CSU213: Lab 9

March 16, 2004

Goals. Familiarize with

- Usage of abstract classes to minimize efforts implementing large interfaces
- Linked list and stack
- Inner classes
- **List** and **ListIterator** interfaces.

1 Work Setup

1. Download **Lab9.java**, **jpt.jar**, and **jpfalt.jar** from the lab's web page.
2. Create a new project. Import the above files.
3. Add JAR files. (Project→Properties→Java Build Path→Libraries→Add JARs...)
4. (Optional, for only those who still have troubles exporting to Zip file):
   (a) Export the project to a Zip file.
   (b) Delete the current project. Create a new project using the exported Zip file.

2 Your own Linked List

Goal. Build your own linked list, which is simpler and therefore slightly more efficient than that of standard Java.

2.1 Understanding existing Java's LinkedList

1. Look at the specification of standard Java's **LinkedList** class in the Sun Java API Specification (http://java.sun.com/j2se/1.4.2/docs/api/). Make sure you understand the following things:
   - Its relation with **Collection** and **List** interfaces.

- Which methods that **Collection** and **List** interfaces have.
- How many ancestor classes does **LinkedList** have ? For each ancestor class, investigate:
  - How the relations with **Collection** and **List** interfaces change from one class to another.
  - Which method of the interfaces it implements, and which it doesn't (still declares as **abstract** methods).

2.2 Adding features to your own linked list

A partly-completed class is provided as **MyLinkedList**. You need to add more features to it to complete.

1. The inner class **MyLLIterator** represents an iterator over **MyLinkedList**. Re-write methods **MyLLIterator.hasNext()** and **MyLLIterator.next()** so that they have the desired behaviors. Notes:
   - You are NOT allowed to use the standard Java's **LinkedList** class to implement these methods.
   - Don't worry about testing right now. You will know how to test in the next step.
2. The testing is provided in **Lab9.testMyLinkedList()**. Uncomment it. See why this method does not compile.
3. In order to make method **testMyLinkedList()** compile, you need to make the class **MyLinkedList** implement the **Collection** interface.
   (a) How many methods will you have to implement if you choose to implement **Collection** interface directly ?
   (b) To reduce to burden, choose an appropriate ancestor class of the standard **LinkedList** class to extend from, and add necessary methods to make it work.
   Hints:
   - You need to implement **Collection** interface, but not **List** interface.
   - You need to implement methods that are declared **abstract** in the abstract class.
   - The iterator class for **MyLinkedList** class is already there: **MyLLIterator**.
   - There is already a data member **size** representing the size of the list.
4. Run **Lab9.testMyLinkedList()** to check if **MyLinkedList** works correctly.

3 Extending Stack

Goal. Extend the standard **Stack** class so that we can "peek" not only the top of the stack, but also the objects below the top of stack.

1. Look at the specification of the standard **Stack** class. Understand how **peek()**, **push()**, **pop()** methods work.

2. Implement method **MyStack.peek(int depth)** so that it has the desired behavior. You can use any methods of the standard **Stack** class.
   (Hints: you would be able to implement it using only the **peek()**, **push()**, and **pop()** methods of **Stack** class.)

3. Run testing method **Lab9.testMyStack()** to check if it works correctly.

4 List and ListIterator

Make **MyLinkedList** class implement the **List** interface.

- Similarly to part 2, it is painful to implement **List** interface directly. You should find an appropriate abstract class to extend from.

- The testing is provided in **Lab9.testListIteratorOfLinkedList()**. Uncomment it, do necessary things to make it work.