ECS510 Algorithms and Data Structures in an Object-Oriented Framework

Exercise Sheet 4: Operations on Lisp Lists

This is a set of exercises to support the "Lisp lists and recursion" section

1) Download the files `LispList.class`, `LispList$Cell.class` and `UseLispLists2.java` from the code folder for this section. Check you can compile and run the code in `UseLispLists2.java`. The other two files are already compiled, you need them to use the type `LispList<E>`, but you do not need the Java code which produced them. The file `UseLispLists2.java` contains a method called `parseIntLispList` which takes a string representing a Lisp list of integers and returns the equivalent object of type `LispList<Integer>`. You can use this to read Lisp lists of integers written in the format used in the questions below (initial `[`, final `]`, commas between integers). In all the questions below, the word "list" is used to mean a `LispList<Integer>` object. Your answers should not involve the use of any other data structure in any way.

2) Write static methods which perform the following operations:

   `length` takes a list and returns the number of integers in it. For example, with [7,3,8,12,9,14] it would return 6.

   `count` takes a list and an integer and returns the number of times the integer occurs in the list. For example, with [2,3,4,2,5,12,2,5] and 2 it would return 3.

   `ordered` takes a list returns `true` if it is in ascending numerical order, `false` otherwise. For example, with [3,7,8,9,12,14] it would return `true`, with [3,7,8,12,9,14] it would return `false`.

   Give <u>both</u> iterative and recursive methods for these operations.

3) Write a static method called `filter` which takes a list and an integer and deletes all integers which are less than the argument integer from the list. For example, if the argument list is [17,11,20,34,5,10,8,19,55,11,13] and the argument integer is 12 it would return [17,20,34,19,55,13].

   You should try to use recursion for this and the following questions, but show also an iterative solution to <u>at least one</u> of them.

4) Write a static method called `multiply` which takes a list and an integer and returns the list obtained by multiplying all the elements of the list by the integer. So if the list is [2,3,4,12,5,12,2,5] and the integer is 5, it will return [10,15,20,60,25,60,10,25].

5) Write a static method called `after` which takes a list and an integer and returns the portion of the list after the first occurrence of that integer. So if the list is [2,3,4,12,5,12,2,5] and the integer is 12, it will return [5,12,2,5].

6) Write a static method called `positions` which takes a list and an integer, and returns a list consisting of all the positions of that integer in the list. For example, if the integer is 2 and the list is [2,3,4,2,5,12,2,5], it will return [0,3,6].

7) Write a static method called `removePos` which takes a list and an integer `n`, and deletes the integer at position `n` in the list. For example, if the list is [7,3,8,12,9,14] and the integer is 2 it will return [7,3,12,9,14].

8) Write a static method called `sublist` which takes two lists and returns `true` if the first list is a sublist of the second, `false` otherwise. For example [4,8,2] is a sublist of [5,6,4,8,2,3,1]. For a list to be a sublist, all the elements must occur in the same order with no elements between.

9) Write a static method called `subset` which takes two lists and returns `true` if the first list is a subset of the second, `false` otherwise. For example [6,3,8,2] is a subset of [5,6,4,8,2,3,1]. For a list to be a subset, all the elements must occur in the other list, but the order they occur in does not matter.

Matthew Huntbach