Time Complexity Analysis of the Furniture Company Classes

Note : I didn't analyze regular methods such as setters, getters, most of constructors etc.

1) Administrator

```java
/**
 * Adds a new branch to company.
 * @param branchID ID of the Branch.
 * @throws IllegalStateException When the given ID is another branch's ID
 * */
public static void addBranch(int branchID) throws IllegalStateException {
    for (Branch branch : branches) {
        if (branch.getBranchID() == branchID)
            throw new IllegalStateException("This branch ID is taken by another branch.");
    }
    branches.add(new Branch(branchID));
}
```

n = Amount of branches

O(n) → $T_W(n) = O(n)$, $T_B(n) = O(1)$

if → O(1)

throw → O(1)

branches.add → O(1)

new Branch(branchID) → O(1)

$T_W(n) = O(n)$ (Addition is successful)

$T_B(n) = O(1)$ (Error at first element)

$T(n) = O(n)$

```java
/**
 * Getter for Branch.
 * @param branchID ID of the brand which will be returned
 * @return Specified branch.
 */
public static Branch getBranch(int branchID){
    for(Branch i : branches){
        if(i.getBranchID() == branchID)
            return i;
    }
    return null;
}
```

n = Amount of branches

O(n) → for(Branch i : branches)

if(i.getBranchID() == branchID) → O(1)

return i; → O(1)

return null; → O(1)

$T(n) = O(n)$

$T_W(n) = O(n)$ (branch doesn't exist)

$T_B(n) = O(1)$ (branch is the first entry)

```java
/**
 * Removes the specified branch.
 * @param branchID ID of the Branch.
 * @throws NoSuchElementException When the given branch doesn't exist.
 */
public static void removeBranch(int branchID)throws NoSuchElementException {
    boolean removeFlag=false;
    ListIterator<Branch> iter = branches.listIterator();
    while (iter.hasNext()){
        if(iter.next().getBranchID() == branchID) {
            iter.remove();
            removeFlag=true;
            break;
        }
    }
    if(!removeFlag)
        throw(new NoSuchElementException("This branch doesn't exist."));
}
```

n = Amount of branches

$T_W(n) = O(n)$ (branch doesn't exist)

$T_B(n) = O(1)$ (branch is the first entry)

$T(n) = O(n)$

```java
/**
 * Adds a new Employee to specified branch.
 * @param branchID ID of the Branch.
 * @param emp Employee which will be added to specified Branch.
 * @throws IllegalArgumentException When the given employee is already working in given branch.
 * @throws NoSuchElementException When the given branch doesn't exist.
 */
public static void addEmployee(int branchID,BranchEmployee emp)throws IllegalArgumentException,NoSuchElementException {
    boolean branchFlag = false;
    for (Branch i : branches) {
        if (i.getBranchID() == branchID) {
            branchFlag = true;
            try {
                i.addBranchEmployee(emp);
                break;
            } catch (IllegalArgumentException e) {
                System.out.println("This employee already works in this branch!");
            }
        }
    }
    if(!branchFlag)
        throw(new NoSuchElementException("This branch doesn't exist."));
}
```

n = Amount of branches

m = Amount of employees

$T_W(n,m) = O(m) \cdot O(n) = O(n \cdot m)$ (Branch doesn't exist.)

$T_B(n,m) = O(m) \cdot O(1) = O(m)$ (Branch is the first entry)

