CustomerAccountReport

Process Design Document

Process Inputs

The following XML documents are inputs to the `CustomerAccountReport` process:

- `custFile.xml`: Contains XML serialized Customer objects
- `securityFile.xml`: Contains XML serialized Security objects

Process Outputs

This process delivers (to the console only) a report of equity positions (stocks and/or mutual funds) owned by each individual customer. The report includes information from the `Customer` object and from each `Security` object associated with that specific Customer.

The process also provides the total assets under management by the company; that is, the total value of all security positions held in trust for customers.

Please refer to the Expected Report Output for details.

Import Statements

This process requires several import statements. Grouping of import statements provides clarity and the ability to check what has or has not been included at any given point. Do not include unnecessary import statements to the code.

Exception Handling

Appropriate try blocks will wrap all I/O operations. (Wrap other functions in try blocks as needed/appropriate.) Use appropriate and **specific** catch blocks. Any possible I/O errors/exceptions will pass to the process main method.

Handle possible exceptions related to `Customer` or `Security` objects appropriately; however, keep in mind that the serialized objects have already passed all validation testing.

Process Plan

Preparation for Processing

- Declare, instantiate, and open inputs
- Deserialize all objects from each XML document (**HINT**: After deserialization, you will probably want to transfer objects to a ***local*** Collection for ease of processing. Use `customerList` for the `Customer` Collection and `securityList` for the `Security` Collection.)
- Close inputs
- Sort Collection objects for processing
    - Ascending order
    - Based on `custNumber` instance value

Process Report

- Establish an object to use in traversing the `Security` Collection
- Retrieve the first `Security` object; store it in a locally declared `Security` object reference.
- Retrieve the `custNumber` instance value from the retrieved `Security` object; store it in a locally declared reference*.
- Using the locally stored `custNumber`* value, call the `findCustomer` method to retrieve the `Customer` to whom the current `Security` object belongs.
- A `try` block will wrap the remaining code, as it will throw an exception when there are no more `Security` objects to process.
- While there continue to be Security objects to process
    - While the `custNumber` of the current `Security` object is equal to the `custNumber` stored above*
        - Output a line describing the `Security` object
        - Add the purchase value of the `Security` (number of shares x purchase price/share) to the aggregated total assets under management
        - Retrieve the next `Security` object
        - End of inner while loop
    - Update the locally stored custNumber*, using the `custNumber` extracted from the recently retrieved `Security` object
    - Call the `findCustomer` method using the (just updated) locally stored `custNumber`*
    - End of outer while loop

Complete Report

Output the aggregated total of assets managed. Calculate the managed value of an asset by multiplying the number of share by the purchase price per share. Add the calculated value for each asset to an accumulator/aggregate defined in your code.

Find Customer (receives a `custNumber` value)

- Establish local storage for two `Customer` objects: one will serve as the template for the search; the second will sever to hold the "right" `Customer` object when found.
- The remaining logic will be wrapped in a `try` block
- Establish the search template object using the `custNumber` value passed into the method
- Declare a local variable to hold the search result
- Perform the search. I recommend using the `binarySearch` method
- Upon a successful search, use the return from the `binarySearch` method to retrieve the appropriate `Customer` object from the Collection.
- Code for the necessary output lines is provided