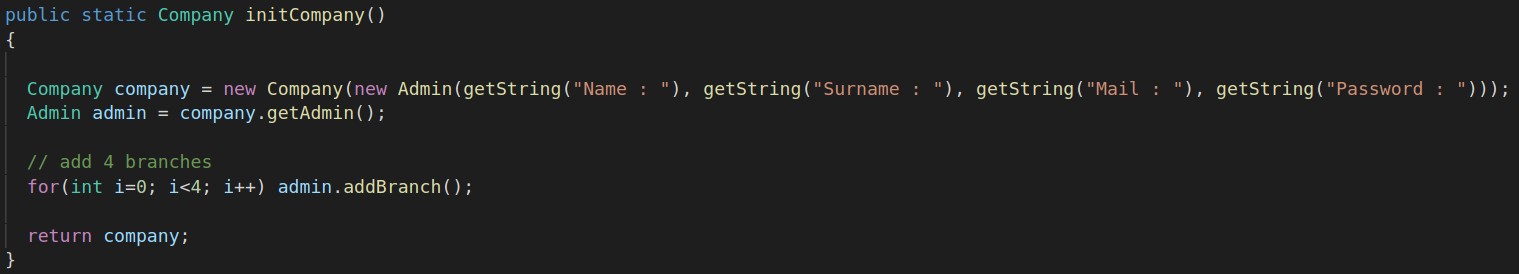
# GTU Department of Computer Engineering CSE 222/505 - Spring 2021

**Homework 3 Report**

# Barış Ayyıldız 1901042252

## SYSTEM REQUIREMENTS

At first, we should initialize a company.

Company Constructor takes an Admin object. Admin constructor takes name,

surname, email and and password as parameters. In this method I use addBranch

method to create 4 branches initially.

## Admin

Admins can add and delete branch/employee/customers.



When deleting a branch, branch id is required. When adding, software creates aunique id so it is not required



When adding a branch employee, an Employee object need to be passed.



When deleting it is only required to pass an employee id



Admin also can list all the employees or all the subscribers(customers)





## Employee

Branch employees can make in-shop sales. sales method takes customerId,

productId and amount of products as parameters

## Company Members

Both Admin and Employee class inherits from CompanyMembers class. They both have access to add and remove a customer.



Add and remove products



And list the products from all the branches that is out of stock



## Customer

User is able to subscribe to the company after creating a customer object.



This method sets customer object’s isSubscribed property to true. And sets the id to a unique integer value. Customer is able to buy online or buy in shop.



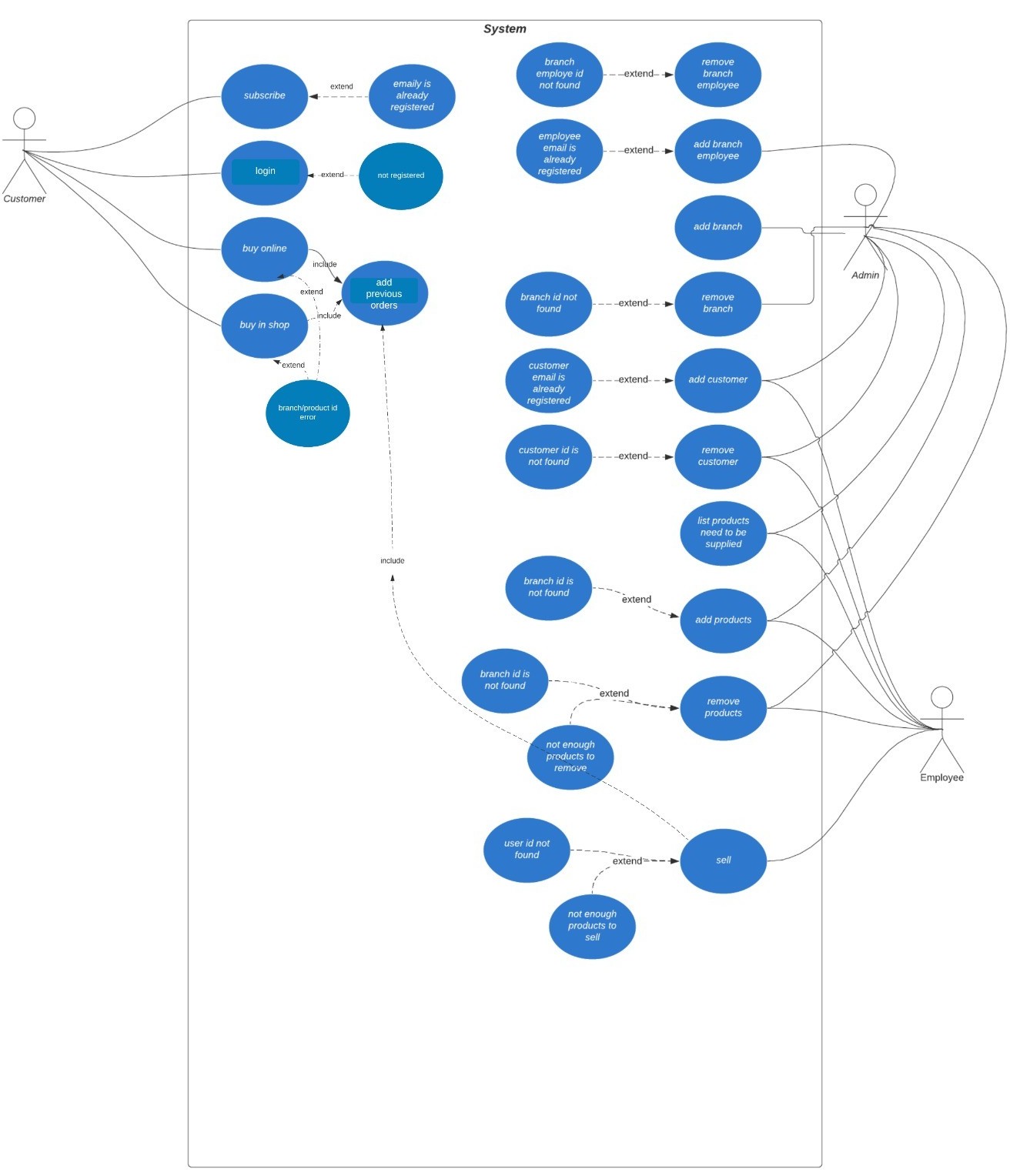
When customer buys online, they only pass productId and amount of products as a parameter. When buying in shop, branch id needs to be

passed as a parameter also. After these methods, new order is inserted to orderHistory array. This array holds all the previous orders of a customer.



A stock object needs to be send as a parameter.

## USE CASE AND CLASS DIAGRAMS

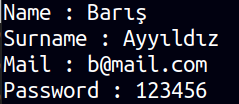


1. **PROBLEM SOLUTION APPROACH**

I have created a ListInterface<T> interface and List<T> class that implements it. Since we were not allowed to use any data structures other than arrays,I developed my own ArrayList like data structure. And it made the whole structure

very clean.

