GTU Department of Computer Engineering
CSE 222/505 - Spring 2021
Homework #1 Report

Ömer Faruk Bitikçioğlu
161044010

1. SYSTEM REQUIREMENTS

1- Introduction

1.1- Purpose

The purpose of this document is to give the intended audience a comprehensive explanation of the software. By using this you can easily understand the concepts of the program. You can also see the related diagrams and test results of the program.

1.2- Intended Audience

Managers, developers, testers, stake holders and other people related to this software.

1.3- Intended Use

The purpose of this software is to provide an automation system for an office furniture selling company. It lets the managers manage their work on the system autonomously, the customers to shop online, employees to keep track of their work.

1.4- Scope

This automation system has information about the products, specifications of the products, stock details of the branches, and customers (name, surname, email, order history). Users of this automation system, depending on their role, can easily automate their work. Use the Javadoc for detailed information.

2- Overall Description

2.1- User Needs

The system has 4 different branches initially.

There are three types of users: Admin, Super Admin, Branch Employee, and Customer.

Administrators can add & remove branches and branch employees to the system. They can also query for whether any product is lacking in the branches and needs to be supplied.
Super Admins are superior to the actual admins and can add & remove ordinary admins to/from the system.

Branch employees can inquire about the products in stock and inform the manager about them. They can add & remove products to/from their branch. They can make sales on the system. They can also access previous order information of a customer using the customer number. They add a new order to this section when they make sale.

Customers have a unique number for each of them. They subscribe to the system, and after that the system assigns a special customer number for them, and the system informs the user about what is their customer number. Customers can search for products, see the list of products and their related branch information. They can shop online by entering their address and phone. They can also view their previous orders.

There are different types of furniture like office chairs, office desks, meeting tables, bookcases, and office cabinets on sale. 7 models and 5 colors for each office chair, 5 models and 4 colors for each office desks, 10 models and 4 colors for each meeting tables, 9 models for bookcases, 6 models for office cabinets. Stocks of these furniture can change depending on the sales and supplies.

2.2- Assumptions and Dependencies

The correctness of the information mostly depends on the users of the system. Customers may give false information about themselves. Branch employees can misinform the system. To gain most of the benefits, every user of the system must be accurate in the information they supply to the system. Also, making queries and supply products on time when needed are the responsibilities of the employees. There is no autonomous work for this.

3- System Features and Requirements

3.1- Functional Requirements

The software needs a java compiler to compile the source code and a machine that has Java Virtual Machine to run the compiled code.

For customers to buy furniture in-store or online, they should be subscribed to the system. For online shopping address and phone number should be given correctly.

3.2- External Interface Requirements

To subscribe to the system, users need an email address.

3.3- System Features

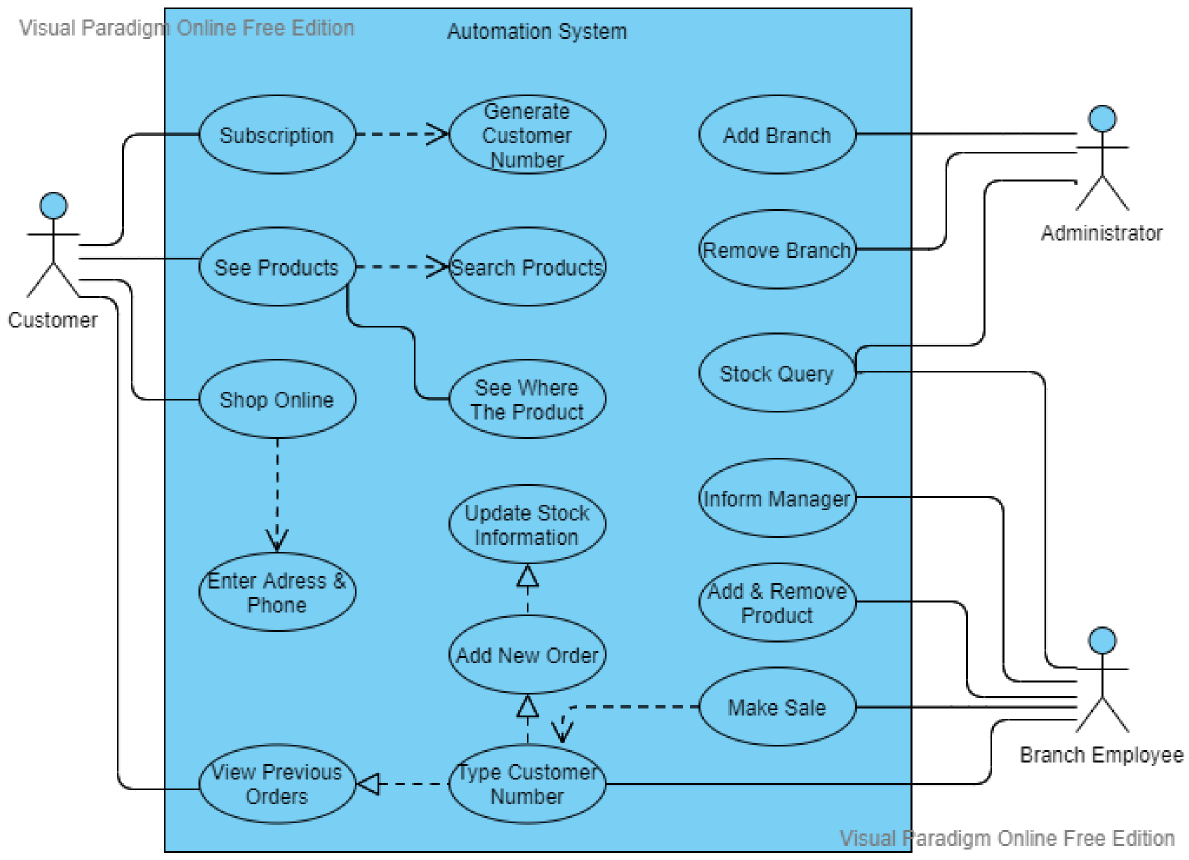
An average computer that has the latest Java version.

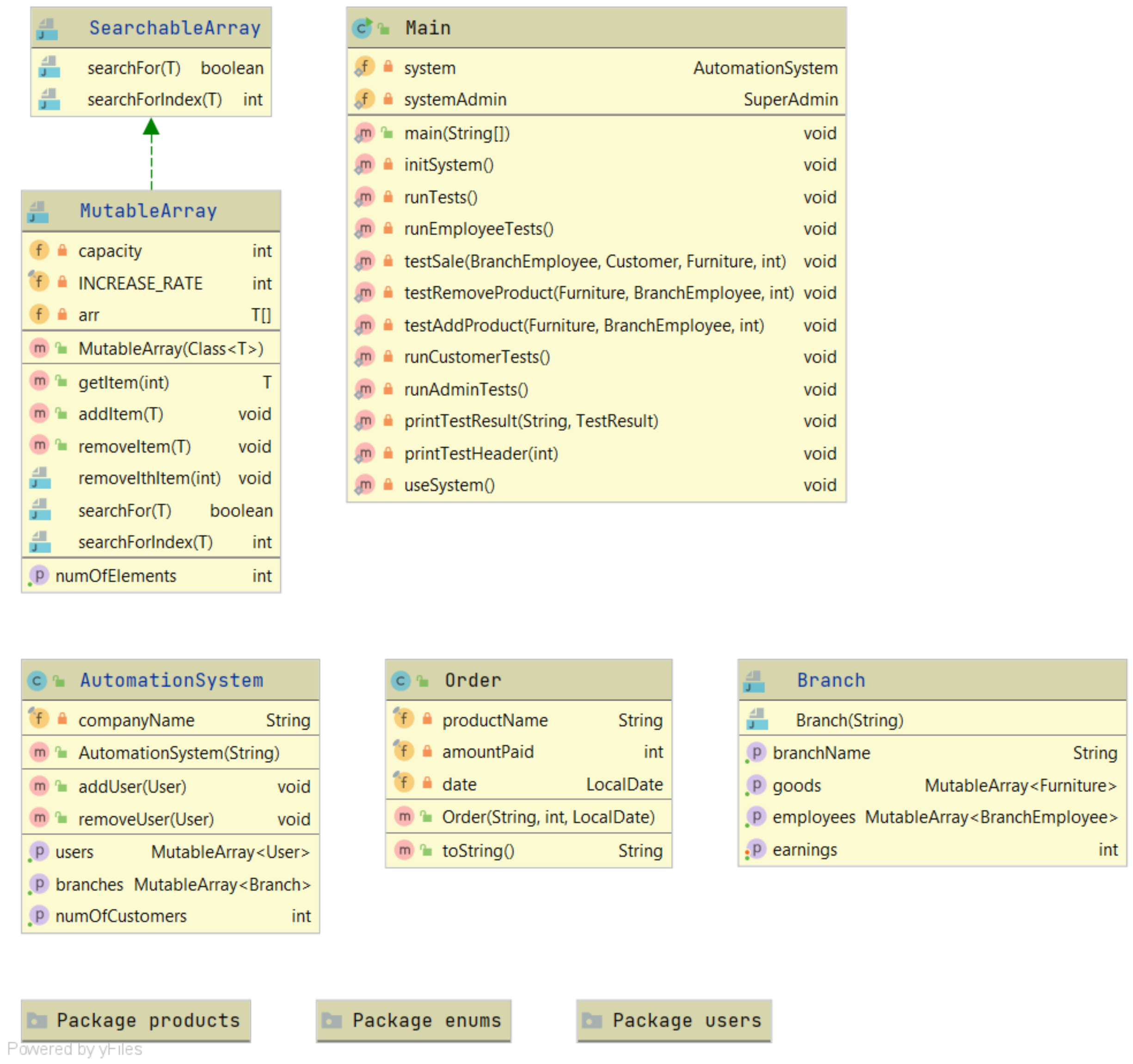
3.4- Nonfunctional Requirements

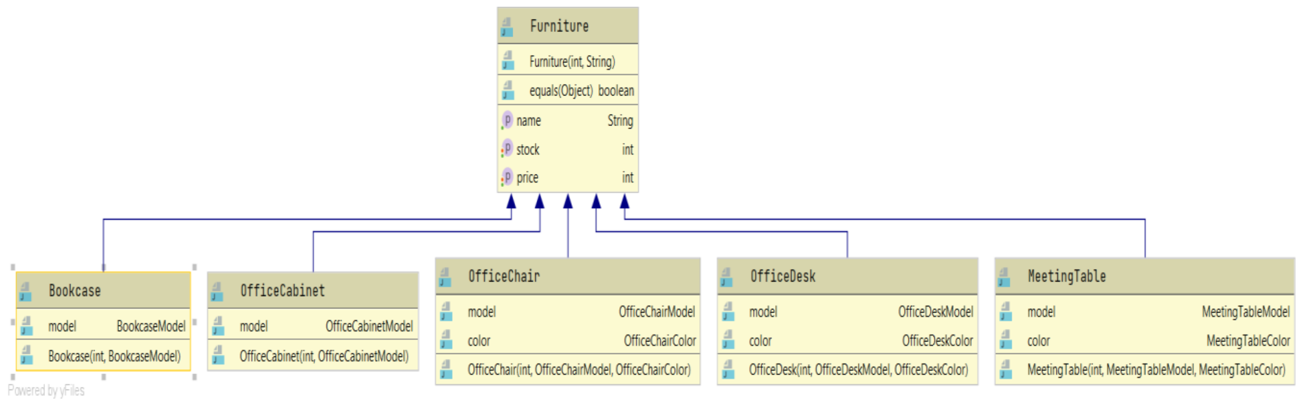
Performance: The system must work efficiently. For instance, making queries should not last forever. It should give results on time.

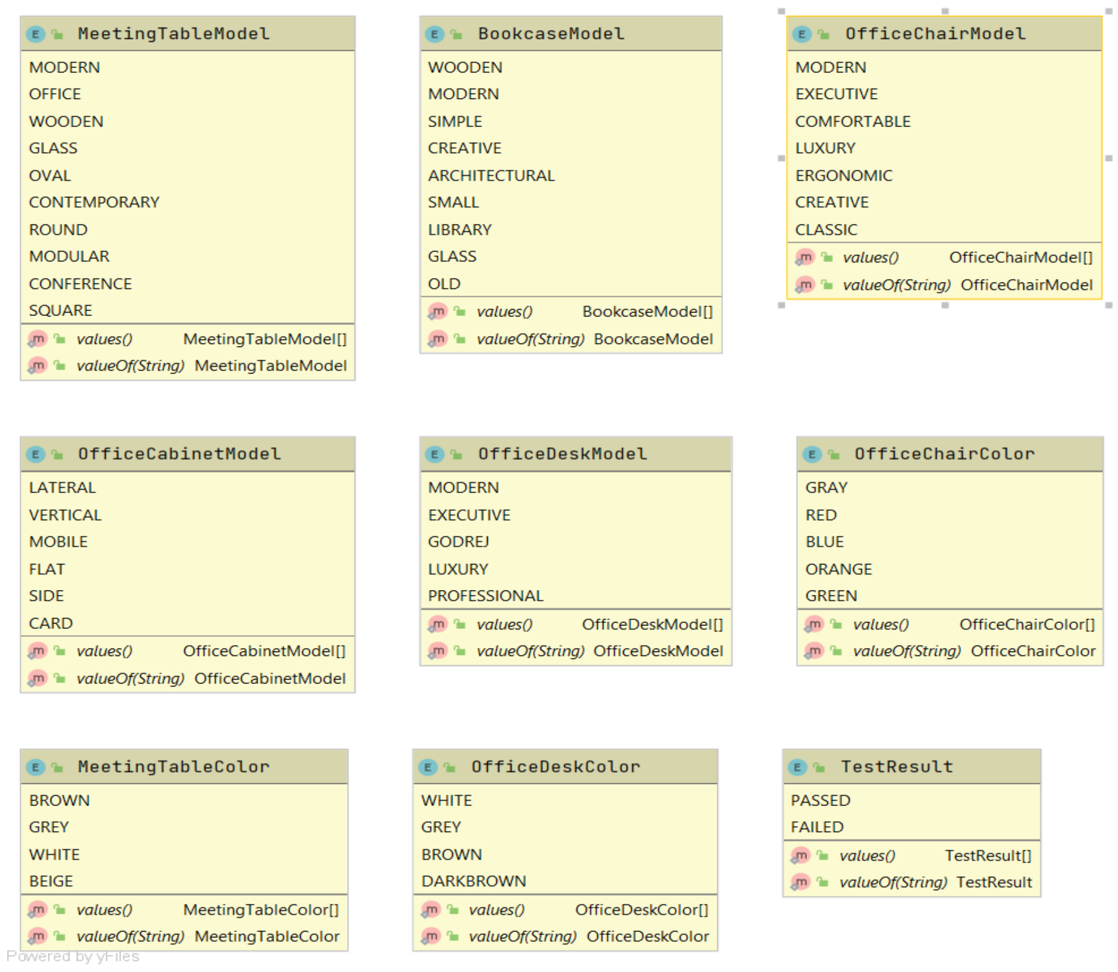
Safety: The information in the system must be safe from external attacks. Also, making online shopping on the system must be safe.

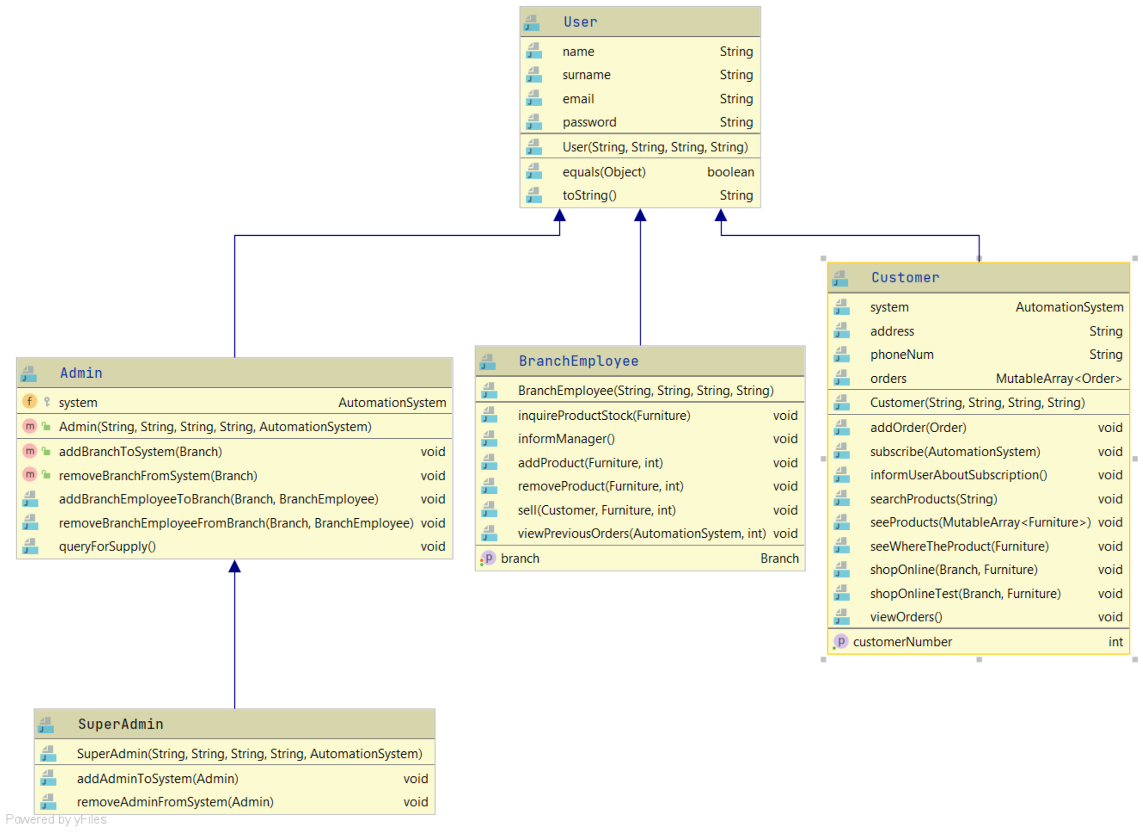
2. USE CASE AND CLASS DIAGRAMS




SearchableArray
searchFor(T) boolean
searchForIndex(T) int
MutableArray
capacity int
INCREASE_RATE int
arr T[]
MutableArray(Class<T>)
getItem(int) T
addItem(T) void
removeItem(T) void
removeIthItem(int) void
searchFor(T) boolean
searchForIndex(T) int
numOfElements int
Main
system AutomationSystem
systemAdmin SuperAdmin
main(String[]) void
initSystem() void
runTests() void
runEmployeeTests() void
testSale(BranchEmployee, Customer, Furniture, int) void
testRemoveProduct(Furniture, BranchEmployee, int) void
testAddProduct(Furniture, BranchEmployee, int) void
runCustomerTests() void
runAdminTests() void
printTestResult(String, TestResult) void
printTestHeader(int) void
useSystem() void
AutomationSystem
companyName String
AutomationSystem(String)
addUser(User) void
removeUser(User) void
users MutableArray<User>
branches MutableArray<Branch>
numOfCustomers int
Order
productName String
amountPaid int
date LocalDate
Order(String, int, LocalDate)
toString() String
Branch
Branch(String)
branchName String
goods MutableArray<Furniture>
employees MutableArray<BranchEmployee>
earnings int
Package products
Package enums
Package users
Powered by yFiles


Furniture
Furniture(int, String)
equals(Object) boolean
name String
stock int
price int
Bookcase
model BookcaseModel
Bookcase(int, BookcaseModel)
OfficeCabinet
model OfficeCabinetModel
OfficeCabinet(int, OfficeCabinetModel)
OfficeChair
model OfficeChairModel
color OfficeChairColor
OfficeChair(int, OfficeChairModel, OfficeChairColor)
OfficeDesk
model OfficeDeskModel
color OfficeDeskColor
OfficeDesk(int, OfficeDeskModel, OfficeDeskColor)
MeetingTable
model MeetingTableModel
color MeetingTableColor
MeetingTable(int, MeetingTableModel, MeetingTableColor)
Powered by yFiles


MeetingTableModel
MODERN
OFFICE
WOODEN
GLASS
OVAL
CONTEMPORARY
ROUND
MODULAR
CONFERENCE
SQUARE
values() MeetingTableModel[]
valueOf(String) MeetingTableModel
BookcaseModel
WOODEN
MODERN
SIMPLE
CREATIVE
ARCHITECTURAL
SMALL
LIBRARY
GLASS
OLD
values() BookcaseModel[]
valueOf(String) BookcaseModel
OfficeChairModel
MODERN
EXECUTIVE
COMFORTABLE
LUXURY
ERGONOMIC
CREATIVE
CLASSIC
values() OfficeChairModel[]
valueOf(String) OfficeChairModel
OfficeCabinetModel
LATERAL
VERTICAL
MOBILE
FLAT
SIDE
CARD
values() OfficeCabinetModel[]
valueOf(String) OfficeCabinetModel
OfficeDeskModel
MODERN
EXECUTIVE
GODREJ
LUXURY
PROFESSIONAL
values() OfficeDeskModel[]
valueOf(String) OfficeDeskModel
OfficeChairColor
GRAY
RED
BLUE
ORANGE
GREEN
values() OfficeChairColor[]
valueOf(String) OfficeChairColor
MeetingTableColor
BROWN
GREY
WHITE
BEIGE
values() MeetingTableColor[]
valueOf(String) MeetingTableColor
OfficeDeskColor
WHITE
GREY
BROWN
DARKBROWN
values() OfficeDeskColor[]
valueOf(String) OfficeDeskColor
TestResult
PASSED
FAILED
values() TestResult[]
valueOf(String) TestResult
Powered by yFiles


User
name String
surname String
email String
password String
User(String, String, String, String)
equals(Object) boolean
toString() String
Customer
system AutomationSystem
address String
phoneNum String
orders MutableArray<Order>
Customer(String, String, String, String)
addOrder(Order) void
subscribe(AutomationSystem) void
informUserAboutSubscription() void
searchProducts(String) void
seeProducts(MutableArray<Furniture>) void
seeWhereTheProduct(Furniture) void
shopOnline(Branch, Furniture) void
shopOnlineTest(Branch, Furniture) void
viewOrders() void
customerNumber int
Admin
system AutomationSystem
Admin(String, String, String, String, AutomationSystem)
addBranchToSystem(Branch) void
removeBranchFromSystem(Branch) void
addBranchEmployeeToBranch(Branch, BranchEmployee) void
removeBranchEmployeeFromBranch(Branch, BranchEmployee) void
queryForSupply() void
BranchEmployee
BranchEmployee(String, String, String, String)
inquireProductStock(Furniture) void
informManager() void
addProduct(Furniture, int) void
removeProduct(Furniture, int) void
sell(Customer, Furniture, int) void
viewPreviousOrders(AutomationSystem, int) void
branch Branch
SuperAdmin
SuperAdmin(String, String, String, String, AutomationSystem)
addAdminToSystem(Admin) void
removeAdminFromSystem(Admin) void
Powered by yFiles

3. PROBLEM SOLUTION APPROACH

I created a class called AutomationSystem to represent the whole system. It holds information of users of the system and branches of the company. To hold the data I used simple arrays but in a special class that I wrote for simplifying common actions on simple arrays.

Implementing such a system with just arrays was a problem of code repetition. Whenever I wanted to add & remove an item to/from an array it was all the same codes repeating again and again. To be able to make operations more easily on a simple array, I created a data structure called MutableArray, similar but a bit simpler of ArrayList. It is a generic structure. You should give the class of the data you want to hold in an array to initialize a MutableArray. After that methods like adding, removing, searching becomes available to you.

MutableArray manages the array inside. It has a capacity property. When we add an item to the array, if there is no capacity enough, it extends its capacity. If there is too much capacity, it reduces its capacity accordingly, to release unnecessary memory. Capacity is initially 5. The increase rate is also 5. Whenever it needs more space, it extends the array with an increased capacity. If the free capacity is equal to the increase rate, it reduces its capacity to the number of elements.

MutableArray is also implementing an interface called SearchableArray. The class that implements SearchableArray must implement methods for searching operations on the array. It gives the ability to find the desired item or check if the inquired item is available in the array.

All different users have some special abilities to perform. I defined some methods for these abilities for each user type in a different class. All these classes extend the abstract class of the user.

For adopting the OOP convention, I created different classes for all different furniture types and all of them extend the abstract class of Furniture. Furniture has model and color properties. Some of them just have model property. These properties are represented by enums.

