College of Computer and Information Sciences

CS 212: Data structure

Phone Book Report – Part 2 [BST]

Section #55027

Student Name
Leader:Renad Alotaibi
Ghazlan aldosari
Lena alhabdan

1.command line menu functionality description

addContact()

Method Input: The method asks the user to enter elements for making a new contact.

Enters: Contact's Name, Contact's Phone Numbe, Contact's E-mail Address, Contact's Address, Contact's Birthday, Notes for the Contact.

Method Output: The method doesn't deliver a direct output that is returned. Instead, it prints messages to the console based on the actions it took.

Method Functionality:

The method verifies the existence of a contact by considering the provided name and phone number. It ensures that there is no redundancy in information. If a contact sharing the same name and phone number is identified, the method displays the message "Contact found!" to notify the user.

- 1. User Input for Contact Details: The method requests the user to provide information such as name, phone number, email address, birthday, and notes for the creation of a new contact.
- 2. Transformation of Input Birthday: The method transforms the input birthday into a Date object.
- 3. Generation of Contact Object: A new Contact object (c) is generated using the given information.
- 4. Contact Inserted: The new contact is added to binary search tree (BST) called "contacts" using the insert method.
- 5. Output Notifies: If the insertion process is successful, the console displays the message "Contact added successfully!

searchContact()

Method Input:

Input: The method asks the user to input a search criteria choice [1 to 5]and the needed information based on the chosen criteria.

Output: The method prints messages indicating whether the contact was found or not, and it prints the details of the found contact.

Method Functionality:

- 1. User Input: The method guides the user to select a search criterion (name, phone number, email, address, or birthday) and asks them to provide the needed details.
- 2. Contact Query: Using the chosen criterion and input information, the method looks for the contact within the (BST) contacts. Distinct search methods are employed based on the selected criterion.
- 3. Output Notification: If the contact is located, it displays "Contact found!" on the console. In the absence of contact, it communicates "Contact not found!".
- 4. Contact Information: In the event of finding the contact, the method prints the contact's details to the console.

deleteContact()

input: the user enters the name of the contact to be deleted.

Output: The method prints a success message, and the details of the deleted contact are displayed, If the specified contact is not found in the contact list, it prints an error message.

Method Functionality:

- 1. Create a new Contact instance using the provided name.
- 2. Verify that the contact list (contacts) is not empty, and the specified contact exists.
- 3. If the contact is found, remove it from the contact list.
 - Handle Events:
 - 3.1: Iterate through the events linked list for the deleted contact.
 - 3. 2: For each associated event:
 - 3. 3: Confirm the event's presence in the global events list (events).
 - 3. 4: If the event is found:
 - 3. 5: Eliminate the deleted contact from the event's list of associated contacts.
- 3. 6: If the event no longer has any associated contacts, remove it from the events list.

• scheduleEvent()

Input: Guides the user in selecting between scheduling an event or an appointment. Collects necessary details including title, contacts, date, time, and location.

Output: Provides informative messages on the outcome of the scheduling process, notifying the user about success or failure. Raises alerts for conflicts or if a specified contact cannot be found.

Method Functionality:

• Processing an event:

- 1. Gathers information including the event title, contacts' names (commaseparated), date, time, and location.
- 2. Verifies conflicts with existing events for each contact.
- 3. If conflict-free, link the event with the contacts and updates both the contact and global events lists.

• Processing an appointment:

- 1. Collects details like the appointment title, contact name, date, time, and location.
- 2. Examines for conflicts with existing events.
- 3. If no conflicts arise, associate the appointment with the contact and update both the contact and global events lists.

• printEvent()

input: Choose search criteria [contact name or event title]to print.

Output: Print events associated with the specified contact or event title.

Method Functionality:

- 1. Takes user input for search criteria.
- 2. Print events based on the chosen criteria.
- 3. Handles cases where contacts or events are not found prints out "not found!".

• printContactsFirstName()

Input: user enters First name for contact search.

Output: Print contacts found by the first name.

Method Functionality:

- 1. Takes user input for a first name.
- 2. Searches and prints contact that first names match the input.