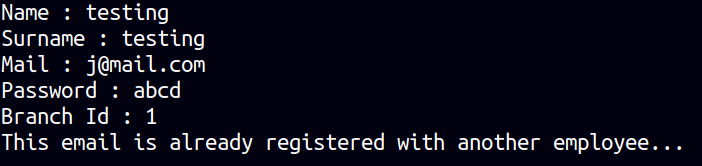
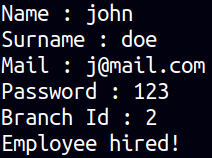
## TEST CASES

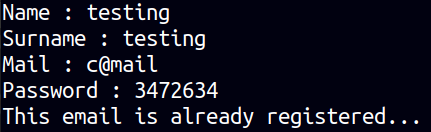
Create a company with administrator

## Admin

Remove branch

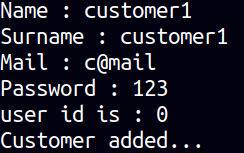


Add branch employee

Remove branch employee



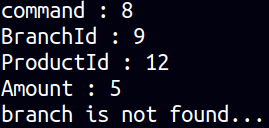
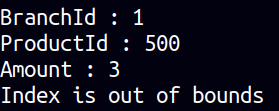
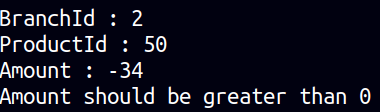
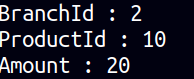
Add customer



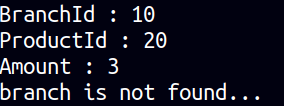
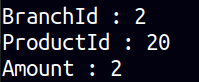
Remove customer

