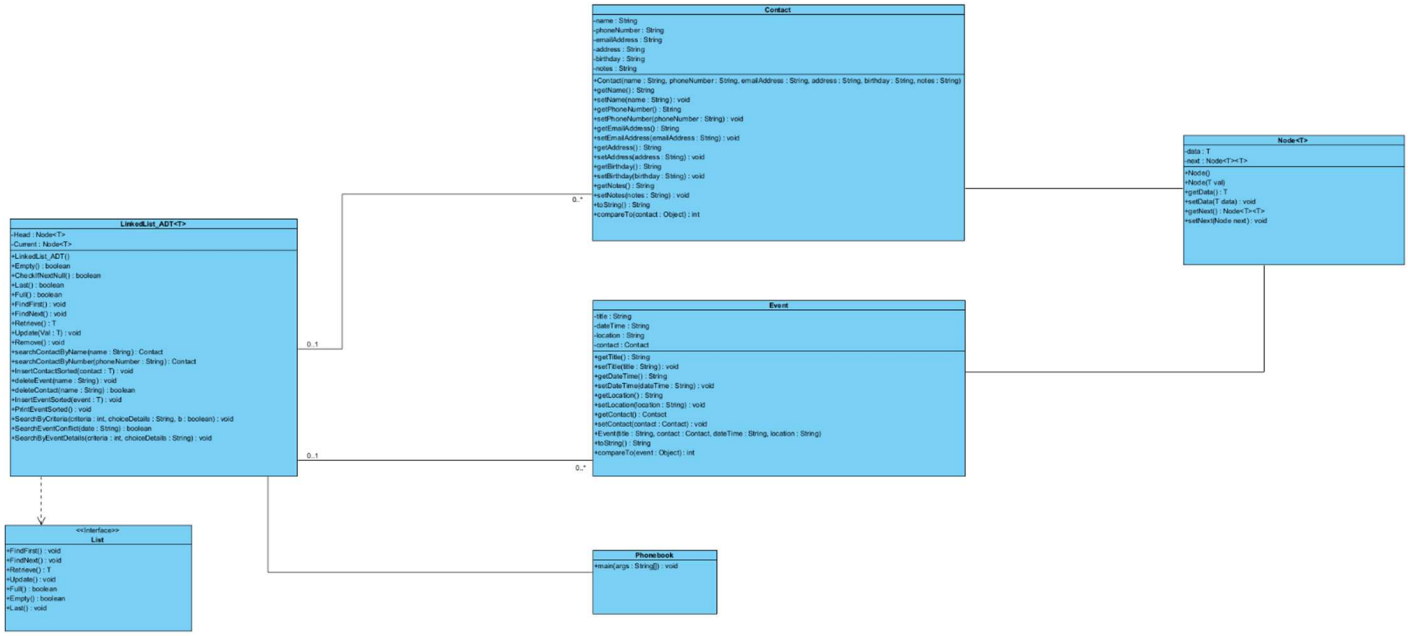
- **UML Class diagram:**



Contact class***:***

- Attributes:
  - ***name***: the name of the contact.
  - ***phoneNumber***: the phone number of the contact.
  - ***emailAddress***: the email address of the contact.
  - ***address*** : the address of the contact.
  - ***birthday***: the birthday of the contact.
  - ***notes***: the notes associated with the contact.
- Methods:
  - ***Contact(name:String, phoneNumber:String, emailAddress:String, address:String, birthday:String, notes:String)*:** constructor with BigO(1) because constructors typically have a time complexity of O(1) because they involve a constant number of operations regardless of the size or complexity.
  - ***toString***():Returns a formatted string representation of the contact.
  - ***compareTo***(Object event):Compares two Contact objects based on their names in a case-insensitive manner and Returns a negative integer, zero, or a positive integer if the name of this contact is less than, equal to, or greater than the title of the specified event, respectively

- ***Getters/Setters*** BigO(1) because Getters and Setters typically have a time complexity of O(1) because they involve a constant number of operations regardless of the size or complexity.

Node<T> class***:***

- Attributes:
  - ***data***: the data stored in the node.
  - ***next***: the next node in the linked list.
- Methods:
  - ***Contact(name:String, phoneNumber:String, emailAddress:String, address:String, birthday:String, notes:String*)**: constructor with BigO(1) because constructors typically have a time complexity of O(1) because they involve a constant number of operations regardless of the size or complexity.
  - ***Getters/Setters*** BigO(1) because Getters and Setters typically have a time complexity of O(1) because they involve a constant number of operations regardless of the size or complexity.

Phonebook class***:***

- Methods:
  - ***main(String[] args):*** main method to let user interact with the phonebook

Event class***:***

- Attributes:
  - ***title***: the id of the contract.
  - ***dateTime:*** the weight of the contract.
  - ***location:*** *the location* of the contract.
  - ***contact:*** *the contact associated with the event.*
- Methods:
  - **Event(String title, Contact contact, String dateTime, String location):** constructor
  - ***Getters/Setters***
  - **toString:** return a formatted string representation of the contact. Big O(1)

- **compareTo(Object event)** : Compares two Event objects based on their titles in a case-insensitive manner and Returns a negative integer, zero, or a positive integer if the title of this event is less than, equal to, or greater than the title of the specified event, respectively

List Interface*:*

- Methods:
  - **findFirst( ):**
  - **findNext( ):**
  - **retrieve( ):**
  - **update(T t):**
  - **full( ):**
  - **empty( ).**
  - **last().**

