Phase I Report

Team: MMA

Authors:

- Muhannad Alasmari, 443102430
- Muhammed Alhithlool, 443101218
- Abdullah Alsalman, 443105689

Content:

Link to GitHub:

https://github.com/mohammedalhazlool/CSC_212_resepritory

Initial Plan

Our initial plan consisted of several stages:

- The first stage was to handle each class separately in terms of abstract planning in the beginning to avoid running into problems with method implementation in each class, but we couldn't separate their planning completely since methods in one class could depend on functionalities in other classes, like the compareTo in class Contact method for example.

- The second stage was to develop a knowledge of the amount of necessary methods for each class, to not provide too much time complexity to each class, and by extension the whole project. We've also wanted to specify which methods (specifically in LinkedListADT and Phonebook) we'll need multiple versions of.

- The third stage was to identify difficulties spread through the project and how it'll be handled as we implement the project, and to revise all classes after implementation to check for their time complexities and to modify conflicts nested within the code that may have gone unnoticed.

Meetings

Meeting 1: Splitting the work – 2023/9/16 – 9AM:

In our first meeting, we've discussed the structure of the phonebook project, and how we're going to implement each class using Java Programming language.

We've also split both our project work and implementation work to make our workflow efficient as follows:

- Class Contact: Muhammed Alhithlool
- Class LinkedListADT: Muhannad Alasmari
- Class Phonebook: Abdullah Alsalman & Muhammed Alhithlool
- Class Event: Muhammed Alhithlool
- Report: Muhannad Alasmari

Meeting 2: Discussing method implementations – 2023/9/25 – 4:30PM:

In our second meeting, we've began discussing the fixed necessary amount of methods for each class, as well as which methods necessary to have one or another version of, as several methods of several functions will be needed for completing the core functionality of the project.

Meeting 3: Discussing the difficulties – 2023/10/2 – 4:00PM:

In our third meeting, we've discussed the difficulties we've faced or we'll eventually face with our implementations. Such as:

- The functionality of sorting the names alphabetically using one of our add methods.
- How the search functionality will be implemented into the project.
- The deletion of the events associated with a contact in the case of deleting them from the list.

Each one was discussed with consideration of the way it should be implemented throughout the project.

Meeting 4: Complete revision – 2023/10/10 – 4:00PM:

In our fourth meeting, we've revised each of our classes after our implementations to check for any conflicts or discrepancies within them and to determine each one's time complexity.

Time Complexity

Class Contact

Methods	Freq.	Total
`public LinkedList<Event> getContact_event(){ return contact_event; }`	1	Total:1 O(1)
`public String getName() { return name; }`	1	Total: 1 O(1)
`public void setName(String name) { this.name = name; }`	1	Total: 1 O(1)
`public String getNumber() { return number; }`	1	Total: 1 O(1)
`public void setNumber(String number) { this.number = number; }`	1	Total: 1 O(1)
`public String getBirthday() { return birthday; }`	1	Total: 1 O(1)
`public void setBirthday(String birthday) { this.birthday = birthday; }`	1	Total: 1 O(1)
`public String getNotes() { return notes; }`	1	Total: 1 O(1)
`public void setNotes(String notes) { this.notes = notes; }`	1	Total: 1 O(1)
`public String getEmail() { return email; }`	1	Total: 1 O(1)
`public void setEmail(String email) { this.email = email; }`	1	Total: 1 O(1)

Methods	Freq.	Total

`public String getAddress() {`
`  return address;`
`}`
	1	Total: 1
		O(1)

`public void setAddress(String address) {`
`  this.address = address;`
`}`
	1	Total: 1
		O(1)

`public int compareTo(String n) {`
`  return (name.compareTo(n));`
`}`
	1	Total: 1
		O(1)

`public void read_contact()`
`  {`
`Scanner read =new Scanner(System.in);`
`System.out.println("Enter the contact's name");`
`name=read.nextLine();`
`System.out.println("Enter the phone number");`
`number=read.nextLine();`
`System.out.println("Enter the birthday");`
`birthday=read.nextLine();`
`System.out.println("Enter the notes");`
`notes=read.nextLine();`
`System.out.println("Enter the email");`
`email=read.nextLine();`
`System.out.println("Enter the address");`
`address=read.nextLine();`
`  }`
	1
	1
	1
	1
	1
	1
	1
	1
	1
	1
	1
	1
	1
		Total: 13
		O(1)

`public String toString() {`
`  return "Contact [name=" + name + ",`
`number=" + number + ", birthday=" + birthday + ",`
`notes=" + notes + ", email="`
`  + email + ", address=" +`
`address + "]";`
`}`
	1	Total: 1
		O(1)

`public void display_contact() {`
`System.out.println("name: "+name);`
`System.out.println("email: "+email);`
`System.out.println("number: "+number);`
`System.out.println("address: "+address);`
`System.out.println("birthday:`
`"+birthday);`
`System.out.println("notes: "+notes);`
`}`
	1
	1
	1
	1
	1
	1
		Total: 6
		O(1)

Class LinkedListADT

Methods	Freq.	Total

`public boolean empty(){`
`  return head == null;`
`}`
	1	Total: 1
		O(1)

`public boolean full(){`
`  return false;`
`}`
	1	Total: 1
		O(1)

`public void findFirst(){`
`  current = head;`
`}`
	1	Total: 1
		O(1)

`public boolean isLast(){`
`  return current.next == null;`
`}`
	1	Total: 1
		O(1)

`public void findNext(){`
`  current = current.next;`
`}`
	1	Total: 1
		O(1)

`public void update(T c){`
`  current.data = c;`
`}`
	1	Total: 1
		O(1)

`public T retrieve(){`
`  return current.data;`
`}`
	1	Total: 1
		O(1)

`public void add(T c){`
`  if(empty())`
`  current = head = new Node<T>(c);`
`  else{`
`  Node<T> tmp;`
`  tmp = current.next;`
`  current.next = new Node<T>(c);`
`  current = current.next;`
`  current.next = tmp;`
`  }`
`}`
	1
	1
	1
	1
	1
	1
	1
	1
		Total: 8
		O(1)

Methods	Freq.	Total

public void addInOrder(T c){
Node<T> n = new Node<T>(c);
if(empty()){
current = n;
head = n;
return;
}
else{
Node<T> n = new Node<T>(c);
 	
if(((Contact)c).compareTo(((Contact)head.data).getName())<0){
n.next = head;
head = n;
}
else{
Node<T> inQueue = null, tmp = head;
while(tmp!=null&&(((Contact)tmp.data).compareTo(((Contact)c).getName())<=0)){
 	
inQueue = tmp;
tmp = tmp.next;
}
inQueue.next = n;
n.next = tmp;
}
}
}
	
	1
	1
	1
	1
	1
	
	1
	1
	
	1
	1
	1
	
	n+1
	
	n
	n
	
	1
	1
		Total:
		3n+15
		O(n)

public void delete(){
if(current==head)
head = head.next;
else{
Node<T> tmp = head;
while(tmp.next!=current)
tmp = tmp.next;
tmp.next = current.next;
}
if(current.next==null)
current = head;
else
current = current.next;
}
	
	1
	1
	1
	1
	n+1
	n
	1
	
	1
	1
	1
	1
		Total:
		2n+10
		O(n)

Methods	Freq.	Total

public boolean search(T c){
Node<T> tmp = head;
while(tmp!=null){
if(tmp.data.equals(c))
return true;
tmp = tmp.next;
}
return false;
}
	
	1
	n+1
	n
	n
	n
	
	1
		Total:
		4n+3
		O(n)

public void display(){
Node tmp;
while(tmp!=null){
System.out.print(tmp.data+"->");
tmp = tmp.next;
}
}
	
	1
	n+1
	n
	n
		Total:
		3n+2
		O(n)

Class Phonebook

Methods	Freq.	Total

public void Add_Sorted_User(Contact d) {
 	
if (all_contacts.empty()) {
 	
all_contacts.add(d);
return;
}
else {
all_contacts.findFirst();
if(d.compareTo(all_contacts.retrieve().getName())<0){
Contact c = new Contact(all_contacts.retrieve());
all_contacts.update(d);
all_contacts.add(c);
return;
}
 	
else {
 	
while(!(all_contacts.retrieve().compareTo(d.getName())<=0)) {
all_contacts.findNext();
}
}
 	
if(all_contacts.retrieve().compareTo(d.getName())>0) {
Contact c = new Contact(all_contacts.retrieve());
all_contacts.update(d);
all_contacts.add(c);
}
else {
 	
d.display_contact();
all_contacts.add(d);
 	
}
}
}
	
	1
	
	9
	1
	
	1
	1
	1
	1
	1
	9
	1
	
	1
	
	n+1
	n
	
	1
	1
	1
	9
	
	1
	
	1
	9
		Total:
		2n+51
		O(n)

Methods	Freq.	Total

public boolean search(Contact c) {
if (all_contacts.empty())
return false;
all_contacts.findFirst();
while (!all_contacts.isLast()) {
if(all_contacts.retrieve().getName().equals(c.getName())||all_contacts.retrieve().getNumber().equals(c.getNumber()))
return true;
all_contacts.findNext();
}
if(all_contacts.retrieve.getContact_name().equals(c.getContact_name())||all_contacts.retrieve().getPhone_num().equals(c.getPhone_num()))
return true;
else
return false;
}
	
	1
	1
	1
	n+1
	n
	
	n
	n
	
	1
	
	1
	1
	1
		Total:
		3n+9
		O(n)

public void add_contact(Contacts c) {
boolean found = search(c);
if (!found) {
contacts.addInOrder(c);
}
}
	
	3n+10
	1
	3n+12
		Total:
		6n + 23
		O(n)

public LinkedListADT<Contact> search_by_First_name(String n){
LinkedListADT<Contact>res = new LinkedListADT<Contact>();
if(all_contact.empty())
return res;
all_contacts.findFirst();
while(!all_contacts.isLast()){
String cur_Full_Name=all_contacts.retrieve().getName();
String First_name=curr_Full_Name.substring(0,cur_Full_Name.indexOf(" ")-1);
if(First_name.equals(n))
res.add(all_contacts.retrieve());
 	
all_contacts.findNext();
}
String cur_Full_Name=all_contacts.retrieve().getName();
String First_name=curr_Full_Name.substring(0,cur_Full_Name.indexOf(" ")-1);
if(First_name.equals(n))
res.add(all_contacts.retrieve());
return res;
}
	
	1
	1
	1
	1
	n+1
	n
	
	n
	
	n
	8n
	
	n
	
	1
	1
	
	1
	1
	1
		Total:
		13n+10
		O(n)

Methods	Freq.	Total

public static void print_Linked_List_Events(LinkedList<Event>A){
if(!A.empty()){
A.findFirst();
while(!A.isLast()){
System.out.println(A.retrieve()+"
linked with contacts: ");
 	
print_contacts_byname(A.retrieve().contact_event);
A.findNext();
}
System.out.println(A.retrieve()+" linked with
contacts: ");
 	
print_contacts_byname(A.retrieve().contact_event);
}
else
System.out.println("is empty");
}
	
	1
	1
	n+1
	n
	
	n
	n
	
	1
	
	3n+6
	
	1
	1
		Total: 7n+12
		O(n)

public Contact search__by_name(String n) {
if(all_contacts.empty())
return null;
all_contacts.findFirst();
while(!all_contacts.isLast()) {
 	
if(all_contacts.retrieve(getContact_name().equals(n))
return all_contacts.retrieve();
all_contacts_findNext();
}
if(all_contacts.retrieve(getName().equals(n))
return all_contacts.retrieve();
return null;
}
	
	1
	1
	1
	n+1
	
	n
	n
	n
	
	1
	1
	1
		Total: 4n+7
		O(n)

public Contact search__by_Phone(String ph) {
if (all_contacts.empty())
return null;
all_contacts.findFirst();
while (!all_contacts.isLast()) {
if (all_contacts.retrieve().getNumber().equals(ph))
res.add(all_contacts.retrieve());
all_contacts.findNext();
}
if (all_contacts.retrieve().getNumber().equals(ph))
res.add(all_contacts.retrieve());
return null;
}
	
	1
	1
	1
	n+1
	n
	8n
	n
	
	1
	8
	1
		Total:
		11n+14
		O(n)

public LinkedListADT<Contact> search_email(string e){
LinkedListADT<Contact> res = new LinkedListADT<Contact>();
if(all_contacts.empty())
return res;
all_contacts.findFirst();
do {
 	
if(all_contacts.retrieve().getEmail().equals(e)){
res.add(all_contacts.retrieve());
all_contacts.findNext();}
 	
}while(!all_contacts.isLast());
if(all_contacts.retrieve().getEmail().equals(e))
res.add(all_contacts.retrieve();
return res;}
	
	1
	1
	1
	1
	n+1
	
	n
	8n
	1
	
	n
	1
	8
	1
		Total:
		11n+16
		O(n)

