Program 4

Due: Thursday, June 25 (by 11:59 PM)

Worth: 50 pts. + 10 pts. Extra Credit

Purpose: This program explores sorting using comparers and the natural ordering of classes.

It is common to want to be able to sort objects of the classes we create into different orders using the sort methods of the **Array** class or the **List** class. In order to define the natural order for objects of a class, one must implement the **IComparable** interface. This will allow objects to be compared to one another to determine if one is less than, greater than, or equal to another. To create additional sort orders (beyond this default order), one must create instances of the **IComparer** interface and use them with the sort methods to establish a different sorting order. You may refer to the following article from MSDN for more information on these interfaces: How to use the IComparable and IComparer interfaces in Visual C# .

For another example, I have created a project that demonstrates these interfaces being used with the **Time2** class from Chapter 10 of the text. It is attached to this program description in the file *SortingDemo.zip* . I recommend that you follow this example closely when solving this assignment. Review Program 4 Discussion recording in Tegrity for a walk through of the demo and discussion of this assignment.

Using the **Parcel** hierarchy from your instructor's solution to Program 1A, rewrite the **Parcel** class to support the **IComparable** interface. Make the default sort order for parcels be ascending by cost. In addition, create another class that implements the **IComparer** interface and will allow parcels to be put in descending order by destination zip code. Write a simple test program that creates a **List** of at least 10 **Parcel** objects (use all the various item types). Display the list of items in the original order then sort the list using the default ordering (ascending by cost). Display the list again. Next, sort the list using your comparer to create descending by destination zip code and display the newly ordered list. Your test application can be a simple console application (as in the attached *SortingDemo*).

Extra Credit - For 10 points of extra credit, for create (and test) a new sort order that will first order by Parcel type (ascending) and then cost (descending) within each grouping. So, all **GroundPackage** items should be together followed by the **Letter** items, then **NextDayAirPackage** items, etc. You will create another class that implements the **IComparer** interface to create this new sort order. The comparer that you create should allow the class hierarchy to be extended without requiring modification. It should not matter what the concrete classes of **Parcel** hierarchy actually are. So, in other words, don't hard code **is** tests for each class type. **Hint**: You may want to explore what method **GetType**() returns. It may prove very helpful for creating your groupings. BTW, this is a multi-level sort ordering, much like the **Time2** demo sorted first by hour, then minute, and finally second.

Please review the documentation requirements specified in the syllabus that are expected for each file.

Rather than giving me floppy or Zip disks, you will upload **all your files** to using the *Assignments* tool. Using a tool like *PKZIP*, zip your entire Visual Studio project directory into a compressed file and upload the .zip file. Before uploading, make sure you can unzip the file, reopen the project, and execute the application. Remember, all assignments are due at the start of class and no late assignments will generally be accepted. For this assignment, I will grade all files electronically. No printouts are required.

Remember, this is an **individual** assignment. Please be mindful of the syllabus' statement on academic dishonesty. If you are unsure about what constitutes academic dishonesty, **ASK!**