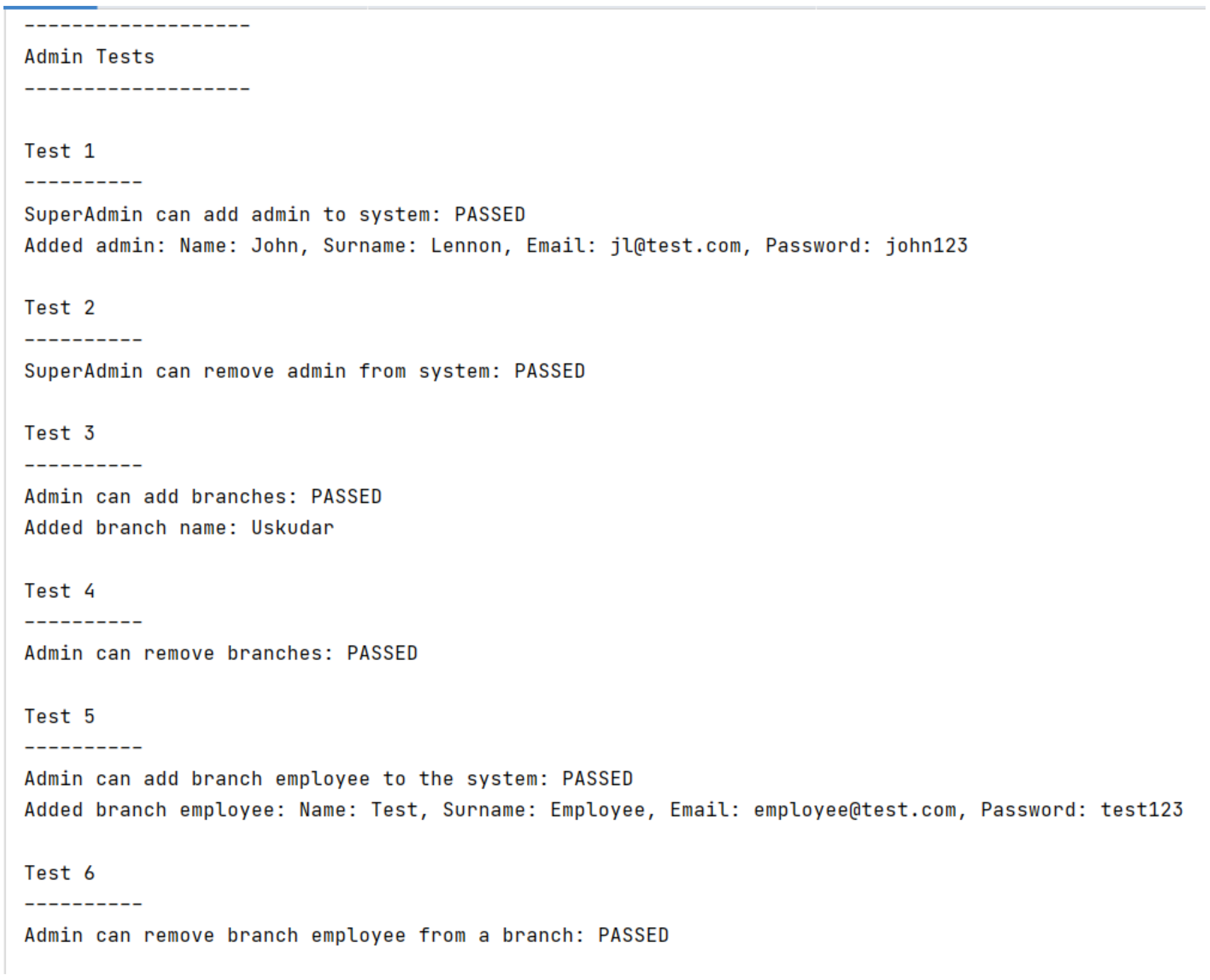
Finally, after implemented the whole system, I wrote a test function to check my test cases and use all the methods I defined earlier in some different conditions.

When the program runs, it asks the user two options. The first option is to see test results, the other one is to use the program on his/ her own. The first option just runs the test function and prints the result on the screen. The second option is an interactive use of the program.

