***Göktuğ Canikli Zeynel Eren Kınalı***

***220313014 220313025***

***Report on the Implementation and Functionality of the Order Management System***

*This is an end-term project report for the Data Structures and Algorithms course, focused on analyzing the algorithms implemented in the project.***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

***1. Node.java***

**The Node class is a basic unit of tree structure and has the following properties:**

**1.1 Variables:**

* ***parent\_node, child\_node, sibling\_node:*** *References that hold the parent, child, and sibling of the node.*
* ***data:*** *A String value representing the data stored in the node.*
* ***amount:*** *Holds the frequency or number of occurrences of the node.*
* ***index:*** *Indicates order level at the orders array(where node placed) of the node.*

**1.2 Methods:**

**• Getter Methods: *Provides access to variables in the node:***

* ***get\_parent\_node():*** *Returns the parent reference of the node.*
* ***get\_child\_node():*** *Returns the child reference of the node.*
* ***get\_sibling\_node():*** *Returns the sibling reference of the node.*
* ***get\_data():*** *Returns the data stored in the node.*
* ***get\_quantity():*** *Returns the number of occurrences of the node.*
* ***get\_index():*** *Returns the order level at the orders array(where node placed) of the node.*

**• *Setter Methods:* Modifies variables in the node:**

* ***set\_parent\_node(Node parent\_node):*** *Sets the parent reference of the node.*
* ***set\_child\_node(Node child\_node):*** *Sets the child reference of the node.*
* ***set\_sibling\_node(Node sibling\_node):*** *Sets the sibling reference of the node.*
* ***set\_data(String data):*** *Sets the data stored in the node.*
* ***set\_quantity(int quantity):*** *Sets the number of node occurrences.*
* ***set\_index(int ​​index):*** *Sets the order level at the orders array(where node placed) of the node.*

**• Other Methods:**

* ***increase\_quaintity():*** *Increases the quantity of the node by one.*
* ***decrease\_quaintity():*** *Decreases the node quantity by one (does not decrease in place of 0).*

**1.3 Purpose of Use:**

*The Node class provides a basic data structure for representing hierarchical relationships in a tree structure. Parent-child and sibling relationships between nodes in the tree can be easily managed.*

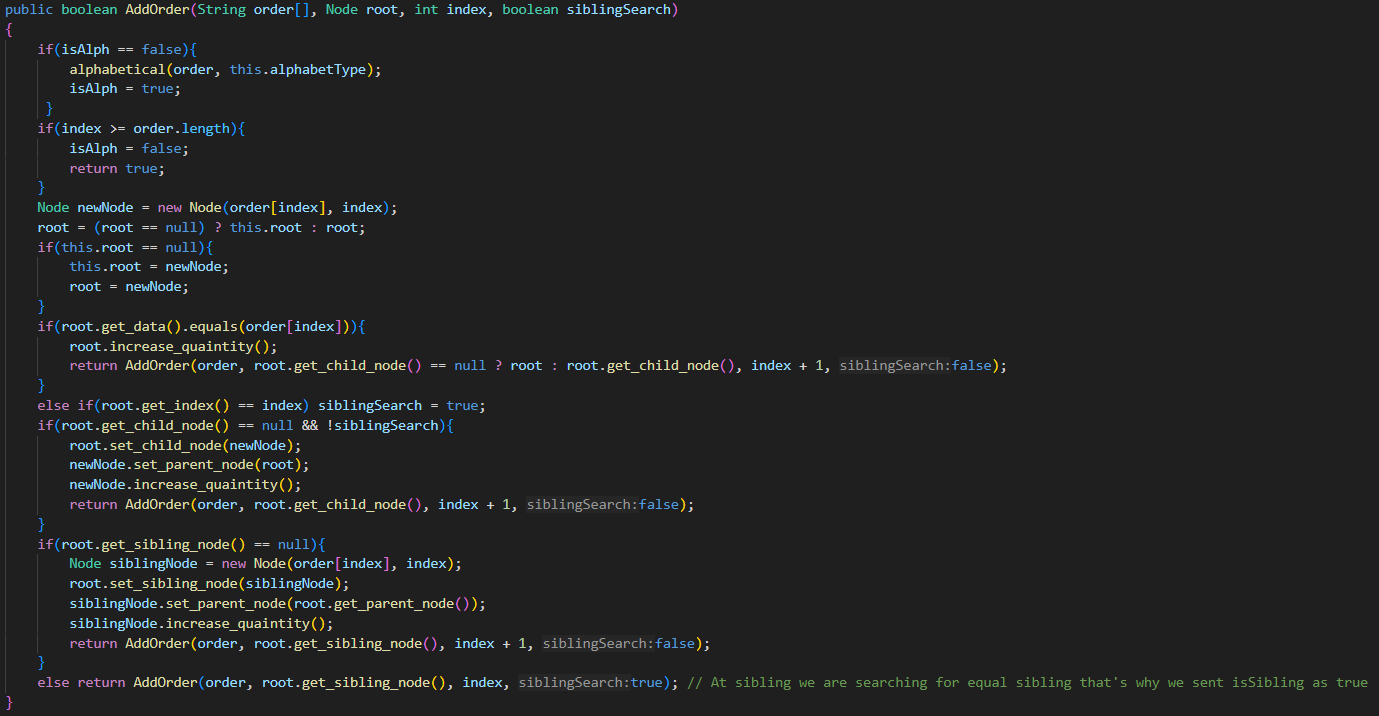
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

