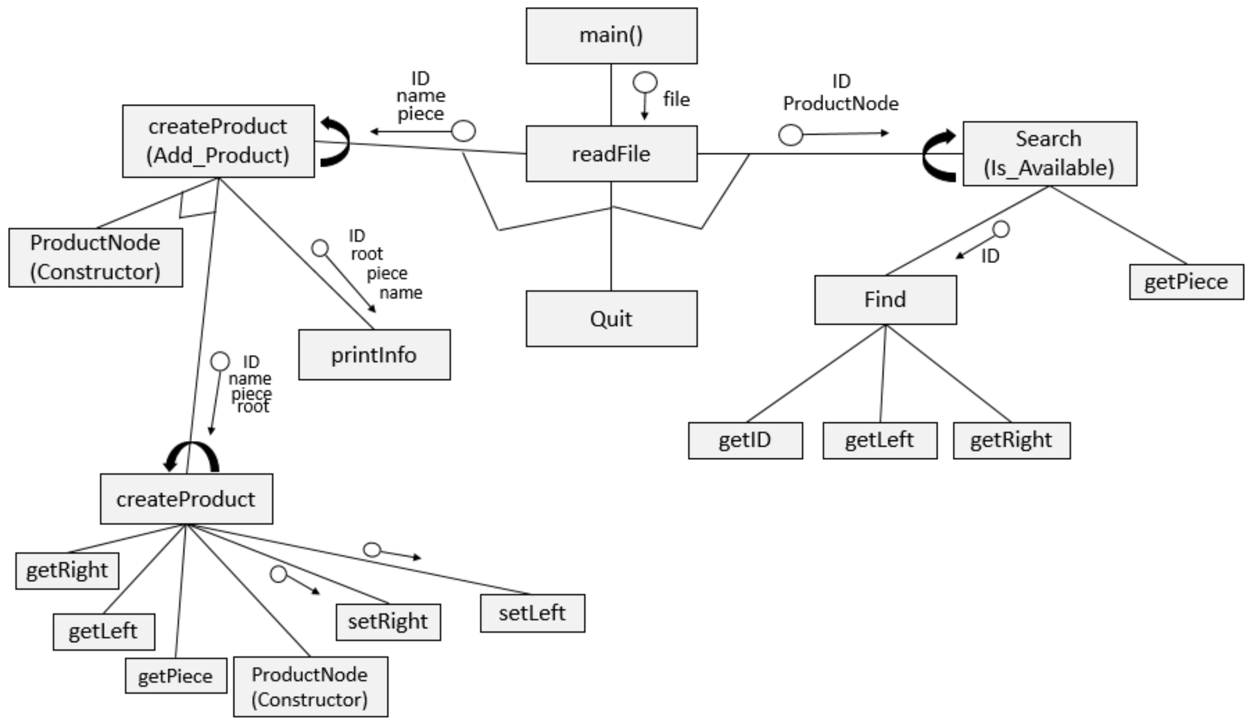
Basme Zantout - Zeynep Sude Bal **Report**

<u>Question 1</u>

➔ Problem Statement and Code Design



➔ Implementation, Functionality

- **main Method:** In the main method, we call the readFile method in order to read the given file. The readFile method takes the file name as input and reads the file line by line. Firstly, the method reads the first String at that line and this determines the ***functionality*** variable. So, we call each method which is compatible by the ***functionality*** we read in switch-case statements. The main functionalities are as follows:
- **Add_product:** This method takes ***ID, name*** and ***piece*** as parameters and adds a new product to Binary Search Tree based on these values by calling the **"*createProduct*"** which is a recursive method finding the correct place to create a new node and link it to the tree. By recursive calls in the **"*createProduct*"** method, if added product's piece is greater than current product's piece, it goes right and if added product's piece is less than current product's piece, it goes left.
- **search:** In the first place, this method calls ***Find*** method which is searching for a given product based on ID in the tree. If there is the product, the ***found*** which is boolean becomes true in the **"Find"** method, and the **"search"** method prints the messages according to this boolean value.

- **printInfo:** It takes a ***root***, ***ID***, ***name*** and ***piece*** as parameters and basically prints the ID*,* name, and piece values.
- **Quit:** This is for printing the message that the program ends.

