Homework Assignment 2

Your assignment for Homework 2 is to finish off the building of a doubly linked list of baskets.

The driver program is given in the code on the website.

You will need to implement the following code so that the `interface`s for `IMyListStructure`, `IDLL`, and `IDLLNode` are all satisfied by your code. These three interfaces are in the website zip file.

STEP ONE

You are required to implement a `MyListStructure` class that for the most part incorporates earlier code in the sample and the previous assignments. Your `MyListStructure` code will need to have methods

- `public boolean add(Basket basket)` that adds into the linked list structure the instance of `Basket` type passed as a parameter to the method.

- `public boolean contains(Basket basket)` that does a lookup on a basket.

  Think for a minute about the nature of the "information" you are using in doing a search. Java by default treats the `==` to be a genuine equality of the two objects, meaning that the two objects are stored in the same memory location and thus the object names are really just synonyms for the same stuff in the computer. That's not usually what you want. Similarly, you don't really want to have to compare the entire basket, although that's what we are doing in this exercise.

  This is one of those issues that needs to be thought out. The list structure code might know about basket and keys, and the basket class could know about keys, but the linked list code in the middle shouldn't have to worry about those details.

- `public boolean remove(Basket basket)` to remove a basket. This should probably call the `contains` method to identify the basket, and then remove it.

  In any such computing, you have to specify what to do in the case of duplicate baskets. The simplest thing here is to specify that you will remove the first instance of a matched basket, if there is a match.

- `toString`

STEP TWO

You are required to implement a `DLL` class that creates and manages a doubly-linked list of `Basket` instances.

Your `DLL` code will need to have a real constructor as well as methods

- `getHead`
- `setHead`
- `getSize`
- `getTail`
- `setTail`
- `add`
- `addAtHead`
- `contains`
- `remove`
- `toString`
- `unlink`

STEP THREE

You are required to implement a `DLLNode` class that creates and manages nodes for a doubly-linked list of `Basket` instances.

Your `DLLNode` code will need to have a real constructor as well as methods

- `getNodeData`
- `getKey`
- `getNext`
- `setNext`

- `getPrev`
- `setPrev`
- `toString`

You can use as sample input and output the files from the website zip file.