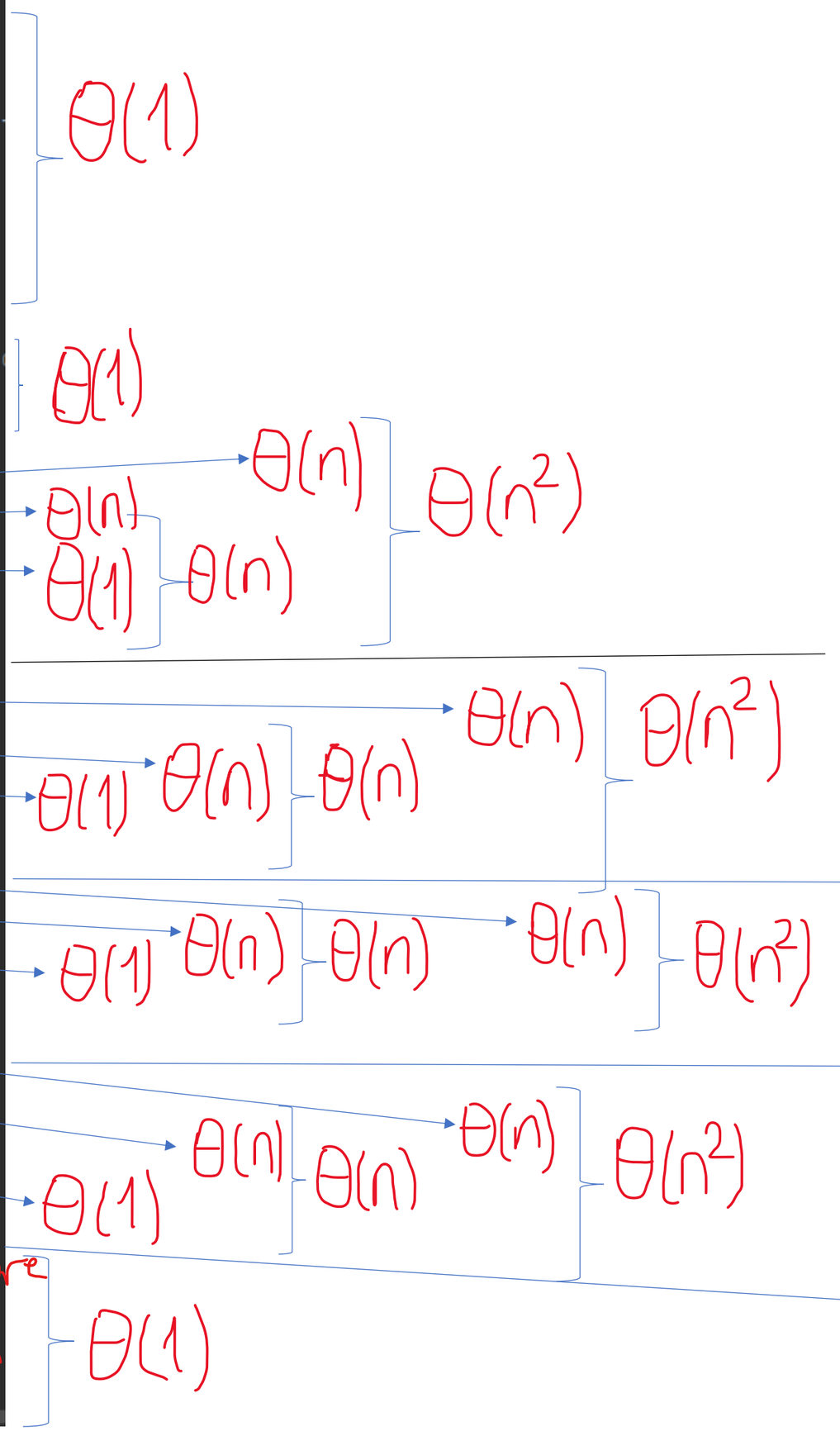
PART 1:

