
ARRAYLIST

```
public class KWArrayList<E> {
```

```
    private static final int INITIAL_CAPACITY = 10;
    private E[] theData;
    private int size = 0;
    private int capacity = 0;
```

$T(n)=\Theta(1)$

```
    public KWArrayList(){
        capacity = INITIAL_CAPACITY;
        theData = (E[]) new Object[capacity];
    }
```

$T(n)=\Theta(1)$

```
    public boolean add(E anEntry){
        if (size == capacity){
            reallocate();
        }
        theData[size] = anEntry;
        size++;
        return true;
    }
```

$T(n)=O(n)$

```
    public void add(int index, E anEntry){
        if (index < 0 || index > size){
            throw new ArrayIndexOutOfBoundsException(index);
        }
        if (size == capacity){
            reallocate();
        }
        for (int i = size; i > index; i--){
            theData[i] = theData[i - 1];
        }
        theData[index] = anEntry;
        size++;
    }
```

$T(n)=\Theta(1)$

```
    public E get(int index){
        if (index < 0 || index >= size){
            throw new ArrayIndexOutOfBoundsException(index);
        }
        return theData[index];
    }
```

$T(n)=\Theta(1)$

```
public E set(int index, E newValue){
    if (index < 0 || index >= size){
        throw new ArrayIndexOutOfBoundsException(index);
    }
    E oldValue = theData[index];
    theData[index] = newValue;
    return oldValue;
}
```

$T(n)=O(n)$

```
public E remove(int index){
    if (index < 0 || index >= size){
        throw new ArrayIndexOutOfBoundsException(index);
    }
    E returnValue = theData[index];
    for (int i = index + 1; i < size; i++) {
        theData[i - 1] = theData[i];
    }
    size--;
    return returnValue;
}
```

$T(n)=\Theta(1)$

```
public int size(){
    return size;
}
```

$T(n)=\Theta(1)$

```
private void reallocate(){
    capacity = 2 * capacity;
    theData = Arrays.copyOf(theData, capacity);
}
```

LINKEDLIST

public class KWSingleLinkedList<E> {

```
    private Node<E> head = null;
    private int size = 0;
```

$T(n)=\Theta(1)$

```
    private static class Node<E> {
        private E data;
        private Node<E> next;
        private Node(E dataItem) {
            data = dataItem;
            next = null;
        }
    }
```

```

    }
    private Node(E dataItem, Node<E> nodeRef) {
        data = dataItem;
        next = nodeRef;
    }
}

```

$T(n)=\Theta(1)$

```

public void addFirst(E item) {
    head = new Node<>(item, head);
    size++;
}

```

$T(n)=\Theta(1)$

```

private void addAfter(Node<E> node, E item) {
    node.next = new Node<>(item, node.next);
    size++; }

```

$T(n)=\Theta(1)$

```

private E removeAfter(Node<E> node) {
    Node<E> temp = node.next;
    if (temp != null) {
        node.next = temp.next;
        size=size-1;
        return temp.data;
    }
    else {
        return null;
    }
}

```

$\Theta(1)$
 $\Theta(1)$
 $\Theta(1)$
 $\Theta(1)$
 $\Theta(1)$

$T(n)=\Theta(1)$

```

private E removeFirst() {
    Node<E> temp = head;
    if (head != null) {
        head = head.next;
    }
    // Return data at old head or null if list is empty
    if (temp != null) {
        size=size-1;
        return temp.data;
    }
    else {
        return null;
    }
}

```

$\Theta(1)$
 $\Theta(1)$
 $\Theta(1)$
 $\Theta(1)$
 $\Theta(1)$

$T(\text{best})(n)=\Theta(1)$
 $T(\text{worst})(n)=\Theta(n)$
 $T(n)=O(n)$

```

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

```

```

    }
    return node;
}

```

$T(n)=O(n)$

```

public E get(int index) {
    if (index < 0 || index >= size) {
        throw new IndexOutOfBoundsException(Integer.toString(index));
    }
    Node<E> node = getNode(index);
    return node.data;
}

```

$T(n)=O(n)$

```

public E set(int index, E newValue) {
    if (index < 0 || index >= size) {
        throw new IndexOutOfBoundsException(Integer.toString(index));
    }
    Node<E> node = getNode(index);
    E result = node.data;
    node.data = newValue;
    return result;
}

```

$T(n)=O(n)$

```

public void add(int index, E item) {
    if (index < 0 || index > size) {
        throw new IndexOutOfBoundsException(Integer.toString(index));
    }
    if (index == 0) {
        addFirst(item);
    }
    else {
        int a=index-1;
        Node<E> node = getNode(a);
        addAfter(node, item);
    }
}

```

$T(n)=O(n)$

```

public boolean add(E item) {
    add(size, item);
    return true;
}