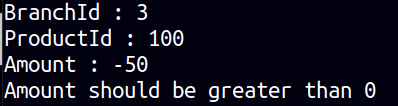


Add products

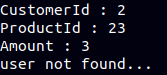


Remove products

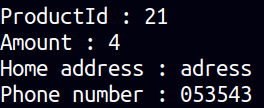




## Employee

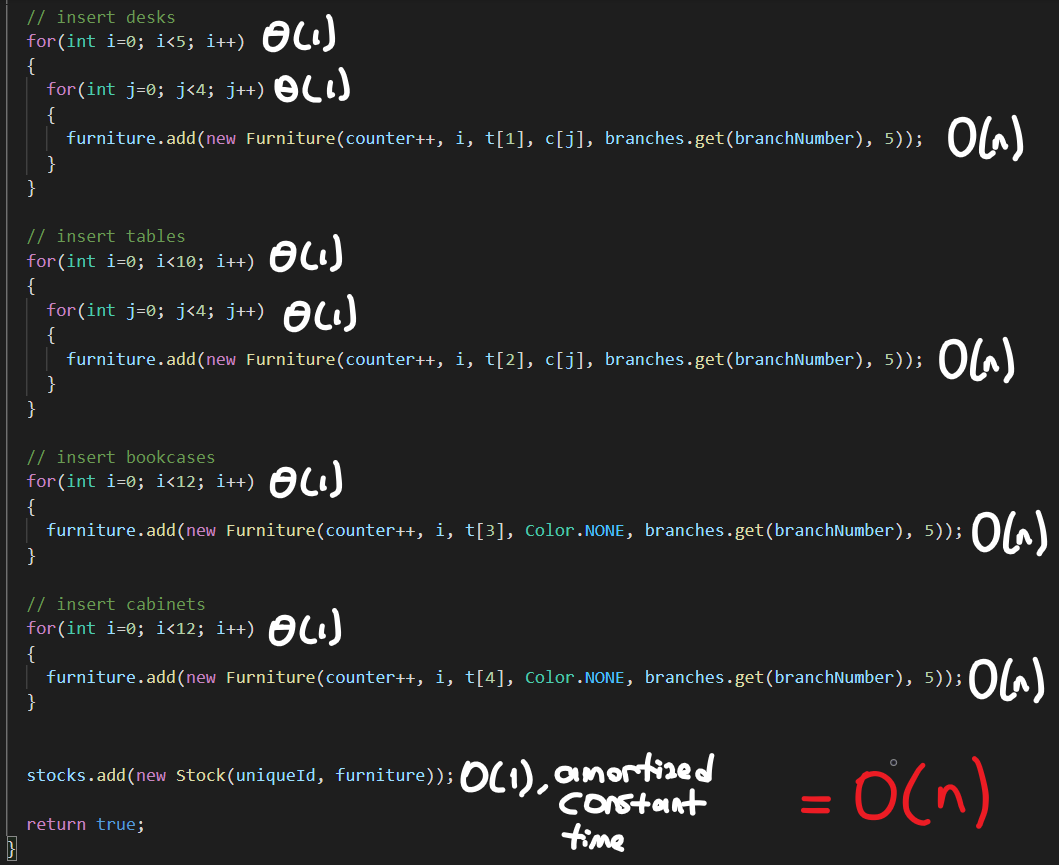
Sell

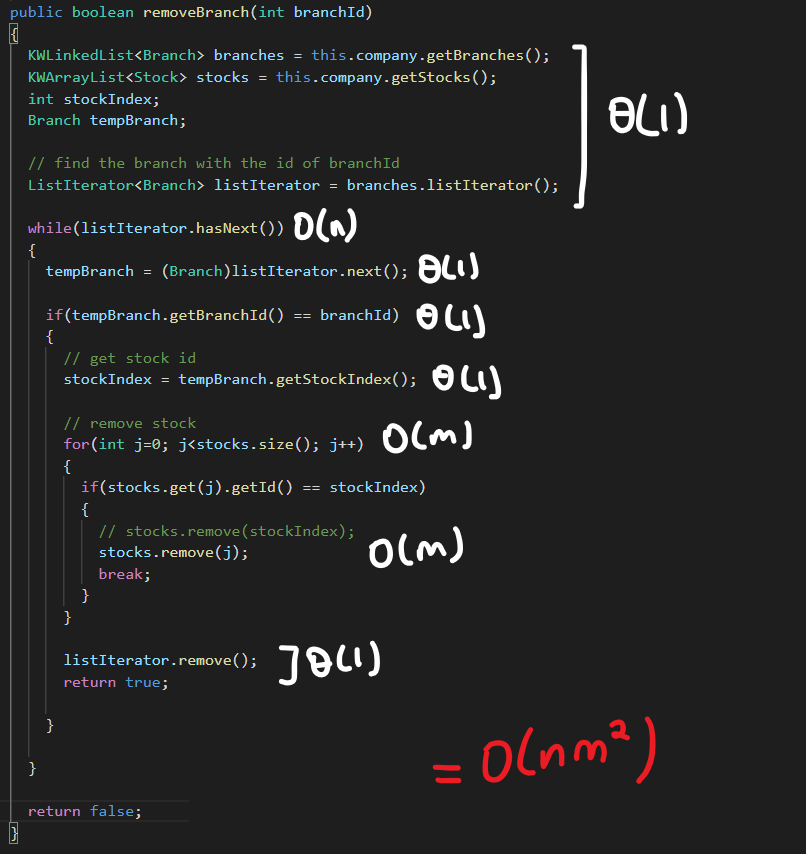
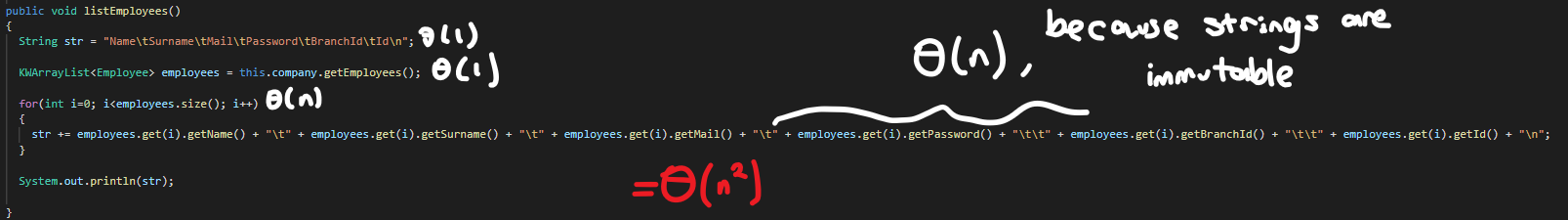
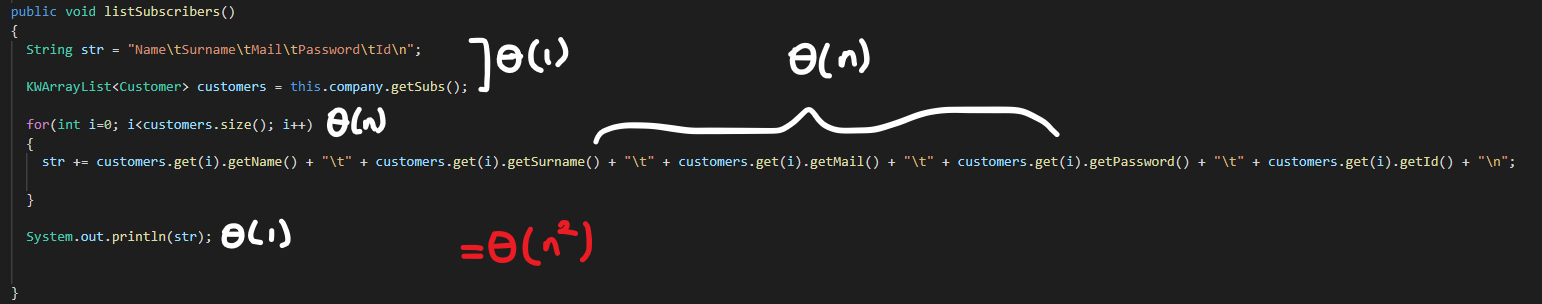
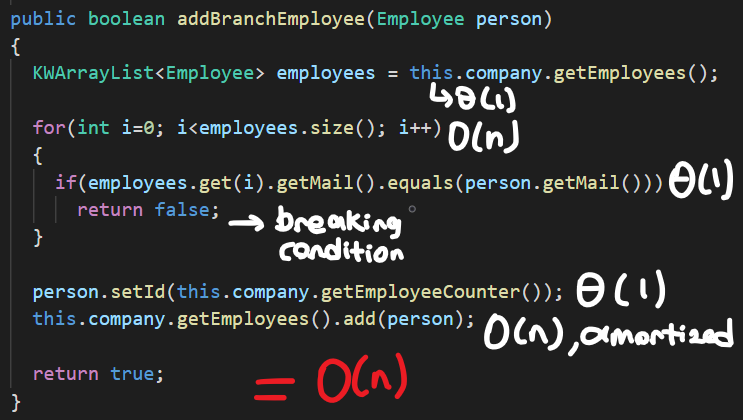
## Customer

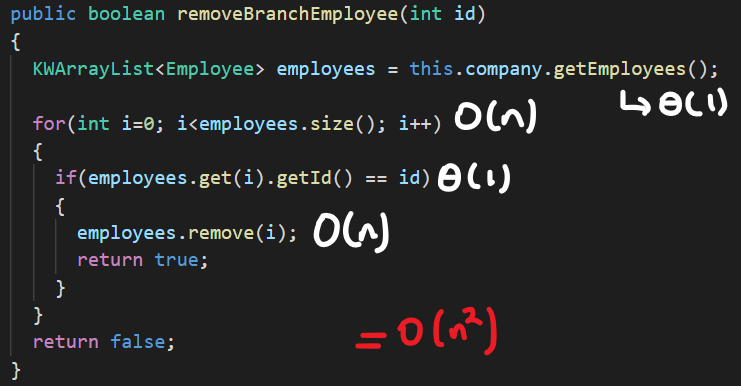
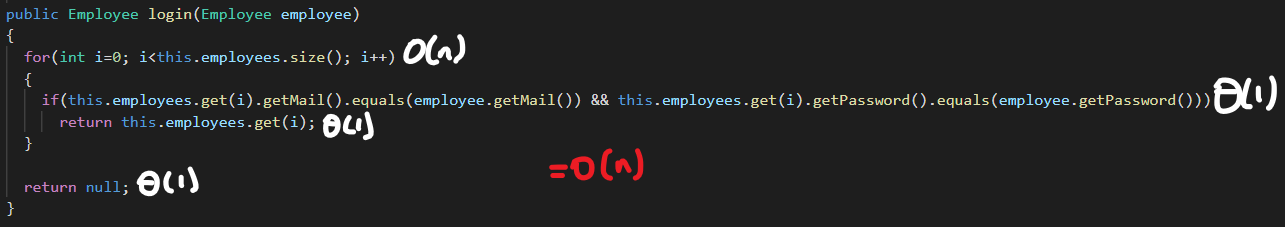
Buy online