$T(n,m) = O(nm)$

```java
/**
 * Removes the specified Employee from specified Branch.
 * @param branchID ID of the Branch.
 * @param id ID of the Employee which will be removed
 * @throws NoSuchElementException When the given employee doesn't exist.
 * @throws NoSuchElementException When the given branch doesn't exist.
 */
public static void removeEmployee(int branchID,int id) throws NoSuchElementException {
    boolean branchFlag = false;
    for (Branch branch : branches) {
        if (branch.getBranchID() == branchID) {
            branchFlag = true;
            try {
                branch.getBranchEmployees().remove(branch.getBranchEmployee(id));
                break;
            }
            catch(NoSuchElementException ex){
                throw new NoSuchElementException("This employee doesn't exist.");
            }
        }
    }
    if(!branchFlag)
        throw(new NoSuchElementException("This branch doesn't exist."));
}
```

n = Amount of branches

m = Amount of employees

$T_W(n,m) = \Theta(n) \cdot \Theta(m) = \Theta(nm)$

$T_B(n,m) = \Theta(1) \cdot \Theta(m) = \Theta(m)$

$T(n,m) = O(nm)$

```java
/**
 * Queries a branch for needs.
 * @param b Branch which will be queried.
 * @return Needed (low on amount) products. null if there is no need to any Product in the specified branch.
 */
public static HybridList<Product> query(Branch b) {
    return b.getNeededProducts();
}
```

$T() = \Theta(1)$

From Previous H.W:

Amount of branches = n

Amount of products at a branch = m

```
public static void createProduct(int branchID,int productId,String name,String color)
                                                    throws IllegalArgumentException,NoSuchElementException{
    boolean branchFlag=false;

    int i;
    for(i=0;i<branches.size();i++){
        if(branchID == branches.at(i).getBranchID()){

            branchFlag=true;
            for(int j=0;j<branches.at(i).getProducts().size();j++){
                if(branches.at(i).getProducts().at(j).getId()==productId)
                    throw new IllegalArgumentException("This product already exists.");
            }
            branches.at(i).getProducts().insert(new Product(name,productId,color));
        }
    }
    if(!branchFlag)
        throw new NoSuchElementException("There is no branch with this ID.");
}
```

$T_W(n,m) = \Theta(n)$, $T_B(n,m) = \Theta(1)$

If a branch with this ID exists:

$T_W(n,m) = O(nm^2)$

If it doesn't:

$T_B(n,m) = \Theta(n)$

$T(n,m) = O(n \cdot m^2) \wedge \Omega(n)$

```
public static void addProduct(int branchID,int productId,int amount) throws NoSuchElementException,IllegalArgumentException{
    if(amount<=0)
        throw new IllegalArgumentException("You can't add zero or less products!");
    else if(getBranch(branchID) == null || getBranch(branchID).getProduct(productId)==null)
        throw new NoSuchElementException("Product or branch doesn't exist!");
    else
        getBranch(branchID).getProduct(productId).addProduct(amount);
}
```

$O(n)$ $O(m)$ $O(1)$

amount $\leq 0 \Rightarrow T_B(n,m) = \Theta(1)$

else $\Rightarrow T_W(n,m) = O(n \cdot m)$

$T(n,m) = O(n \cdot m)$

1) Branch Employee

n = Product Amount

```
public boolean inquireProduct(int productID) throws NoSuchElementException{
    if (employeesBranch.getProduct(productID) == null)
            throw new NoSuchElementException("Product doesn't exist.");

    return employeesBranch.getProduct(productID).getAmount() < 10; //true if new products needed
}
```

$T(n) = O(n)$

```
/**
 * Inform the Admin if a product is needed. Usually used with inquireProduct method.
 * @param productID ID of the product which will bi inquired
 */
@Override
public void informAdmin(int productID) { employeesBranch.addNeededProduct(productID); }
```