```

$T(n)=\Theta(1)$

```

public int size(){
    return size;
}

```

ADMINISTRATOR METHODS

Query_Product_in_Stock Method

temporary-amount-product-num } ArrayList

```
public void Query_Product_in_Stock(){
```

```
    KWSingleLinkedList<KWSingleLinkedList<Integer> > Office_Chairs=getOffice_Chairs();
```

```
    KWSingleLinkedList<KWSingleLinkedList<Integer> > Office_Desks=getOffice_Desks();
```

```
    KWSingleLinkedList<KWSingleLinkedList<Integer> > Meeting_Tables=getMeeting_Tables();
```

```
    KWSingleLinkedList<Integer> Bookcases=getBookcases();
```

```
    KWSingleLinkedList<Integer> Office_Cabinets=getOffice_Cabinets();
```

```
    Company person=new Customer("default","default","default","default");
```

```
    int []Customer_Num_Arr=((Customer)person).getCustomer_Num_Arr();
```

```
    KWSingleLinkedList<KWSingleLinkedList<Integer> > temporary_amount_product_num=((Customer)person).getTemporary_amount_product_num();
```

```
    int [][]temporary_item_number=((Customer)person).getTemporary_item_number();
```

```
    int k=1;
```

```
    for(int i=0;i<Office_Chairs.size();i++){
```

```
        for(int j=0;j<Office_Chairs.get(i).size();j++){
```

```
            product_stock_num[k]=Office_Chairs.get(i).get(j);
```

```
            k++;
```

```
        }
```

```
    for(int i=0;i<Office_Desks.size();i++){
```

```
        for(int j=0;j<Office_Desks.get(i).size();j++){
```

```
            product_stock_num[k]=Office_Desks.get(i).get(j);
```

```
            k++;
```

```
        }
```

```
    for(int i=0;i<Meeting_Tables.size();i++){
```

```
        for(int j=0;j<Meeting_Tables.get(i).size();j++){
```

```
            product_stock_num[k]=Meeting_Tables.get(i).get(j);
```

```
            k++;
```

```
        }
```

```
    for(int i=0;i<Bookcases.size();i++){
```

```
        product_stock_num[k]=Bookcases.get(i);
```

```
        k++;
```

```
    }
```

```
    for(int i=0;i<Office_Cabinets.size();i++){
```

```
        product_stock_num[k]=Office_Cabinets.get(i);
```

```
        k++;
```

```
    }
```

```
    int less_num=0;
```

```
    for(int i=0;i<Customer_Num_Arr.length;i++){
```

```
        int counter=0;
```

```
        if(Customer_Num_Arr[i]==0){ break; }
```

```
        System.out.println("Orders for Customer that special customer number's is "+Customer_Num_Arr[i]+".");
```

```
        for(int j=0;j<temporary_item_number[i].length;j++){
```

```
            if(temporary_item_number[i][j]==0){
```

```
                System.out.println("!!!! There is No Order And Supplying Does Not Necessary !!!!!");
```

```
                counter++;
```

```
                break;
```

```
            }
```

```
            if(temporary_item_number[i][j]!=0){
```

```
                break;
```

```
            }
```

```
            if(product_stock_num[temporary_item_number[i][j]]<temporary_amount_product_num.get(Customer_Num_Arr[i].get(j)){
```

```
                less_num=temporary_amount_product_num.get(Customer_Num_Arr[i].get(j))-product_stock_num[temporary_item_number[i][j]];
```

```
                System.out.println("Dear Employee "+temporary_item_number[i][j]+", Product less : "+ less_num + " Therefore please supply it !!!");
```

```
                counter++;
```

```
            }
```

```
        }
```

```
        if(counter==0){
```

```
            System.out.println("All order's amounts are enough !!!");
```

```
        }
```

```
        System.out.println();
```

```
    }
```

$\Theta(1)$

(Simple Statement)

$\Theta(1)$

$O(t^4)$

(inner loop)

$O(t^5)$

(All Loop)

$O(c^4)$

(inner loop)

$O(c^5)$

(All Loop)

$O(p^4)$

(inner loop)

$O(p^5)$

(All loop)

$O(a^2)$

$O(a^2)$

$O(b^2)$

$O(b^2)$

$T_{1(cw)}(n) = \Theta(n)$

$T_{1(cb)}(n) = \Theta(n)$

$T_1(n) = O(n)$

$T_2(w) = \Theta(m)$

$T_2(b) = \Theta(1)$

$T_2(m) = O(m)$

All loop $T(n,m):$

$\Rightarrow T_1(n), T_2(m)$

$\Rightarrow O(n,m)$

$\Rightarrow O(n,m)$

$\Rightarrow O(n,m)$

$\Rightarrow O(n,m)$

$\Rightarrow O(n,m)$

$\Rightarrow O(n,m)$

$\Rightarrow O(n,m)$

$\Rightarrow O(n,m)$

$\Rightarrow O(n,m)$

$\Rightarrow O(n,m)$

$\Rightarrow O(n,m)$

All Algorithm $\Rightarrow T(n,m,t,c,p,a,b) \Rightarrow O(t^5) + O(c^5) + O(p^5) + O(a^2) + O(b^2) + O(n,m) + \Theta(1)$
 $\Rightarrow O(t^5 + c^5 + p^5 + a^2 + b^2 + n,m)$