Methods	Freq.	Total

public LinkedList<Contact> search_address(String a){
LinkedList<Contact> res =new LinkedList<Contact>();
if(all_contacts.empty())
return res;
all_contacts.findFirst();
while(!all_contacts.isLast()) {
 	
if(all_contacts.retrieve().getAddress().equals(a));
res.add(all_contacts.retrieve());
all_contacts.findNext();
}
if(all_contacts.retrieve().getAddress().equals(a));
res.add(all_contacts.retrieve());
return res;
}
	
	1
	1
	1
	1
	n+1
	
	n
	8n
	n
	
	1
	8
	1
		Total:
		11n+15
		O(n)

public LinkedList<Contact> search_birth(String b){
LinkedList<Contact> res=new LinkedList<Contact>();
if(all_contacts.empty())
return res;
all_contacts.findFirst();
while(!all_contacts.isLast()) {
 	
if(all_contacts.retrieve().getBirthday().equals(b))
res.add(all_contacts.retrieve());
 	
all_contacts.findNext();
}
if(all_contacts.retrieve().getBirthday().equals(b))
res.add(all_contacts.retrieve());
return res;
 	
}
	
	1
	1
	1
	1
	n+1
	
	n
	8n
	
	n
	
	1
	8
	1
		Total:
		11n+15
		O(n)

public boolean is_conflict(Event e,Contact c) {
LinkedList<Event>contacts_event=c.contact_event;
if(contacts_event.empty())
return false;
 	
contacts_event.findFirst();
while(!contacts_event.isLast()) {
 	
if(e.getEvent_date().equals(contacts_event.retrieve().getE
vent_date())&&e.getEvent_time().equals(contacts_event.retrieve().getE
vent_time()))
return true;
contacts_event.findNext();
}
 	
if(e.getEvent_date().equals(contacts_event.retrieve().getE
vent_date())&&e.getEvent_time().equals(contacts_event.retrieve().getE
vent_time()))
return true;
else
return false;
}
	
	1
	1
	1
	
	1
	n+1
	
	n
	
	n
	n
	
	1
	
	1
	1
	1
		Total: 4n+9
		O(n)

Methods	Freq.	Total

public void schedule_event(Event e, String contact_name) {
Contact this_contact = search__by_name(contact_name);
if(this_contact==null) {
System.out.println("can not schedule this event because this contact does not exist"+contact_name);
return;
}
 	
boolean is_conflict=is_conflict(e,this_contact);
if(this_contact!=null&&!is_conflict) {
System.out.println("schedlulling contact"+this_contact.getName()+"'"+e.getEvent_title());
 	
this_contact.contact_event.add(e);
e.contact__event.add(this_contact);
add_Event(e);
 	
this_contact.event_in_contact=e;
 	
this_contact.event_in_contact.contact__event.add(this_contact);
}
}
	1
	
	1
	1
	
	1
	
	4n+9
	
	1
	1
	
	3n+12
	3n+12
	7n+23
	
	1
	
	8
		Total:
		17n+71
		O(n)

public static void print_contacts(Phonebook p1){
if(!p1.all_contacts.empty()){
p1.contacts.findFirst();
while(!p1.all_contacts.isLast()){
 	
p1.all_contacts.retrieve().display_contact();
p1.all_contacts.findNext();
}
p1.all_contacts.retrieve().display_contact();
}
else
System.out.println("is empty");
}
	1
	1
	n+1
	
	6n
	n
	
	6
	
	1
	1
		Total: 8n+5
		O(n)

public static void print_contacts_byname(LinkedListADT<Contact>A){
if(!A.empty()){
A.findFirst();
while(!A.isLast()){
 	
System.out.println(A.retrieve().get_Name());
A.findNext();
}
System.out.println(A.retrieve().get_Name());
}
else
System.out.println("is empty");
}
	1
	1
	n+1
	
	n
	n
	
	1
	
	1
	1
		Total: 3n+6
		O(n)

Methods	Freq.	Total

`public static void print_List_Events(LinkedListADT<Event>A){`
`if(!A.empty()){`
`A.findFirst();`
`while(!A.isLast()){`
`System.out.println(A.retrieve()+" linked with contacts: ");`
 	
`print_contacts_byname(A.retrieve().contact_event);`
`A.findNext();`
`}`
`System.out.println(A.retrieve()+" linked with contacts: ");`
 	