$T() = O(1)$

```
/**
 * Make a sale to a customer face-to-face (in store.)
 * @param productID ID of the product which will be selled.
 * @param customerNum ID of the customer.
 * @param amount Amount of the product.
 * @throws NoSuchElementException when there is no customer with this ID.
 * @throws IllegalArgumentException you can't sell zero or less products.
 * @throws IllegalStateException when there is not enough product  to sell
 */
@Override
public void makeSale(int productID, int customerNum,int amount)
        throws NoSuchElementException, IllegalArgumentException,IllegalStateException {
    if(employeesBranch.getCustomer(customerNum) == null)
        throw new NoSuchElementException("There is no account for this customer. Create on
    else if(employeesBranch.getProduct(productID) == null)
        throw new NoSuchElementException("There is no product with this ID");
    else if(amount<=0)
        throw new IllegalArgumentException("You can't sell zero or less product!");
    else if(amount <= employeesBranch.getProduct(productID).getAmount()) {
        employeesBranch.getProduct(productID).decreaseAmount(amount);
        employeesBranch.getCustomer(customerNum).addToPreviousOrders(
                new Order(employeesBranch.getProduct(productID),amount));
    }
    else throw new IllegalStateException("There is not enough products.");
}
```

n = Product Amount

m = Customer Amount

$T_W(m,n) = O(m+n)$

$T_B(m,n) = O(1)$

$T(m,n) = O(m+n)$

```java
/**
 * Create subscription for a new customer. Only used for face-to-face sells. Online sells handles this automatically.
 * @param newCus New Customer.
 * @throws IllegalArgumentException If the customer already has a subscription.
 */
public void createSubscription(BrandCustomer newCus) throws IllegalArgumentException{
    if(employeesBranch.getCustomers().contains(newCus))
        throw new IllegalArgumentException("This customer already has a subscription.");
    employeesBranch.addNewCustomer(newCus);
}
```

n = customer Amount

→ O(n)

→ O(1)

→ O(n)

T(n) = O(n)

```java
/**
 * Getter for previous orders of a customer.
 * @param customerNum ID of the customer
 * @return Previous orders of the customer. Null if there isn't any.
 * @throws NoSuchElementException if the given customer doesn't exist."
 */
@Override
public KWArrayList<Order> getPreviousOrders(int customerNum) throws NoSuchElementException{
    if(employeesBranch.getCustomer(customerNum)==null)
        throw new NoSuchElementException("This customer doesn't exist.");
    return employeesBranch.getCustomer(customerNum).getPreviousOrders();
}
```

n = customer Amount

→ O(n)

→ O(1)

O(n)

T(n) = O(n)

3) Brand Customer

```java
/**
 * Constructor for Brand Customer
 * @param name Name of the Customer.
 * @param surname Surname of the Customer.
 * @param phoneNumber Phone number of the Customer.
 * @param eMail E-mail of the Customer.
 * @param password Password of the Customer.
 */
public BrandCustomer(String name, String surname,String phoneNumber,String eMail,String password) {
    super(name, surname, generateID(eMail));
    this.phoneNumber = phoneNumber;
    mails.add(eMail);
    this.eMail = eMail;
    this.password = password;
    previousOrders= new KWArrayList<>();


    loggedIn=false;
}
```

n = Number of Users

→ O(n) (because of ID check)

→ O(1)

→ O(n) (because of mail check)

→ O(1)

→ O(1)

→ O(1) = 0

T(n) = O(n) + O(n) = O(n)

```java
private static int generateID(String eMail){
    while(User.getIds().contains(customerNum)){      // O(n)
        customerNum++;                                 // → Θ(1)
    }
    if(mails.contains(eMail))                          // → O(n)
        throw new IllegalArgumentException("This mail is already signed up!");
    System.out.println("----------------Your auto-generated ID: "   // ↳ Θ(1)
            +customerNum + "----------------");        // → Θ(1)
    return customerNum;                                // → Θ(1)
}
```

n = Amount of users
= " " mails
= " " ID's

$T(n) = O(n) + O(n) = O(n)$

```java
/**
 * Login to the system.
 * @param mail E-mail of the customer.
 * @param pw Password of the customer.
 */
public void login(String mail,String pw){
    if (!loggedIn && mail.equals(eMail) && pw.equals(password)) {   // → Θ(1)
        loggedIn = true;                                            // → Θ(1)
        System.out.println("You have successfully logged in.");     // → Θ(1)
    }
    else if(loggedIn)                                               // → Θ(1)
        System.out.println("You are already logged in.");           // → Θ(1)
    else
        System.out.println("Wrong info!");                          // → Θ(1)
}
```

