**CSC 205 PROJECT 3 FALL 2012**

**T.M. Rao DUE: December 5, 2012 Points 30**

**ContactList: An application of Doubly Linked Lists**

In this project you are asked to create an *address book,* application program. An address book should be implemented as a doubly linked list of *Contact* objects. Each Contact consists of the following information:

**Contact**: lastName, firstName, middleName, monthOfBirth, dayOfBirth, cellPhone, homePhone, eMail, address, city, state, zipcode.

Sample Data: (“Smith”, “John”, “R”, 10, 22, “111-222-3333”, “111-222-4444”, [jsmith@gmail.com](mailto:jsmith@gmail.com), “22 A Street”, “Brockport”, “NY”, “14420”)

An **ContactList** is a collection of such contacts, maintained in the alphabetical order of last names. (Within the same last names, they are sorted in the alphabetical order of first names, and then according to middle names and cell-phone numbers).

The AddressBook application should have the following capabilities:

INSERT

The user should be able to insert a new contact into the ContactList in alphabetical order

DELETE

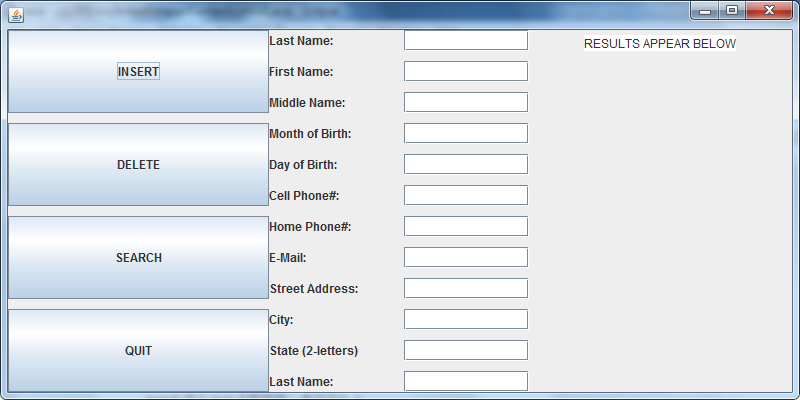
The user should be able to delete a contact from the Contact list

SEARCH

To search for something, the user enters partial or complete search criteria and the software displays all Contacts that “meet” the criteria. The input screen for the search criteria is the same as the one for inserting a new contact. The software will capture the data entered and computes the list of contacts that “meet” the criteria. What we mean by “meeting” the criteria is:

* For string data, the piece of data in the search criteria is a substring of the corresponding data in the table. This means that, if data entered for last-name is “Sm” then all last names which contain “Sm” as a substring meet the criteria. (So, Smith, Smalley, Jasmin, etc. will all qualify)
* For numerical data, the number should match exactly. So, if you want people who are born in October, you should enter 10, precisely.

The user interface for this looks like this:



The application is supposed to work like this:

* When the program launches, it will read a list of contacts that are available in a file called contacts.txt. This file is formatted like this: first line contains a number like 20 which represents the number of contacts in the list. This is followed by 20 lines, each of which contains all the information about one contact. It starts with an empty DoublyLinkedList and inserts each of the addresses in alphabetical order. And then, the above GUI is presented to the user
* When the user clicks the **insert** button, data is captured from the text fields, a new Contact object is created and inserted into the list in alphabetical order. We first make sure that at least the last name and first name are not empty. We make sure that the new contact is NOT identical to an existing contact. (No two contacts can have the same last name and first name). We inform the user about the success of the insert operation in the text area to the right.
* When the **delete** button is clicked, we capture the data and instantiate a Contact object, and search for this object in the list using last name and first name. When something that has the same last name and first name is found, we load that contacts full info into the text fields and ask the user (Using a JOptionPane) to confirm the delete operation. If the answer is, yes, we delete that contact from the doubly-linked-list. We inform the user about the success of the delete operation in the text area to the right.
* When the **search** button is clicked, we collect all the contacts whose data ‘matches’ the data that is in the text fields. We create a sublist of such contacts and display it in the textarea. What we mean by matches is: if lastName field contains “sm”, all last names which contain “sm” as substring will ‘match’. Thus if, we put “sm” in the last name field and hit the search button, we get all the info about Smith, Smalley, Besmer, etc
* When the **quit** button is clicked, we write the current contents of the doubly-linked-list back to the same file (contacts.txt) and quit.