`print_contacts_byname(A.retrieve().contact_event);`
`}`
`else`
`System.out.println("is empty");`
`}`
	
	1
	1
	n+1
	n
	
	n(3n+6)
	n
	
	1
	
	3n+6
	
	1
	1
		Total:
		$3n^2+12n+12$
		$O(n^2)$

`public void add_Event(Event e) {`
`Event f = search_event_title(e.getEvent_title());`
`if(f==null) {`
`Events.Add_sorted_Event(e);`
`}`
 	
`}`
	4n+7
	
	1
	3n+15
		Total:
		7n+23
		O(n)

`public LinkedListADT<Contact> contacts_event(String a){`
`Event f = search_event_title(a);`
`if(f!=null)`
`return f.contact__event;`
`return new LinkedListADT<Contact>();`
 	
`}`
	4n+7
	1
	1
	1
		Total:
		4n+10
		O(n)

`public LinkedListADT<Event> events_contact(String c){`
`Contact g = search__by_name(c);`
`if(c!=null)`
`return g.contact_event;`
`return new LinkedListADT<Event>();`
 	
`}`
	4n+7
	1
	1
	1
		Total:
		4n+10
		O(n)

Methods	Freq.	Total

public void delete_events_with_contacts(String s, LinkedList<Event>A){
System.out.print(" ");
while(!A.empty()){
String E_title= A.retrieve().getEvent_title();
delete_event(E_title, s);
A.delete();
}
}
	
	1
	n+1
	n
	9n+21
	1
		Total: 11n+24
		O(n)

public static void search_criteria_for_searching() {
System.out.println("Enter search criteria:\n"+ "1. Name\n" + "2. Phone Number\n" + "3. Email Address\n" + "4. Address\n" + "Birthday");
}
	1	Total: 1
		O(1)

public static void print_Linked_List_of_all_contacts(LinkedList<Contact>L) {
if(L.empty())
System.out.println("empty list");
else {
L.findFirst();
while(!L.isLast()) {
L.retrieve().display_contact();
System.out.println("");
L.findNext();
}
L.retrieve().display_contact();
System.out.println("");
}
}
	
	1
	1
	1
	1
	n+1
	6
	1
	1
	
	6
	1
		Total: n+20
		O(n)

public void Linked_List_of_Events(LinkedList<Event>L) {
if(L.empty())
System.out.println("empty list");
else {
L.findFirst();
while(!L.isLast()) {
System.out.println(L.retrieve());
System.out.println("this event has the following contacts");
 	
print_contacts_by_name(L.retrieve().contact_event);
System.out.println("");
L.findNext();
}
}
}
	
	1
	1
	1
	1
	n+1
	n
	n
	
	n(3n+6)
	n
	n
		Total:
		$3n^2+11n+5$
		$O(n^2)$

Methods	Freq.	Total

public void delete_contact(String s) {
if(Contacts.empty())
System.out.println(" list is empty ");
Contacts.findFirst();
while(!Contacts.last()) {
 	
if(Contacts.retrieve().getName().equals(s)) {
 	
LinkedListADT<Event>A=getEvents_contact(s);
delete_events_with_contacts(s,A);
System.out.println(" contact has been removed ");
Contacts.delete();
}
Contacts.findNext();
}
if(Contacts.retrieve().getName().equals(s)) {
 	
LinkedList<Event>A=getEvents_contact(s);
delete_events_with_contacts(s,A);
System.out.println(" contact has been removed ");
Contacts.delete();}
 	
}
	
	1
	1
	1
	n+1
	
	n
	
	n
	n(11n+24)
	n
	
	n(2n+10)
	
	n
	
	1
	
	1
	11n+24
	1
	
	2n+10
		Total:
		$13n^2+52n+41$
		$O(n^2)$

public Event search_event_title(String s) {
if(all_events.empty())
return null;
Events.findFirst();
while(!all_events.isLast()) {
 	
if(all_events.retrieve().getEvent_title().equals(s))
return all_vents.retrieve();
Events.findFirst();
}
if(all_events.retrieve().getEvent_title().equals(s))
return all_events.retrieve();
return null;
}
	