➔ Testing

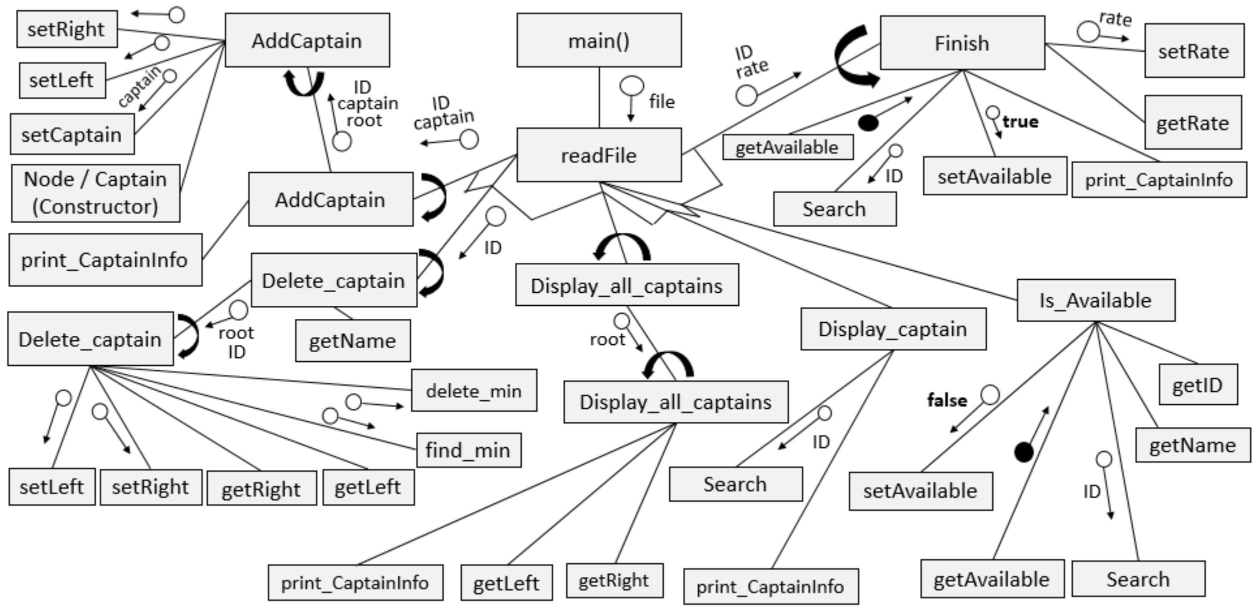
While testing this class, we tested the code by altering the ID, name, and piece of some products in order to ensure that it continued to produce the desired output. Also, we considered changing the functionalities' positions in the input file to check whether the code works or not. Moreover, we checked the code's adaptability to various file structures and functionalities' ordering by adding new functionalities and editing others. On the other hand, while the code works perfectly in Eclipse, we have faced ***"cannot find symbol"*** errors relating to the TreeNode.java class for the getID***()*** method from the ProductNode.java class. While calling getPiece() and getName() methods in the same way, we did not get the same error.

➔ Final Assessments

The challenging part in this assignment was to cope with the errors which VPL produce as "cannot find symbol". We have had trouble understanding what is the problem in the codes related to this error. On the other hand, this assignment was completely nice to consolidate the subject of Binary Search Trees in terms of approaching the subject and its implementation.

Question 2

➔ Problem Statement and Code Design



➔ Implementation, Functionality

- **main Method:** we read the text file using a method called **"readFile"** which takes the file name as input and reads it line by line. When reading each line the method first reads the first

String at that line which determines the *functionality* that should be implemented. So, in a switch-case statement, we call the method that has to be implemented based on the *functionality* we read. The main functionalities are as follows:

- **AddCaptain:** adds a new Captain to the Binary Search Tree based on the key value (ID). It calls a recursive **"Add_Captain"** method which finds the right place to create a new node and link it to the tree.
- **Delete_captain:** this method deletes a Captain from the tree based on the key value (ID). It calls a recursive **"Delete_captain"** method which finds the node that will be deleted (if it exists) and returns the new link to the tree root. (Note: this methods also uses **"delete_min"** and **"find_min"** methods that find and delete a specific node)
- **Search:** this method is a private method used by **"Is_Avaialable"**, **"Display_captain"**, and **"Finish"** methods to search for a given captain in the tree (using his ID).
- **Is_Avaialable:** the method **"Search"es** for the node with the given ID and rents a car with the Captain if he/she is available. Otherwise, it prints an error message indicating that the Captain is either unavailable or is not found.
- **Display_captain:** the method**"Search"es** for the node with the given ID and displays his/her information (if the captain exists). Otherwise, it prints an error message indicating that the Captain is not found.
- **Finish:** the method **"Search"es** for the node with the given ID and finishes the ride with the given captain (if the captain exists). Otherwise, it prints an error message indicating that the Captain either was not on a ride or was not found**.**
- **Display_all_captains:** this method is an "In-order Traversal" method that traverses through every node in the list in ascending order to print out all the Captains' information

➔ Testing

For testing this class we tried to switch the places of the functionalities in the input file, we also tested the code by adding more of the functionalities, deleting some, and editing some to make sure that the code is flexible on different types of structures of the files and order of functionalities. Moreover, we tested the code by changing the data of some Captains like the ID, Name, and rate, to test whether the code still gives the wanted output. For example, we changed the rate to make sure that it doesn't go beyond its range from 0 to 5.

➔ Final Assessments

The trouble points in completing this assignment was the VPL section. So, the parts that were most challenging were trying to figure out how to solve the errors that appeared when submitting the code on VPL. However, what we liked about the assignment is that we learned how to handle such troubles. For example, by observing the expected output we understood that we had to scan the input file name and by looking over the error messages we understood that the package import had to be deleted and we learned how to resolve the "symbol cannot be accessed" error.