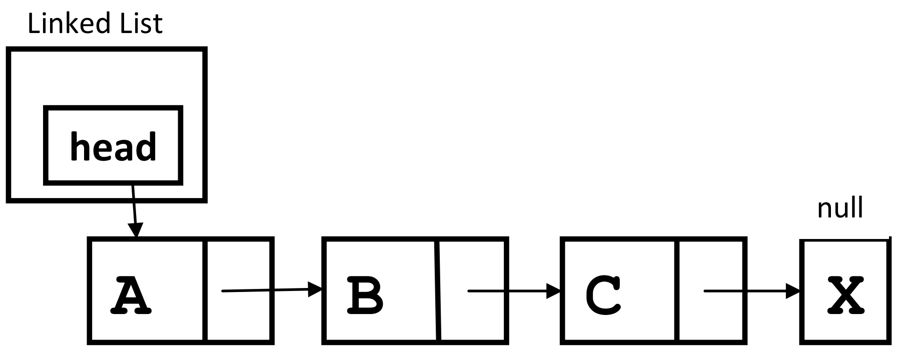
HW3: Writing a Linked List

DUE DATE: Friday February 15, 2008

A linked list is a dynamic data structure that stores information through nodes. Each node is a data structure itself that stores information and the next node that it references. Unlike an array, a linked list can be adjusted without moving any data over to a new array or without shifting over any information. Therefore, it can be beneficial in many ways depending on the task at hand.

As a linked list, you only know about the first node. That node has its data element and then points you to the next node. To find the last element in the list you need to start at the first node and traverse the list node by node until you run out of elements (if the next of a node points to null, then you are at the end of the list)



For Homework 3 you will be creating a linked list that stores contact information for your friends. You will be turning in 3 files: `ContactList.java`, `ContactNode.java`, `ContactEntry.java`.

ContactEntry.java

Since we will be making a linked list of contacts, it makes sense to write an Object that can hold information required for a friend. You will be a simple address book entry, but you still need three data fields to store name, phone number, and email address. These data fields must be private and you will use accessors and modifiers to view and modify them. You will also need constructors to initialize your `ContactEntry`. You will need the following constructors:

```
public ContactEntry(){..}
public ContactEntry(String name, String phone, String email){..}
```

You may have more constructors then that, but those are the required elements. You will also need a `toString()` method which returns a `String` summarizing the information about your contact.

ContactNode.java

This is your node that will make up your linked list. It needs to have a data element (of type `ContactEntry`) and it will need to have a data element that will be the next node in the list (of type `ContactNode`). These elements also have to be private and have the appropriate accessors and modifiers. You will be required to have a constructor, `public ContactNode(ContactEntry data),` that creates a stand alone node with `data` as its data element. You will also be required to have a `toString()` that summarizes the information in your data element (the contact).

ContactList.java

This is the actual linked list and will be your entry point into your address book. It will not have access to every element, just the first one (typically called head). Once again your head data element will be private and will need to include the appropriate accessors and modifiers. This is where you will insert elements and remove elements from your address book. Be sure to include a constructor that initializes the list by setting head to null. `ContactList` will need to have the following methods:

```
public ContactList(){…}
public ContactList(ContactNode head){…}

public ContactNode getHead(){…}
public void setHead(ContactNode head){…}

public void add(ContactNode contactToInsert){…}
public boolean insertAfter(ContactNode node, ContactNode
contactToInsert){…}
public boolean delete(ContactNode node){…}

public ContactEntry searchByName(String name){…}
public ContactEntry searchByPhone(String phone){…}
public ContactEntry searchByEmail(String email){…}

public int size(){…}
public String toString(){…}
```

Bonus (+10pts)

Allow your users to sort your list. Add methods `sortByName()`, `sortByPhone()`, and `sortByEmail()` to your `ContactList`.

What to Turn In

- ContactEntry.java
- ContactNode.java
- ContactList.java

How to Turn In

- Turn in via TSquare