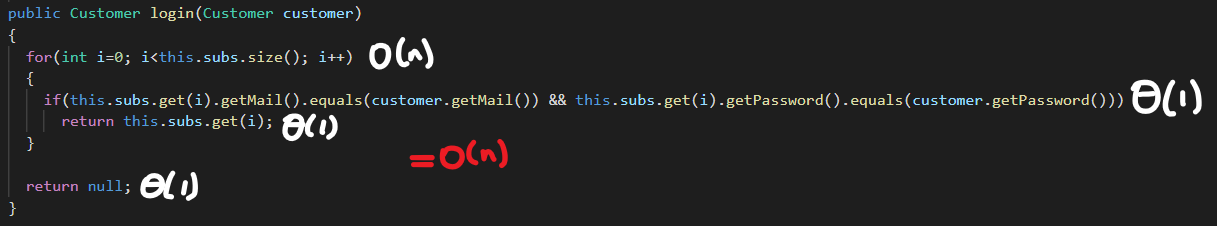
**Part 2 :**

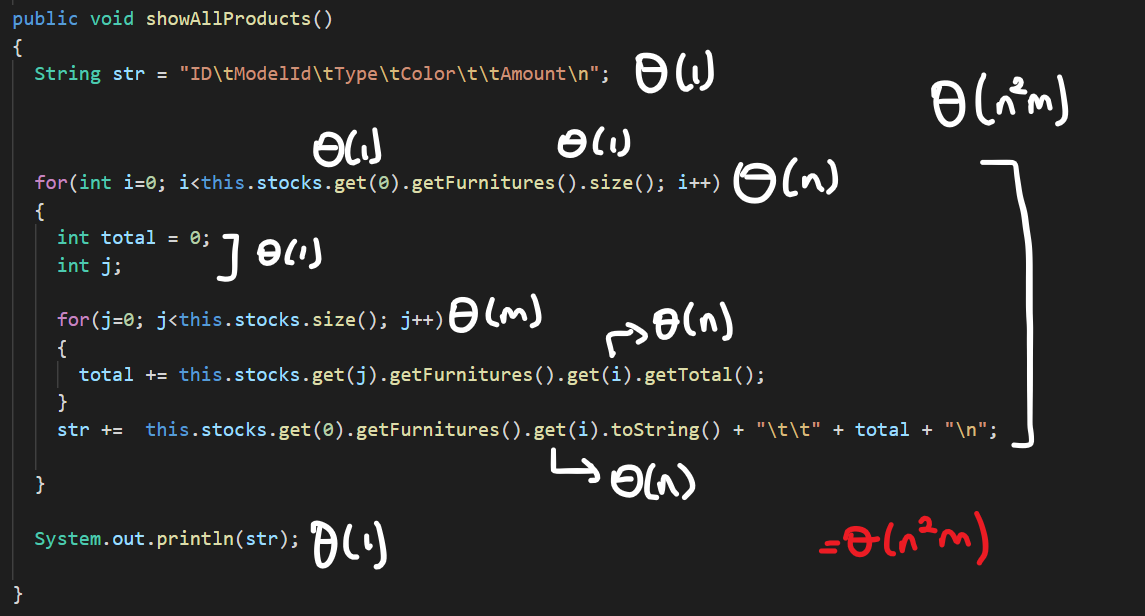
I did not implement the constant time getters and setters

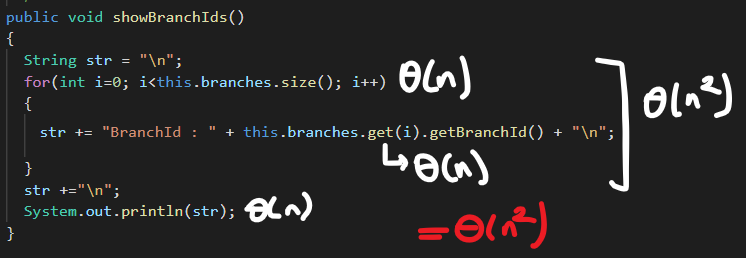


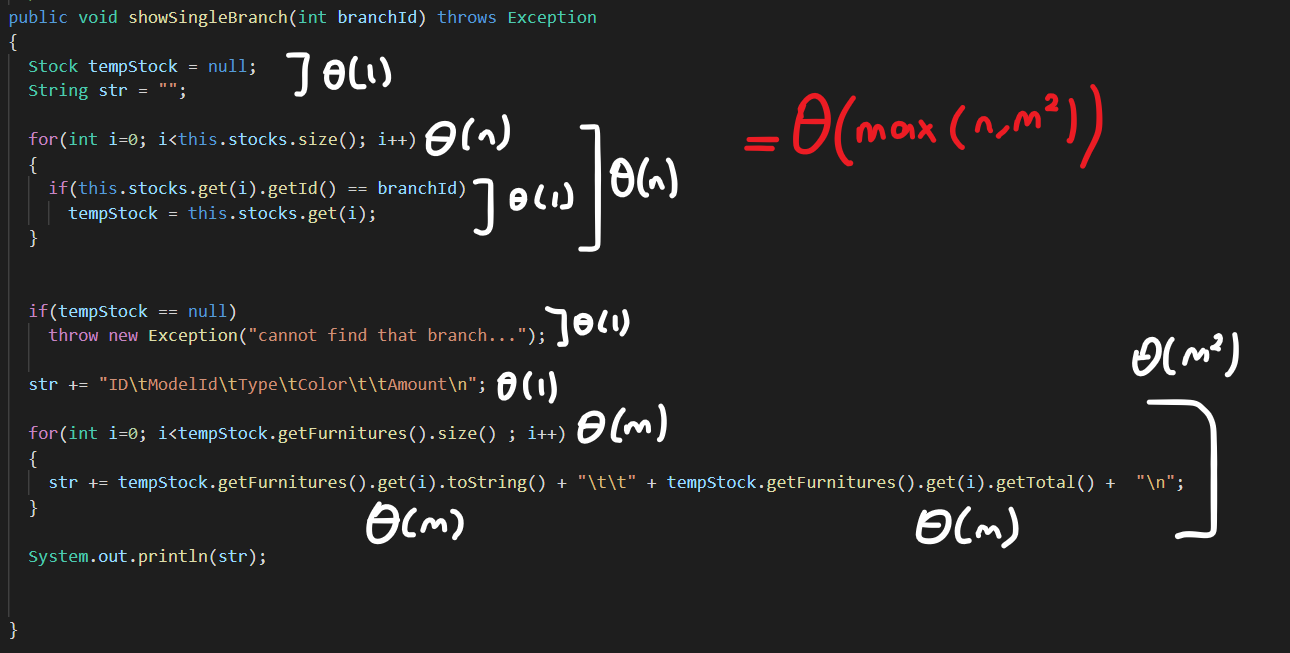
 n is the number of branches and m is the size of the individual stocks

