| | |
|---|---|
| **Course Name** | Algorithms and Data Structures |
| **Course Code** | SFT206 |
| **Instructor** | Dr. Wafaa Samy |
| **Project Name** | Contacts Management Application. (Using Doubly Linked List) |

| **Student Name** | **ID** |
|---|---|
| Mohamed Altaf Abd El Malek | 19-00077 |
| Mohamed Helmy Mahmoud | 19-00402 |
| Rania Refaat Abd El Rahman | 19-00346 |
| Hifzy Atef Mosa | 18-00023 |

## Table of contents

## Tools:-

1-Jdk

2-netBeans

## Introduction

This project can demonstrate the working of contact book applications and also teach you about data structures and algorithms. Typically, phone book management includes the following operations:

- Inserting
- Updating
- Searching (by contact name, and by phone number)
- Sorting
- Deleting

**1-Inserting:-**

The user is allowed to enter his phone number and name .

**2-Deleting:-**

User is allowed to delete contact and phone number

**3-Updating:-**

The user is allowed to make changes to the number or contact that was entered.

Extra features of the search queries:

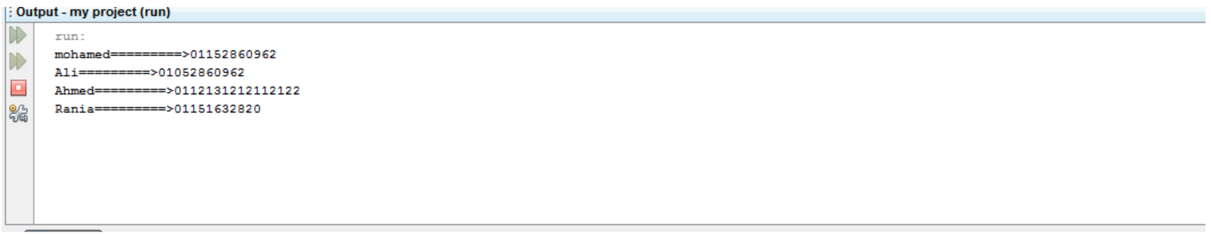
- ☒ The user can see suggestions from the contact list after entering each character.
- ☒ The user can search with “starts with”, “any part of” or “whole words only” of the contact name or phone number.

## Inserting:-

function

```
public void insert(phone data)
{
        Node newNode = new Node();
        newNode.data = data;
        newNode.next = null;
        newNode.prev=tail;
        if(tail!=null)
            tail.next=newNode;
        tail = newNode;
        if(head==null)
            head=newNode;
        size++;
}
```

Output:



Output - my project (run)

```
run:
mohamed==========>01152860962
Ali==========>01052860962
Ahmed==========>0112131212112122
Rania==========>01151632820
```

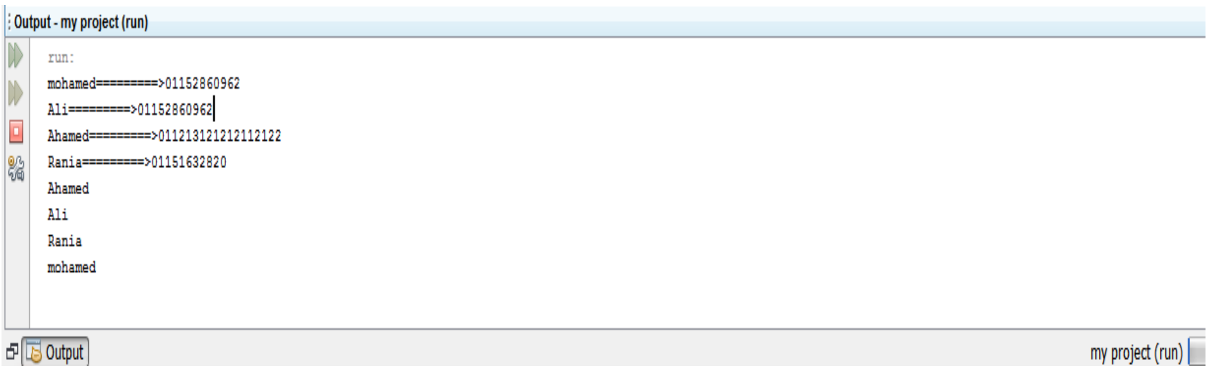
Through this function, you can add a number or a name to add it to the contact, through a list that appears for him to enter the name and then the number and then press OK.

