Programming Assignment 04

Background

We now have the beginnings of a functioning application. The next step is to prepare a process for reporting the investments held by each of our customers.

The provided code includes the backbone of the reporting logic. You must furnish certain, specific portions of the logic in order to complete the report processing code.

Provided to you

The following are provided to you:

- These instructions
- A Process Plan
- UML Class Diagrams (**NOTE:** These are unchanged from PA03)
- Serialized files and supporting ***executables*** (in a zipped folder):
  - `customerFile.xml`
  - `securityFile.xml`
  - `Customer.class`
  - `Security.class`
  - `MutualFund.class`
  - `Stock.class`
  - `CustomerException.class`
  - `SecurityException.class`
  - `CostBasis.class`
- An Excel workbook showing the data contained in the two serialized files
- The expected results from the completed `CustomerAccountReport.java` code
- `CustomerAccountReport.java` containing the base logic for the match/merge algorithm and which you will complete, compile, and debug as needed
- A grading rubric for the assignment

Deliverables

Submit your completed **`CustomerAccountReport.java`, AND source files for all your supporting code** in a zipped folder. **DO NOT submit** class code (compiled/built code), edit backups (.java~ files), any provided code, or **anything other than the required code**, **in the format specified**.

Requirements

Deserializing Objects

The provided XML documents contain serialized objects of `Customer` and `Security` types. Successfully deserializing these objects requires appropriate "helper" code for each class/hierarchy. Your "helper" code must follow these standards:

- Appropriate naming of source module
- XML schema must be correctly defined
- Correctly named, declared and instantiated Collection object
- Correctly named method for returning a reference to the Collection object

I recommend creating Collection objects local to the process code to hold your `Customer` and `Security` objects after successful deserialization. I recommend using the names `customerList` and `securityList` for these two Collection objects.

Sorting the Deserialzed Objects

Once the deserialization tasks are complete, sort each Collection into **ascending order**, based on the **`custNumber`** instance variable. You may choose how to complete this task. **If custom comparator(s) are used, that source code MUST be included in the zipped folder submitted for grading.**

CustomerAccountReport.java

Please refer to the provided Process Plan document.

Additional Requirements

The submitted code must follow the published Coding Standards.