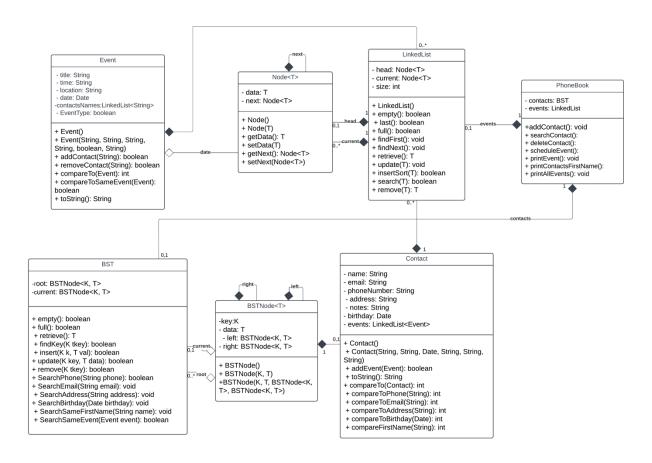
• printAllEvents()

Input: None.

Output: Prints all events alphabetically.

Method Functionality: Prints all events in alphabetical order.

2.Phone Book project UML class diagram



3.time complexity calculation

Method	S/E	Freq.	Total
private static void addContact() {	0	-	0
System.out.println("\nEnter the contact's name:");//1	1	1	1
String name = input.nextLine();	1	1	1
System.out.println("\nEnter the contact's phone number:");//1	1	1	1
String phoneNumber = input.nextLine();	1	1	1
if (!contacts.empty() &&	1	n	n
contacts.findKey(name)&&contacts.SearchPhone(phoneNumber))//1			
{	0	-	0
System.out.println("Contact found!");//1	1	1	1
return;	1	1	1
}	0	-	0
System.out.println("\nEnter the contact's email address:");//1	1	1	1
String emailAddress = input.nextLine();	1	1	1
System.out.println("\nEnter the contact's address:");//1	1	1	1
String address = input.nextLine();	1	1	1
System.out.println("\nEnter the contact's birthday:");//1	1	1	1
String birthday = input.nextLine();	1	1	1
Date birthdayDate = new Date(birthday);	1	1	1
System.out.println("\nEnter any notes for the contact:");//1	1	1	1
String notes = input.nextLine();	1	1	1
Contact c = new Contact(name, emailAddress, birthdayDate, phoneNumber,	1	1	1
address, notes); //create a new contact	1	1	1
if (contacts.insert(name,c))// insert the new contact	1	n	n
System.out.println("Contact added successfully!");	1	1	1
}	0	-	0
			2n
Big o()			O(n)
DIR O()			O(II)
		1	

Method	S/E	Freq.	Total
public static void searchContact(){	1	1	1
System.out.println("Enter search criteria:");//1	1	1	1
System.out.println("1.Name");//1	1	1	1
System.out.println("2.Phone Number");//1	1	1	1
System.out.println("3.Email Address");//1	1	1	1
System.out.println("4.Address");//1	1	1	1
System.out.println("5.Birthday");//1	1	1	1
System.out.println("\nEnter your choice: ");	1	1	1
int choice2 = input.nextInt();	1	1	1
1 0/	0	-	0
if (!contacts.empty()) {//1	1	1	1
switch(choice2)	1	1	1
1	0		0
case 1 : {		1	1
case 1. (0	1	0
Custom out print/"Enter the contestic name. "\.		1	_
System.out.print("Enter the contact's name: ");	1	1	1
input.skip("\r\n [\n\r\u2028\u2029\u0085]");	1	1	1
String name = input.nextLine();	1	1	1
	0	-	0
if (!contacts.empty() && contacts.findKey(name))	1	n	n
{	0	-	0
System.out.println("Contact found!");	1	1	1
	0	-	0
System.out.println(contacts.retrieve().toString());	1	1	1
break;	1	1	1
}	0	-	0
System.out.println("Contact not found!");	1	1	1
	0	_	0
break;	1	1	1
}	0	1-	0
1	0	_	0
case 2 : {	1	1	1
System.out.print("Enter the contact's phone number:");		1	1
input.skip("\r\n [\n\r\u2028\u2029\u0085]");		1	1
String phonenumber = input.nextLine();	1	1	1
	0	-	0
<pre>if (!contacts.empty() && contacts.SearchPhone(phonenumber))</pre>	1	n	n
{	0	-	0
System.out.println("Contact found!");	1	1	1
	0	-	0
System.out.println(contacts.retrieve());	1	1	1
break;	1	1	1
}	0	-	0
System.out.println("Contact not found!");	1	1	1
}	0	-	0
break;	1	1	1
, , , , , , , , , , , , , , , , , , ,	0		0
case 3 : {	1	1	1
System.out.print("Enter the contact's email address: ");	1	1	1
input.skip("\r\n [\n\r\u2028\u2029\u0085]");	1	1	1
String emailaddress = input.nextLine();	1	1	1
String emanaturess – imputatextente(),		1	
:f / leavete ata annutu // \			0
if (!contacts.empty())	1	1	1
{	0	[]	0
contacts.SearchEmail(emailaddress);	1	1	1
System.out.println(contacts.retrieve());	1	1	1