## Sorting:-

- function

```java
    public String[] sort()
    {
        int i = 0;
        Node current = head;
        while(current != null)
        {
            current = current.next;
            i++;
        }
        String[] x = new String [i];
        current = head;
        int j = 0;
        while(current != null)
        {
            x[j++] = current.data.getName();
            current = current.next;
        }
        return x;
    }
}
```

- Output:

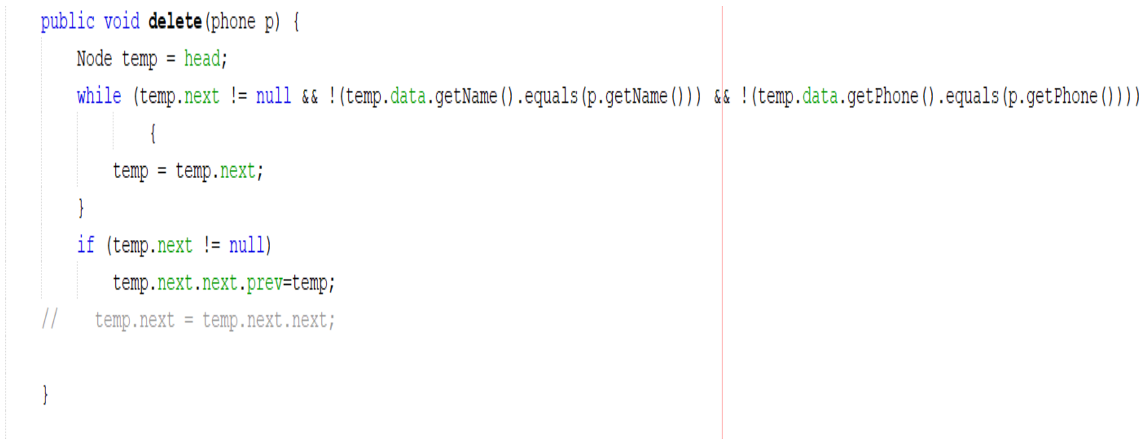


```
run:
mohamed=========>01152860962
Ali=========>01152860962
Ahamed=========>011213121212112122
Rania=========>01151632820
Ahamed
Ali
Rania
mohamed
```

This function arranges all the names in the contacts by alphabetical letters .

## Deleting:

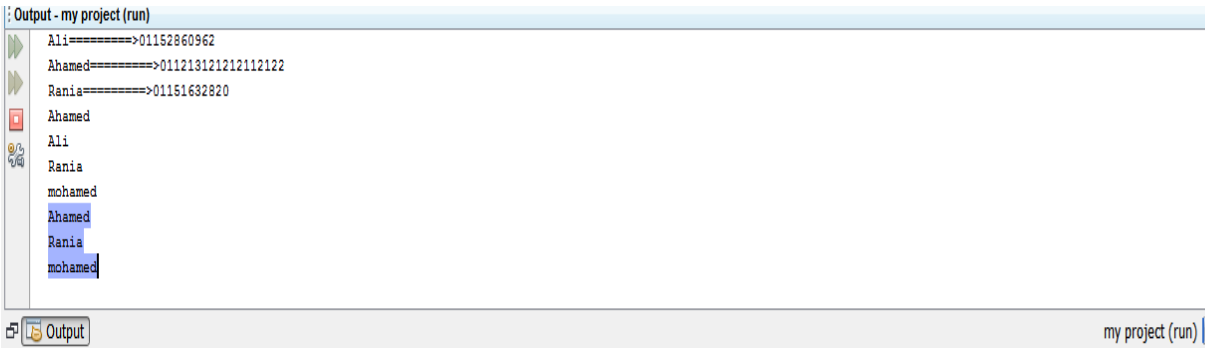
Function:



```
public void delete(phone p) {
    Node temp = head;
    while (temp.next != null && !(temp.data.getName().equals(p.getName())) && !(temp.data.getPhone().equals(p.getPhone())))
        {
        temp = temp.next;
    }
    if (temp.next != null)
        temp.next.next.prev=temp;
//    temp.next = temp.next.next;

}
```

Output:



