Assignment 3: Unordered Lists

**Due:** The seventh week of classes, with due date and time as per the course information sheet.

**Hints, announcements, and starter code:** On the course webpage.

**Purpose**

To make you work with linked lists and Java packages.

**Overview**

For this assignment, you will be working with *unordered lists*. Objects in an unordered list are stored in an arbitrary, unknown order, which may be different from the one in which the Objects were added to the list. In that sense, an unordered list is like a set (in that the order of elements is unimportant), except that an Object may appear more than once in an unordered list.

To implement unordered lists in Java, we have already provided a class `BasicUnorderedList` that stores Objects in a doubly-linked list and provides a few simple methods. This class, and the helper classes that it depends on, are part of a Java package that we defined, called `unorderedlist`. (Have a look at the starter code to see exactly how the package is defined, and how the basic classes are defined.)

As we already remarked in the lecture notes, the advantage of using a linked list instead of an array to store the Objects is that the size of the list is not constrained in any way (except by the size of the computer’s memory, of course). Unfortunately, this has the disadvantage that searching for an Object in the list must be done using a linear search, which can be quite time-consuming. If the list is used in a setting where many searches will be performed, it becomes important to make the `contains` method as fast as possible.

Since the order of elements in an unordered list does not matter, consider the following strategy for making the `contains` method operate faster: once Object “obj” is found in the list, the list is reorganized so that the node that contains obj is moved to the front of the list. This way, Objects that are frequently looked for in the list should stay close to the front of the list, which will speed up the search when these Objects are searched for again.

For example, if we start with the first unordered doubly-linked list below and search it for Object "hello", we should end up with the second list below (and return true since "hello" was found). Searching for an Object that does not appear in the list should leave the list unchanged.

Before searching for "hello":

```
head: [ ] "good" [ ] 3125 [ ] "hello" [ ] false [ ]
```

After searching for "hello":

```
head: [ ] "hello" [ ] "good" [ ] 3125 [ ] false [ ]
```
Your Task: extend BasicUnorderedList

Write a new class FastUnorderedList that extends class BasicUnorderedList in two ways. First, method contains should be rewritten in class FastUnorderedList so that it implements the strategy discussed above. Second, class FastUnorderedList should add a new method with signature

    public boolean remove(Object obj)

that removes the specified Object from the list (if it is there). Method remove should return a boolean value to indicate whether the Object was removed or not (i.e., it should return true if obj was found and removed, and false if obj was not found in the list). Note that if obj appears more than once in the list, only one occurrence should be removed.

You should also modify the Driver class to perform testing of your new class, although we will not mark your testing directly.

What to Submit

Submit all of your .java files electronically by the due date. Your file Driver.java must be submitted directly to the folder A3, and the rest of your files (part of the unorderedlist package) must be submitted to the subdirectory A3/unorderedlist.