SE 2AA4 Winter 2007

Software Design Exercise 5

Instructor: William M. Farmer

Assigned:	16 March 2007
Files due:	30 March 2007
Lab report due:	5 April 2007
Revised:	28 March 2007

The purpose of this software design exercise is to write a Java program that creates an abstract data type (ADT) of *lists*. The program will consist of a package named `exercise5` that contains the following four files:

1. `List.java`
2. `ListPlus.java`
3. `Element.java`
4. `BadIndexException.java`

The first two files will be modules represented as a Java classes. They will be written by the student. The third and fourth files are posted on the course web site with this exercise.

Background

A *list* is a data structure that stores a finite sequence of values. The *empty list* is the empty sequence, i.e., the sequence that has no members. nil is the 0-ary function that returns the empty list. cons is the function that, given a value x and a list $[x_0, \ldots, x_n]$, returns the list $[x, x_0, \ldots, x_n]$. member is the function that, given an integer i and a list k, returns the i-th member of the k. If $k = [x_0, \ldots, x_n]$, then $\mathsf{member}(i, k)$ is x_i if $0 \leq i \leq n$ and is undefined if k is the empty list, $i < 0$, or $n < i$. take is the function that, given an integer i and a list k, returns the list that is obtained by taking the first i members of k and dropping the rest. If $k = [x_0, \ldots, x_n]$, then $\mathsf{take}(i, k) = [x_0, \ldots, x_{i-1}]$ if $0 \leq i \leq n$, $\mathsf{take}(i, k) = k$ if $n + 1 \leq i$, and is undefined if $i < 0$. drop is the function that, given an integer i and a list k, returns the list that is obtained by dropping the first i members of k and taking the rest. If $k = [x_0, \ldots, x_n]$, then $\mathsf{drop}(i, k) = [x_i, \ldots, x_n]$ if $0 \leq i \leq n$, $\mathsf{drop}(i, k)$ is the empty list if $n + 1 \leq i$, and is undefined if $i < 0$.

Step 1

Write a module that is a class named `List`. It should:

1. Define an object of type `List` that represents a list of objects of type `Element` indexed by values of type `int`.

2. Define four class methods for constructing objects of type `List`.
3. Implement the interface `Element` so that lists of lists can be constructed.

The `List` class should contain no public fields and only the following public methods:

- A selector

  ```
  public Element getMember(int i)
    throws BadIndexException;
  ```

  that implements the member function defined above. (Notice that this is a method of a `List` object.)

- A constructor

  ```
  public static List nil();
  ```

  that implements the nil function defined above.

- A constructor

  ```
  public static List cons(Element x, List k);
  ```

  that implements the cons function defined above.

- A constructor

  ```
  public static List take(int i, List k)
    throws BadIndexException;
  ```

  that implements the take function defined above.

- A constructor

  ```
  public static List drop(int i, List k)
    throws BadIndexException;
  ```

  that implements the drop function defined above.

- The methods

  ```
  public boolean same(Element e);
  ```

  and

  ```
  public String toString();
  ```

  from the `Element` interface.

The implementation of your module is required to:

1. Store all constructed objects of type `List` in a linked list. Create your own linked list data structure. In particular, do not use the Java class `java.util.LinkedList`.
2. The objects in the linked list do no need to be sorted.
3. For any two objects k_1 and k_2 of type `List` stored in the linked list, k_1.`same`(k_2) must be false. This means that, if a method is required to return a `List` object that is not the "same" as an object that has already been created, the method should create the `List` object and insert it into linked list at the correct place. Otherwise, the `List` object should be found in the linked list.
4. Nothing in your implementation should limit the number of `List` objects that can be stored in the linked list.

Step 2

Write a module that is a subclass and definitional extension of `List` named `ListPlus`. It should contain no public fields and only the following public methods:

- A predicate

  ```
  public static boolean isEmpty(List k);
  ```

  that returns true iff `k` is the empty list.

- A selector

  ```
  public static Element getHead(List k)
    throws BadIndexException;
  ```

  that returns `getMember(0,k)`.

- A selector

  ```
  public static List getTail(List k);
  ```

  that returns `drop(1,k)`.

- A selector

  ```
  public static int getLength(List k);
  ```

  that returns the length of `k`. The length of the empty list is 0.

- A selector

  ```
  public static int getHeight(List k);
  ```

  that returns the height of `k` viewed as a binary tree where, if k is not empty, `getHead`(k) and `getTail`(k) are the left and right subtrees of k, respectively. The height of the empty list is 0.

- A constructor

  ```
  public static List reverse(List k);
  ```

  that returns the object of type `List` whose members are the reverse of the members of `k`.

Step 3

Submit your `List.java` and `ListPlus.java` files using subversion. Your `List.java` will be tested with a `ListPlus.java` file written by the TAs, and your `ListPlus.java` will be tested with a `List.java` file written by the TAs. You must finish this step no later than 23:59 on Friday, March 30, 2007.

Step 4

Write a report that includes the following:

1. Your name and MAC ID.
2. Your `List.java` and `ListPlus.java` files.
3. A discussion of what you learned doing the exercise. List any problems you had with (1) your program and (2) the specification of the two modules. Discuss how these problems could have been avoided.
4. A copy of the part of your log book relevant to this software design exercise.

The lab report is due no later than the beginning of the lecture on Thursday, April 5.

Notes:

1. Please put your name and MAC ID at the top of each of your source files.
2. Your program must work on birkhoff when compiled by `javac`.
3. All fields and methods in your classes should be designated as either `private` or `public` except for the constructor for the `List` class which should have no designation.