Singly Linked Lists

Cosc 241 Assignment 2

Due: 4pm Thursday May 16th 2013

Overview

The Linked List is a fundamental data structure which is widely used throughout Computer Science. A Singly Linked List consists of zero or more Nodes, where each Node has an item of data and a reference to the next Node. In this assignment you will be implementing a Singly Linked List similar to what you have seen in lectures and labs. You will then add the following operations:

- `Split` — splits a list into two or more lists.
- `Join` — joins two or more lists into a single list.
- `Reverse` — reverses the order of a list.

You must implement all three operations in two different ways. Firstly, by producing a "copy" version, where the original list(s) remain unaltered. And secondly, a "modify" version where no new Nodes are created; instead references to existing Nodes are altered.

It is likely that you will find the "copy" versions easier to implement, so we suggest that you do those first.

Required methods

Your `SLList` class must contain at least the following methods:

```
public void addFirst(E item)
public void addLast(E item)
public E getFirst()
public E getLast()
public boolean isEmpty()
public static <E> SLList<E> joinAll(List<SLList<E>> lists)
public static <E> SLList<E> joinAllCopy(List<SLList<E>> lists)
public E removeFirst()
public E removeLast()
```

```
public SLList<E> reverse()
public SLList<E> reverseCopy()
public String reverseString()
public int size()
public List<SLList<E>> split(List<Integer> indexes)
public List<SLList<E>> splitCopy(List<Integer> indexes)
```

The "copy" methods (*joinAllCopy, reverseCopy, splitCopy*) must not modify the original list(s). The equivalent methods (*joinAll, reverse, split*) must work by manipulating references in the original list(s) and not create any new Nodes.

The *reverseString* method must use **recursion** to traverse the list from *last* to *first*.

Provided Files

We have provided you with two files to help you implement your solution. The first is a skeletal version of `SLList.java` which contains incomplete methods, as well as a few small helper methods. The second is an application class, `ListApp.java`, that you will use to manipulate your *SLList*. You can find these files in the directory `/home/cshome/coursework/241/Asgn2`.

Submission

Use the package name **asgn2** for this assignment. You need to submit your work electronically by 4pm on the due date. Your program should compile using the **ant** build file which has been provided:

```
/home/cshome/coursework/241/files/build.xml
```

See lab 9 for more details about using *ant* and the changes you need to make to the build file. To submit your files change into the top level directory of your assignment and run the command:

```
ant submit
```

Check that the list of submitted files is correct. Marks will be deducted for late, or incorrectly named assignments and files (this includes incorrect case for your source files).

Any assignments submitted after the due date and time will lose marks at a rate of 10% per day late.

This assignment should be done individually. DO NOT work with other students, or ask the demonstrators for help. Treat it as a take-home exam. However, feel free to come and discuss any questions you may have with Iain Hewson.

All programs will be checked for similarity.

Marking

This assignment is worth 9% of your final mark for Cosc 241. It is possible to get full marks. In order to do this you must write correct, well-commented code which meets the specifications.

Marks are awarded for your program based on both implementation and style (although it should be noted that it is very bad to style to have an implementation that doesn't work).

Allocation of marks	
Implementation	5
Style/Readability	4
Total	9

In order to maximise your marks please take note of the following points:

- Your code should compile without errors or warnings using ant.

  If your code does not compile, it is a considered to be a very, very bad thing!
- Your program should use good Java layout as demonstrated in the code provided for the third lab.
- Make sure each file is clearly commented.
- No line should be more than 80 characters long, to avoid wrapping when printing.
- Most of your comments should be in your method headers. A method header should include:
  - A description of what the method does.
  - A description of all the parameters passed to it.
  - A description of the return value if there is one.
  - Any special notes.

Part of this assignment involves you clarifying exactly what your program is required to do. Don't make assumptions, only to find out that they were incorrect when your assignment gets marked.

If you have any questions about this assignment, or the way it will be assessed, please see Iain or send an email to ihewson@cs.otago.ac.nz.