```java
public void choiceCatalog(int name,int model,int color,int amount,Branch branch)throws RuntimeException{

    //int model,color,amount;
    System.out.println("PLEASE ONLY WRITE THE NUMBER OPPOSITE OF THE ITEM YOU WANT TO CHOOSE.( chair = '0'

    //Scanner input = new Scanner(System.in);
    //name = input.nextInt();

    switch(name){  // we choice the type of furniture type in this switch-case.

        case 0:
            pocket[pocketCount][0] = "chair";
        break;

        case 1:
            pocket[pocket[pocketCount][0] = "desk";
        break;

        case 2:
            pocket[pocketCount][0] = "table";
         break;

        case 3:
            pocket[pocketCount][0] = "bookcase";
        break;

        case 4:
            pocket[pocketCount][0] = "cabinets";
        break;

        default:
            throw new RuntimeException("Invalid Value!!!");
    }
```

$\Theta(1)$

```java
//In this place we entered the furniture features.
System.out.println("PLEASE ENTER THE MODEL YOU WILL CHOOSE AS A NUMBER.(FOR EXAMPLE Model1 = '1', Model2 = '2' ...):  ");

//model = input.nextInt();

pocket[pocketCount][1] = "Model " + Integer.toString(model);

System.out.println("PLEASE ENTER THE COLOR YOU WILL CHOOSE AS A NUMBER.(FOR EXAMPLE Color1 = '1', Color2 = '2' ...): ");
//color = input.nextInt();

pocket[pocketCount][2] = "Color " + Integer.toString(color);

System.out.println("Please determine the amount of furniture ");

//amount = input.nextInt();

pocket[pocketCount][3] = Integer.toString(amount);

branch.decrease(name, model, color, amount);


pocketCount++;

extendArray();
```

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

Time complexity = $\Theta(n)$

```java
public void add(int index, E anEntry) {
    if (index < 0 || index > size) {
    throw new ArrayIndexOutOfBoundsException(index);
    }
    if (size == capacity) {
    reallocate();
    }
    // Shift data in elements from index to size - 1
    for (int i = size; i > index; i--) {
    theData[i] = theData[i - 1];
    }
    // Insert the new item.
    theData[index] = anEntry;
    size++;
    }
```

$O(n)$

$T_b = \Theta(1)$
$T_w = \Theta(n)$
$O(n)$

$\Theta(1)$

-For loop in this method has worst and best condition best condition is Θ(1) the worst condition is Θ(n) so this method time complexity is O(n).

$O(n)$

```java
public class Furniture{

    private int[][] stock;

    public int getTotal(){

        int sum=0;

        for(int i=0;i<stock.length;i++){
            for(int j=0;j<stock[i].length;j++){
                sum += stock[i][j];
            }
        }

        return sum;
    }
```

$\Theta(1)$

n

n

$\Theta(1)$

$\Theta(n^2)$

$\Theta(1)$

time complexity is $\Theta(n^2)$ because of nested for loops.

```java
*/
public void rePlace(){


    branch.add(0,new Branch(shrink));
    branch.add(1,new Branch(shrink2));
    branch.add(2,new Branch(shrink3));
    branch.add(3,new Branch(shrink4));
}
```

$O(1)$ $O(1)$ $O(1)$ $O(1)$

$O(1)$

```java
@Override
public void Register(){
    String nameSur;
    String eMail;
    String passW;
    Scanner in = new Scanner(System.in);
    Customer object;

            System.out.println("Enter Customer name:");
            nameSur = in.next();
            System.out.println("Enter Customer Eposta: ");
            eMail = in.next();
            System.out.println("Enter Customer Password: ");
            passW = in.next();
            this.setAccount(nameSur,eMail,passW);
}
```

$\Theta(1)$

$\Theta(1)$

$\Theta(1)$ = Time complexity

```
*/
public void add(E obj){


    if(size==0){
        storage.add(size/MAX_NUMBER,new KWArrayList<E>());
    }


    ListIterator<KWArrayList<E>> linkArray =storage.listIterator();
    KWArrayList<E> temp = linkArray.next();



    while(linkArray.hasNext()){
        temp = linkArray.next();
    }
    /*
    *The number of elements in each ArrayList should be less than MAX
    */
    if(MAX_NUMBER == temp.size()){
        storage.add(size/MAX_NUMBER,new KWArrayList<E>());

        linkArray = storage.listIterator();
        temp = linkArray.next();

        while(linkArray.hasNext()){
            temp = linkArray.next();
        }

    }

    temp.add(obj);
    size++;
}
```

O(1)

O(n)

O(1)

O(n)

O(n)

O(1)

O(n) → time complexity

```java
@Override
public void stockTotal(){


shrink[0] = new int[7][5];
shrink[1] = new int [5][4];
shrink[2] = new int [10][4];
shrink[3] = new int [12][1];
shrink[4] = new int [12][1];

    int[] array = new int[5];


    for(int i=0;i<shrink.length;i++){
        for(int j=0;j<shrink[i].length;j++){
            for(int k=0;k<shrink[i][j].length;k++){
                shrink[i][j][k] = 5;
            }
        }
    }
```

$\Theta(1)$

$\Theta(1)$

$\Theta(i \cdot j \cdot k)$

```java
public E get(int index) {

    int x,y;

    x = index / MAX_NUMBER;
    y = index % MAX_NUMBER;

    ListIterator<KWArrayList<E>> linkArray =storage.listIterator();
    KWArrayList<E> temp = linkArray.next();


    if (index < 0 || index >= size) {
        throw new ArrayIndexOutOfBoundsException(index);
    }

    for(int i=0;i<x;i++){
        temp = linkArray.next();

    }

    return temp.get(y);
}
```

$\Theta(1)$

n

$T_b = \Theta(1)$

$T_w = \Theta(n)$

$O(n)$

$O(n)$

Time complexity = $O(n)$

```java
@Override
public void Welcome(){
    System.out.println();
    System.out.println("*******WELCOME TO THE FURNITURE STORE:*********");
    System.out.println("***PLEASE SELECT THE NUMBER FOR MENU:***");
    System.out.println(" 1)-Press 1 to New Register: ");
    System.out.println(" 2)-Press 2 Already Customer: ");
    System.out.println(" 3)-Press 3 if you are Branch Employee: ");
    System.out.println(" 4)-Press 4 if you are Administrator: ");
    System.out.println(" 0)-Press 0 if you want to exit the store: ");
}
```

$\Theta(1)$

$\Theta(1)$

```java
    /**
    *Call the manager if there is need product in store. .
    */
  public void callManager() { System.out.println(" PLEASE PROVIDE PRODUCT. "); }

}
```

$\Theta(1)$

```java
//model and color special of furniture and I keep the this information in this indexes.
public void decrease(int model,int color,int count){

    if(stock[model][color]-count >= 0){
        stock[model][color] -=count;
    }

}
```

$\Theta(1)$

$\Theta(1)$

```java
/**
 * To remove employees from the store.
 */
public void exit() { Administrator.countEmployee--; }

/**
 * @return count of employee.
 */
public static int numberEmployee() { return Administrator.countEmployee; }

/**
 * To remove branch from the system.
 */
public void quitBranch() { Administrator.countBranch--; }

/**
 * @return count of Branch.
 */
public static int numberBranch() { return Administrator.countBranch; }
```

$\Theta($

$\Theta(1)$

$1)$ $\Theta(1)$

$\Theta(1)$

$\Theta(1)$

```java
*/
public String amountPocket(){
    StringBuilder a = new StringBuilder();

    for(int i=0;i<pocket.length-1;i++){

        a.append(" Name: ");
        a.append(pocket[i][0]);
        a.append(", Model: ");
        a.append(pocket[i][1]);
        a.append(", Color: ");
        a.append(pocket[i][2]);
        a.append(", Amount: ");
        a.append(pocket[i][3]);
        a.append("\n");
    }

    return a.toString();
}
```

n

$\Theta(1)$

$\Theta(n)$

$\Theta(n)$

Answer is $\Theta(n) + \Theta(n) = \Theta(n)$

```java
*/
public int getStock(String stock){

    switch(stock){

        case "chair":
        return theData.get(0).getTotal();
                O(n)    Θ(n^2)

        case "desk":
        return theData.get(1).getTotal();
                O(n)    Θ(n^2)

        case "table":
        return theData.get(2).getTotal();
                O(n)    Θ(n^2)

        case "bookcase":
        return theData.get(3).getTotal();
                O(n)    Θ(n^2)

        case "cabinet":
        return theData.get(4).getTotal();
                O(n)    Θ(n^2)

        default:
            throw new NoSuchElementException();
    }
}
```

Θ(n^2) because getTotal method's time complexity is Θ(n^2) it has nested for loop and I assume that for size is normally get() method in linkedlist is O(n) but Θ(n^2) bigger than O(n) so answer is Θ(n^2).

$\Theta(n^2)$

```java
/**
*Here I use the 3D array to downsize to a single dimension,
*and here I use HybridList for the insertion process.
*/
public Branch(int [][][] arrayStock){


    if(arrayStock.length > 5 || arrayStock == null)
        throw new IllegalStateException();

        theData = new HybridList<Furniture>();


        theData.add(new officeDesk(arrayStock[0]));



        theData.add(new officeBookcase(arrayStock[1]));


        theData.add(new officeChair(arrayStock[2]));


        theData.add(new officeCabinet(arrayStock[3]));


        theData.add(new officeTable(arrayStock[4]));


}
```

Time complexity is O(1) because I use the linked list while I writing add() method in hybrideList.

And I explain the other page add() method in detail.

O(1)

```java
/**
* In this function, I am expanding the size of the directory used for our shopping cart.
*/

private void extendArray(){

    String[][] temp;

    temp = pocket;

    pocket =  new String[temp.length+1][4];

    for(int i=0;i<temp.length;i++){
        pocket[i] = temp[i];
    }
}
```

$\Theta(1)$

$\Theta(1)$

n

$\Theta(1)$

This method's time complexity is Θ(n) because of for loop and I assume that temp.length is 'n'.

$\Theta(n)$