System out println/"Contact found!"	1	1	1
System.out.println("Contact found!");	1	1	1
break;	1	1	1
}	0	-	0
System.out.println("Contact not found!");	1	1	1
}	0	-	0
break;	1	1	1
	0		0
case 4 : {	1	1	1
System.out.print("Enter the contact's address: ");	1	1	1
input.skip("\r\n [\n\r\u2028\u2029\u0085]");	1	1	1
String address = input.nextLine();	1	1	1
String address = input.next.tine(),	0	1	0
:f /leastacha assutu())		1	
if (!contacts.empty())	1	1	1
{	0	-	0
contacts.SearchAddress(address);	1	1	1
System.out.println("Contact found!");	1	1	1
break;	1	1	1
}	0	-	0
System.out.println("Contact not found!");	1	1	1
]	0	-	0
break;	1	1	1
2.54.1,	0	-	0
case 5 : {	1	1	1
System.out.print("Enter the contact's Birthday:");	1	1	1
	0		0
Date birthday = new Date(input.next());	1	1	1
	0		0
if (!contacts.empty())	1	1	1
{	0	-	0
contacts.SearchBirthday(birthday);	1	1	1
System.out.println("Contact found!");	1	1	1
break;	1	1	1
}	0	-	0
System.out.println("Contact not found!");		1	1
System.out.printing Contact not lound: J,			
J harali	0	-	0
break;	1	1	1
}	0	-	0
}	0	-	0
else	1	1	1
System.out.println("Contact not found!");	1	1	1
	0		0
}	0	-	0
			2n
Big o()			O(n)

Method	S/E	Freq.	Total
public static void deleteContact(){	0	-	0
System.out.print("Enter the contact's name: ");	1	1	1
//input.skip("\r\n [\n\r\u2028\u2029\u0085]");	0	1	0
String name= input.nextLine();	1		1
Contact c = new Contact();// create new contact	1	1	1
c.name = name;// set new caontact name	1	1	1
	0		0
if (! contacts.empty())	1	n	n
{	1		1
if (!contacts.findKey(c.name)){	0	-	0
System.out.println("Contact not found!");	1	1	1
return;	1	1	1
}	1	-	1
c = contacts.retrieve();	0	1	0
contacts.remove(c.name);	1	1	1
if (! c.events.empty())	1		1
{	1	1	1
c.events.findFirst();	0		0
for (int i = 0 ; i < c.events.size ; i++)	1	n + 1	n+1
{	1		1
Event e = c.events.retrieve();	0	n	n
if ((!events.empty()) && events.search(e)){	1	n(n)	n(n)
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1	0	, ,	0
if((events.retrieve().date.compareTo(e.date)== 0) &&	1	n(n)	n(n)
(events.retrieve().time.compareTo(e.time)==0)	1		1
&& (events.retrieve().location.compareTo(e.location)==0) &&	1		1
(events.retrieve().EventType== e.EventType))	1		1
{	0		0
Event updateEvent = events.retrieve();// the event that was found with the same information	1	n	n
	0		0
updateEvent.removeContact(c.name);// remove contact from event	1	n	n
if (updateEvent.contactsNames.empty())	1	n	n
{	0		0
events.remove(e);// remove event	1	n(n)	n(n)
System.out.println("Event is deleted ");	1	n	n
}	0		0
else	1		1
events.update(updateEvent);	1	n	n
}}	0		0
c.events.findNext();	1	n	n
}	0	["	0
}	0		0
System.out.println("Contact Deleted Successfully!");	1	1	1
System.out.printing Contact Deleted Successibility: 7, System.out.printing Contact Deleted Successibility: 7,	1	1	1
}else	1	-	1
System.out.println("Contact not found!");	1	1	1
System.out.printing contact not found:],	0	1	0
J	0	+	
			2n^2 +8n+1
Big o()			O(n^2)
U - W		ĺ	-

