Program 0

This assignment explores the use of composition (HAS-A) and inheritance as relationships between classes.

In this assignment you will use what you have learned through Chapter 12 to design and implement a simple class hierarchy for package-delivery services, such as UPS or FedEx. The hierarchy will consist of several abstract and concrete classes. At the top of the hierarchy is the abstract base class Parcel. It will contain the To and From addresses for the parcel. This will be done via composition with the Address class. Each address consists of a name, two address lines, city, state, and zip code. Each parcel will also support a CalcCost method that will return the cost of shipping the item. Derived from Parcel are two classes, Letter and Package. Letters will be delivered for a fixed price. The Package class and others will be developed in the next assignment. Follow the naming, return types, and parameter lists specified below carefully. You may include private/protected helper methods/properties but no additional public methods or properties may be added without agreement from your instructor.

Address

- Constructor that specifies the name (a string), address line 1 (a string), address line 2 (a string), city (a string),state (a string), and zip code (an integer, from 0 - 99999), in this order.
- Overloaded constructor with same parameters as above EXCEPT address line 2. So, name (a string), address line 1 (a string), city (a string), state (a string), and zip code (an integer, from 0 - 99999), in this order. Address2 should be set to an empty string in this scenario.
- Get and set property for each of the data fields. Use the following property names: Name, Address1, Address2, City, State, and Zip. Name, Address1, City, and State must not be null or empty even after the value has been trimmed to remove leading and trailing whitespace. String methods IsNullOrWhiteSpace and Trim will be helpful. Zip must be between 0 - 99999. If the value received for any property is invalid, throw an ArgumentOutOfRangeException with appropriate error message as shown in the text.
- ToString method that will return all the data fields as a formatted string consisting of three or four lines. The name and two address lines (one line, if Address2 is empty) are on separate lines with city, state and zip code on the final line. [Hint: How do you get zip codes to print leading zeros if the integer is less than 10000? Try D5 format.]

Parcel (abstract)

- Constructor that specifies the origin address and the destination address (as Address objects).
- Get and set property for each of the data fields. Use the following names: OriginAddress and DestinationAddress.
- Abstract CalcCost method (no parameters) that returns the given cost of the parcel as a decimal.
- ToString method that will return the all the values of the data fields as a formatted string. All costs and fees should be formatted to display using currency formatting.

Letter - derived from Parcel

- Constructor that specifies the origin address, the destination address and the fixed cost (non-negative decimal), in dollars. The cost may vary from store to store due to local competition, so it will be specified in the constructor.
- Override the CalcCost method to return the fixed cost specified in constructor.
- Override the ToString method as needed to return the all the values of the data fields as a formatted string. All costs and fees should be formatted to display using currency formatting.
- Note, there is no public property for the fixed cost in these specs. If you wish to have a property for this, use a different visibility modifier (private or protected, perhaps).

In addition to the classes described above, you will need to create a simple console application (.NET Framework) to test your letters and addresses. In your application file (where the Main method is located), construct at least 3 Letter objects (you may hard code your test data) using as least 4 Address objects. Add your Letter objects to a List of Parcel objects (List<Parcel>) using the List class introduced in Chapter 9 (and used previously in CIS 199). Print all the letters and associated cost to the console. Include other statements as necessary to test the properties included in the classes.

Be sure to add appropriate comments in your code for each file, including your Grading ID (not name nor student ID number), program number, due date, course section, and description of each file's class. Each variable used in your program needs a comment describing its purpose. These requirements are expected for every program and are listed in the syllabus. Preconditions and postconditions are required, as well. So, for each constructor, method, get accessor, and set accessor a pair of comments describing the precondition and postcondition must be provided. Please review the PowerPoint presentation (under Course Documents) for further details about preconditions and postconditions.