LinkedList

```
public void add(int index, E item) //Inserts itemect item into the list at position index
{
     if (index < 0 || index > size) {
         throw new IndexOutOfBoundsException(Integer.toString(index));
     }
     if (index == 0) {addFirst(item);}
     else if(index == size) {addLast(item);}
     else {
         Node<E> current =getNode(index-1);
         Node<E> newNode = new Node<>(item);
         newNode.next=current.next;
         current.next=newNode;
         newNode.prev=current;
         if(newNode.next!=null)
             newNode.next.prev=newNode;
         size++;
     }
}
```

The getnode method takes n time in the worst case, adding elements before and after is a fixed time.

$T_B = \theta(1)$

$T_W = \boldsymbol{\theta}(n)$

T(n)=O(n)

```
public void addFirst(E item) //Inserts itemect item as the first element of the list
{
    Node<E> newNode = new Node<>(item);
    if (isEmpty()) {
       addInitial(newNode);
    } else {
        head.prev = newNode;
        newNode.next = head;
        newNode.prev = null;
        head = newNode;

        size++;
    }
}
```

T(n)= $\boldsymbol{\theta}$(1)

```java
public void addLast(E item) //Adds itemect item to the end of the list
{

    Node<E> newNode = new Node<>(item);
     if (isEmpty()) {
        addInitial(newNode);
     }else {
        tail.next = newNode;
        newNode.prev = tail;
        tail = newNode;
        size++;
     }


}
```

T(n)= θ(1)

```java
private Node<E> getNode(int index) {
     Node<E> node = head;
     for (int i = 0; i < index && node != null; i++) {
         node = node.next;
     }
     return node;
}
```

İndex = n

T(n)= θ(n)

```java
public E get(int index) //Returns the item at position index
{
    if (index < 0 || index >= size)
        throw new IndexOutOfBoundsException(Integer.toString(index));

    if(index==0) getFirst();
    if(index==size-1) getLast();

   Node<E> node = getNode(index);
   return node.data;
}
```

getNode= θ(n)

$T_B = \theta(1)$

$T_W = \theta(n)$

T(n)=O(n)

```
public boolean remove(E item) //Removes the first occurrence of it
{
     Node<E> current = head;

     if(isEmpty())
            return false;

     if (head.data.equals(item) && tail.data.equals(item))
    {
        head=null;
        tail=null;
        size--;
        return true;
    }

    else if(head.data.equals(item))
    {
        removeFirst();
    }

    else if(tail.data.equals(item))
    {

        removeLast();
    }
    else {
         for(int i=0;i<size;i++) {
             if(current.data.equals(item))
                 break;
             current = current.next;
         }

        current.prev.next = current.next;
        if (current.next != null)
        {
            current.next.prev = current.prev;
        }
        current = null;
        size--;
    }
    return true;
```

remoFirst= $\theta(1)$ **removeLast=** $\theta(1)$ **so** $T_B = \theta(1)$

other remove = $\theta(n)$

$T_W = \theta(n)$ so that **T(n)=O(n)**

```
private boolean removeLast()
{
     tail = tail.prev;
     tail.next = null;
     size--;
     return true;
}
/**
 *
 * @return the first element from this list
 */
private boolean removeFirst()
{

    head = head.next;
    head.prev = null;
    size--;
    return true;
}
```

T(n) = $\boldsymbol{\theta}$(1) for two method

```
public KWListIter(int i)
{   if (i < 0 || i > size)
    {     throw new IndexOutOfBoundsException("Invalid index " + i); }
    lastItemReturned = null;

    if (i == size)
    {     index = size;     nextItem = null;      }
    else
    {   nextItem = head;
        for (index = 0; index < i; index++)  nextItem = nextItem.next;
    }
}
```

T_B = $\boldsymbol{\theta}$(1) T_W = **$\boldsymbol{\theta}$(n)**

T(n)=O(n)

```
public E next()
{   if (!hasNext()) {  throw new NoSuchElementException();   }
    lastItemReturned = nextItem;
    nextItem = nextItem.next;
    index++;
    return lastItemReturned.data;
}
/**
 * @return the previous element in the list
 * @throws if the iteration has no previous element
 */
public E previous()
{   if (!hasPrevious()) {   throw new NoSuchElementException();   }
    if (nextItem == null) { nextItem = tail;  }
    else {  nextItem = nextItem.prev;  }
    lastItemReturned = nextItem;
    index--;
    return lastItemReturned.data;
}
```