public static void schedule/centif	Method	S/E	Freq.	Total
System.out.println["1.event"]	public static void scheduleEvent(){	0		0
System out.prind(r) = appointment';	System.out.println("Enter type:");			
	System.out.println("1. event");			
int choice4 = input.nextint(1); String type(Choice; Contact < = new Contact(2); Event = = new Event(1); switch(choice4){ Case : [boolean updatedEvent = false bype(Choice = "Vennt"; syltent = new Event(1); Sylten out print("Enter event title:"); Sylten out print("Enter event title:"); Sylten out print("Enter contacts name separated by comma: "); String Title = input.nextin(2); String Title = input.nextin(2); String Title = input.nextin(3); String Title = input.nextin(4); String I mane = name split(","); If (Econtacts empty()) (System out.print("Enter event date and time (MM/DD/YYY HH.MM): "); String I mane = name split(","); String E imme = input.next(1); String I me = input.next(1); String I me = input.next(1); String E imme = input.next(1); Input.skip("\n'\n \n'\n'\n'\n2028\n2029\n2029\n2038\n2039\n2039\n2038\n2039\n2039\n2038\n2039\n2039\n2038\n2039\n2039\n2038\n2039\n2039\n2038\n2039\n2039\n2038\n2039\n2039\n2038\n2039\n2039\n2038\n2039\n2039\n2039\n2039\n2039\n2039\n2039\n2039\n2039\n2039\n2039\n2039\n2039\n2039\n2039				1
String typechoice;		1	1	1
Contact = new Contact(); Event = new Event(); switch(inholes){ case 1:{ hoolean updatedEyent = false 1 1 1 typeChoice = "Event"; System.out.print("Enter event title: "); imputs/skip("\n") [\n"\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\		1	1	1
Event e = new Eventl(): switch(holices) case 1:{		1	1	1
Switch(choice4)		1	1	1
Case 1:		1	1	1
Case 1.5				
TypeChoice = "Event"; 1				
System.out.print("Enter event title: ");	·			
Input.skip("\M [\mathbb{N} \ \mathbb{N} \ \	···			
String Title = input.nextLine(); etitle=Title; System_out.print("Enter contacts name separated by comma: "); String name = (input.nextLine()); String name = (input.nextLine()); String (inmes = name.spillt(")); if (icontacts.empty()) { System_out.print("Enter event date and time (MM/DD/YYYY HH:MM): "); String etime = input.next(); e.date = new Date(eDate); e.time = etime; System_out.print("Enter event date and time (MM/DD/YYYY HH:MM): "); String etime = input.next(); e.date = new Date(eDate); e.time = etime; System_out.print("Enter event location: "); input.skip("\n\n [\n\n\u2028\u2029\u0085]"); String locate = input.nextLine(); e.time = etime; System_out.print("Enter event location: "); input.skip("\n\n [\n\n\u2028\u2029\u0085]"); String loc = input.nextLine(); 1				1
Extitle=Title;		1	1	1
System.out.print("Enter contacts name separated by comma: ");		1	1	1
String name = (input.nextLine());	· ·	1	1	1
String Name		1	1	1
If ((contacts.empty()) {		1	1	
System.out.print("Enter event date and time (MM/DD/YYYY HH:MM): "); String eDate = input.next(); String eTime = input.next(); e.date = new Date(eDate); e.time = eTime; system.out.print("Enter event location: "); string loc = input.nextline(); 1				
String eDate input.next(); e.date = new Date(eDate); e.time = eTime; system.out.print("Enter event location: "); input.skip("\n\n\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\				
String eTime = input.next(); e.date = new Date(eDate); e.time = eTime; System.out.print("Enter event location: "); system.out.print("Enter event location: "); input.skip("\n [\n\\2028\u2029\u0085]"); 1				
e.date = new Date(eDate); e.time = eTime; System.out.print("Enter event location: "); input.skip("\r\n \n\r\u2028\u2029\u0085]"); String loc = input.nextLine(); e.location = loc; for(int i = 0; i <names.length;i++){ &&="" ((levents.empty())="" (c.addevent(e))="" (events.<="" (events.retrieve().date.compareto(e.location)="=0)" (events.retrieve().trine.compareto(e.time)="=0)&&" c="contacts.retrieve();" c.name="names[i].trim();" contacts.findkey(c.name))="" events.search(e)="" if="" if(lcontacts.empty()="" td="" {=""><td>9</td><td>1</td><td>1</td><td>1</td></names.length;i++){>	9	1	1	1
e.time = eTime; System.out.print("Enter event location: "); input.skip("\n\n\(\n\corr \u2028\u2029\u0085]"); String loc = input.nextLine(); e.location = loc; 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		1	1	1
System.out.print("Enter event location: "); input.skip("\r\n {\n\r\u2028\u2029\u0085]"); 1		1	1	1
String loc = input.skip("\r\n \[\n\r\u2028\u2029\u0085]");		1	1	1
String loc = input.nextLine();		1	1	
String loc = input.nextLine(); 1 1 1 e.location = loc; 1 1 1 c = contacts.retrieve(); 1 1 1 for(int i = 0; i <names.length;i++){< td=""> 1 n+1 1 c.name = names[i].trim(); 1 1 1 if(!contacts.empty() && contacts.findKey(c.name)) { 1 1 1 c = contacts.retrieve(); 1 1 1 if (.addEvent(e)) { 1 1 1 contacts.update(c.name,c); 1 1 1 if (!levents.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) 1 1 && (events.retrieve().time.compareTo(e.time)==0)&&(events.retrieve().location.compareTo(e.location)==0) 1 1 1 Event eventFound.contactsNames.insertSort(c.name); 1 1 1 1 eventFound.contactsNames.insertSort(c.name); 1 1 1 1</names.length;i++){<>				
e.location = loc; c = contacts.retrieve(); for(int i = 0; i <names.length;i++){ &&="" ((!events.empty())="" ((.addevent(e))="" (c.addevent(e))="" (events.retrieve().date.compareto(e.date)="=0)" (events.retrieve().eventtype="=" (events.retrieve().location.compareto(e.location)="=0)" (events.retrieve().time.compareto(e.time)="=0)&&" 1="" 1<="" c="contacts.retrieve();" c.name="names[i].trim();" contacts.findkey(c.name))="" contacts.update(c.name,c);="" e.eventtype))="" event="" eventfound="events.retrieve();" events.search(e)="" if="" if(!contacts.empty()="" td="" {=""><td>String loc - input novtling():</td><td></td><td></td><td></td></names.length;i++){>	String loc - input novtling():			
e.location = loc; c = contacts.retrieve(); 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	String lot - input.frextellie(),			
c = contacts.retrieve(); 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	e location – loc			
C = contacts.retrieve();	e.iocation – ioc,	1	1	1
for(int i = 0 ; i < names.length; i++){ c.name = names[i].trim(); if(lcontacts.empty() && contacts.findKey(c.name)) { c = contacts.retrieve(); if (c.addEvent(e)) { contacts.update(c.name,c); if ((levents.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) && (events.retrieve().EventType== e.EventType)) { Event eventFound = events.retrieve(); eventFound.contactsNames.insertSort(c.name); 1	c = contacts ratriova():	1	1	1
