COMP 348: Principles of Programming Languages

Assignment 4 on Java, C#, and LINQ

Winter 2022, section E

March 29, 2022

Contents

1 General Information

Date posted: Wednesday March 30th, 2022.

Date due: Monday, April 18th, 2022, by 23:59[1].

Weight: 5% of the overall grade.

2 Introduction

This assignment targets 1) Functional Programming using Java, 2) Advanced Object-Oriented Programming using C#, and 3) Language Integrated Query (LINQ).

3 Ground Rules

You are allowed to work on a team of 3 students at most (including yourself). Each team should designate a leader who will submit the assignment electronically. See Submission Notes for the details.

ONLY one copy of the assignment is to be submitted by the team leader. Upon submission, you must book an appointment with the marker team and demo the assignment. All members of the team must be present during the demo to receive the credit. Failure to do so may result in zero credit.

This is an assessment exercise. You may not seek any assistance from others while expecting to receive credit. **You must work strictly within your team).** Failure to do so will result in penalties or no credit.

4 Your Assignment

Your assignment is given in two parts, as follows. 1) Functional Programming using Java, 2) Object Queries using C# and LINQ.

[1] see Submission Notes

4.1 Functional Programming using Java

In assignment I, 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 I.

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.

4.2 Object Queries using C# and LINQ

Q 3. Implement the following classes and interfaces in C#. Implement all classes in a single code file.

i. Shape (interface)

- `Name`: property, to return object's name. Provide a default implementation that returns the object's Runtime class name[3].
- `GetPerimeter()`: a method to return shape's perimeter.
- `GetArea()`: a method to return shape's area.

ii. Rectangle (class, implements Shape)

- Two sides as double: define using properties.
- Implement standard constructors: no-arg as well as specialized constructor that receives the sides.
- `ToString()`: override and return the shape name (by calling the method in the base class), followed by the two values for the sides (separated by comma).
- Implement `GetPerimeter` and `GetArea` methods inherited from the `Shape` interface.

iii. Circle (class, implements Shape)

- : `Radius` as double: implemented as property.

[3]Use may use `this.GetType().Name`

- Implement standard constructors: no-arg as well as specialized constructor that receives the radius.
- : `ToString()`: override and return the shape name (by calling the method in the base class), followed by the value of the radius (separated by comma).
- Implement `GetPerimeter` and `GetArea` methods inherited from the `Shape` interface.

Q 4. Implement the following additions to the classes you implemented in the previous section. Use a separate code file for the additions. Include all additions in a single file.

i. `Shape.Print()`: a void method, added to the Shape interface via "Extension methods", to print object's info to the console.
   Use Object's toString() to get the shape info.

ii. `Rectangle.Parse()`: A static method that receives an input string and returns an instantiated `Rectangle` whose sides are initialized with the values in the input string. The input string is in comma separated format,i.e.: "`Rectangle,2,3.5`". The method returns the object as `Rectangle`.

iii. `Circle.Parse()`: A static method that receives an input string and instantiates and returns a `Circle` object whose radius is initialized with the value in the input string. The input string is in comma separated format,i.e.:
   "`Circle,1`". The method returns the object as `Circle`. Use partial classes to implement this. This method must NOT be implemented in the same file that the Circal class was previously implemented.

Q 5. Implement a static[4] class called `TextFileProcessor` that has the following methods and event:

i. `LineRead`: a static event that is fired when a line is read in the `Read()` method. The event handler has a single `string` argument that is set to the line that is currently read by the read() method.

ii. `Read(file)`: a void method and reads the content of a file. The method reads the file, line by line until it reaches the end of file. Use `File.ReadAllText()` and `String.Split()` methods to read all lines form a file. Use `ForEach()` to raise the `LineRead` event for each line.

Q 6. Using LINQ, write a small 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). Use the `TextFileProcessor` class you created in the previous question.
- 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.

[4]Note that in C#, a class cannot be static and sealed at the same time.

5 What to Submit

The whole assignment is submitted by the due date under the corresponding assignment box. Your instructor will provide you with more details. It has to be completed by ALL members of the team in one submission file.

Submission Notes

Clearly include the names and student IDs of all members of the team in the submission. Indicate the team leader.

IMPORTANT: You are allowed to work on a team of 3 students at most (including yourself). Any teams of 4 or more students will result in 0 marks for all team members. If your work on a team, ONLY one copy of the assignment is to be submitted. You must make sure that you upload the assignment to the correct assignment box on Moodle. No email submissions are accepted. Assignments uploaded to the wrong system, wrong folder, or submitted via email will be discarded and no resubmission will be allowed. Make sure you can access Moodle prior to the submission deadline. The deadline will not be extended.

Naming convention for uploaded file: Create one zip file, containing all needed files for your assignment using the following naming convention. The zip file should be called a#_studids, where # is the number of the assignment, and studids is the list of student ids of all team members, separated by (_). For example, for the first assignment, student `12345678` would submit a zip file named `a1_12345678.zip`. If you work on a team of two and your IDs are `12345678` and `34567890`, you would submit a zip file named `a1_12345678_34567890.zip`.
Submit your assignment electronically on Moodle based on the instruction given by your instructor as indicated above: `https://moodle.concordia.ca`
Please see course outline for submission rules and format, as well as for the required demo of the assignment. A working copy of the code and a sample output should be submitted for the tasks that require them. A text file with answers to the different tasks should be provided. Put it all in a file layout as explained below, archive it with any archiving and compressing utility, such as WinZip, WinRAR, tar, gzip, bzip2, or others. You must keep

a record of your submission confirmation. This is your proof of submission, which you may need should a submission problem arises.

6 Grading Scheme

Q1	10 marks
Q2	20 marks
Q3	20 marks
Q4	10 marks
Q5	20 marks
Q6	20 marks

Total: 100 pts.

References

1. Assignment I: Q1, Section "4.1 Object-Oriented Programming using Java"
2. Assignment III: Q5, Section "4.1.3 Files and Text Processing"