$T() = \Theta(1)$

```java
/**
 * List all the products in specified branch.
 * @param branchID ID of the branch. This branches products will be listed.
 * @throws IllegalStateException if the customer is not logged in.
 * @throws NoSuchElementException if the branch doesn't exist.
 */
public void listProducts(int branchID) throws  NoSuchElementException,IllegalStateException{
    if(loggedIn) {                                                   // → Θ(1)
        boolean branchFlag=false;                                    // → Θ(1)
        for(Branch branch : Administrator.branches){                 // → Θ(n)      Θ(n²)
            if(branch.getBranchID() == branchID){                    // → Θ(1)
                branchFlag = true;                                   // → Θ(1)
                ListIterator<Product> e = branch.getProducts().hybridIterator();   // Θ(1) Θ(1) → Θ(1)
                while(e.hasNext()){                                  // → Θ(1)     Θ(m)
                    System.out.println(e.next() + "\n");             // → Θ(1)
                }
            }
        }
        if(!branchFlag)                                              // → Θ(1)
            throw new NoSuchElementException("This branch doesn't exist");   // → Θ(1)
    }
    else throw new IllegalStateException("This user is not logged in!");    // → Θ(1)
```

n = Amount of branches.
m = Amount of products
(varies from branch to branch)

$T(n) = \Theta(n).\Theta(n) = \Theta(n^2)$

```java
/**
 * Search a product in all branches.
 * @param productName Product name to search.
 * @throws IllegalStateException when the user is not logged in.
 */
public void search(String productName) throws IllegalStateException{
    boolean productFlag = false;
    if(loggedIn) {
        for(Branch branch : Administrator.branches){
            ListIterator<Product> e = branch.getProducts().hybridIterator();
            while(e.hasNext()){
                Product product = e.next();
                if(product.getName().equals(productName)){
                    productFlag = true;
                    System.out.println("Branch code " + branch.getBranchID() +
                            " has " + product.getAmount() + " of this product.");
                }
            }
        }
        if(!productFlag)
            System.out.println("This product doesn't exist right now.");
    }
    else throw new IllegalStateException("This user is not logged in!");
}
```

$T(n) = O(n) \cdot O(n) = O(n^2)$

```java
public void buyOnline(int branchID,int productID,int amount)
        throws NoSuchElementException,IllegalStateException,IllegalArgumentException{

    if(loggedIn) {

        if(Administrator.getBranch(branchID) == null)
            throw new NoSuchElementException("This branch doesn't exist!");

        else if(Administrator.getBranch(branchID).getProducts() == null)
            throw new NoSuchElementException("This product doesn't exist!");

        else if(amount<=0)
            throw new IllegalArgumentException("You can't buy zero or less products!");

        try {
            if (Administrator.getBranch(branchID).getProduct(productID).getAmount() >= amount) {

                Administrator.getBranch(branchID).getProduct(productID).decreaseAmount(amount);

                System.out.println("Products will be shipped to you. You can pay only at door for now.\n");

                if (getPreviousOrders().size() == 0 && !Administrator.getBranch(branchID).getCustomers().contains(this))
                    Administrator.getBranch(branchID).addNewCustomer(this);

                addToPreviousOrders(new Order(Administrator.getBranch(branchID).getProduct(productID), amount));

            } else
                System.out.println("There is not enough Product in this branch. Try again later.\n");

        }catch(NullPointerException ex){
            throw new NoSuchElementException("This product doesn't exist!");
        }
    }
    else throw new IllegalStateException("This user is not logged in!");
}
```

m = Amount of customers
n = Amount of branches
t = Amount of products

getProduct = O(t)
getBranch = O(n)
getCustomers = O(m)

$T(m,n,t) = O(m+n+t)$