c.name = names[i].trim(); if(!contacts.empty() && contacts.findKey(c.name)) { c = contacts.retrieve(); if (c.addEvent(e)) { contacts.update(c.name,c); if ((!events.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) && (events.retrieve().time.compareTo(e.time)==0)&& (events.retrieve().location.compareTo(e.location)==0) && (events.retrieve().EventType== e.EventType)) { Event eventFound = events.retrieve(); eventFound.contactsNames.insertSort(c.name); 1	c - contacts.retrieve(),	1	1	1
c.name = names[i].trim(); if(!contacts.empty() && contacts.findKey(c.name)) { c = contacts.retrieve(); if (c.addEvent(e)) { contacts.update(c.name,c); if ((!events.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) && (events.retrieve().time.compareTo(e.time)==0)&& (events.retrieve().location.compareTo(e.location)==0) && (events.retrieve().EventType== e.EventType)) { Event eventFound = events.retrieve(); eventFound.contactsNames.insertSort(c.name); 1	for(int i =0 · i <names <="" longth:i++)="" td=""><td>1</td><td>n+1</td><td>1</td></names>	1	n+1	1
c.name = names[i].trim(); 1 1 1 if(!contacts.empty() && contacts.findKey(c.name)) 1 1 1 c = contacts.retrieve(); 1 1 1 if (c.addEvent(e)) 1 1 1 contacts.update(c.name,c); n 1 1 if (!levents.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) 1 1 1 && (events.retrieve().time.compareTo(e.time)==0)&&(events.retrieve().location.compareTo(e.location)==0) 1 1 1 && (events.retrieve().EventType== e.EventType)) { 1 1 1 1 Event eventFound = events.retrieve(); 1 1 1 1 eventFound.contactsNames.insertSort(c.name); 1 1 1 1	ioi(iiit i =0 , ixiidilles.ieiigui,i++)(_		-
if(!contacts.empty() && contacts.findKey(c.name)) { c = contacts.retrieve(); if (c.addEvent(e)) { contacts.update(c.name,c); if (!events.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) && (events.retrieve().EventType== e.EventType)) { Event eventFound = events.retrieve(); eventFound.contactsNames.insertSort(c.name);	c name = names[i] trim().	1		1
if(!contacts.empty() && contacts.findKey(c.name)) 1 1 1 c = contacts.retrieve(); 1 1 1 if (c.addEvent(e)) 1 1 1 if (c.addEvent(e)) 1 1 1 contacts.update(c.name,c); n 1 1 if ((!events.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) 1 1 && (events.retrieve().time.compareTo(e.time)==0)&&(events.retrieve().location.compareTo(e.location)==0) 1 1 && (events.retrieve().EventType== e.EventType)) { 1 1 1 Event eventFound = events.retrieve(); 1 1 1 eventFound.contactsNames.insertSort(c.name); 1 1 1	C.Hame – Hames[ij.dim(),			
c = contacts.retrieve(); if (c.addEvent(e)) { contacts.update(c.name,c); if ((levents.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) && (events.retrieve().time.compareTo(e.time)==0)&&(events.retrieve().location.compareTo(e.location)==0) && (events.retrieve().EventType== e.EventType)) { Event eventFound = events.retrieve(); eventFound.contactsNames.insertSort(c.name); 1	if(contacts ampty() 8.8, contacts find(outs name))			1
c = contacts.retrieve(); if (c.addEvent(e)) { contacts.update(c.name,c); if ((levents.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) && (events.retrieve().time.compareTo(e.time)==0)&&(events.retrieve().location.compareTo(e.location)==0) && (events.retrieve().EventType== e.EventType)) { Event eventFound = events.retrieve(); eventFound.contactsNames.insertSort(c.name); 1 1 1 1 1 1 1 1 1 1 1 1 1	intermetis.empty() && contacts.inukey(c.name))	1	1	1
if (c.addEvent(e)) { contacts.update(c.name,c); if ((levents.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) && (events.retrieve().time.compareTo(e.time)==0)&&(events.retrieve().location.compareTo(e.location)==0) && (events.retrieve().EventType== e.EventType)) { Event eventFound = events.retrieve(); eventFound.contactsNames.insertSort(c.name); 1 1 1 1 1 1 1 1 1 1 1 1 1	c = contacts retrieve():	1	1	1
contacts.update(c.name,c); if ((!events.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) && (events.retrieve().time.compareTo(e.time)==0)&&(events.retrieve().location.compareTo(e.location)==0) && (events.retrieve().EventType== e.EventType)) { Event eventFound = events.retrieve(); eventFound.contactsNames.insertSort(c.name); 1 1 1 1 1 1 1 1 1 1 1 1 1	c – contacts.retrieve(),	1		1
contacts.update(c.name,c); if ((!events.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) && (events.retrieve().time.compareTo(e.time)==0)&&(events.retrieve().location.compareTo(e.location)==0) && (events.retrieve().EventType== e.EventType)) { Event eventFound = events.retrieve(); eventFound.contactsNames.insertSort(c.name); 1 1 1 1 1 1 1 1 1 1 1 1 1	if (c addEvent(e))		1	
contacts.update(c.name,c); if ((levents.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) && (events.retrieve().time.compareTo(e.time)==0)&&(events.retrieve().location.compareTo(e.location)==0) && (events.retrieve().EventType== e.EventType)) { Event eventFound = events.retrieve(); eventFound.contactsNames.insertSort(c.name); 1 1 1 1 1 1 1 1 1 1 1 1 1	ii (c.audzveiit(e)) \	1		1
if ((levents.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) && (events.retrieve().time.compareTo(e.time)==0)&&(events.retrieve().location.compareTo(e.location)==0) && (events.retrieve().EventType== e.EventType)) { Event eventFound = events.retrieve(); eventFound.contactsNames.insertSort(c.name); 1 1 1 1 1 1 1 1 1 1 1 1 1	contacts undatals name sly	1	n .	1
if ((levents.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0) && (events.retrieve().time.compareTo(e.time)==0)&&(events.retrieve().location.compareTo(e.location)==0) && (events.retrieve().EventType== e.EventType)) {	contacts.upuate(c.name,c),		11	
&& (events.retrieve().time.compareTo(e.time)==0)&&(events.retrieve().location.compareTo(e.location)==0) && (events.retrieve().EventType== e.EventType)) {	if / (loyonts ampty/)) 8.8 ayonts search(a) 8.8 (ayonts ratrioval) data comparaTo(a data)==0)	1		1
&& (events.retrieve().EventType== e.EventType)) { 1 1 1 Event eventFound = events.retrieve(); 1 1 1 eventFound.contactsNames.insertSort(c.name); 1 1 1 1 1 1 1			1	
1			1	
eventFound.contactsNames.insertSort(c.name); 1 1 1 eventFound.contactsNames.insertSort(c.name); 1 1 1	αα (events, retrieve(), ενεπειγρε ε.ενεπειγρε)) (1	1	1
eventFound.contactsNames.insertSort(c.name); 1 1 1 eventFound.contactsNames.insertSort(c.name); 1 1 1	Front eventFound - events retrieve():	1	1	
eventFound.contactsNames.insertSort(c.name); 1 1 1	Event eventi ound – events.retrieve(),			
	eventFound contactsNames insertSert(c name)			
	eventround.contactsnames.insertsort(c.name); events.update(eventFound);			
undated Front - true 3	, , , , , , , , , , , , , , , , , , , ,			
if (JundatedEvent) {		1		1
e.contactsNames.insertSort(c.name);		1	1	1
events.insertSort(e); }		1		1
System.out.println("Event scheduled successfully! " + c.name); } 1		1		1

