CSSE 371: Lab 1-1

Background

Assume that you work for a company that uses Business Intelligence (BI) data from multiple sources. Your company initially had a contract with Google and Microsoft. In order to use their data appropriately, your company needed a standard data format.

Here is an example of the Google's BI data format (also found in the **Lab1-1/input_output/io.gogl** file):

```
google
geo1 - 100
geo2 - 450
geo3 - 90
geo4 – 750
```

Here is an example of the Microsoft's BI data format (also found in the **Lab1-1/input_output/io.ms** file):

```
microsoft
ms1,100
ms2,450
ms3,90
ms4,750
```

You company requires the data to be in the following standard format to make a good use of it (Please run the application to see the standard data format):

company
<field1> : 100
<field2> : 450
<field3> : 90
<field4> : 750

Your company is expanding and it is hoping to work with other companies beside Google and Microsoft. After investigating the sample data of several other companies, such as Amazon, Groupon, Ebay, and many more, the in-house BI team came up with the conclusion that the only difference in the data format among these companies lied in how fields were represented. The first line always had the company name. Your executives are now asking you to re-design and implement the Data Standardizer application to allow addition of new data source with low maintenance headaches. It should also be able to switch the data source at runtime as it is currently doing.

Design

Create **Lab1-1/docs/Answer.pdf** with answers to the following problems:

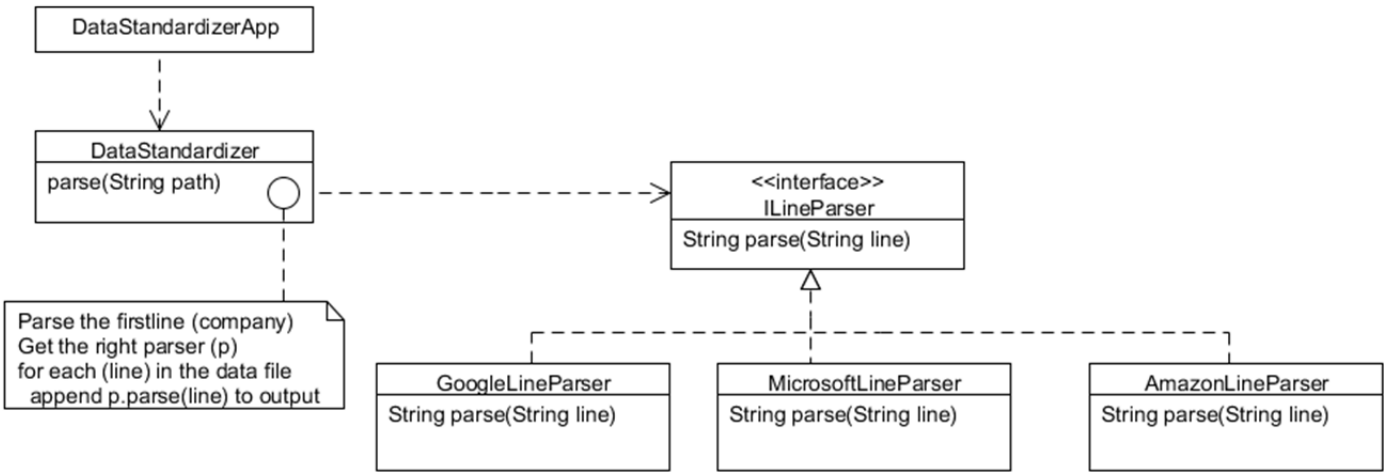
Q1. Identify and explain problems prevalent in the existing design. [**5 points**]

The project has not encapsulated the regions in code that change from the regions that remain unchanged. As a result, every time we add a support for a new company, we need to modify the *DataStandardizer* class.

Q2. Explain why sub-classing DataStandardizer and overriding the parse() method is not the best idea. [**5 points**]

Parse method has code that can be reused. The only block that changes is within the while loop of the *parse()* method. Overriding the *parse()* method creates code duplicates in subclasses.

Q3. Create a UML Class Diagram to present your new design idea and explain it in a few lines. [**10 points**]



The if-else statement that switches parsing logic in the *DataStrandardizer* class can be encapsulated using the *Strategy* pattern. After reading the first line of the data, the parse() method can choose which line parser to use. After that the appropriate parser will parse the line and standardize it. As an implementation details, to choose of the right kind of parser, we can pre-populate a *Map<String, ILineParser>* in the *DataStardardizerApp.main()* method and pass it to the *DataStrandardizer* class during initialization.

Implementation

Q4. Implement your new solution in the **Lab1-1/src/problem** package. Use the following data source from Amazon as a proof of concept in addition to the Google and Microsoft's data: [**20 points**]

```
amazon
aws1 ttl 100, aws2 ttl 450
aws3 ttl 90, aws4 ttl 750
aws5 ttl 900, aws6 ttl 200
```

[**Note**: You must create an **io.aws** file with this data in the **Lab1-1/input_output** folder.]

Testing

Q5: Implement necessary test cases in the **Lab1-1/test/problem** package. [**10 points**]

[**Note**: If you would like to experiment things, then you can do so in the **Lab1-1/src/temp** package. We will not grade your work in this package.]

Deliverable

Bundle you project in the **zip** format [**not rar**] and turn it in on Moodle.