CUSTOMER METHODS

List_Of_Products Method

$$\Rightarrow T(a, b, c, d, e)$$

```

public void List_Of_Products(){
    KWSingleLinkedList<KWSingleLinkedList<Integer> > Office_Chairs=getOffice_Chairs();
    KWSingleLinkedList<KWSingleLinkedList<Integer> > Office_Desks=getOffice_Desks();
    KWSingleLinkedList<KWSingleLinkedList<Integer> > Meeting_Tables=getMeeting_Tables();
    KWSingleLinkedList<Integer> Bookcases=getBookcases();
    KWSingleLinkedList<Integer> Office_Cabinets=getOffice_Cabinets();
    int ctr=0;
    System.out.println();
    System.out.println("*****");
    System.out.println("-----Office Chair List-----");
    System.out.println("*****");
    for(int i=0; i<7; i++){
        System.out.println();
        System.out.println("-----Office Chair "+(i+1)+". Model-----");
        for(int j=0; j<Office_Chairs.get(i).size(); j++){
            System.out.println((ctr+1)+". -> Office Chair "+ (i+1) + ".Model "+(j+1)+".Color");
            product_list_num[ctr]=ctr+1;
            ctr++;
        }
        System.out.println();
        System.out.println("*****");
        System.out.println("-----Office Desk List-----");
        System.out.println("*****");
        for(int i=0; i<5; i++){
            System.out.println();
            System.out.println("-----Office Desk "+(i+1)+". Model-----");
            for(int j=0; j<Office_Desks.get(i).size(); j++){
                System.out.println((ctr+1)+". -> Office Desk "+ (i+1) + ".Model "+(j+1)+".Color");
                product_list_num[ctr]=ctr+1;
                ctr++;
            }
            System.out.println();
            System.out.println("*****");
            System.out.println("-----Meeting Table List-----");
            System.out.println("*****");
            for(int i=0; i<10; i++){
                System.out.println();
                System.out.println("-----Meeting Table "+(i+1)+". Model-----");
                for(int j=0; j<Meeting_Tables.get(i).size(); j++){
                    System.out.println((ctr+1)+". -> Meeting Table "+ (i+1) + ".Model "+(j+1)+".Color");
                    product_list_num[ctr]=ctr+1;
                    ctr++;
                }
                System.out.println();
                System.out.println("*****");
                System.out.println("-----Bookcase List-----");
                System.out.println("*****");
                for(int i=0; i<Bookcases.size(); i++){
                    System.out.println((ctr+1)+". -> Bookcase "+ (i+1) + ".Model");
                    ctr++;
                    product_list_num[ctr]=ctr+1;
                }
                System.out.println();
                System.out.println("-----Office Cabinet List-----");
                System.out.println("*****");
                for(int i=0; i<Office_Cabinets.size(); i++){
                    System.out.println((ctr+1)+". -> Office Cabinet "+ (i+1) + ".Model");
                    product_list_num[ctr]=ctr+1;
                    ctr++;
                }
            }
        }
    }
}

```

$$\Rightarrow O(a^2) + O(b^2) + O(c^2) + O(d) + O(e) + O(1)$$

$$\Rightarrow O(a^2 + b^2 + c^2 + d + e)$$

$$O(b^2)$$

$$O(c^2)$$

$$O(d)$$

$$O(e)$$

Stock Method

```

public void Stock()
{
    KWSingleLinkedList<KWSingleLinkedList<Integer>> Office_Chairs=getOffice_Chairs();
    KWSingleLinkedList<KWSingleLinkedList<Integer>> Office_Desks=getOffice_Desks();
    KWSingleLinkedList<KWSingleLinkedList<Integer>> Meeting_Tables=getMeeting_Tables();
    KWSingleLinkedList<Integer> Bookcases=getBookcases();
    KWSingleLinkedList<Integer> Office_Cabinets=getOffice_Cabinets();

    int ctr=0;

    System.out.println();
    System.out.println("*****");
    System.out.println("-----Office Chair Stock-----");
    System.out.println("*****");
    for(int i=0;i<7;i++){
        System.out.println();
        System.out.println("-----Office Chair "+(i+1)+" Model-----");
        for(int j=0;j<Office_Chairs.get(i).size();j++){
            System.out.println((ctr+1)+" -> Office Chair "+ (i+1) +".Model "+(j+1)+".Color : "+Office_Chairs.get(i).get(j));
            ctr++;
        }
        System.out.println();
        System.out.println("*****");
        System.out.println("-----Office Desk Stock-----");
        System.out.println("*****");
        for(int i=0;i<5;i++){
            System.out.println();
            System.out.println("-----Office Desk "+(i+1)+" Model-----");
            for(int j=0;j<Office_Desks.get(i).size();j++){
                System.out.println((ctr+1)+" -> Office Desk "+ (i+1) +".Model "+(j+1)+".Color : "+Office_Desks.get(i).get(j));
                ctr++;
            }
            System.out.println();
            System.out.println("*****");
            System.out.println("-----Meeting Table Stock-----");
            System.out.println("*****");
            for(int i=0;i<10;i++){
                System.out.println();
                System.out.println("-----Meeting Table "+(i+1)+" Model-----");
                for(int j=0;j<Meeting_Tables.get(i).size();j++){
                    System.out.println((ctr+1)+" -> Meeting Table "+ (i+1) +".Model "+(j+1)+".Color : "+Meeting_Tables.get(i).get(j));
                    ctr++;
                }
                System.out.println();
                System.out.println("*****");
                System.out.println("-----Bookcase Stock-----");
                System.out.println("*****");
                for(int i=0;i<Bookcases.size();i++){
                    System.out.println((ctr+1)+" -> Bookcase "+ (i+1) +".Model : "+Bookcases.get(i));
                    ctr++;
                }
                System.out.println();
                System.out.println("-----Office Cabinet Stock-----");
                System.out.println("*****");
                for(int i=0;i<Office_Cabinets.size();i++){
                    System.out.println((ctr+1)+" -> Office Cabinet "+ (i+1) +".Model : "+Office_Cabinets.get(i));
                    ctr++;
                }
            }
        }
    }
}

```

Handwritten annotations on the code:

- $O(1)$ for the initial list assignments.
- $O(1)$ for the separator lines.
- $O(n^4)$ for the Office Chair stock loop (7 models, each with up to 7 chairs).
- $O(n^2)$ for the Office Desk stock loop (5 models, each with up to 5 desks).
- $O(n^4)$ for the Meeting Table stock loop (10 models, each with up to 10 tables).
- $O(n^2)$ for the Bookcase stock loop (up to 10 bookcases).
- $O(n)$ for the Office Cabinet stock loop (up to 10 cabinets).
- $O(1)$ for the final separator lines.

All Algorithm

$$T(a, b, c, d, e) = O(a^4) + O(b^4) + O(c^4) + O(d^2) + O(e^2) + O(1)$$

\Rightarrow

$$\Rightarrow O(a^4 + b^4 + c^4 + d^2 + e^2)$$

Shopping_on_Online Method

address - arr, phone - number - arr, product - list - num, temporary - item - num, permanent - item - number,
temporary - amount - product, permanent - amount - product - num

Array list

```

public void Shopping_on_Online(String address, String phone_number, int num_of_product, int piece_of_product) throws ClassException {
    // 1)
    int count3=0, count4=0, count5=0;
    for(int i=0; i<this.address_arr.size(); i++){
        if(address.equals(this.address_arr.get(i))){
            count3++;
        }
    }
    for(int i=0; i<this.phone_number_arr.size(); i++){
        if(phone_number.equals(this.phone_number_arr.get(i))){
            count4++;
        }
    }
    if(count3==0 && count4==0){ throw new ClassException("!!! Given phone number using for other customer !!!"); }
    for(int i=0; i<product_list_num.length; i++){
        if(num_of_product==product_list_num[i]){
            count5++;
        }
    }
    if(count5==0){ throw new ClassException("!!! Given product number does not belong the product list !!!"); }
    else{
        int count2=0;
        for(int i=0; i<this.phone_number_arr.size(); i++){
            if(phone_number==this.phone_number_arr.get(i)){
                count2++;
            }
        }
        if(count2==0){
            this.address_arr.add(index_of_address_and_phone_number_arr, address);
            this.phone_number_arr.add(index_of_address_and_phone_number_arr, phone_number);
            index_of_address_and_phone_number_arr++;
        }
        int ctr2=0;
        for(int j=0; j<temporary_item_number[customer_num].length; j++){
            if(temporary_item_number[customer_num][j]==0){
                break;
            }
            ctr2++;
        }
        temporary_item_number[customer_num][ctr2]=num_of_product;
        ctr2=0;
        for(int j=0; j<permanent_item_number.get(customer_num).size(); j++){
            if(permanent_item_number.get(customer_num).get(j)==0){
                break;
            }
            ctr2++;
        }
        permanent_item_number.get(customer_num).add(ctr2, num_of_product);
        ctr2=0;
        for(int j=0; j<temporary_amount_product_num.get(customer_num).size(); j++){
            if(temporary_amount_product_num.get(customer_num).get(j)==0){
                break;
            }
            ctr2++;
        }
        temporary_amount_product_num.get(customer_num).add(ctr2, piece_of_product);
        ctr2=0;
        for(int j=0; j<permanent_amount_product_num.get(customer_num).size(); j++){
            if(permanent_amount_product_num.get(customer_num).get(j)==0){
                break;
            }
            ctr2++;
        }
        permanent_amount_product_num.get(customer_num).add(ctr2, piece_of_product);
    }
}

```

All Algorithm

$T(a, b, c, d, e, k, t)$

$\Rightarrow \theta(a) + \theta(b) + \theta(c) + \theta(b) + \theta(a+b) + \theta(d) + \theta(e) + \theta(e) + \theta(k) + \theta(k) + \theta(t) + \theta(k) + \theta(1)$

$\Rightarrow \theta(a+b+c+d+e+k+t)$

previous_order Method

permanent_item_number, permanent_amount_product_num Array list

```
public void previous_order(){  
    System.out.println("Your Previous Order (Customer Number "+customer_num+"):");
```

$\rightarrow \Theta(1)$

```
    for(int i=0; i<Customer_Num_Arr.length; i++){
```

$\Theta(1)$

```
        int ctr2=0;
```

$\Theta(1)$

```
        if(Customer_Num_Arr[i]==customer_num){  
            for(int j=0; j<permanent_item_number.get(customer_num).size(); j++){  
                if(permanent_item_number.get(customer_num).get(j)==0){  
                    if(ctr2==0){  
                        System.out.println("!!!! There is No Order !!!!!");  
                    }  
                    break;
```

$\Theta(1)$

$\Theta(1)$

$\Theta(1)$

```
                if(j<completed_order_number[i][0]){  
                    System.out.println(permanent_item_number.get(customer_num).get(j)  
                        +", Product in the List of Product and Amount of Product :"  
                        +permanent_amount_product_num.get(customer_num).get(j)+" (DONE)");  
                    ctr2++;
```

```
                if(j>=completed_order_number[i][0]){  
                    System.out.println(permanent_item_number.get(customer_num).get(j)  
                        +", Product in the List of Product and Amount of Product :"  
                        +permanent_amount_product_num.get(customer_num).get(j));  
                    ctr2++;  
                }  
            }  
        }  
    }  
}
```

$T_{1(b)}(n) = \Theta(1)$
 $T_{1(w)}(n) = \Theta(n)$

$T_1(n) = O(n)$

All Loop

$T_{2(b)}(m) = \Theta(1)$
 $T_{2(w)}(m) = \Theta(m)$

$T_2(m) = O(m)$

$T_1(n), T_2(m)$
 $\Rightarrow O(n, m)$

All Algorithm

$$T(n, m) = O(n, m) + \Theta(1)$$

$$\Rightarrow O(n, m)$$

EMPLOYEE METHODS

Add_Product_to_Stock Method

```

public void Add_Product_to_Stock(int num_of_product, int piece_of_product) {
    KWSingleLinkedList<KWSingleLinkedList<Integer>> Office_Chairs=getOffice_Chairs();
    KWSingleLinkedList<KWSingleLinkedList<Integer>> Office_Desks=getOffice_Desks();
    KWSingleLinkedList<KWSingleLinkedList<Integer>> Meeting_Tables=getMeeting_Tables();
    KWSingleLinkedList<Integer> Bookcases=getBookcases();
    KWSingleLinkedList<Integer> Office_Cabinets=getOffice_Cabinets();

    product_stock_num[num_of_product] += piece_of_product;

    int k=1;
    for(int i=0; i<7; i++){
        for(int j=0; j<Office_Chairs.get(i).size(); j++){
            product_stock_num[k]=Office_Chairs.get(i).get(j);
            k++;
        }
    }
    for(int i=0; i<5; i++){
        for(int j=0; j<Office_Desks.get(i).size(); j++){
            product_stock_num[k]=Office_Desks.get(i).get(j);
            k++;
        }
    }
    for(int i=0; i<10; i++){
        for(int j=0; j<Meeting_Tables.get(i).size(); j++){
            product_stock_num[k]=Meeting_Tables.get(i).get(j);
            k++;
        }
    }
    for(int i=0; i<Bookcases.size(); i++){
        product_stock_num[k]=Bookcases.get(i);
        k++;
    }
    for(int i=0; i<Office_Cabinets.size(); i++){
        product_stock_num[k]=Office_Cabinets.get(i);
        k++;
    }
    setOffice_Chairs(Office_Chairs);
    setOffice_Desks(Office_Desks);
    setMeeting_Tables(Meeting_Tables);
    setBookcases(Bookcases);
    setOffice_Cabinets(Office_Cabinets);
}
    
```

$\Theta(1)$
 $O(n^2)$
 $O(n^2)$
 $O(n^4)$ (inner loop)
 $O(7n^4) = O(n^4)$ (All loop)
 $O(m^2)$
 $O(m^2)$
 $O(m^4)$
 $O(k^2)$
 $O(k^2)$
 $O(k^4)$
 $O(a)$
 $O(a)$
 $O(a^2)$
 $O(b)$
 $O(b)$
 $O(b^2)$

$$\begin{aligned}
 T(n, m, k, a, b) &= O(n^4) + O(m^4) + O(k^4) + O(a^2) + O(b^2) + \Theta(1) \\
 &= O(n^4 + m^4 + k^4 + a^2 + b^2)
 \end{aligned}$$

pre_cust_order Method

temporary-amount-product-num } Arraylist

```

public void pre_cust_order(int number) throws ClassNotFoundException {
    int counter=0;
    for(int i=0; i<Customer_Num_Arr.length; i++){
        if(Customer_Num_Arr[i]==0){
            break;
        }
        if(number==Customer_Num_Arr[i]){
            counter++;
        }
    }
    if(counter==0){
        throw new ClassNotFoundException("!!! Given number is not belong any special customer numbers !!!");
    }
    System.out.println("Orders for Customer that special customer number's is "+ number+".");
    int counter=0;
    for(int i=0; i<Customer_Num_Arr.length; i++){
        if(Customer_Num_Arr[i]==number){
            for(int j=0; j<temporary_item_number[number].length; j++){
                if(temporary_item_number[number][j]==0){
                    if(counter==0){
                        System.out.println("!!!! There is No Order !!!!!");
                    }
                    break;
                }
            }
            System.out.println(temporary_item_number[number][j]+" . Product in the List of Product and Amount of Product : "+temporary_amount_product_num.get(number).get(j));
            counter++;
        }
    }
}

```

$\Theta(1)$ $\left\{ \begin{array}{l} T_w(m) = \Theta(m) \\ T_b(m) = \Theta(1) \end{array} \right\} O(m)$
 $\Theta(1)$ $\left\{ \begin{array}{l} T_{1(b)}(m) = \Theta(1) \\ T_{1(w)}(m) = \Theta(m) \end{array} \right\} T_1(m) = O(m)$
 $\Theta(1)$ $\left\{ \begin{array}{l} T_{2(b)}(k) = \Theta(1) \\ T_{2(w)}(k) = \Theta(k) \end{array} \right\} T_2(k) = O(k)$
 $\Theta(1)$ $\left\{ \begin{array}{l} \Theta(1) \\ \Theta(1) \end{array} \right\} \Theta(1)$

$$T(m, k) = O(m) + O(m.k) + \Theta(1)$$

$$\Rightarrow O(m.k)$$

check_The_Product_Stock Method

temporary-amount-product-num} ArrayList

```

public void checkTheProductStock(){
    KWSingleLinkedList<KWSingleLinkedList<Integer>> Office_Chairs=getOffice_Chairs();
    KWSingleLinkedList<KWSingleLinkedList<Integer>> Office_Desks=getOffice_Desks();
    KWSingleLinkedList<KWSingleLinkedList<Integer>> Meeting_Tables=getMeeting_Tables();
    KWSingleLinkedList<Integer> Bookcases=getBookcases();
    KWSingleLinkedList<Integer> Office_Cabinets=getOffice_Cabinets();
    int k=1;
    for(int i=0;i<7;i++){
        for(int j=0;j<Office_Chairs.get(i).size();j++){
            product_stock_num[k]=Office_Chairs.get(i).get(j);
            k++;
        }
    }
    for(int i=0;i<5;i++){
        for(int j=0;j<Office_Desks.get(i).size();j++){
            product_stock_num[k]=Office_Desks.get(i).get(j);
            k++;
        }
    }
    for(int i=0;i<10;i++){
        for(int j=0;j<Meeting_Tables.get(i).size();j++){
            product_stock_num[k]=Meeting_Tables.get(i).get(j);
            k++;
        }
    }
    for(int i=0;i<Bookcases.size();i++){
        product_stock_num[k]=Bookcases.get(i);
        k++;
    }
    for(int i=0;i<Office_Cabinets.size();i++){
        product_stock_num[k]=Office_Cabinets.get(i);
        k++;
    }
}
    
```

$O(a^4)$

$O(b^4)$

$O(c^4)$

$O(d^2)$

$O(t^2)$

$T(a,b,c,d,t,n,m)$

$\Rightarrow O(a^4) + O(b^4) + O(c^4) + O(d^2) + O(t^2) + O(n,m) + O(1)$

$\Rightarrow O(a^4 + b^4 + c^4 + d^2 + t^2 + n,m)$

```

    int less_num=0;
    for(int i=0;i<Customer_Num_Arr.length;i++){
        int counter=0;
        if(Customer_Num_Arr[i+1]==0){
            break;
        }
        try{
            pre_cust_order(Customer_Num_Arr[i]);
        }
        catch(ClassException e){
            System.out.println();
        }
        for(int j=0;j<temporary_item_number[i].length;j++){
            if(temporary_item_number[Customer_Num_Arr[i]][j]==0){
                counter++;
                break;
            }
            if(temporary_item_number[Customer_Num_Arr[i]][j]==0){ break; }
            if(product_stock_num[temporary_item_number[Customer_Num_Arr[i]][j]]<temporary_amount_product_num.get(Customer_Num_Arr[i].get(j))){
                less_num=temporary_amount_product_num.get(Customer_Num_Arr[i].get(j))-product_stock_num[temporary_item_number[Customer_Num_Arr[i]][j]];
                System.out.println("Dear Administrator "+temporary_item_number[Customer_Num_Arr[i]][j]+". Product less : "+less_num+" Therefore I am adding !!!");
                counter++;
                Add_Product_to_Stock(temporary_item_number[Customer_Num_Arr[i]][j],less_num);
            }
        }
        if(counter==0){
            System.out.println("!!!All Order's Amounts are enough in the Stock!!!");
        }
        System.out.println();
    }
}
    
```

$T_1(b)(n) = \Theta(1)$
 $T_1(w)(n) = \Theta(n)$
 $T_1(n) = O(n)$

$T_2(b)(m) = \Theta(1)$
 $T_2(w)(m) = \Theta(m)$
 $T_2(m) = O(m)$

All Loop

$T_1(n), T_2(m)$
 $\Rightarrow O(n,m)$

Create_Order_List_For_Market_Customer Method

$\left. \begin{array}{l} \text{address_arr, phone_number_arr, temporary_amount_product_num, permanent_item_number} \\ \text{permanent_amount_product_num} \end{array} \right\} \text{Array list}$

```

public void Create_Order_List_For_Market_Customer(String address, String phone_number, int num_of_product, int piece_of_product) throws ClassException {
    int count3=0, count4=0, count5=0;
    O(1) Company person = new Customer("default", "default", "default", "default");

    for (int i=0; i<address_arr.size(); i++) {
        if (address.equals(address_arr.get(i))) {
            count3++;
        }
    }
    for (int i=0; i<phone_number_arr.size(); i++) {
        if (phone_number.equals(phone_number_arr.get(i))) {
            count4++;
        }
    }
    O(1) if (count3==0 && count4==0) {
        throw new ClassException("!!! Given phone number using for other customer !!!");
    }
    for (int i=0; i<product_list_num.length; i++) {
        if (num_of_product==product_list_num[i]) {
            count5++;
        }
    }
    O(1) if (count5==0) {
        throw new ClassException("!!! Given product number does not belong the product list !!!");
    }
    else {
        int index_of_address_and_phone_number_arr = ((Customer)person).getIndex_of_address_and_phone_number_arr();
        address_arr.add(index_of_address_and_phone_number_arr, address);
        // address_arr.add(index_of_address_and_phone_number_arr, address);
        phone_number_arr.add(index_of_address_and_phone_number_arr, phone_number);
        // phone_number_arr.add(index_of_address_and_phone_number_arr, phone_number);
        index_of_address_and_phone_number_arr++;
        ((Customer)person).setIndex_of_address_and_phone_number_arr(index_of_address_and_phone_number_arr);

        // temporary_item_number.get(customer_num).add(index_of_permanent_and_temporary_arrays, num_of_product);
        temporary_item_number[customer_num][index_of_permanent_and_temporary_arrays] = num_of_product;
        temporary_amount_product_num.get(customer_num).add(index_of_permanent_and_temporary_arrays, piece_of_product);

        permanent_item_number.get(customer_num).add(index_of_permanent_and_temporary_arrays, num_of_product);
        permanent_amount_product_num.get(customer_num).add(index_of_permanent_and_temporary_arrays, piece_of_product);

        index_of_permanent_and_temporary_arrays++;

        O(1) {
            ((Customer)person).setAddress(address_arr);
            ((Customer)person).setPhone_num(phone_number_arr);
            ((Customer)person).setTemporary_item_number(temporary_item_number);
            ((Customer)person).setTemporary_amount_product_num(temporary_amount_product_num);
            ((Customer)person).setPermanent_item_number(permanent_item_number);
            ((Customer)person).setPermanent_amount_product_num(permanent_amount_product_num);
        }
    }
}
    
```

$$T(n, m, k, p, t, a) = \Theta(n) + \Theta(m) + \Theta(k) + O(n) + O(m) + O(p) + O(t) + O(a) + \Theta(1)$$

$$\Rightarrow O(n + m + p + t + a + k)$$

Selling_Ordered_Products Method

temporary-amount-product-num, temporary-amount-product-num } Array list

```

public void Selling_Ordered_Products() {
    KWSingleLinkedList<Integer> > Office_Chairs=getOffice_Chairs();
    KWSingleLinkedList<Integer> > Office_Desks=getOffice_Desks();
    KWSingleLinkedList<Integer> > Meeting_Tables=getMeeting_Tables();
    KWSingleLinkedList<Integer> > Bookcases=getBookcases();
    KWSingleLinkedList<Integer> > Office_Cabinets=getOffice_Cabinets();
    Company person=new Customer("default","default","default","default");
    for(int i=0;i<Customer_Num_Arr.length;i++){
        int counter=0;
        if(Customer_Num_Arr[i]==0){
            System.out.println(".....EMPLOYEE SAY THAT:");
            System.out.println(".....");
            System.out.println("!!!!!!!!!!!!!!!!!!!!!!ALL ORDERS ARE SOLD!!!!!!!!!!!!!!!!!!!!!!");
            System.out.println();
            break;
        }
        for(int j=0;j<temporary_item_number[i].length;j++){
            if(temporary_item_number[Customer_Num_Arr[i]][j]==0){
                //completed_order_number=get(i).add0,completed_order_number.get(i).get(0).counter;
                completed_order_number[i][0]+=counter;
                break;
            }
            product_stock_num[temporary_item_number[Customer_Num_Arr[i]][j]]+=temporary_amount_product_num.get(Customer_Num_Arr[i]).get(j);
            temporary_item_number[Customer_Num_Arr[i]][j]=0;
            temporary_amount_product_num.get(Customer_Num_Arr[i]).add(j,0);
            counter++;
        }
        ((Customer)person).setCompleted_order_number(completed_order_number);
        ((Customer)person).setTemporary_item_number(temporary_item_number);
        ((Customer)person).setTemporary_amount_product_num(temporary_amount_product_num);
    }
    for(int k=1;k<7;k++){
        for(int i=0;i<7;i++){
            for(int j=0;j<Office_Chairs.get(i).size();j++){
                product_stock_num[k]=Office_Chairs.get(i).get(j);
                k++;
            }
        }
        for(int i=0;i<5;i++){
            for(int j=0;j<Office_Desks.get(i).size();j++){
                product_stock_num[k]=Office_Desks.get(i).get(j);
                k++;
            }
        }
        for(int i=0;i<10;i++){
            for(int j=0;j<Meeting_Tables.get(i).size();j++){
                product_stock_num[k]=Meeting_Tables.get(i).get(j);
                k++;
            }
        }
        for(int i=0;i<Bookcases.size();i++){
            product_stock_num[k]=Bookcases.get(i);
            k++;
        }
        for(int i=0;i<Office_Cabinets.size();i++){
            product_stock_num[k]=Office_Cabinets.get(i);
            k++;
        }
        setOffice_Chairs(Office_Chairs);
        setOffice_Desks(Office_Desks);
        setMeeting_Tables(Meeting_Tables);
        setBookcases(Bookcases);
        setOffice_Cabinets(Office_Cabinets);
    }
}
    
```