else			
System.out.println("You have a conflict Event "+ c.name);	4		
} else	1	1	1
System.out.println("contact not found " + c.name) }}	1	1	1
}break;	1	1	1
case 2:	1	1	1
typeChoice = "Appoinment";	1	1	1
e.EventType = false;	1	1	1
System.out.print("Enter appoinment title: ");	1	1	1
input.nextLine();			
e.title = input.nextLine();	1	1	1
System.out.print("Enter contact name: ");	1	1	1
c.name = input.nextLine();	1	1	1
if (!contacts.empty() && contacts.findKey(c.name) == true) {	1	n	n
c = contacts.retrieve();	1	1	1
System.out.print("Enter appoinment date and time (MM/DD/YYYY HH:MM): ");	1	1	1
e.date = new Date (input.next())	1	1	1
e.time = input.next();			
System.out.print("Enter appoinment location: ");	1	1	1
input.nextLine();	1	1	1
e.location = input.nextLine();	1	1	1
if ((!events.empty()) && events.search(e) && (events.retrieve().date.compareTo(e.date)==0)&&	1	n	n
(events.retrieve().time.compareTo(e.time)==0) && (events.retrieve().location.compareTo(e.location)==0	1	1	1
&& (events.retrieve().EventType== e.EventType)) {	1	1	1
System.out.println("Appointment had been scheduled previously, could not add more contacts,	1		
try again "); }		1	1
u y ugum //)	1	1	1
Else	1	1	1
Lise	1	1	1
if (c.addEvent(e) {			
11 (0.0002-0.11(0))	1	1	1
contacts.update(c.name,c);	-	1	_
contacts.aparte(e.name,e),			
e.contactsNames.insertSort(c.name);			
e.contactsivames.insertsort(c.name),	1	1	1
events.insertSort(e);	1	1	1
events.insertsuride),	1	1	1
System.out.println("The Appoinment is scheduled successfully! "); }	1	1	1
System.out.printing The Appoinment is scheduled successibility: 7, 3		_	
alsa	1	1	1
else	1	1	1
Contain and uninter/like and in a sufficient format (Augustus and L. III).			
System.out.println("there is a conflict Event/Appoinment! "); } }	1	1	1
also	1	1	1
else			1
Contains and printle (II Contact and formed III)	1	1	1
System.out.println("Cantcat not found !"); } }	1	1	1
	<u> </u>	1	
			4n
			1
			1
Big o()			O(n)
		l	L