1-)SEARCHING A PRODUCT:

```
try{
    switch(select){

        case 1:
            System.out.println("Please write that which you furniture you want "+"FOR EXAMPLE: "+
            System.out.println();
            System.out.println("PLEASE ENTER THE FURNITURE YOU WANT WHICH IN CATALOG:");

        Scanner input3 = new Scanner(System.in);
        shopping = input3.nextLine();
    try{
        switch(shopping){   //Here, if the customer chooses case 1, she/he will be able to see ho

            case "chairs":

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

            for (int i = 0; i < a.Branch[1][0].length ; i++){
                for(int j=0;j<a.Branch[1][0][i].length;j++){
                    sum2 += a.Branch[1][0][i][j];
                }
            }
            for (int i = 0; i < a.Branch[2][0].length ; i++){
                for(int j=0;j<a.Branch[2][0][i].length;j++){
                    sum3 += a.Branch[2][0][i][j];
                }
            }
            for (int i = 0; i < a.Branch[3][0].length ; i++){
                for(int j=0;j<a.Branch[3][0][i].length;j++){
                    sum4 += a.Branch[3][0][i][j];
                }
            }


            System.out.println("Sum of the office Chairs are in Branch1:" + sum);
            System.out.println("Sum of the office Chairs are in Branch2:" + sum2);
            System.out.println("Sum of the office Chairs are in Branch3:" + sum3);
            System.out.println("Sum of the office Chairs are in Branch4:" + sum4);
        break;
```



-I assigned this stock number as the default value for each color, number and store. For example, here I looked for the number of chairs in which store and used it in my function when adding other details.

So $\theta(n^2)$ = time complexity

2-)ADD/REMOVE PRODUCT:

n shows the number of types of products.

-(I myself assigned values to 4-dimensional arrays for the number of furniture and features for example(1-Branch Count, 2-furnitures, 3-model, 4- Color).

```
@Override
public void stockTotal(){


    int[] array = new int[5];

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


    for(int i=0;i<n;i++){

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

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

                Branch[0][i][j][k] = 5;
            }
        }

    }


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

    for(int i=0;i<n;i++){
        for(int j=0;j<Branch[1][i].length;j++){

            for(int k=0;k<Branch[1][i][j].length;k++){

                Branch[1][i][j][k] = 3;
            }
        }

    }
```

θ(1)

θ(1) θ(m) θ(m l) θ(m.l.n)

θ(1)

θ(1) θ(m) θ(m l) θ(m l n)

“For example, there are 3 of each color of each model of each product type of the 2nd store.

There are 5 of each color of each model of each product type of the first store.

-I throws numbers one after the other in order, how may stores there are.