T(n) = $\theta(1)$ for two method

ARRAYLIST

```
public void add(E anEntry)
{
    if (size >= capacity) {
        reallocate();
    }
    theData[size] = anEntry;
    size++;
}
```

T(n) = $\theta(1)$ amortized

```
public void add (int index, E anEntry) {
    // check bounds
    if (index < 0 || index > size) {
        throw new ArrayIndexOutOfBoundsException(index);
    }
    // Make sure there is room
    if (size >= capacity) {
        reallocate();
    }
    // shift data
    for (int i = size; i > index; i--) {
        theData[i] = theData[i-1];
    }
    // insert item
    theData[index] = anEntry;
    size++;
}
```

T(n)=O(n)

ADMİNİSTRATOR

```
public void setBranchEmployee(BranchEmployee newEmployee){
    dataBase.branchEmpList.add(newEmployee);
    dataBase.setbranchEmployeeSize(getbranchEmployeeSize()+1);

}
```

T(n) = $\boldsymbol{\theta}$(1) add method constant time for arraylist

```
public void removeBranchEmployee(BranchEmployee deleteBranchEmployee) throws Exception{
    int delete = 0;
    boolean control = false;
    for(int i=0;i<getbranchEmployeeSize();i++){
        if(dataBase.branchEmpList.get(i).getUsername().equals(deleteBranchEmployee.getUsername())){
            delete = i;
            control = true;
        }
    }
    if(control){
        dataBase.setbranchEmployeeSize(getbranchEmployeeSize()-1);
        dataBase.branchEmpList.remove(delete);
        System.out.println("Branch Employee successfully removed. ");
    }
    else
        throw new Exception("Error : This branch employee does not exist !!!");
}
 /**
```

employeeSize=n

get()= $\boldsymbol{\theta}(1)$

for loop $T_B = \boldsymbol{\theta}(1)$ $T_W = \boldsymbol{\theta}(n)$

remove method $T_B = \boldsymbol{\theta}(1)$ $T_W = \boldsymbol{\theta}(n)$

T(n)=O(n)

```
public void setBranch(Branch newBranch){
    dataBase.branchList.addLast(newBranch);
    dataBase.setbranchSize(getbranchSize()+1);
}
```

addLast = $\boldsymbol{\theta}(1)$

T(n) = $\boldsymbol{\theta}(1)$

```
public void addBranch(Branch newBranch) throws IllegalArgumentException{

     boolean control = true;

    for(int i=0;i<getbranchSize();i++){
        if(dataBase.branchList.get(i).getName().equals(newBranch.getName()))
            control = false;
    }
    if(control) {
        setBranch(newBranch);
        System.out.println("Branch successfully added. ");
    }
    else
        throw new IllegalArgumentException("Already existing " + newBranch.getName() +" in the set object cannot be added.");


}
```

branchList.get(i)----> T_B = θ(1) T_W = θ(n)

for loop = T_B = θ(1) T_W = θ(n)

so that for loop T(n) = O(n^2)

if(control) = θ(1)

T(n) = O(n^2)

```
public void removeBranch(Branch deleteBranch)throws Exception{
    int delete = 0;
    boolean control = false;
    for(int i=0;i<getbranchSize();i++){
        if(dataBase.branchList.get(i).getName().equals(deleteBranch.getName())) {
            delete = i;
            control = true;
        }
    }
    if(control){
        dataBase.branchList.remove(deleteBranch);
        dataBase.setbranchSize(getbranchSize()-1);
        System.out.println("Branch successfully removed. ");
    }
    else
        throw new Exception("Error : There is no " + deleteBranch.getName() +" so it cannot be deleted");
}
```

branchList.get(i)----> T_B = θ(1) T_W = θ(n)

for loop = T_B = θ(1) T_W = θ(n)

so that for loop T(n) = O(n^2)

if(control) = θ(1) because remove method = θ(1)

