Thank you for taking the time to code the "Adfenix recruitment challenge"!

Included in this task are the following files:

- This "readme" file
- An interface called SaleObjectConsumer
- Three example files with different formats, but similar data: SaleObjects.csv, SaleObjects.xml, SaleObjects.json

The goal of this fictive task is to write an application that reads the content from files with content written in different formats, but containing similar data, and reporting that data to our partner companies. As the data is similar, the content of the files may be reported using the same interface, called "*SaleObjectConsumer*" (which is included in this task).

To fulfill the task, you must:

- Accept the filenames that should be parsed as an argument to your application.
- Parse the data files, containing meta-data (called *SaleObjects*) describing houses and apartments.
- Report all the parsed SaleObjects to our partners systems by using the interface in the following order:

1. Call getPriorityOrderAttribute() to get the attribute to order the SaleObjects by before reporting them.
2. Call startSaleObjectTransaction() before reporting any SaleObjects.
3. Call reportSaleObject() for each sale object in prioritized order.
4. Call commitSaleObjectTransaction() when done reporting SaleObjects.

- Remember to read the documentation in the SaleObjectConsumer interface file.

Note! As of now we have not received any implementation of the SaleObjectConsumer-interface from our partners, but we are confident that your solution will work regardless of their chosen implementation.

Note! You may report any amount of SaleObjects to the interface after having called the method startSaleObjectTransaction, but after calling commitSaleObjectTransaction you must start over from step #2 if you wish to send additional SaleObjects.

There are three different file formats included: CSV, XML and Json. You do not have to parse all of them. Please choose the two formats you are most comfortable parsing (and feel free to use any third-party parsing library of your choosing if you want to).

Final words

We hope that you have a fun time coding this challenge! Think of your work as a good starting point for a library that will be used and continued on by other developers. Below are some hints of some of the aspects that we will look at in your solution (not ordered):

- Have the task been solved?

- How easy would it be for other developers to:
    - Understand and consume your API?
    - Contribute further to your solution?
- How is the code structured in regards to Object Oriented Design?
- How error prone is it?
- How easy would it be to extend it with new input formats?
- e.t.c.

We recommend implementing the challenge using a console application, but you may solve it using spring if you please.

Before sending your solution to us, please rename the root folder containing your solution to your full name.

Good Luck!
Best wishes,
The Adfenix developers

PS. Observe that both the interface and the file specifications have been designed with only this challenge and its quirks in mind. It does not represent Adfenix in terms of code-standards or code architecture. DS.

Below is an explanation of the data found in the csv file format (if you choose to implement that). It contains the following post types, with each data field separated by a comma sign:

[1..1] START-post.
* The tag "START"
* The date of when the file was created.

[0..*] A-posts. (A stands for Apartment)
* The tag "A". (char)
* Apartment size in square meters. (int)
* Apartment price. (long)
* The City of residence. (string)
* The street. (string)
* The floor number at which the apartment is located. (int)

[0..*] H-posts. (H stands for House)
Like apartment, but with the tag "H" and without floor information.

[1..1] END-post
* The tag "END".
* The sum of A and H posts in the file.