LinkedList_ADT<T> class*:*

- Attributes:
  - ***head***: pointer for first element in list.
  - ***current:*** pointer for the current element in list.
- Methods:
  - ***LinkedList_ADT()*****:** constructor with BigO(1) because constructors typically have a time complexity of O(1) because they involve a constant number of operations regardless of the size or complexity.
  - ***empty():*** Check whether the list is empty or not , BigO(1) because it will return one Boolean value and it involve a constant number of operations .
  - ***checkIfNull():*** Check whether the current is null or not , BigO(1) because it will return one Boolean value and it involve a constant number of operations .
  - ***last():*** Check whether the current is in the last of the list or not, BigO(1) because it will return one Boolean value and it involve a constant number of operations .
  - ***full():*** check whether the list is full or not, BigO(1) because it will involve one operation which is returning false.

- ***findFirst():*** put current in the first element on the list, BigO(1) because it involve a constant number of operations.
- ***findNext():*** put current in the next element of the current ,BigO(1) because it involve a constant number of operations.
- ***retrieve():*** return the data of the Node , BigO(1) because it involve a constant number of operations.
- ***update(T Val):*** Update data in the given Node, BigO(1) because it involve a constant number of operations.
- ***searchContactByName(String name):*** This method searches for a contact in the linked list by name. It takes the name of the contact as input and returns the corresponding '**Contact**' object if found, or '**null**' if not found, BigO(n) because n will represent the number of elements(contacts) and it will iterates the linked list for n time.
- ***searchContactByNumber(String phoneNumber):*** This method searches for a contact in the linked list by phone number. It takes String (phone number) as input and returns the corresponding 'Contact' object if found, or 'null' if not found, BigO(n) because n will represent the number of elements(contacts) and it will iterates the linked list for n time.
- ***insertContactSorted(T contact):*** This method inserts a new contact into the linked list in a sorted order based on the contact's name. It takes a new contact of type 'T'(Generic) as input, checks if a contact with the same name or phone number already exists, and inserts the new contact at the appropriate position. If the contact is successfully inserted, it prints a success message. This method has no output (void), BigO(n) because n will represent the number of elements(contacts) and it will iterates the linked list for n time.

- ***deleteEvent(String name):*** This method checks if the event list is empty. If it is empty, it prints a message and returns. If the list is not empty, it iterates through the linked list of events and compares the name of the contact associated with each event to the given String (name). If a match is found, the event is deleted from the list. If no events are found for the given contact name, appropriate messages are printed, BigO(n) because n will represent the number of elements(events) and it will iterates the linked list for n time.
- ***deleteContact(String name):*** This method checks if the contact list is empty. If it is empty, it prints a message and returns 'false'. If the list is not empty, it iterates through the linked list of contacts and compares the name of each contact to the given String (name). If a match is found, the contact is deleted from the list. If no contact is found with the given String (name), appropriate messages are printed. The method returns 'true' if the contact is successfully deleted, and 'false'

otherwise, BigO(n) because n will represent the number of elements(contacts) and it will iterates the linked list for n time.

- ***insertEventSorted(T event):*** This method creates a new node with the given event. If the event list is empty or the given event title compared to the first event title in the list is less than zero, the new node becomes the new head. Otherwise, the method iterates through the linked list to find the appropriate position to insert the event while maintaining the sorted order. Once the position is found, the new node is inserted into the list. A success message is printed, BigO(n) because n will represent the number of elements(events) and it will iterates the linked list for n time.
- ***printEventSorted():*** This method checks if the event list is empty. If it is empty, it prints a message and returns. If the list is not empty, it iterates through the linked list of events and prints each event. After printing each event, a separator line is printed, BigO(n) because n will represent the number of elements(event) and it will iterates the linked list for n time.
- ***searchByCriteria(int criteria,String choiceDetails, Booleanb):*** This method checks if the current contact matches the specified criteria and choice details.by using switch, in every case from 1 to 5 It compares the details of the current contact with the given choice details based on the specified criteria (name, phone number, email address, address, and birthday). If a match is found, it prints a message and returns 'true'. If no match is found. If none of the cases match, it returns 'false', BigO(n) because n will represent the number of elements(contacts) and it will iterates the linked list for n time.
- ***searchEventConflict(String date):*** This method checks if the event list is empty. If it is empty, it returns 'false'. If the list is not empty, it iterates through the event list and compares the date of each event with the given date. If a conflict is found (i.e., an event with the same date exists), it returns 'true'. If no conflicts are found, it continues to the next event until the end of the list is reached. It returns 'false' if no conflicts are found, BigO(n) because n will represent the number of elements(event) and it will iterates the linked list for n time.
- ***searchByEventDetails(int criteria, String choiceDetails):*** The method checks if the event list is empty. If it is empty, it prints a message and returns. If the list is not empty, it searches for events (by using switch) that match the given criteria and choice details. It iterates through the event list, comparing the specified criteria (contact name, event title) with the corresponding details of each event. If a match is found, the event is printed. The method prints a separator line before returning, BigO(n) because n will represent the number of elements(event) and it will iterates the linked list for n time.

CONCLUSION

In this phase, we successfully designed and implemented a phonebook tree using an ADT List data structure. The phonebook tree provides a comprehensive set of functionalities for storing, searching, deleting contacts, as well as storing, searching, deleting, and printing events.