T(n) = O(n^2)

```
public void productSupplied()
{
    boolean control = true;
    for(int i=0; i<getbranchSize();i++)
    {
        if(dataBase.branchList.get(i).getBranchManagement().getwarningStock())
        {
            for(int j=0;j<dataBase.branchList.get(i).getBranchManagement().getProductTypeSize();j++)
            {
                if(dataBase.branchList.get(i).getBranchManagement().requiredProduct.getcolor() !=null) {
                    if(dataBase.branchList.get(i).getBranchManagement().products.get(j).getmodel().equals(dataBase.branchList.get(i).getBranchManagement().requiredProduct
                            dataBase.branchList.get(i).getBranchManagement().products.get(j).getcolor().equals(dataBase.branchList.get(i).getBranchManagement().requiredPr
                                    dataBase.branchList.get(i).getBranchManagement().products.get(j).getProductType().equals(dataBase.branchList.get(i).getBranchManagemen
                    {
                        dataBase.branchList.get(i).getBranchManagement().products.get(j).setstock(5);
                        dataBase.branchList.get(i).getBranchManagement().setwarningStock(false);
                        System.out.println("The product has been successfully supplied.");
                        break;
                    }
                }
                else {
                    if(dataBase.branchList.get(i).getBranchManagement().products.get(j).getmodel().equals(dataBase.branchList.get(i).getBranchManagement().requiredProduct
                            dataBase.branchList.get(i).getBranchManagement().products.get(j).getProductType().equals(dataBase.branchList.get(i).getBranchManagement().requ
                    {
                        dataBase.branchList.get(i).getBranchManagement().products.get(j).setstock(5);
                        dataBase.branchList.get(i).getBranchManagement().setwarningStock(false);
                        System.out.println("The product has been successfully supplied.");
                        break;
                    }

                }
                control =false;
            }
        }
    }
    if(control)
        System.out.println("There is no product to be supplied.");
```

8 branchlist.get (i) methods are used in the loop and this is O(8n).

For loop=O(n)

So that T(n)=O(8n^2)=O(n^2)

```
public void listBranch(){
    int i=0;
     @SuppressWarnings({ "unchecked", "unused" })
    Iterator<Branch> iterator =dataBase.branchList.listIterator();
    while(iterator.hasNext())
    {
        System.out.printf("%d. Branch) %s\n",i+1,iterator.next().getName());
        i++;
    }
}
```

İterator.next= $\theta(1)$ **İterator.hasnext**= $\theta(1)$

While = $\theta(n)$

T(N)= $\theta(N)$

```
public void listBranchEmployee(){
    for(int i=0;i<getbranchEmployeeSize();i++)
        System.out.printf("%d. Branch Employee) %s - %s branch\n",i+1,dataBase.branchEmpList.get(i).getUsername(),dataBase.branchEmpList.get(i).getBranchName());
}
```

T(n)= $\boldsymbol{\theta}$(n)

BRANCHEMPLOYEE

```
public void addCustomer(Customer newCustomer) throws Exception{
    int i;

    boolean control = true;

    for(i=0;i<dataBase.getcusotomerSize();i++){
        if(dataBase.customerList.get(i).getName().equals(newCustomer.getName()) &&
                dataBase.customerList.get(i).getlastName().equals(newCustomer.getlastName()))
            control = false;
    }
    if(control) {
        setCustomers(newCustomer);
        System.out.println("You have successfully registered, your customer number is "+dataBase.customerList.get(i).getCustomerNumber());
    }
    else
        throw new Exception("Error : This customer is available");

}
/**
```

Customer list all get() method = $\boldsymbol{\theta}$(1)

So that **for loop =** T_B = $\boldsymbol{\theta}$(1) T_W = $\boldsymbol{\theta}$**(n)**

T(n)=O(n)

```java
public void removeCustomer(Customer deleteCustomer) throws Exception{
        int delete = 0;
        boolean control = false;
        for(int i=0;i<dataBase.getcusotomerSize();i++){
            if(dataBase.customerList.get(i).getName().equals(deleteCustomer.getName()) &&
                    dataBase.customerList.get(i).getlastName().equals(deleteCustomer.getlastName()))
            {
                delete = i;
                control = true;
            }
        }
        if(control){
            dataBase.customerList.remove(delete);
            dataBase.setcusotomerSize(dataBase.getcusotomerSize()-1);
            System.out.println("Customer has been successfully deleted from the system.");

        }
        else {
            throw new Exception("Error : This customer is not registered in the system.");
        }
}
/**
```