***2. OrderSystem.java***

**The OrderSystem class is used to organize and process orders over a tree structure.**

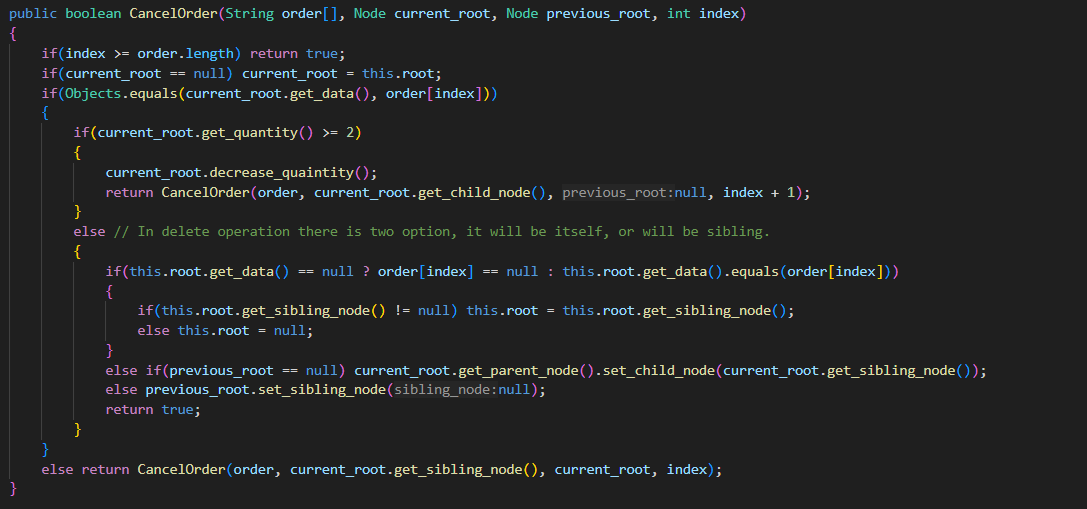
**2.1 Algorithms:**

**Add Order:**

****

* **Purpose:** *to select the specified order (in series) from the tree structure.*
* **Processing:**
  + ***Alphabetical Sorting:*** *The ordered array is sorted according to the alphabet type ("tr" or "en") defined by the system.*
  + **Root Node Check:** *If there is no root node in the tree yet, the first-order element is connected as the root.*
  + ***Matching with Existing Node:***
    - *If the node already exists, its quantity is increased by one.*
    - *Otherwise, a new node connection and parent-child connection are established.*
  + ***Sibling Node Search:*** *If there are other nodes in the same way, the new node is connected to their sibling.*
  + ***Continue Recursive Addition:*** *The process continues for the next element in the order sequence.*