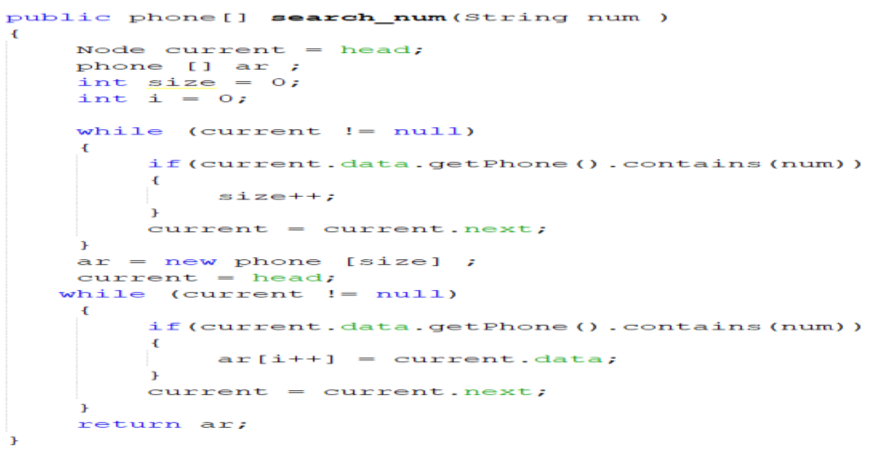
Output - my project (run)

```
Ali=========>01152860962
Ahamed=========>011213121212112122
Rania=========>01151632820
Ahamed
Ali
Rania
mohamed
Ahamed
Rania
mohamed
```

Through this function, it is possible to delete any contacts from the beginning of the list or at the end of the list and any place in the list.

## Searching (by contact number)

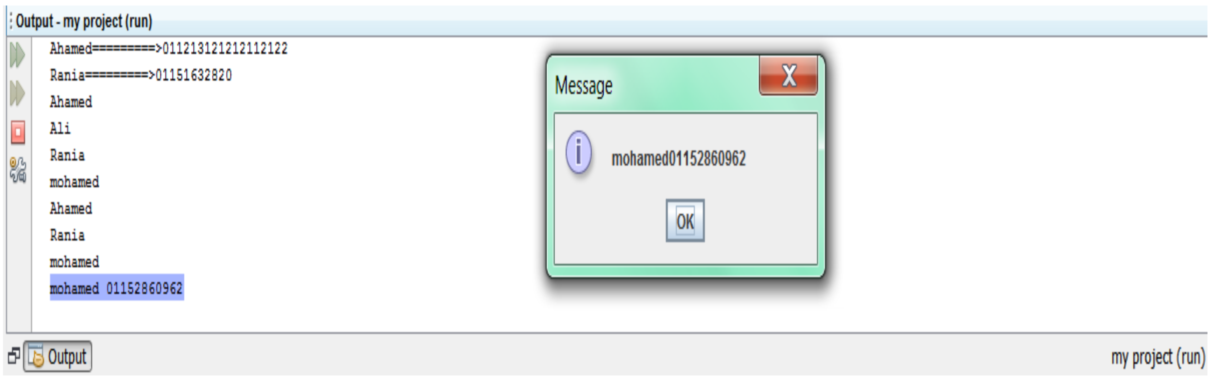
Function:



```
public phone[] search_num(String num )
{
      Node current = head;
      phone [] ar ;
      int size = 0;
      int i = 0;

      while (current != null)
      {
            if(current.data.getPhone().contains(num))
            {
                  size++;
            }
            current = current.next;
      }
      ar = new phone [size] ;
      current = head;
    while (current != null)
      {
            if(current.data.getPhone().contains(num))
            {
                  ar[i++] = current.data;
            }
            current = current.next;
      }
      return ar;
}
```

Output:



Output - my project (run)

```
Ahamed=========>011213121212112122
Rania=========>01151632820
Ahamed
Ali
Rania
mohamed
Ahamed
Rania
mohamed
mohamed 01152860962
```