for loop = $T_B = \theta(1)$ $T_W = \theta(n)$

remove method = $T_B = \theta(1)$ $T_W = \theta(n)$

T(n)=O(2n)=O(n)

```
public void saleProduct(Customer customerObj,Product newProduct,Branch brancjObj)throws Exception
{

    int i,temp = 0;
    boolean control = true;
    for(i=0;i<dataBase.getcusotomerSize();i++){
       if(dataBase.customerList.get(i).getName().equals(customerObj.getName()) &&
              dataBase.customerList.get(i).getlastName().equals(customerObj.getlastName()))
       {
           control = false;
           temp=i;
       }
    }

   if(control) {
       addCustomer(customerObj);
       temp=dataBase.getcusotomerSize()-1;

   }

   if(( brancjObj.getBranchManagement().products.get(getIndexbyProduct(newProduct, brancjObj)).getstock())>0) {

       dataBase.customerList.get(temp).newOrder(newProduct);
       //dataBase.customer[temp].newOrder(newProduct);
       removeProduct(newProduct, brancjObj, 1);
       System.out.println("The product has been successfully sold.");
   }
   else {
       System.out.println("not stock enough");
   }


}
/**
```

for loop = T_B **=** $\theta(1)$ T_W **=** θ**(n)**

if(control) = T(n) = $\theta(1)$

getIndexProduct=O(n^2)

products.get= T_B = $\theta(1)$ T_W = θ**(n)**

remove = T_B = $\theta(1)$ T_W = θ**(n)**

T(n) = O(n^2+3n) =O(n^2)

```
public void addProduct(Product productObj,Branch brancjObj,int stockSize)
{
    brancjObj.getBranchManagement().products.get(getIndexbyProduct(productObj, brancjObj)).
       setstock(brancjObj.getBranchManagement().products.get(getIndexbyProduct(productObj, brancjObj)).getstock()+stockSize) ;
    System.out.println("PRODUCT ADD SUCCESSFUL");
}
```

get()= T_B = $\theta(1)$ T_W = θ**(n)**

getIndexProduct=O(n^2)

T(n)=O(2n^2+2n)=O(n^2)

```
public void removeProduct(Product newProduct,Branch brancjObj,int stockSize)
{
    brancjObj.getBranchManagement().products.get(getIndexbyProduct(newProduct, brancjObj)).
       setstock(brancjObj.getBranchManagement().products.get(getIndexbyProduct(newProduct, brancjObj)).getstock()-stockSize) ;
    System.out.println("PRODUCT REMOVE SUCCESSFUL");
}
```

get()= T_B = θ(1) T_W = θ(n)

getIndexProduct=O(n^2)

T(n)=O(2n^2+2n)=O(n^2)

```
public int getIndexbyProduct(Product productObj,Branch brancjObj)
{
    int IndexbyProduct;
    if(productObj.getProductType() == "Chair" || productObj.getProductType() == "OfficeDesk" ||
            productObj.getProductType() == "MeetingTable" )
    {

        for(int i=0;i<brancjObj.getBranchManagement().getProductTypeSize();i++) {
           if( brancjObj.getBranchManagement().products.get(i).getProductType().equals(productObj.getProductType()) &&
                   brancjObj.getBranchManagement().products.get(i).getcolor().equals(productObj.getcolor()) &&
                       brancjObj.getBranchManagement().products.get(i).getmodel().equals(productObj.getmodel()))
           {

               IndexbyProduct = i;
               return IndexbyProduct;
           }

        }

    }

    else
    {
        for(int i=0;i<brancjObj.getBranchManagement().getProductTypeSize();i++) {
           if( brancjObj.getBranchManagement().products.get(i).getProductType().equals(productObj.getProductType()) &&
                   brancjObj.getBranchManagement().products.get(i).getmodel().equals(productObj.getmodel()))
           {
               IndexbyProduct = i;
               return IndexbyProduct;
           }

        }
    }
   return 0;
}
```

get()= T_B = θ(1) T_W = θ(n)

for = T_B = θ(1) T_W = θ(n)

T(n)=O(n^2)

```
public void listProduct(Product productObj,Branch branchObj){

    int index;

    index = getIndexbyProduct(productObj, branchObj);

   if(branchObj.getBranchManagement().products.get(index).getcolor() != null) {
       System.out.printf("\nBranch : %s \nProduct Type : %s \nProduct Model : %s \nProduct Color: %s \nProduct Stock : %d\n",branchObj.getName(),
               branchObj.getBranchManagement().products.get(index).getProductType(),branchObj.getBranchManagement().products.get(index).getmodel(),
                   branchObj.getBranchManagement().products.get(index).getcolor(),branchObj.getBranchManagement().products.get(index).getstock());
   }
   else {
       System.out.printf("\nBranch : %s \nProduct Type : %s \nProduct Model : %s \nProduct Stock : %d\n",branchObj.getName(),branchObj.getBranchManagement().products.get(
               branchObj.getBranchManagement().products.get(index).getmodel(),branchObj.getBranchManagement().products.get(index).getstock());
   }
```

get()= T_B = θ(1) T_W = θ(n)

