Generic Lists in Dart

Due Date: October 2nd @11:59pm

Description:

This goal of this project is to implement a portion of a library for collection classes in Dart. The library contains an abstract Collection class and one concrete subclass. Your collections will contain values of an arbitrary type T. No duplicate values will be allowed.

Implementation Details:

Abstract superclass Collection<T> is the root of all concrete collection subclasses, where T is the type of the elements stored in the collection. This class defers instance implementation to its concrete subclasses; however, it does define some concrete methods that are will not be redefined in the subclasses. You must define Collection<T> in such a way that only classes extending (subclassing) Dart predefined class Comparable can be used to instantiate T.

Concrete Collection<T> subclass LinkedList<T>, will implement a collection as a linked list.

Class Collection<T> declares the following (public) deferred methods:

- add()—This method takes as input an element of type T to be inserted in the receiver (i.e., an instance of a concrete Collection subclass). This method is defined in the subclasses. The modified receiver is returned.
- copy()—This no argument method returns a Collection subclass instance that is a deep copy of the receiver. The returned collection must be of the same type as the receiver.

- operator[]()—This is the indexing operator. This function takes as input an integer index and returns the element at the index position in the receiver. If the index is out of bounds, an error message is printed on the standard error stream and an exception is thrown. (Do not worry about catching the exception; you can let program execution terminate as a result of this exception.)
- printString()—This function prints all the elements in the receiver collection in their order on the console.

In addition the Collection<T> class defines the following concrete functions:

- A no-arg constructor.
- mapC()—This public function takes as input a function parameter fn. The parameter function fn takes as input a type-T object and returns also a T object. Function map() applies function fn to all elements contained in the receiver. The values returned by each execution of function fn are stored in a new linked list in the order in which the values are returned by each fn execution. The linked list is returned.
- containsC()—This public function takes as input an arbitrary object (not necessarily of type T) and returns a boolean indicating whether the receiver contains the argument object or not.
- equals()—This the logical equivalence operator. This function takes as input an arbitrary object. The function returns a boolean indicating whether the receiver and the argument are logically equivalent. The two objects will be logically equivalent if (1) they are exactly the same data type, and (2) they contain the same values in the same order.
- Finally, class Collection<T> declares a private, integer data member size, which returns the number of elements contained in the receiver. Collection<T> defines a

public getter for size. In addition Collection<T> defines a concrete method incrementSize() that adds 1 variable size. Only functions in Collection<T> are allowed to modify this variable.

Implementation requirements:

Define class Collection<T> to be a subclass of Dart class Object. You can use data type Object for objects of arbitrary classes. Except for no-arg functions, all your methods must include at least keyword (aka named) parameter. Finally, method add() must be coded in such a way that their invocations can be cascaded. You are not allowed to use Dart data-structure libraries, e.g., for your linked list implementations. You may add additional classes, data members and functions as needed.

Extra Credit: 10 points!

For the extra 10 points, you will want to make your LinkedList iterable, like any data structure should be. This will be a direct implementation of the iterator design pattern; very similar to what you had to do on project #1 in CS342. You will need to extend **Iterable** and provide a class that extends **Iterator**. If you implement this properly, you should be able to use the forEach loop to iterate through your LL.

```
myLinkedList.forEach((val)=> print(val));
```

The above line of code will iterate through each item in your LL and apply the supplied function for each item.

You should not define your own forEach function in your class.

Electronic Submission:

You only need to zip your .dart files together and name it with your netid + HW1: for example, I would have a submission called mhalle5HW1.zip, and submit it to the link on Blackboard course website.

Assignment Details:
Late work **is NOT accepted**.

Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them, this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml.

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing

your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml.