CSE222 HW2 REPORT

OĞULCAN KALAFATOĞLU

1801042613

1.System Requirements

From the problem definition we can see that we need some kind of data holder class like database and log-in system

Adminstrator can add branch, remove branch, list branches, add branch employee, remove branch employee and list a branch's employees.

Adminstrators can login using admin id.

Branch Employee can add product, remove product, list products, access customers order list, add sale(offline sale), list previous customers who visited the branch and add stock to a product.

Branch Employees can login using Employee id.

Customer can log-in using mail and password, register, search for products by entering information of product, list all products on all branches, shop online using address and phone number, view their current order.

2.Class Diagram

3. Problem Solutions approach

For automation system, Problem was requiring me to use database-like classes, so I created a class called Database and hold all users and branch informations there.

```
public class Database{
    private KWArrayList<Adminstrator> admins;
    private KWLinkedList<Branch> branch;
    private KWArrayList<BranchEmployee> employee;
    private KWArrayList<Customer> customer;

    public Database(){
        admins = new KWArrayList<Adminstrator>();
        branch = new KWLinkedList<Branch>();
        employee = new KWArrayList<BranchEmployee>();
        customer = new KWArrayList<Customer>();
    }
```

Inside Branch, I hold branchEmployee list, furniture(Products) list and visited customer's information

```
public class Branch {
    private String name;
    private KWArrayList<BranchEmployee> branchEmp;
    private HybridList products;
    private KWArrayList<Customer> visitedCustomer;
```

And I created an static Database object inside Adminstrator class to make my job easier.

```
public class Adminstrator extends Person {
    public static Database db = new Database();

    public Adminstrator(String name, String surname, int id){
        super(name, surname, id);
    }
```

After accessing all data became easier, All I did was writing methods according to the problem.

For hw2 part, after implementing linked list and array list, since their functions were doing same job as my old dynamicArray functions, changing to them was not hard.

For implementation of HybridList

```
public class HybridList {

    KWLinkedList<KWArrayList<Product>> hybridData;
    int size;
    private int currentIndex;
    private static final int MAX_NUMBER = 100;
```

I hold its data using linkedlist and arraylist classes, hold another index for which node to add furnitures.

```
public Product getProduct(int index){
    int nodeIndex = index / MAX_NUMBER;
    int temp = index;
    if(nodeIndex > 0){
        temp = index - MAX_NUMBER * nodeIndex;
    }
    return hybridData.get(0).get(temp);
}
```

Node index calculation is done with index/MAX_NUMBER, and after inside arrayList's size is equal to MAX_NUMBER,

```
public boolean addFurniture(Product item){
    if(hybridData.get(currentIndex).size() == MAX_NUMBER) {
        currentIndex++;
        addNode();
    }
    return hybridData.get(currentIndex).add(item);
}
```

Another node is created for linkedList.

```
public Product removeProduct(int index, int nodeIndex){
    Product temp;
    int tempIndex = index - nodeIndex * MAX_NUMBER;
    temp = hybridData.get(nodeIndex).remove(tempIndex);
    if(hybridData.get(nodeIndex).size() == 0){
        hybridData.remove(nodeIndex);
        size--;
    }
    return temp;
}
```

In the process of removing products, If size of arrayList is 0, that node is removed from Linked list.

4. Running and results

```
1 - Log in as administrator
2 - Log in as branch employee
3 - Log in as customer
4 - Register as customer
5 - Exit

1
```

```
---ADMIN---
1- Add Branch
2- Remove Branch
3- List Branches
4- Add Branch Employee
5- Remove Branch Employee
6- List a Branch's Employees
0- Logout
1
```

```
Enter name of the branch to be created:
Malatya
---ADMIN---
1- Add Branch
2- Remove Branch
3- List Branches
4- Add Branch Employee
5- Remove Branch Employee
6- List a Branch's Employees
0- Logout
3
```

```
---ADMIN---
1- Add Branch
2- Remove Branch
3- List Branches
4- Add Branch Employee
5- Remove Branch Employee
6- List a Branch's Employees
0- Logout
2
```

```
0 - Ankara
1 - Istanbul
2 - Izmir
3 - Antalya
4 - Malatya
Select id of the branch to be removed:
1
```

```
---ADMIN---
1- Add Branch
2- Remove Branch
3- List Branches
4- Add Branch Employee
5- Remove Branch Employee
6- List a Branch's Employees
0- Logout
3
0 - Ankara
1 - Izmir
2 - Antalya
3 - Malatya
```

```
---ADMIN---
1- Add Branch
2- Remove Branch
3- List Branches
4- Add Branch Employee
5- Remove Branch Employee
6- List a Branch's Employees
0- Logout
4
```

```
Enter name of the employee to be added:
Ahmet
Enter surname:
Yılgın
0 - Ankara
1 - Izmir
2 - Antalya
3 - Malatya
Select branch:
2
```

```
Id of the employee is : 4
---ADMIN---
1- Add Branch
2- Remove Branch
3- List Branches
4- Add Branch Employee
5- Remove Branch Employee
6- List a Branch's Employees
0- Logout
```

```
---ADMIN---
1- Add Branch
2- Remove Branch
3- List Branches
4- Add Branch Employee
5- Remove Branch Employee
6- List a Branch's Employees
0- Logout
6
0 - Ankara
1 - Izmir
2 - Antalya
3 - Malatya
Select branch: 2
```

```
Select branch: 2
3 - Mehmet Yılmaz
5 - Ahmet Yılgın
---ADMIN---
1- Add Branch
2- Remove Branch
3- List Branches
4- Add Branch Employee
5- Remove Branch Employee
6- List a Branch's Employees
0- Logout
```

Part2

Database.addProduct

```
public void addProduct(Product.type_t type, Product.color_t color, String model, String branchName, int stockNumber){
    int branchIndex = getIndexOfBranch(branchName);
    int productIndex;
    if(searchProduct(type,color,model,branchName) == -1)
        getBranch().get(branchIndex).getProducts().addFurniture(new Product(type, color, model));
    productIndex = searchProduct(type,color,model,branchName);
    getBranch().get(branchIndex).getProductByIndex(productIndex).increaseStock(stockNumber);

}
```

O(n^2)

```
public boolean addFurniture(Product item){
    if(hybridData.get(currentIndex).size() == MAX_NUMBER) {
        currentIndex++;
        addNode();
    }
    return hybridData.get(currentIndex).add(item);
}
```

O(n^2)

Branch.getProductByIndex

```
public Product getProductByIndex(int index){
    int nodeIndex = index / 100;
    if(index > products.size()) return null;
    return products.getProduct(index, nodeIndex);
}
```

O(n)

```
public Product getProduct(int index, int nodeIndex){
    return hybridData.get(nodeIndex).get(index);
}
```

O(n)

```java
public void createAdmin(String name, String surname, int id){
    admins.add(new Adminstrator(name, surname, id));
}
```

O(n)

```java
public void createBranch(String name){
    getBranch().add(new Branch(name));
}

public void deleteBranch(String name){
    int branchIndex = getIndexOfBranch(name);
    getBranch().remove(branchIndex);
}
```

theta(n)

```java
public void addBranchEmployee(String name, String surname, int id, String branchName){
    int branchIndex = getIndexOfBranch(branchName);
    getBranch().get(branchIndex).getBranchEmployee().add(new BranchEmployee(name, surname, id, getBranch().get(branchIndex)));
}
```

theta(n^2)

```java
public void eraseProduct(Product.type_t type, Product.color_t color, String model, String branchName){
    int branchIndex = getIndexOfBranch(branchName);
    int productIndex = searchProduct(type, color, model, branchName);
    if(productIndex != -1)
        getBranch().get(branchIndex).getProducts().remove(productIndex);
}
```

O(n)