

CS_M71 Assignment 2 2008/2009

Set 09/04/2009

Deadline 08/05/2009. 11:55 p.m.

1 Outline

The purpose of this assignment is to give you practice in implementing some well known Design Patterns using generic code. By the end you should have a clear understanding of the advantages offered by Design Patterns in general, and how using Generic Programming you can promote many type errors from run-time to compile-time errors.

2 The task

You are going to implement a generic Ordered Linked-List. The list structure will have the following operations defined upon it:

1. **add(Node n)** - which adds the node *n* to the queue in its appropriate place.
2. **empty()** - which states that the list is empty or not.
3. **size()** - which returns the number of nodes in the list.
4. **toString()** - which walks through the list printing the values held in the nodes. (This method **must** override the `toString()` method defined in `Object`).

Note: the list is ordered, therefore the type parameter will need to be bounded.

Note: you may NOT use an `Array` or an `ArrayList` to implement your List.

The list will also have an **accept** method, which takes a visitor. This method invokes the visitor's **visit** method on the head of the list. The visitor will then operate upon all nodes in the list and incorporate an arbitrary extra operation.

The list will also have a resident **Strategy**. This means that a constructor must exist allowing us to pass the strategy as the list is constructed. This strategy will operate upon all nodes in the list making permanent changes to the nodes.

Now define **two** different classes which can be held in your list. They will of course need to be **comparable**. (Remember the type bound in the list definition). **Your two classes must be defined by you. Marks will be deducted for using pre-defined types such as Integers or Strings.**

Now define two different **strategies** which will be passed into the list when it is constructed. These strategies will have a single method which, when invoked, will permanently change one or more of the values held at the nodes. It doesn't matter how they work, what they do, or in which order they operate.

Define two different **Visitors** which can be passed into the list structure's **accept** method. These will dynamically perform some kind of action on the list. What they do is up to you but you must document your classes well so that it is easy for the marker to see what is supposed to happen.

Finally write test code which thoroughly tests your code. This test code should show that the list can be constructed using both of your types. It should show the visitor patterns working and show the list structure both before and after the strategies have operated. For full marks you should demonstrate that an inappropriate strategy and visitor will be refused compilation because the code has been written generically, i.e. that writing generic code means that errors which would in non-generic code have been reported at run-time will now be reported at compile-time. This means that you will have to think about your user defined classes very carefully.

3 Submission

All files should be contained within a single directory. This directory should be archived and a single file submitted to Blackboard in the usual way. Your submission should include (electronic) printouts of the compilation commands and the results of running the test code. If your compiler reports unchecked

warnings which you suppress or ignore you will be deducted marks unless you can give a very good reason for allowing them.

4 Assessment

- For a Linked List with correctly bounded type parameters - 20%
- For each user defined type - 10%
- For each strategy - 10%
- For each visitor - 10%
- For the test code - 20%

Marks will be awarded *up to* the given total, according to how accurate and well thought out the code is. Note that marks are assigned to reward **thought** as well as **implementation**. This is deliberately done because it will help you to think about Design Patterns and Generic Programming in a general way.

As usual marks will be deducted at a rate of 10% per day after the deadline. Work submitted more than one week late will not normally be marked.