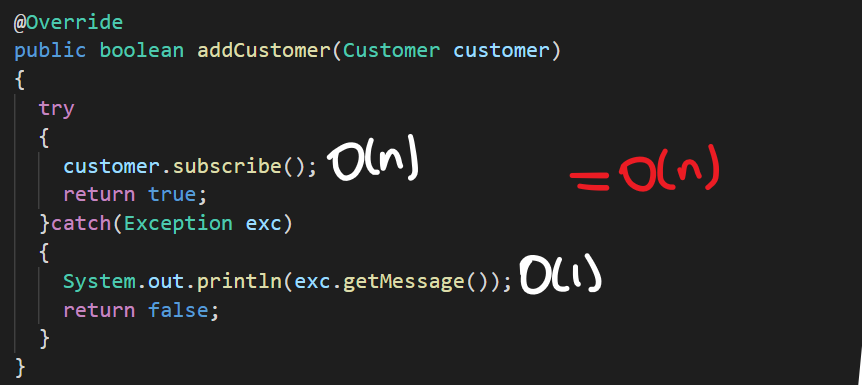


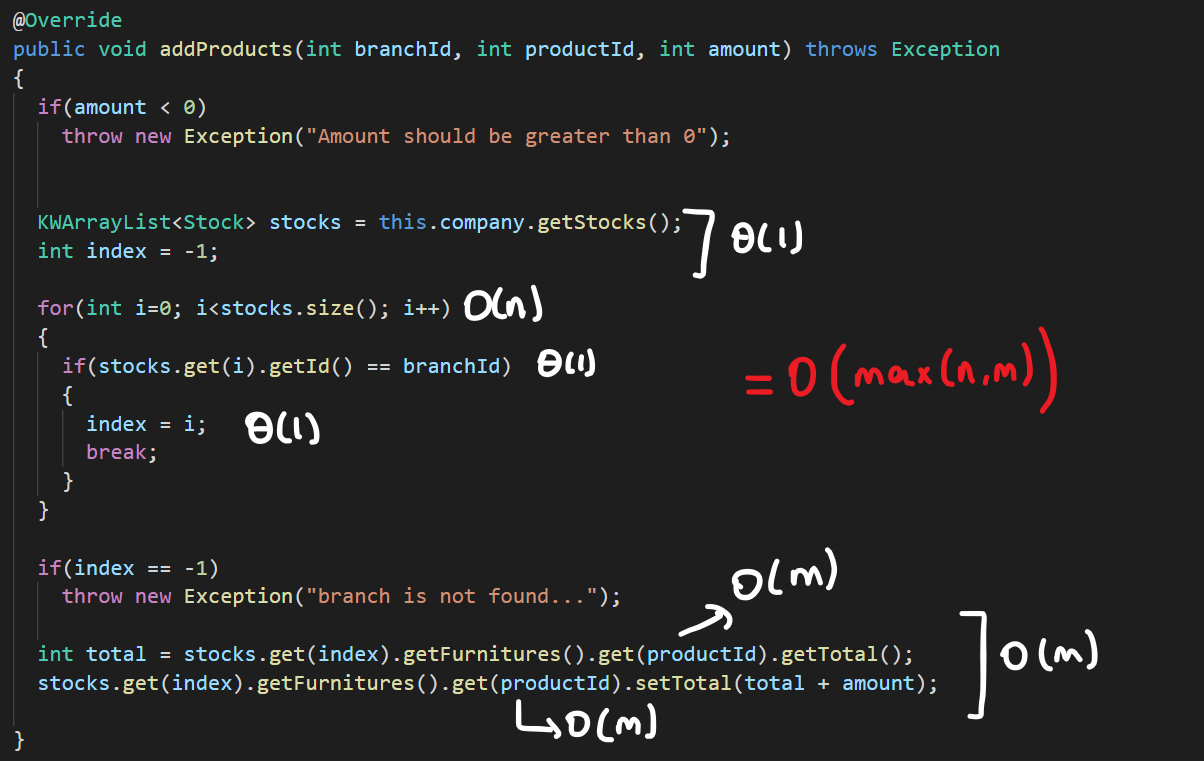
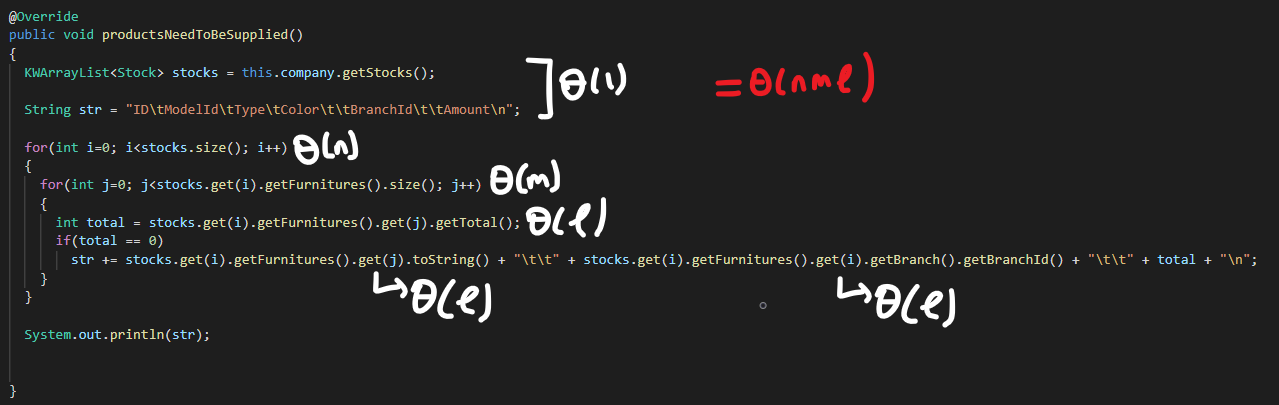


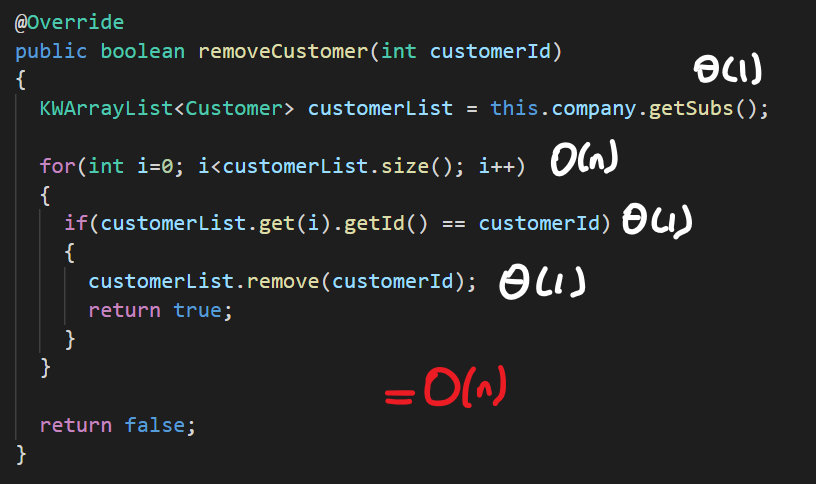
n is the number of furnitures and m is the number of stocks

