CSCE 156 – Assignment 6

Fall 2013

Phase V – Integrating a Data Structure

You will again extend the design of your application by designing, implementing and incorporating a sorted list Abstract Data Type (ADT). Specifically, you will design and implement a list class (or classes) to hold an arbitrary number of portfolio objects which will *maintain* an ordering of each of the portfolios according to some criteria. You will need to provide implementations (or variations on implementations) to support each of the following three orders:

- Last name/first name of the portfolio owner in alphabetic ordering
- Total value of the portfolio highest-to-lowest
- An ordering that groups all portfolios by their managers, first by the type of manager (Junior or Expert) then by last name/first name.

You are not allowed to use any of the standard JDK collections objects or algorithms, nor are you allowed to exploit sorting functionality provided by your MySQL database or to implement a sorting algorithm outside your list class.

In prior assignments, you may have maintained a list of portfolios using an array or some `List` implementation. You will need to update your code to take advantage of your new list implementation instead.

Requirements

As with the last assignment, your program will still connect to your MySQL database on cse. You can (and should) use any test data that you wish, but when you hand in the final homework we will use our test cases.

As in the previous assignments, you will print out a summary table as before. However, to demonstrate that all three of your orderings work, in this assignment, you will print out all three orderings in three different tables.

Design Process

The details of how you implement your list ADT are a design decision that you need to make. Some possibilities may include the following.

- Your list ADT may be array-based, linked-list based, or something else entirely.
- You should implement the standard list ADT methods to facilitate adding, removing, and retrieving/iterating over elements. Keep in mind that *order should be maintained, not imposed or updated/changed by a method call*.

- You can/should implement any convenience methods (batch adds, size, get-by-index, etc.) that you feel would simplify interactions with your list ADT.
- You can/should make your implementation *generic* by parameterizing it.
- To unify your design you may consider a design that allows a user to construct your list along with a `Comparator` which your list implementation then uses to maintain ordering.
- You may consider making your list implement the `Iterable` interface so that it can be used in an enhanced for-loop.

Artifacts

You will turn in all of your code as a self-executing JAR file using the CSE webhandin (`PortfolioReport.jar`) with your source code included. Any library files (such as the mysql connector JAR) must also be included in your JAR file. If you follow the instructions as before, this should be automatic. Turn in a hardcopy of the grading rubric with your name(s) and login(s) filled out.

Design Write-up

You will update and modify your Design Document draft to include details on your new Sorted List ADT. You must hand in an updated printed hardcopy will be handed in 1 week prior to the assignment due date.

In the new section, make sure to given enough details so that a technically competent individual would be able to reasonably reproduce an implementation of your design. Be sure to address *at least* the following.

- What is/are the public interface(s) to your list ADT class? How is it constructed?
- How is your list class(es) implemented internally? Is it implemented using an array? A linked list? Other?
- How does the list's capacity expand and/or contract in response to the insertion or removal of elements?
- How is order maintained in your list ADT?
- How are adds/removes taken care of in your list? How do you handle additions to the list?
- Is your implementation general—that is could it hold other types rather than just portfolio objects or is it limited?

Common Errors, Issues, & Questions

The following is a list of common errors and issues that past students have encountered and lost points on. These are issues that you will need to avoid and/or address.

1. You are not allowed to use the standard JDK collections library for your sorted list ADT (and your class should not simply be a "wrapper" to any other class). However, you may use the standard JDK collections in other respects or internally to other classes.
2. Some designs have pre-sorted objects and then given them to the "sorted list" ADT. This is an explicitly bad and incorrect design as it defeats the purpose of having a *sorted list* entirely. The purpose of such an ADT is that the particulars of how it represents and maintains and order is

encapsulated in the class and we deal only with the abstract add, remove, and retrieve methods.

3. Some designs have attempted to sort by requiring the user of the class to call an explicit sort-by method for whatever ordering you wanted to impose. This is not an Object-Oriented Design; in OOP we design objects with semantic meaning that should be invariant throughout an object's life cycle (recall the square/rectangle or ellipse/circle problem). An object's definition should not change based on the program state up to that point (this would be a Structural Paradigm, not OOP). Defining and implementing a list that sorts by the same ordering throughout its life cycle is more in-line with OOP principles.
4. Many write-ups fail to adequately describe how the list dynamically expands/contracts as elements are added/removed from the list. Make sure that you are addressing all issues as described in the rubric.