional classes or interfaces are allowed. The Shape and its sub-classes may not implement the `comparable` interface.

[2]Only a driver class with a single static `main()` method is allowed.

- Use Java stream API to process the input file. You may use `String.spilt()` to transform the input lines as line array, and eventually into an array of shapes.
- Use Java stream API to display the summary information. No explicit loops may be used.
- Use at least one "method-reference" in your code.
- **IMPORTANT:** It is recommended not to use explicit `throws` clauses in method declarations. Throwing a `RuntimeException` might be useful. In any case, make sure all [runtime] exceptions are eventually caught in the `main()` method.