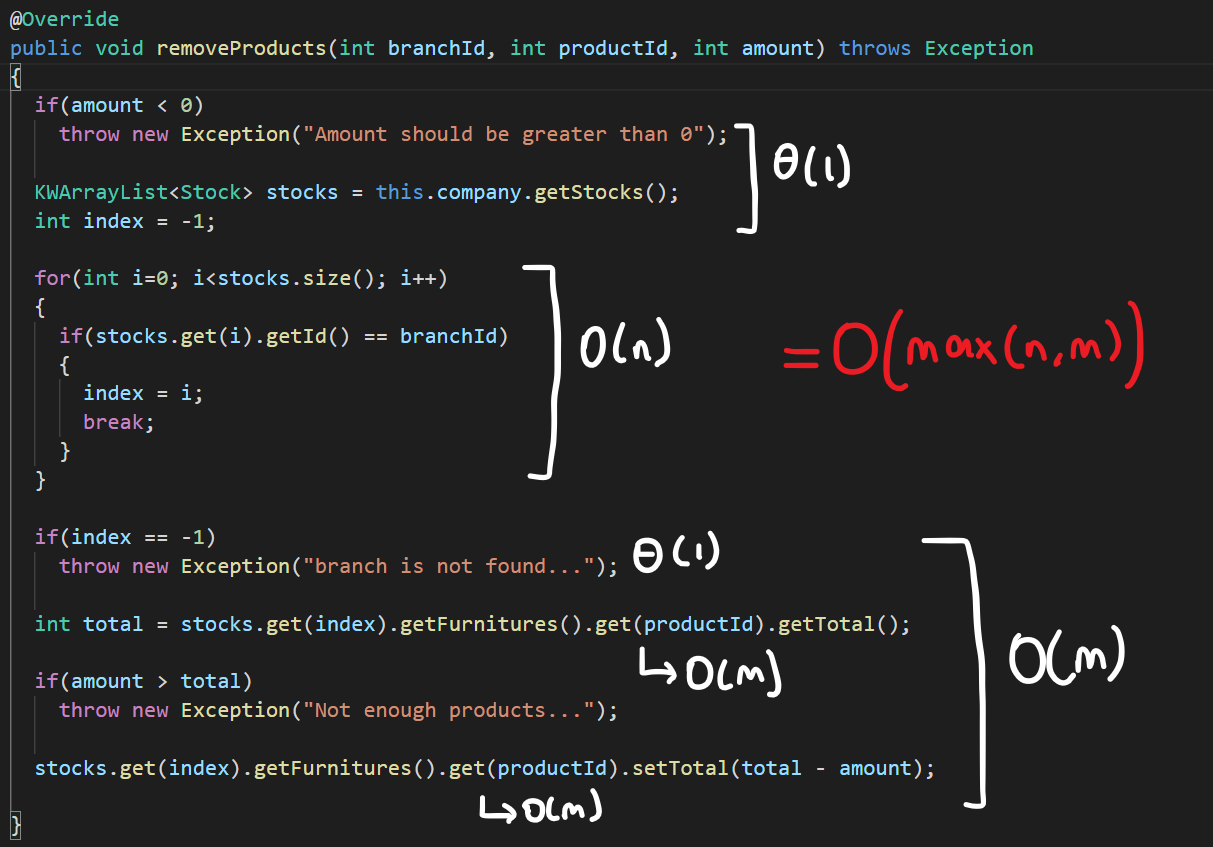


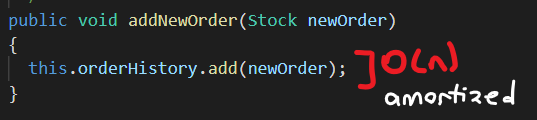
n is the number of stocks, m is the size of the furnitures