Method	S/E	Freq.	Total
public static void printEvent(){	0	-	
System.out.println("Enter search criteria:");	1 1	1	1 1
System.out.println("1.contact name");	1	1	1
System.out.println("2.Event title");	1		1
System.out.println("\nEnter your choice: ");		1	
int choice3 = input.nextInt()	1 1	1	1 1
if(!events.empty()){	1	1	1
switch (choice3) {			
case 1 {	1	1	1
Contact c = new Contact();	1	1	1 1
System.out.print("Enter the contact name:");	1	1	1
input.skip("\r\n [\n\r\u2028\u2029\u0085]");	1	_	1
c.name = (input.nextLine())			
if (!contacts.empty() {	1	1	1
if (contacts.findKey(c.name) == true) {	1 1	n 1	n 1
System.out.println("Contact found!");	1 1	1	1 1
c = contacts.retrieve();			
c.events.findFirst();			
for (int i = 0 ; i < c.events.size ; i++) {	1	n+1 1	n+1
Event e = c.events.retrieve();	1 1	1	1
if (!events.empty() && events.search(e))	1 1	n	1 n
System.out.println(events.retrieve());			
c.events.findNext(); }			
if (c.events.empty())			
System.out.println("There is no events for this contact"); }	1 1	1	1 1
else	1 1	1	1 1
System.out.println("Contact not found!"); }	1	1	1
else			