θ(m l n) → time complexity

3-QUERYING THE PRODUCTS THAT NEED TO BE SUPPLIED:

```
*/
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

θ(1)

θ(n)

θ(n)

θ(1)

Thanks to this function, the number, color and model of the products purchased by the customers from the store are recorded, and new products are brought instead of the sold product.

I make purchases with the help of another function and record the purchases made through this function.

θ(n) → time complexity

Part 2) a) The O-notation is used to provide an upper bound, so when $f(n) = O(g(n)) \iff \exists c_1, n_0 : \forall n > n_0 \quad f(n) \leq c_1 \cdot g(n)$

So, saying that an algorithm A is at least $O(n^2)$ is meaningless

b) $\max(f(n), g(n)) = \Theta(f(n) + g(n))$

we consider $f(n) + g(n)$. In this if degree $f(n)$ > degree $g(n)$ so degree $(f(n)+g(n))$ = degree $(f(n))$ and if degree $g(n)$ > degree $f(n)$ then degree $(f(n)+g(n))$ = degree $g(n)$.

$\Theta(f(n)+g(n)) = \max$ [highest degree of $f(n)$, highest degree of $g(n)$]

$\Theta(f(n) + g(n)) = \max[f(n), g(n)]$

c) I) $$\lim_{n \to +\infty} \frac{2^{n+1}}{2^n} = \lim_{n \to +\infty} \frac{2^n \cdot 2}{2^n} = \lim_{n \to +\infty} 2 = 2$$

It is known that if $\lim_{n \to +\infty} \frac{f(n)}{g(n)} = C \in \mathbb{R}$ then $f(n) = \Theta(g(n))$

so $2^{n+1} = \Theta(2^n)$ so this is true

II) $2^{2n} = \Theta(2^n)$ → this statement is false I take the limit and

$$\lim_{n \to \infty} \frac{2^{2n}}{2^n} = \frac{2^{2n+1} \cdot \log(2)}{2^n \cdot \ln 2} = \infty$$ so this statement is false

If we find constant, it would true.

III) we have $f(n) \leq C \cdot n^2$ together with $C_1 \cdot n \leq g(n) \leq C_2 \cdot n$. We can $f \cdot g \leq C_0 n^3$. We cannot say $fg \in \Theta(n^3)$. Counter example: $f(n) = n$ and $g(n) = \frac{1}{n^2}$. Clearly $f \in \Theta(n)$ and $g \in O(n^2)$, but statement close to zero

So we can only indicate $\Theta(n) \cdot O(n^2) \subset O(n^3)$

And I disprove $f(n)^p g(n) = \Theta(n^4)$

Part 3-) $n^{1.01}$, $n\log^2 n$, 2^n, $\sqrt{n}$, $(\log n)^3$, $n2^n$, 3^n, 2^{n+1}, $5^{\log_2 n}$, $\log n$, Order of growth this functions.

I use L'Hospital rule for this order. So I will take the limits of the functions I want to compare and apply the L hospital rule, and list their growth according to the result

for exp $\lim_{x\to\infty} \frac{f(x)}{g(x)} = \begin{cases} 0, & \text{if } f \text{ is slower than } g \\ \infty, & \text{if } f \text{ is faster than } g \\ \text{const}, & \text{if } f \text{ and } g \text{ are comparable} \end{cases}$

and we know $O(1) < O(\log n) < O(n) < O(n\log n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$
(constant) < (Logarithmic) < (linear) < (log-linear) < (Quadratic) < (cubic) < (Exponential) < (Factorial)

$\lim_{n\to\infty} \frac{(2^n+1)'}{(2^n)'} = \frac{2.2^n.\ln 2}{2^n.\ln 2} = 2$ so $\to 2^n+1 = 2^n$

$\lim_{n\to\infty} \frac{(3^n)'}{(n.2^n)'}$, $= \infty \to$ it is infinite in this expression when we go towards infinity
so $3^n > n.2^n$

$\lim_{n\to\infty} \frac{(n.2^n)'}{(2^{n+1})'} = \frac{2^n + n.2^n.\ln 2}{2.2^n.\ln 2} = \lim_{n\to\infty} \frac{1+n.\ln 2}{2.\ln 2} = +\infty$ so $n.2^n > 2^{n+1}$

$\lim_{n\to\infty} \frac{((\log n)^3)'}{(\log n)'} = \frac{3\log^2(n)}{\frac{n}{1}} = \lim_{n\to\infty} 3\log^2(n) = +\infty$

$\lim_{n\to\infty} \frac{(\sqrt{n})'}{((\log n)^3)'} = \frac{\frac{1}{2\sqrt{n}}}{\frac{3\log^2(n)}{n}} = \frac{n}{2\sqrt{n}.3\log^2(n)} \Rightarrow \infty$, so $\sqrt{n} > (\log n)^3$

$\lim_{n\to\infty} \frac{(n.\log^2 n)'}{(\sqrt{n})'} = \frac{\log(n)(\log(n)+2)}{\frac{1}{2\sqrt{n}}} = \lim_{n\to\infty} 2\sqrt{n}.\log(n).(\log(n)+2) \Rightarrow \infty$, so $n.\log^2 n > \sqrt{n}$

$\lim_{n\to\infty} \frac{n^{1.01}}{n.\log^2 n} = \frac{1,01.n^{1,01}}{\log(n).(\log(n)+2)} = \infty$ ($n^{1.01}$ → linear; $\log(n).(\log(n)+2)$ → logarithmic)
linear > logarithmic, so $n^{1.01} > n\log^2 n$

$\lim_{n\to\infty} \frac{5^{\log_2 n}}{n^{1.01}} = \frac{n^{\log 5}}{n^{1.01}} = \frac{n^{2,3219}}{n^{1,01}} = \infty$ so $5^{\log_2 n} > n^{1,01}$

According to my explanations and proofs above, the order is as follows
$\log(n) < (\log n)^3 < \sqrt{n} < n\log^2 n < n^{1.01} < 5^{\log_2 n} < 2^n = 2^{n+1} < n.2^n < 3^n$

PART 4:

Minimum - Valued item:

1) public class Minimum{

public static void main (String[] args)

{

ArrayList<Integer> array = new ArrayList<>();

int minimum = array.get(0) ⟶ Θ(1)

int n = array.size() ⟶ Θ(1)

for i=1 to n ⟶ Θ(n)

If array.get(i) smaller than minimum ⟶ Θ(1)

minimum = array.get(i) ⟶ Θ(1)

Try Again

End If

End For

⇒ Θ(n) linear time

print "minimum"

Θ(n) → time complexity

}

Median

2-)