	1
	1
	1
	n+1
	
	n
	
	n
	n
	
	1
	
	1
	1
		Total:
		4n+7
		O(n)

public static void menu() {
System.out.println("Welcome to the Linked Tree Phonebook!");
System.out.println("Please chose an option");
System.out.println("1. Add a contact");
System.out.println("2. Search for a contact");
System.out.println("3. Delete a contact");
System.out.println("4. schedule an event");
System.out.println("5. Print event details");
System.out.println("6. Print contacts by first name");
System.out.println("7. Print all events alphabetically");
System.out.println("8. Exit");
System.out.println("\nEnter your choice: ");
}
	
	1
	1
	1
	1
	1
	1
	1
	1
	1
	1
	1
	1
		Total: 11
		O(1)

Methods	Freq.	Total

public LinkedList<Event> getEvent_contact(String n){
Contact this_contact = search__by_name(n);
if(this_contact!= null)
return this_contact.getContact_event();
return new LinkedList<Event>();
}
	
	4n+7
	1
	1
	1
		Total:
		4n+10
		O(n)

public LinkedList<Contact> getContact_event(String n){
Event this_event = search_event_title(n);
if(this_event!= null)
return this_event.getEvent_contact();
return new LinkedList<Contact>();
}
	
	4n+7
	1
	1
	1
		Total:
		4n+10
		O(n)

public void delete_event(String tit, String n){
LinkedList<Contact>contacts_with_cur_event=getCont
acts_Event(tit);
print_contacts_by_name(contacts_with_cur_event);
contacts_with_cur_event.findFirst();
while(!contacts_with_cur_event.isEmpty&&!contacts_
with_cur_event.isLast){
 	
if(contacts_with_cur_event.retrieve().getName().eq
uals(n){
 	
contacts_with_cur_event.delete();
break;
}
contacts_with_cur_event.findNext();
}
if(contacts_with_cur_event.retrieve().getName().eq
uals(n){
contacts_with_cur_event.delete();
}
}
	1
	
	3n+6
	1
	n+1
	
	n
	
	n(2n+10)
	n
	
	n
	
	1
	
	2n+10
		Total:
		$2n^2+19n+14$
		$O(n^2)$

Class Event

Methods	Freq.	Total

`public String getEvent_title() {`
`  return Event_title;`
`}`
	1	Total: 1
		O(1)

`public void setEvent_title(String event_title) {`
`  Event_title = event_title;`
`}`
	1	Total: 1
		O(1)

`public String getEvent_Location() {`
`  return Event_Location;`
`}`
	1	Total: 1
		O(1)

`public void setEvent_Location(String event_Location) {`
`  Event_Location = event_Location;`
`}`
	1	Total: 1
		O(1)

`public String getEvent_name() {`
`  return Event_name;`
`}`
	1	Total: 1
		O(1)

`public void setEvent_name(String event_name) {`
`  Event_name = event_name;`
`}`
	1	Total: 1
		O(1)

`public String getEvent_date() {`
`  return Event_date;`
`}`
	1	Total: 1
		O(1)

`public void setEvent_date(String event_date) {`
`  Event_date = event_date;`
`}`
	1	Total: 1
		O(1)

`public String getEvent_time() {`
`  return Event_time;`
`}`
	1	Total: 1
		O(1)

`public void setEvent_time(String event_time) {`
`  Event_time = event_time;`
`}`
	1	Total: 1
		O(1)

`public String toString() {`
`  return "Event [Event_title=" + Event_title + ", Event_Location=" + Event_Location + ", Event_name=" + Event_name`
`  + ", Event_date=" + Event_date + ", Event_time=" + Event_time + "]";`
`}`
	1	Total: 1
		O(1)

`public LinkedListADT<Contact> getContact__event() {`
`  return contact__event;`
`}`
	1	Total: 1
		O(1)

Assumptions

Our assumptions were made based on our execution of initial plan, and our discussion of the difficulties during some of our meetings. Some of our initial assumptions included:

- Since the phonebook class represents the application itself, it should be the main link between the other three classes; Meaning that it'll involve methods and attributes from all classes on a certain degree.
- Since the linked list data structure involves generics and could deal with many types of data, there should be a linked list within the Phonebook class and the Event class to help sync the events with the contacts scheduling them.
- Given there are multiple criteria for searching, we've assumed that it would be best to have multiple other search methods within the phonebook class to handle each of the non-unique criteria to not cause a surge in time complexity by letting one method handle all of them.
- We assume there is a many-to-many relationship between the Contact and Event classes, as a contact can schedule many events and an event can be scheduled by many contacts.