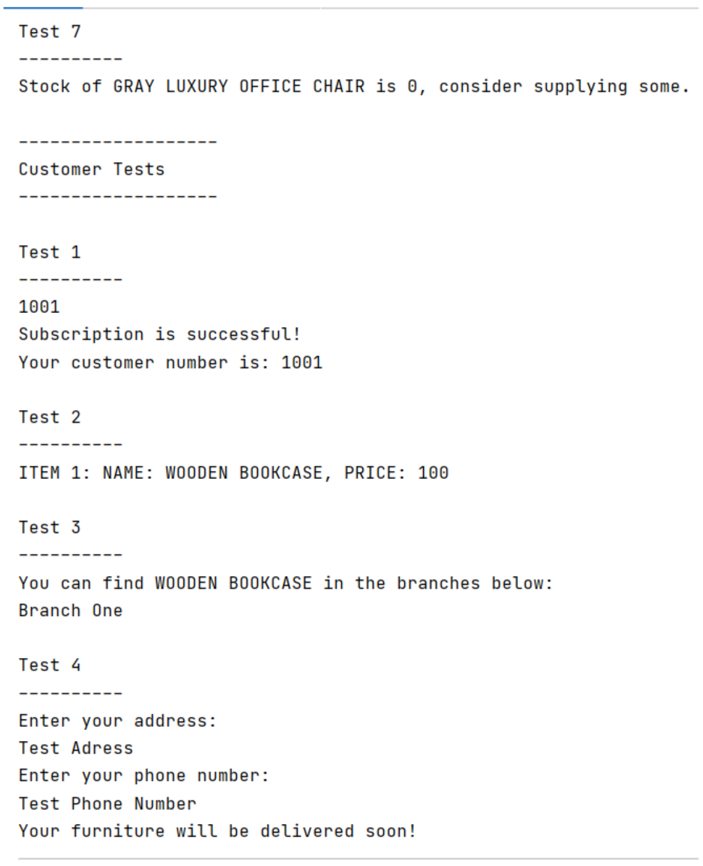
4. TEST CASES

- Super admin can add administrator to the system.
- Super admin can remove administrator from the system.
- Admin can add branch to the system.
- Admin can remove branch from the system.
- Admin can add branch employee to a branch.
- Admin can remove branch employee from a branch.
- Admin can query for products that need to be supplied.
- Customer can subscribe to system and get a customer number.
- Customer can search for a product and see the list of products they searched.
- Customer can see which store has the product they look for.
- Test online shopping with existing product in the test branch.
- Test online shopping with non-existing product in the test branch.
- Customer can view his/her previous orders.
- Branch employee can inquire about stock information of a product.
- Branch employee can inform the managers.
- Branch employee can add an existing product to the branch.
- Branch employee can add a non-existing product to the branch.
- Remove existing number of products.
- Remove all stock of the product.
- Remove non-existing number of products.
- Remove more than existing product.
- Sell existing number of products.
- Sell non-existing number of products.
- Branch employee can view previous orders of customer by using customer number.

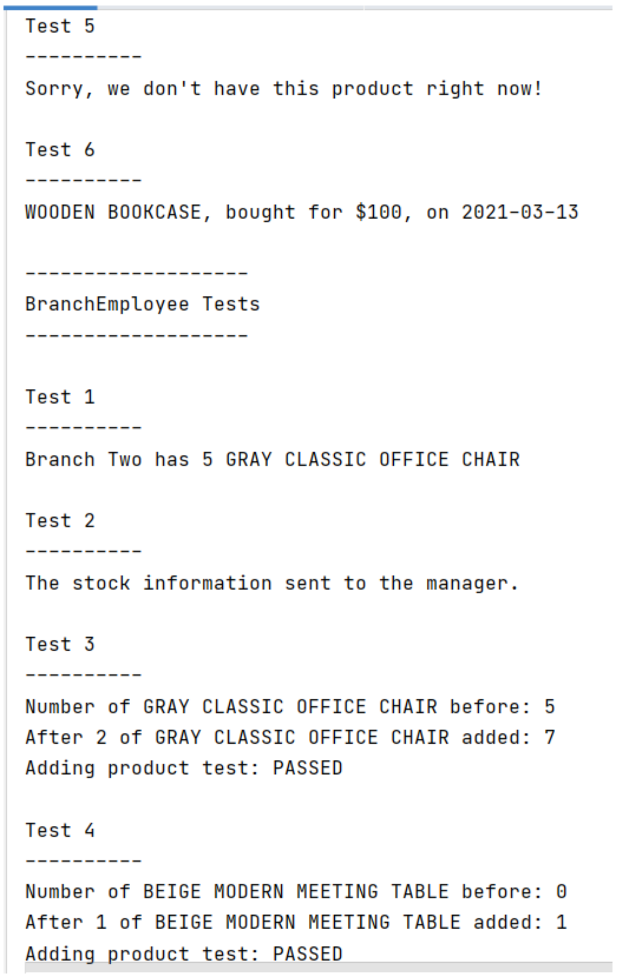
5. RUNNING AND RESULTS



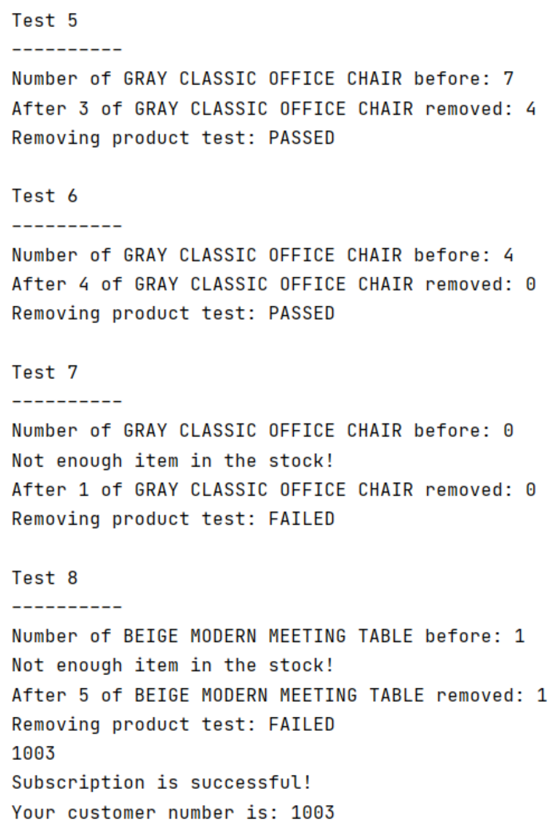
```
--------------------
Admin Tests
--------------------

Test 1
----------
SuperAdmin can add admin to system: PASSED
Added admin: Name: John, Surname: Lennon, Email: jl@test.com, Password: john123

Test 2
----------
SuperAdmin can remove admin from system: PASSED

Test 3
----------
Admin can add branches: PASSED
Added branch name: Uskudar

Test 4
----------
Admin can remove branches: PASSED

Test 5
----------
Admin can add branch employee to the system: PASSED
Added branch employee: Name: Test, Surname: Employee, Email: employee@test.com, Password: test123

Test 6
----------
Admin can remove branch employee from a branch: PASSED
```