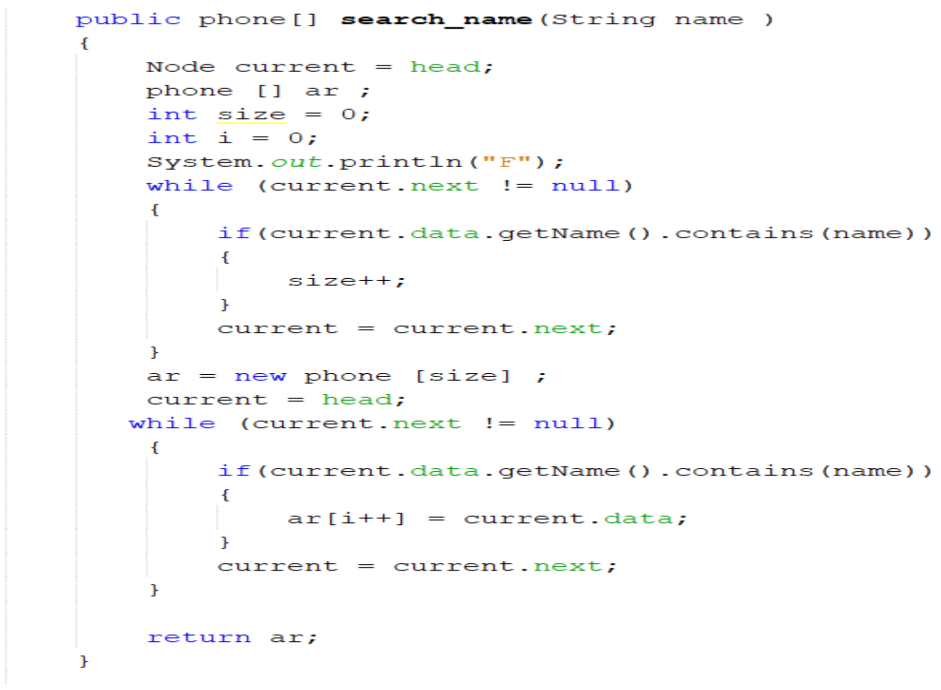
Message

mohamed01152860962

Through this function, the user can search for the contact that he entered, and he can search by the phone number.

## Searching (by contact name)

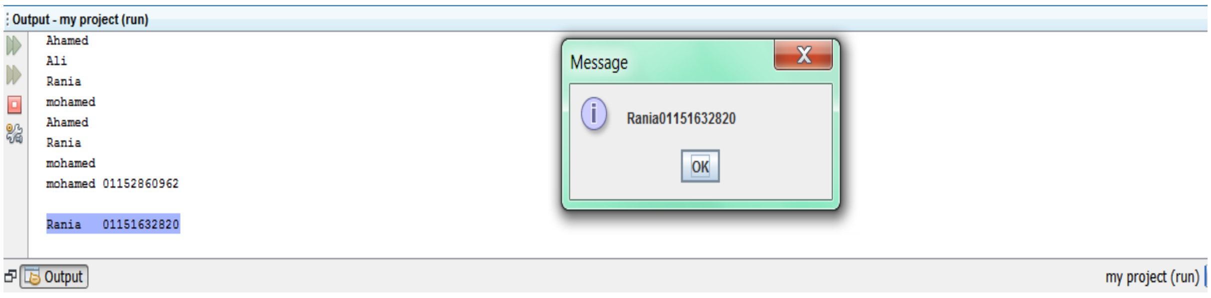
### Function:



```
public phone[]  search_name(String name )
{
     Node current = head;
     phone [] ar ;
     int size = 0;
     int i = 0;
     System.out.println("F");
     while (current.next != null)
     {
          if(current.data.getName().contains(name))
          {
               size++;
          }
          current = current.next;
     }
     ar = new phone [size] ;
     current = head;
    while (current.next != null)
     {
          if(current.data.getName().contains(name))
          {
               ar[i++] = current.data;
          }
          current = current.next;
     }

     return ar;
}
```

### output



```
Output - my project (run)
Ahamed
Ali
Rania
mohamed
Ahamed
Rania
mohamed
mohamed 01152860962

Rania   01151632820
```

```
Message
Rania01151632820
OK
```

Through this function, the user can search for the contact that he entered, and he can search by name.

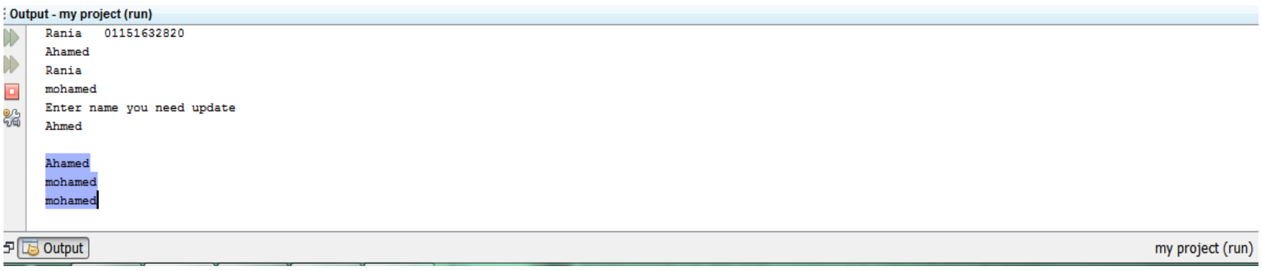
## Updating:

### Function:

```
public void update (String name , phone new_data)
{
    Node current = head;
    while (current.next != null)
    {
        if (name.equals(current.data.getName()))
        {
            break;
        }
        current = current.next;
    }
    current.data.setName (new_data.getName());
    current.data.setPhone(new_data.getPhone());

}
```

### Output



Output - my project (run)

```
Rania    01151632820
Ahamed
Rania
mohamed
Enter name you need update
Ahmed

Ahamed
mohamed
mohamed
```

Through this function, the user can modify the person's name in case there is a mistake in the name. It can also modify the phone number with this function in case the phone number is wrong .