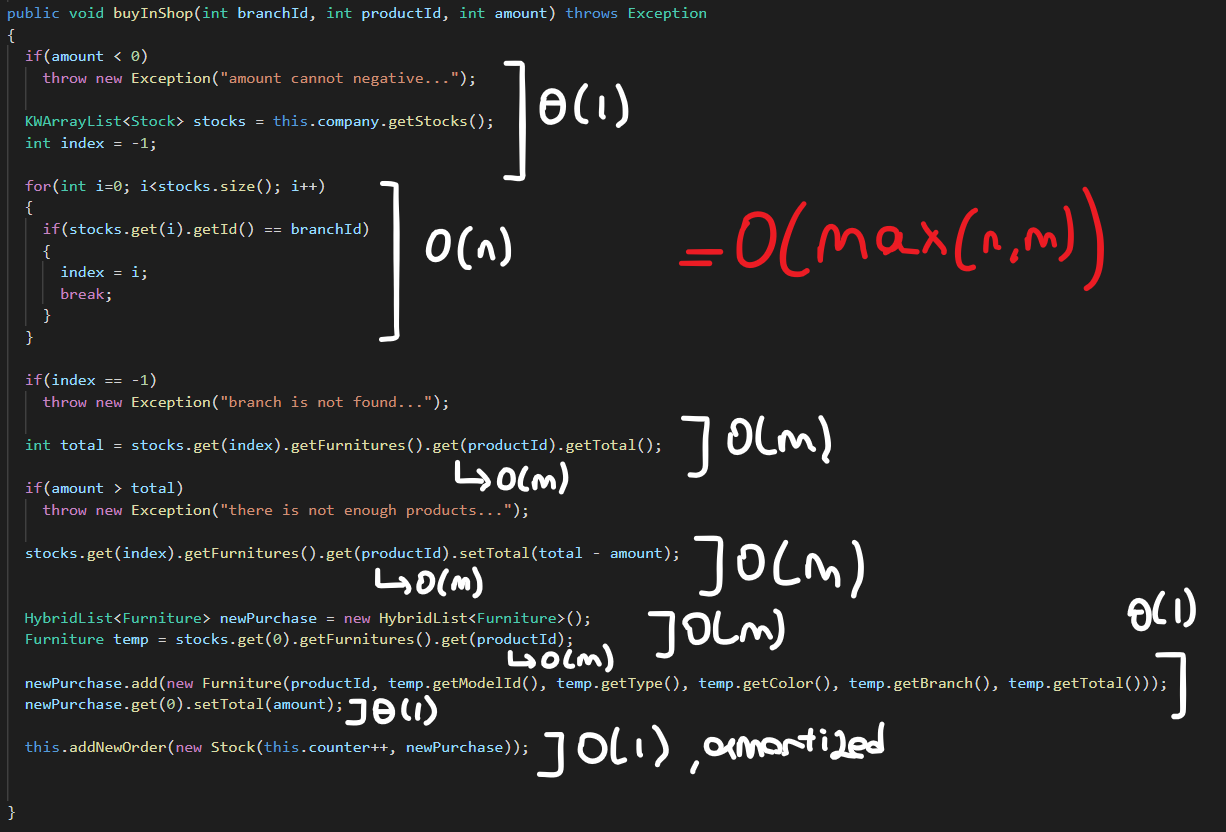


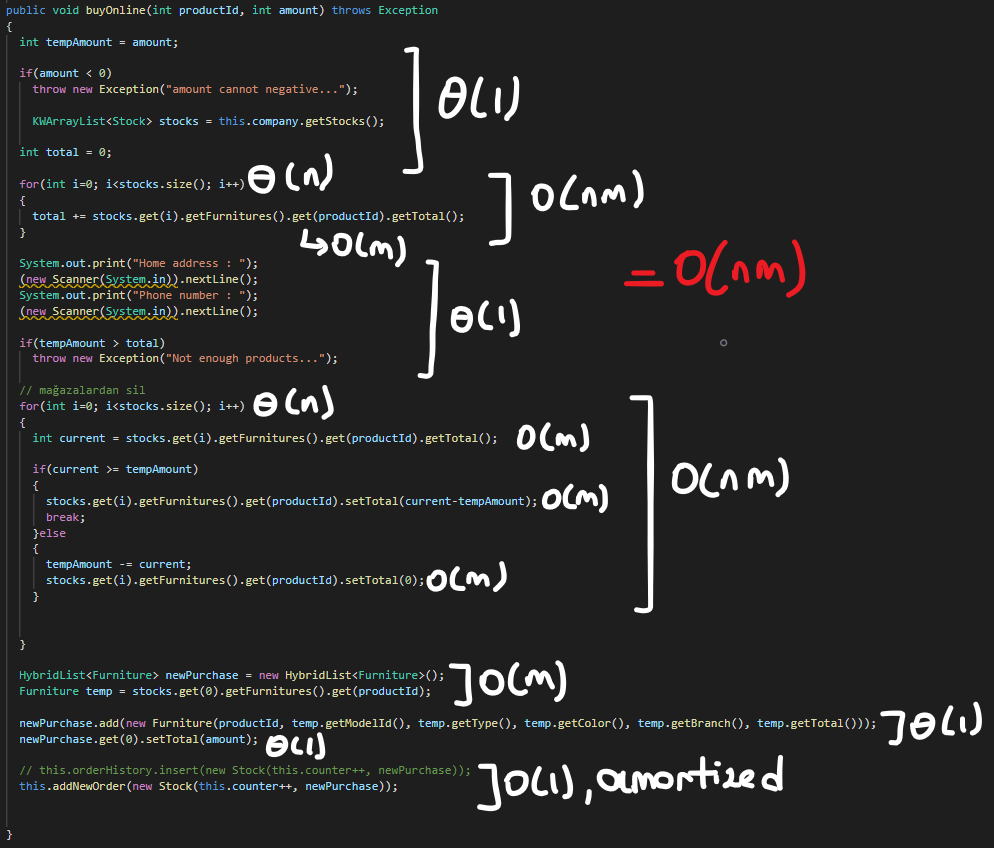


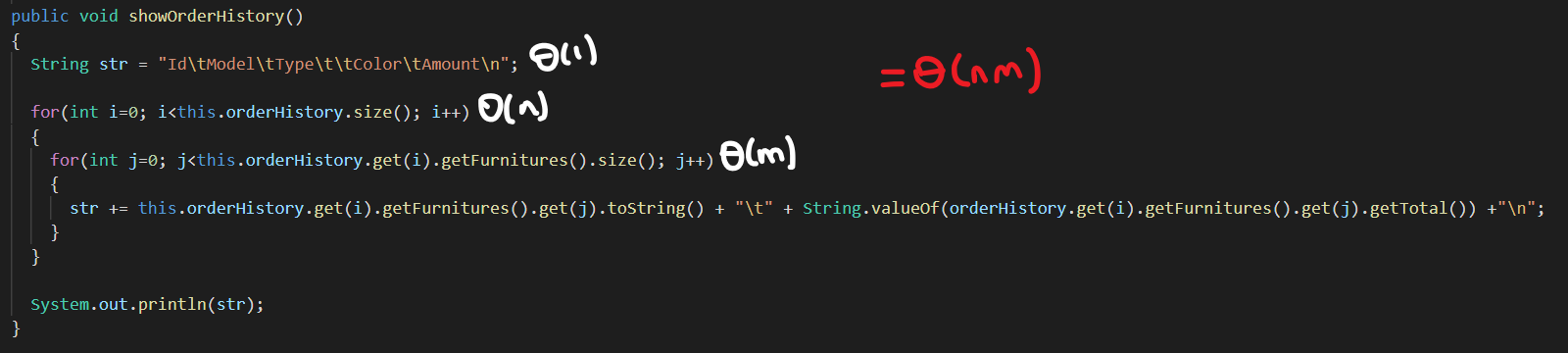


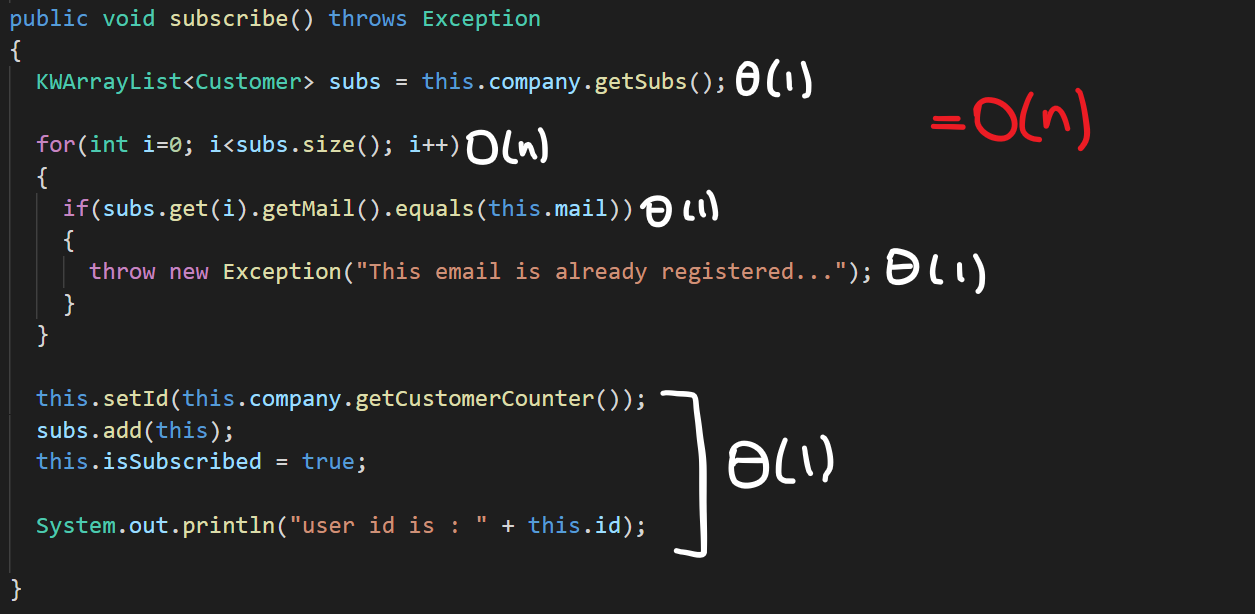
n is the number of stocks, m is the number of furnitures in a stock



 n is the number of stocks, m is the number of furnitures in a stock

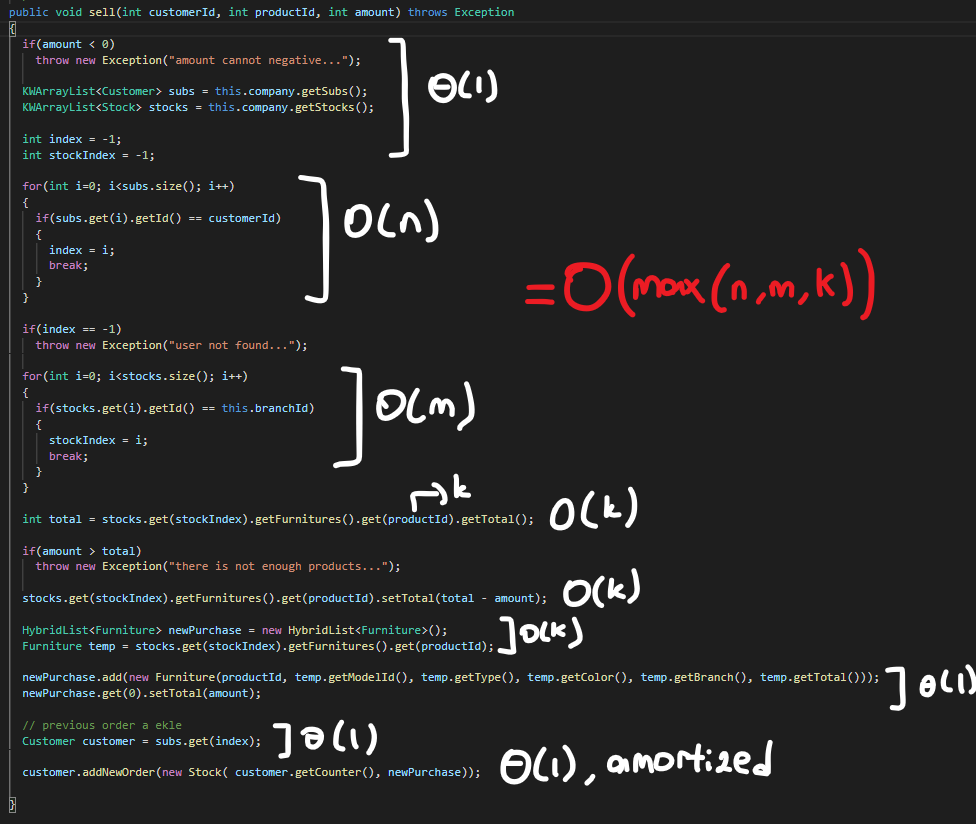
 n is the number of stocks, m is the number of furnitures in a stock