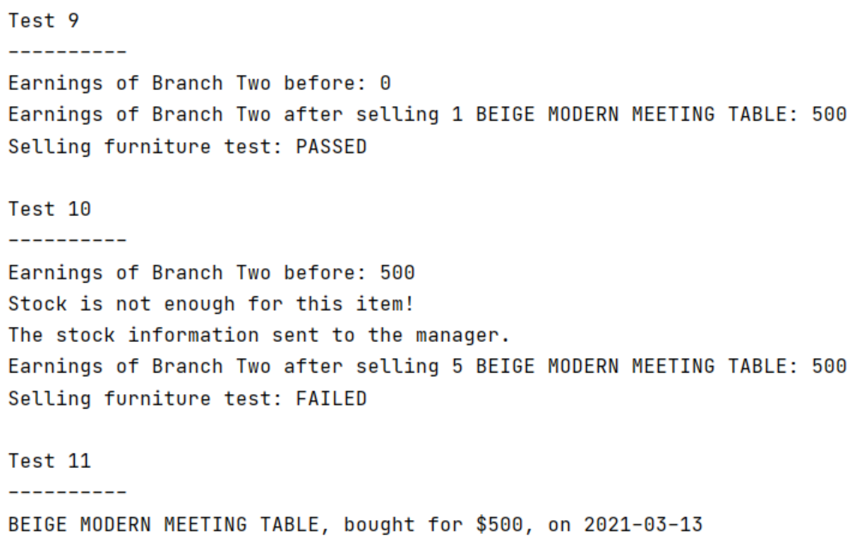
```
Test 7
----------
Stock of GRAY LUXURY OFFICE CHAIR is 0, consider supplying some.

-------------------
Customer Tests
-------------------

Test 1
----------
1001
Subscription is successful!
Your customer number is: 1001

Test 2
----------
ITEM 1: NAME: WOODEN BOOKCASE, PRICE: 100

Test 3
----------
You can find WOODEN BOOKCASE in the branches below:
Branch One

Test 4
----------
Enter your address:
Test Adress
Enter your phone number:
Test Phone Number
Your furniture will be delivered soon!
```



```
Test 5
----------
Sorry, we don't have this product right now!

Test 6
----------
WOODEN BOOKCASE, bought for $100, on 2021-03-13

-------------------
BranchEmployee Tests
-------------------

Test 1
----------
Branch Two has 5 GRAY CLASSIC OFFICE CHAIR

Test 2
----------
The stock information sent to the manager.

Test 3
----------
Number of GRAY CLASSIC OFFICE CHAIR before: 5
After 2 of GRAY CLASSIC OFFICE CHAIR added: 7
Adding product test: PASSED

Test 4
----------
Number of BEIGE MODERN MEETING TABLE before: 0
After 1 of BEIGE MODERN MEETING TABLE added: 1
Adding product test: PASSED
```



```
Test 5
----------
Number of GRAY CLASSIC OFFICE CHAIR before: 7
After 3 of GRAY CLASSIC OFFICE CHAIR removed: 4
Removing product test: PASSED

Test 6
----------
Number of GRAY CLASSIC OFFICE CHAIR before: 4
After 4 of GRAY CLASSIC OFFICE CHAIR removed: 0
Removing product test: PASSED

Test 7
----------
Number of GRAY CLASSIC OFFICE CHAIR before: 0
Not enough item in the stock!
After 1 of GRAY CLASSIC OFFICE CHAIR removed: 0
Removing product test: FAILED

Test 8
----------
Number of BEIGE MODERN MEETING TABLE before: 1
Not enough item in the stock!
After 5 of BEIGE MODERN MEETING TABLE removed: 1
Removing product test: FAILED
1003
Subscription is successful!
Your customer number is: 1003
```



```
Test 9
----------
Earnings of Branch Two before: 0
Earnings of Branch Two after selling 1 BEIGE MODERN MEETING TABLE: 500
Selling furniture test: PASSED

Test 10
----------
Earnings of Branch Two before: 500
Stock is not enough for this item!
The stock information sent to the manager.
Earnings of Branch Two after selling 5 BEIGE MODERN MEETING TABLE: 500
Selling furniture test: FAILED

Test 11
----------
BEIGE MODERN MEETING TABLE, bought for $500, on 2021-03-13
```