System.out.println("Contact not found!");			
	1	1	1
} break			
	1	1	1
case 2 {	1	1	1
Frank a manu Frank().			
Event e = new Event();	1	1	1
System.out.print("Enter the event title:");	1	1	1
System outprint (Enter the event time.),	1	1	1
input.skip("\r\n [\n\r\u2028\u2029\u0085]");	1	1	1
e.title = input.nextLine();			
Citité - inputitionettine(),	1	n	n
if (!events.empty()&&events.search(e)) {		1	1
	1	1	1
System.out.println("Event found!");	1		
System.out.println(events.retrieve()); }			
else		1	1
	1	1 1	1 1
System.out.println("Event not found!") }	1	1	1
	1	_	1
break; }			
}else		1	1
	1	1 1	1 1
System.out.println("There are no events."); }	1	1	1
	1		
			3n
			3.7
Big o()			
			O(n)

Method	S/E	Freq.	Total
<pre>public static void printContactsFirstName(){</pre>	0	-	-
System.out.print("Enter the first name:");	1	1	1
String fname = input.nextLine();	1	1	1
if (contacts.empty())	1	1	1
System.out.println("There is no Contacts found");	1	1	1
else	1	1	1
contacts.SearchSameFirstName(fname);	1	1	1
}	0	-	-
			6
Big o()			O(1)

Method	S/E	Freq.	Total
public static void printAllEvents(){	0	-	-
if(!events.empty())	1	1	1
System.out.println(events.toString());	1	n	1
else	1	1	1
System.out.println("no events found!"); }	1	1	1
}	0	-	-
			n+4
Big o()			O(n)