**Cancel Order:**

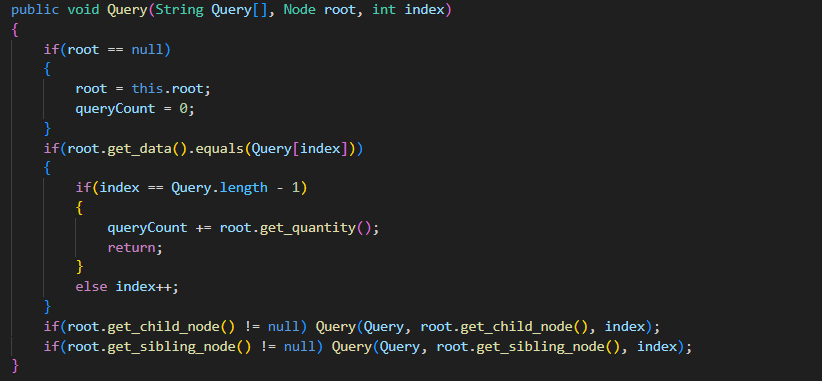
****

**• Purpose:** *Add the specified order (in alphabetical series) to the tree structure.*

**• How it Works:**

* **Finding Node:** *The relevant node is found by following the order sequence.*
* **Quantity Check:**
  + *If the quantity of the node is larger than one, the quantity is reduced".*
  + *Otherwise, the node is completely deleted.*
* **Reference Update:**
  + *If the deleted node is connected to a parent or sibling, the references are reviewed again.*
  + *The parent's child connection or sibling connections are checked.*
* **Child Nodes:** *Since the deletion process will add child nodes, the necessary arrangements are made.*

**Query:**

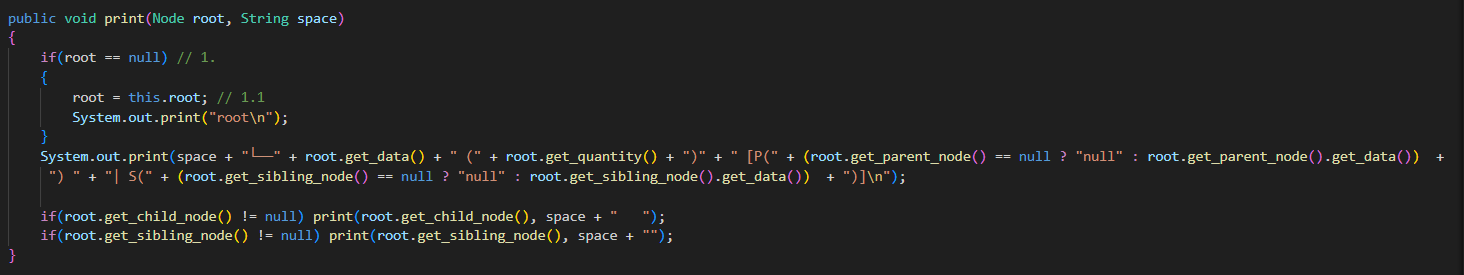
****

**• Purpose:** *to find the quantity of a specified query node.*

**• How it Works:**

* ***Following the Query Path:*** *By following the query sequence, the tree's structure is progressed for each element.*
* ***Node Validation:*** *The relevant data is checked at each node.*
* ***Last Element Check:*** *When the last element of the query series is reached, the node quantity is added to the total query quantity.*

**Print:**

****

**• Purpose:** *To print the current properties of the tree structure.*

**• How it Works:**

* **Starting from Root Node:** *The coloring process starts from the root node.*
* **Node Formatting:** *Each node is printed with its quantity information and parent-sibling relationships.*
* ***Recursive Printing:*** *Recursive printing is applied to child nodes and siblings.*

**2.2 Purpose of Use:**

*Provides an effective way to organize, query, and process orders. The hierarchical structure also shows how to manage the children of orders.*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**3. Main.Java**

**This file controls the order management system by calling the methods of the OrderSystem class after the user enters it.**

**3.1 Algorithms:**

* **Purpose:** *To ensure that the process is carried out according to the user's choices.*
* **How it Works:**
  + ***Language and Alphabet Selection:*** *The application asks the user to choose the "tr" or "en" alphabet. The selected alphabet is used to sort the orders.*
  + ***Menu Display:*** *Five main operations are presented to the user:*
  + ***Add Order:*** *Receives an order from the user and calls the AddOrder method.*
  + ***Cancel Order:*** *Calls the CancelOrder method to cancel one of the existing orders.*
  + ***Querying the Product Set:*** *Receives the query series from the user and uses the query method using the relevant data in the tree.*
  + ***Print:*** *Calls the print method to print the tree.*
  + ***Exit:*** *Exits the application.*

**3.2 Purpose of Use:** *Application management that has main menu via using main menu methods.*

**3.3 Error Conditions:**

*When an invalid entry or menu option is entered, the user:*

*given error message, the situation that needs to be done, and the menu reload is done*