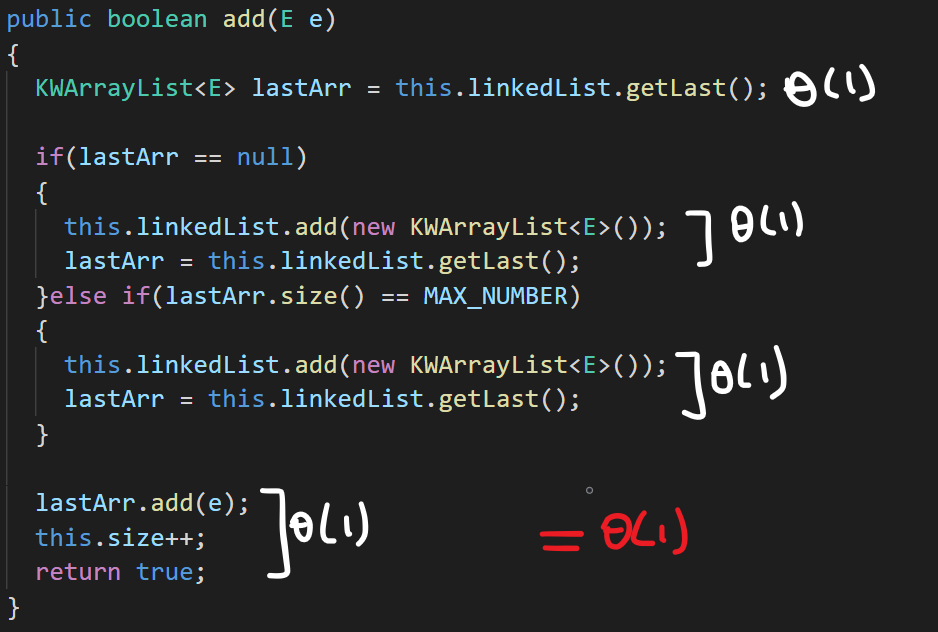
 n is the number of stocks, m is the number of furnitures in a stock

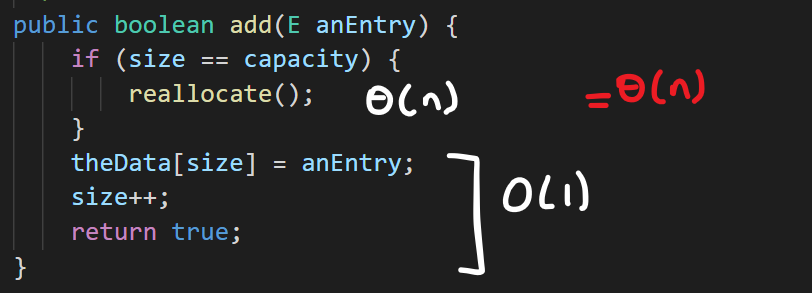


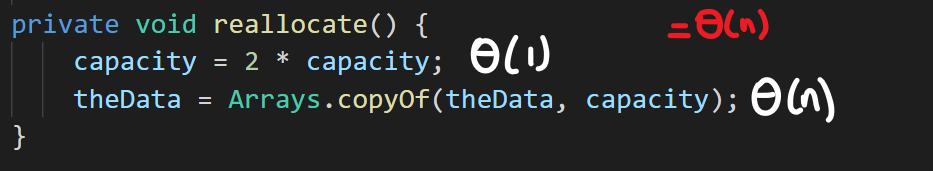
n is the number of subscribers(customers),

m is the size of the stocks

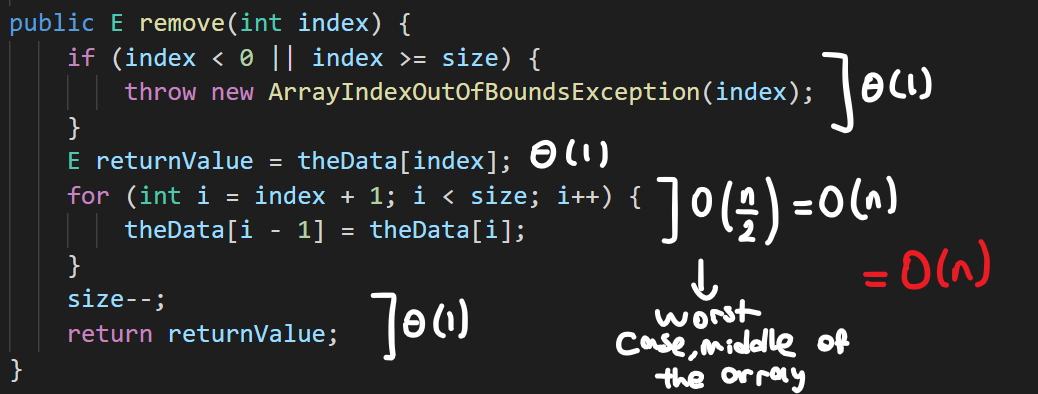
 k is the number of furnitures in a stock

 HybridList add method

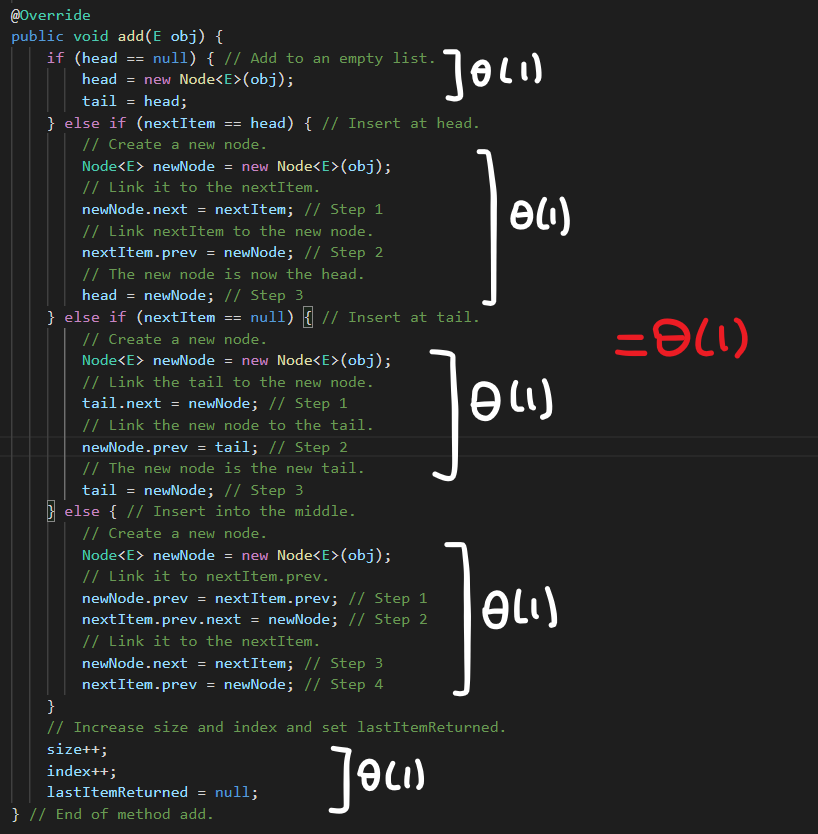
KWArrayList add method

KWArrayList reallocate method

KWArrayList remove method



KWLinkedList add method



KWLinkedList’s iterator’s remove method