```
int n equal to array.size()
for i to n ------------------------> Θ(n)
  for j=i+1 to n ------------------> Θ(n)      Θ(n²)
    Initialize tmp to zero
      If array.get(i) greater than array.get(j)
        tmp equal to array.get(i)
        array.set(i, array.get(j)) → Θ(1)
        array.set(j, tmp) → Θ(1)              Θ(1)
        End if
        End for
        End for
  if n%2 equal to 1
      print "array.get(n/2)"                  Θ(1)
    else → Θ(1)
      print (array.get(n/2-1) + array.get(n/2)) / 2.0)
      End if
```

Θ(n²)

time complexity: $\Theta(n^2)$

3-) Two elements whose sum is equal to a given value.

```
int n equal to array.size()
Declare an integer variable i, j and sum.
for i=0 to n ------------------------> Θ(n)
    for j=i+1 to n <=> Tw = Θ(n), Tb = Θ(1) > O(n)
        if array.get(i) + array.get(j) = sum --> Θ(1)
          print array[i] + array[j] ---> Θ(1)
          return
          End if
        End for
    End for
```

O(n) × O(n) = O(n²) ; Θ(1) + Θ(1) = Θ(1) ; O(n²) × Θ(1) = O(n²)

time complexity = $O(n^2)$

4-) Merge two ordered array lists and to get a single list

Declare an integer variable i and j

Declare Integer new ArrayList array1, array2, array3

```
int n = array1.size()
int m= array2.size()                                   Θ(1)

for i=0 to n j=0 to m if i<n && j<m
    if array1.get(i) < array2.get(j)                   Θ(1)
        array3.add(array1.get(i))
        increment i

    else
        array3.add(array2.get(j))                      Θ(1)
        increment j
        End if
        End else
        End For                                        Θ(m.n)

while variable i less than array1.size()               Θ(n)
        array3.add(array1.get(i++))
    end while

while variable j less than array2.size()               Θ(m)
    array3.add(array2.get(j++))
    end while

    for i=0 to n+m                                     Θ(n+m)
        print array3.get(i)
        end FOR
```

Θ(m.n)

time complexity = Θ(m.n)

a) PART5 :

```
int p_1 (int array[]):
{
        return array[0] * array[2])
}
```

→ Θ(1)

Time complexity = Θ(1)

→ space complexity: O(1)

```
int p_2 (int array[], int n):
{
        Int sum = 0
        for (int i = 0; i < n; i=i+5)
                sum += array[i] * array[i])
        return sum
}
```

θ(1)
θ(n)
θ(1)
θ(1)

θ(n) = Time complexity

→ Space complexity: O(1)

c)

- I started from j=1

```
void p_3 (int array[], int n):

{

        for (int i = 0; i < n; i++)
                for (int j = 1; j < i; j=j*2)
                        printf("%d", array[i] * array[j])

}
```

θ(n)

θ(logn)

θ(1)

θ(n.logn) = time complexity

space complexity = O(1)

Space complexity is O(1) because there is no additional memory allocation that depends on some variable n.

d)

```
void p_4 (int array[], int n):
{
        If (p_2(array, n)) > 1000)        } Θ(n)
                p_3(array, n)             } Θ(n.logn)
        else
Θ(n)    { printf("%d", p_1(array) * p_2(array, n))
}                          Θ(1)     Θ(n)
```

$T_w = \Theta(n.\log n)$

$T_b = \Theta(n)$

Time complexity = $O(n.\log n)$

Space Complexity = $O(1)$

*Space complexity is O(1) because there is no additional memory allocation that depends on some variable n.