You have to write:

* class Contact: This just represents one Contact, and has get and set methods for all the fields and the toString() method.
  + It should have a public int compareTo(Contact other) method that results -1 if this less-than other, 0 if this equals other, and 1 if this greater other. To do this comparison, we use the concatenation of last name and the first name.
  + It should have a public boolean matches(Contact other) method. This method returns true if this object matches the other object. (Match as we described above, first name of this is a substring of the first name of the other, etc.)
* class ContactList: This represents a collection of Contact objects.
  + It will have an instance variable called **cList** which will be a DoublyLinkedList<Contact> cList = new DoublyLinkedList<Contact>();
  + It will have a constructor that reads the data from the file contacts.txt
  + It will have: public boolean insert(Contact c) method which inserts the object c into the alphabetically sorted **cList.** We first make sure that it is not already in the cList. If it is we return false. Otherwise, we use the compareTo method of Contact and compute how many objects in the list are less than the object you are trying to insert. If this number is k, then we use the insertAt method of the DLL to insert it at location k.
  + It will have a public boolean delete(Contact c) method. We find the first contact that has the same last name and first name (We keep a counter as we go). If one is found, we then ask the user for confirmation of the delete. (Use JOptionPane). If it is confirmed, we delete the contact using the deleteNth method of the DLL and return true. If no such contact was found, we return false.
  + It will have a: public DoublyLinkedList<Contact> search(Contact c) method that searches the cList and creates a new list of all contacts that ‘match’ the contact c. (Match as defined above).
  + It will have a save method that writes the current contents of the cList into the file contacts.txt
* class ContactListGUI: A large portion of this class will be given to you. You have to fill in the action listeners.

**Turn in:** The code for all the classes, and screen captures of the program in action (at least one insert, one delete, one search). Show the data file before program runs, and after the program terminates.

