**Class 1:**

package task2;

/\*\*

\* Tests class Skeleton by calling all methods on a Skeleton object.

\*/

public class TestContactList {

public static void main(String args[]) {

mainContactAppLoop newContactList = new mainContactAppLoop();

newContactList.mainLoop();

}

}

**Class 2:**

package task2;

import java.io.PrintStream;

import java.util.Scanner;

public class Contact {

private String firstName;

private String lastName;

private String streetAddress;

private String emailAddress;

private String phoneNumber;

private String notes;

private static final String firstNamePrompt = "FirstName:";

private static final String lastNamePrompt = "lastName:";

private static final String streetAddressPrompt = "StreetAddress:";

private static final String emailAddressPrompt = "EmailAddress:";

private static final String phoneNumberPrompt = "PhoneNumber:";

private static final String notesPrompt = "Notes:";

public void readContact(PrintStream out, Scanner in) {

/\*\*

\* read a contact from in using the prompt strings, print to out

\* keeps prompts and everything else in one place

\*/

}

public String toString(){

/\*\*

\* check that the contacts are valid

\*/

return null;

}

public String getLastName(){

return null;

/\*\*

\* basic method, just return lastName

\*/

}

}

**Class 3:**

package task2;

/\*\*

\* ContactList

\* Operations for managing a list of contacts. This includes the key add method

\* plus ways of printing the whole list or just finding and printing matching entries by matching

\* last name. Also includes methods for exporting whole list to disk and importing it back

\* again when program is restarted.

\*/

import java.io.PrintStream;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

public class ContactList {

/\*\*

\* suggested implementation; use an ArrayList - because it can automatically adjust its size and

\* can be serialized to disk and back saving us time on figuring out how to do that.

\* Other alternatives might be a data structure that maintains sort order so we don't need to re-sort

\* every time the list is printed.

\*/

List<Contact> myContactList = new ArrayList<Contact>();

public void addContact(Contact newPerson) {

/\*\*

\* TODO add a new contact to the list, which would use the .add method for an ArrayList

\*/

}

public void printContactList(PrintStream out) {

/\*\*

\* TODO first the array needs to be sorted alphabetically by last name, and then first name

\* case INsensitive, then we can iterate through the sorted list and print that to out using

\* the toString method on each contact.

\*

\*/

}

public void printContactByName(PrintStream out, String name) {

/\*\*

\* TODO use an Iterator to cycle through Contacts in contact list, if

\* the last name matches name (case INsensnsitive) then print out the Contact using its

\* toString method. If not found then print "Could not find" message.

\*/

}

public void exportContactList(String filename) {

/\*\*

\* TODO save/export the contactList into filename. We need to look at the serialization

\* capabilities of the ArrayList class to see how this can be easily done.

\*/

}

public Boolean importContactList(String filename) {

/\*\*

\* TODO import the ContactList from filename where it was previously saved. Again, we need to

\* look at ArrayList serialization. Some error checking if filename does not exist also needed.

\* Return true/false depending on whether import succeeded.

\*/

return true;

}

};