$T_1(w)(n) = \Theta(n)$
 $T_1(b)(n) = \Theta(1)$
 $T_1(n) = O(n)$
 $O(m, k) > O(n)$
 $\Rightarrow O(k, m, n)$
 (All loop)

$T_2(w)(m) = \Theta(m)$
 $T_2(b)(m) = \Theta(1)$
 $T_2(m) = O(m)$
 (inner loop)
 $T_2(m), T_3(k)$
 $= O(m, k)$

$T_3(k) = O(k)$
 $\Theta(1)$

$O(a^4)$
 $O(b^4)$
 $O(c^4)$
 $O(d^2)$
 $O(t^2)$

$T(n, m, k, a, b, c, d, t) =$

$\Rightarrow O(m, n, k) + O(a^4) + O(b^4) + O(c^4) +$
 $+ O(d^2) + O(t^2) + \Theta(1)$

$\Rightarrow O(m, n, k + a^4 + b^4 + c^4 + d^2 + t^2)$

ex) [

setOffice_Chairs(Office_Chairs);

setOffice_Desks(Office_Desks);

setMeeting_Tables(Meeting_Tables);

setBookcases(Bookcases);

setOffice_Cabinets(Office_Cabinets);

Other Method

```
public Customer Create New Account For Market Customer(String name,String surname,String e_mail,String password){
    Company person2=new Customer("default","default","default","default");
    customer_num=((Customer)person2).getRandom_special_customer_num();
    Company person=new Customer(name,surname,e_mail,password,customer_num);
    return ((Customer)person);
}
```

`public void SupplyOfficeChairs(int number, int num_of_model, int num_of_color){`
`KWSingleLinkedList<KWSingleLinkedList<Integer>> Office_Chairs=getOffice_Chairs();`
`Office_Chairs.add(new KWSingleLinkedList<Integer>());`
`Office_Chairs.get(num_of_model).add(num_of_color, number);`
`setOffice_Chairs(Office_Chairs);`
`}`

$\rightarrow O(1)$ $\rightarrow O(n)$ $\rightarrow O(n^2)$ $\rightarrow O(1)$

$\left. \begin{array}{l} \rightarrow O(1) \\ \rightarrow O(n) \\ \rightarrow O(n^2) \end{array} \right\} O(n^2)$

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

`public void Supply_Office_Desks(int number, int num_of_model, int num_of_color){`
`KWSingleLinkedList<KWSingleLinkedList<Integer>> Office_Desks=getOffice_Desks();`
`Office_Desks.add(new KWSingleLinkedList<Integer>());`
`Office_Desks.get(num_of_model).add(num_of_color, number);`
`setOffice_Desks(Office_Desks);`
`}`

$\rightarrow \theta(1)$
 $\rightarrow \theta(m)$
 $\rightarrow \theta(m^2)$
 $\rightarrow \theta(1)$

$\boxed{\theta(1) \quad \theta(m^2) \quad T(m) = \theta(m^2)}$

`public void Supply Meeting Tables(int number,int num_of_model,int num_of_color){`
`KMSingleLinkedList<KMSingleLinkedList<Integer>> Meeting_Tables=getMeeting_Tables();` → $O(1)$
`Meeting_Tables.add(new KMSingleLinkedList<Integer>());` → $O(1)$
`Meeting_Tables.get(num_of_model).add(num_of_color,number);` → $O(n^2)$
`setMeeting_Tables(Meeting_Tables);` → $O(1)$

$\left. \begin{array}{l} O(1) \\ O(1) \\ O(n^2) \end{array} \right\} O(n^2)$

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

`public void Supply_Bookcases(int number,int num_of_model){`
`this.LinkedList<Integer> Bookcases=getBookcases();`
`Bookcases.add(num_of_model,number);`
`setBookcases(Bookcases);`
`}`

Handwritten analysis:
 - `this.LinkedList<Integer> Bookcases=getBookcases();` → $O(1)$
 - `Bookcases.add(num_of_model,number);` → $O(b)$
 - `setBookcases(Bookcases);` → $O(1)$
 Total complexity: $T(b) = O(b)$

public void SupplyOfficeCabinets(int number, int num_of_model){

KWSingleLinkedList<Integer> Office_Cabinets=getOfficeCabinets(); → OC1

Office_Cabinets.add(num_of_model,number); → OCC

setOffice_Cabinets(office_cabinets); → OCC

→ OC1 } OCC

TCC = OCC