**package linkedlistapp;**

**//System Imports**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.\*;**

**//Project Imports**

**import linkedlist.\*;**

**public class ContactListGUI extends JFrame {**

**private JPanel navigationPanel;**

**private JPanel dataEntryPanel;**

**private JPanel resultsPanel;**

**private JTextArea resultsArea;**

**private JButton insertButton;**

**private JButton deleteButton;**

**private JButton searchButton;**

**private JButton quitButton;**

**private JLabel lastNameLabel = new JLabel("Last Name:");**

**private JLabel firstNameLabel = new JLabel("First Name:");**

**private JLabel middleNameLabel = new JLabel("Middle Name:");**

**private JLabel monthOfBirthLabel = new JLabel("Month of Birth:");**

**private JLabel dayOfBirthLabel = new JLabel("Day of Birth:");**

**private JLabel cellPhoneNumberLabel = new JLabel("Cell Phone#:");**

**private JLabel homePhoneNumberLabel = new JLabel("Home Phone#:");**

**private JLabel eMailLabel = new JLabel("E-Mail:");**

**private JLabel addressLabel = new JLabel("Street Address:");**

**private JLabel cityLabel = new JLabel("City:");**

**private JLabel stateLabel = new JLabel("State (2-letters)");**

**private JLabel zipcodeLabel = new JLabel("Last Name:");**

**private JTextField lastNameTextField = new JTextField(10);**

**private JTextField firstNameTextField = new JTextField(10);**

**private JTextField middleNameTextField = new JTextField(10);**

**private JTextField monthOfBirthTextField = new JTextField(10);**

**private JTextField dayOfBirthTextField = new JTextField(10);**

**private JTextField cellPhoneNumberTextField = new JTextField(10);**

**private JTextField homePhoneNumberTextField = new JTextField(10);**

**private JTextField eMailTextField = new JTextField(10);**

**private JTextField addressTextField = new JTextField(10);**

**private JTextField cityTextField = new JTextField(10);**

**private JTextField stateTextField = new JTextField(10);**

**private JTextField zipcodeTextField = new JTextField(10);**

**private ContactList myList;**

**public ContactListGUI(ContactList cL)**

**{**

**myList = cL;**

**instantiateGUIComponents();**

**buildGUI();**

**addListeners();**

**setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);**

**setSize(800,400);**

**setVisible(true);**

**}**

**//--------------------------------------------------------------**

**private void instantiateGUIComponents()**

**{**

**navigationPanel = new JPanel();**

**dataEntryPanel = new JPanel();**

**resultsPanel = new JPanel();**

**resultsArea = new JTextArea();**

**resultsArea.append("RESULTS APPEAR BELOW");**

**insertButton = new JButton("INSERT");**

**deleteButton = new JButton("DELETE");;**

**searchButton = new JButton("SEARCH");**

**quitButton = new JButton("QUIT");**

**}**

**//--------------------------------------------------------------**

**private void buildGUI()**

**{**

**//--------------Get Content Pane to add components**

**Container c = getContentPane();**

**navigationPanel.setLayout(new GridLayout(4, 1, 10, 10));**

**dataEntryPanel.setLayout(new GridLayout(12, 2, 10, 10));**

**navigationPanel.add(insertButton);**

**navigationPanel.add(deleteButton);**

**navigationPanel.add(searchButton);**

**navigationPanel.add(quitButton);**

**dataEntryPanel.add(lastNameLabel);**

**dataEntryPanel.add(lastNameTextField);**

**dataEntryPanel.add(firstNameLabel);**

**dataEntryPanel.add(firstNameTextField);**

**dataEntryPanel.add(middleNameLabel);**

**dataEntryPanel.add(middleNameTextField);**

**dataEntryPanel.add(monthOfBirthLabel);**

**dataEntryPanel.add(monthOfBirthTextField);**

**dataEntryPanel.add(dayOfBirthLabel);**

**dataEntryPanel.add(dayOfBirthTextField);**

**dataEntryPanel.add(cellPhoneNumberLabel);**

**dataEntryPanel.add(cellPhoneNumberTextField);**

**dataEntryPanel.add(homePhoneNumberLabel);**

**dataEntryPanel.add(homePhoneNumberTextField);**

**dataEntryPanel.add(eMailLabel);**

**dataEntryPanel.add(eMailTextField);**

**dataEntryPanel.add(addressLabel);**

**dataEntryPanel.add(addressTextField);**

**dataEntryPanel.add(cityLabel);**

**dataEntryPanel.add(cityTextField);**

**dataEntryPanel.add(stateLabel);**

**dataEntryPanel.add(stateTextField);**

**dataEntryPanel.add(zipcodeLabel);**

**dataEntryPanel.add(zipcodeTextField);**

**resultsPanel.add(resultsArea);**

**c.setLayout(new GridLayout(1,3));**

**c.add(navigationPanel);**

**c.add(dataEntryPanel);**

**c.add(resultsPanel);**

**}**

**//--------------------------------------------------------------**

**private void addListeners()**

**{**

**insertButton.addActionListener(**

**new ActionListener()**

**{**

**public void actionPerformed(ActionEvent e)**

**{**

**}**

**});**

**deleteButton.addActionListener(**

**new ActionListener()**

**{**

**public void actionPerformed(ActionEvent e)**

**{**

**}**

**});**

**searchButton.addActionListener(**

**new ActionListener()**

**{**

**public void actionPerformed(ActionEvent e)**

**{**

**}**

**});**

**quitButton.addActionListener(**

**new ActionListener()**

**{**

**public void actionPerformed(ActionEvent e)**

**{**

**System.*exit*(0);**

**}**

**}**

**);**

**}**

**/\*\***

**\* @param args**

**\*/**

**public static void main(String[] args) {**

**ContactList cL = new ContactList();**

**new ContactListGUI(cL);**

**}**

**}**