T(n)=O(n)

```java
public void orderList(int customerNumber)
{
    for(int i=0; i<dataBase.getcusotomerSize(); i++)
    {
        if(dataBase.customerList.get(i).customerNumber == customerNumber)
            dataBase.customerList.get(i).orderList(customerNumber);
    }
}
```

for = T_B = θ(1) T_W = θ(n)

orderList()=O(n*(m^2))

T(n)=O((n^2)*(m^2))

```java
public void listCustomers(){
   for(int i=0;i<dataBase.getcusotomerSize();i++)
       System.out.printf("%s %s , %s ,CustomerNumber : %d\n",dataBase.customerList.get(i).getName(),dataBase.customerList.get(i).getlastName(),dataBase.customerList.get(i)
}
```

T(n)= θ(n)

CUSTOMER

```java
public void customerLogin(Customer customer)
{
    boolean control = true;
    for(int i=0;i< dataBase.getcusotomerSize();i++)
    {
        if(dataBase.customerList.get(i).getName().equals(customer.getName()))

        {
            System.out.println("Customer Login successful.");
            control=false;
        }

    }
    if(control)
        System.out.println("Username is wrong.");

}
```

T(n)= θ(n)

```java
public void saleOnlineProduct(Customer customerObj,Product newProduct,Branch brancjObj)throws Exception
{
    int i,temp = 0;
    for(i=0;i<dataBase.getcusotomerSize();i++) {
       if(dataBase.customerList.get(i).getName().equals(customerObj.getName()) &&
               dataBase.customerList.get(i).getlastName().equals(customerObj.getlastName()))
       {

           temp=i;
       }
    }

    if(brancjObj.getBranchManagement().products.remove(getIndexbyProduct(newProduct, brancjObj)).getstock()>0) {
       dataBase.customerList.get(temp).newOrder(newProduct);
       removeProduct(newProduct, brancjObj, 1);
       System.out.println("The product has been successfully sold.");
    }
    else {
       throw new Exception("Error : No stock enough");

    }
}
```

For= $\boldsymbol{\theta}$**(n)**

remove = T_B = $\boldsymbol{\theta}$(1) T_W = $\boldsymbol{\theta}$**(n)**

getIndexProduct=O(n^2)

T(n)=O(n^2)

```java
public void listProduct(Product productObj,Branch branchObj){

    int index;
    index = getIndexbyProduct(productObj, branchObj);


    if(branchObj.getBranchManagement().products.get(index).getcolor() != null) {
       System.out.printf("\nBranch : %s \nProduct Type : %s \nProduct Model : %s \nProduct Color: %s \n",branchObj.getName(),
               branchObj.getBranchManagement().products.get(index).getProductType(),branchObj.getBranchManagement().products.get(index).getmodel(),
                   branchObj.getBranchManagement().products.get(index).getcolor());
    }
    else {
       System.out.printf("\nBranch : %s \nProduct Type : %s \nProduct Model : %s \n",branchObj.getName(),branchObj.getBranchManagement().products.get(index).getProduct
               branchObj.getBranchManagement().products.get(index).getmodel());
    }
```

getIndexProduct=O(n^2)

get()=O(n)

T(n)=O(n^2)

```
public void orderList(int customerNumber)
{

     for(int i=0; i<dataBase.getcusotomerSize(); i++)
     {
         if(dataBase.customerList.get(i).customerNumber == customerNumber)
         {
         System.out.println("Customer number "+dataBase.customerList.get(i).getCustomerNumber());

          for(int j=0;j<orderCount;j++)
            {
                if(dataBase.customerList.get(i).orderArr.get(j).getcolor() != null) {
                    System.out.printf("%s %s %s\n",dataBase.customerList.get(i).orderArr.get(j).getProductType(),
                            dataBase.customerList.get(i).orderArr.get(j).getmodel(),dataBase.customerList.get(i).orderArr.get(j).getcolor());
                }
                else {
                    System.out.printf("%s %s\n",dataBase.customerList.get(i).orderArr.get(j).getProductType(),
                            dataBase.customerList.get(i).orderArr.get(j).getmodel());
                }
            }
         }
     }

}
/**
```

customersize = n

orderCount = m

orderArr.get()= $T_B = \theta(1)$ $T_W = \theta(n)$

T(n,m) = O((n^2)*m)