Course Project ADT

To implement the course project’s functionality, I have defined a total of fifteen Abstract Data Types, divided into two categories. The first category consists of data types describing entities within each list. This includes the following ADTs:

1. Product – Each Product instance has an ID, a condition, which can either be new or refurbished, a name, and a delivery time. All the properties can be accessed and mutated through their respective getter and setter functions. Products can also vary in prices depending on their condition, which the function getTotalPrice will consider when calculating the cost.
2. Complaint – Complaints encompass an ID, a title, and a description, accessible and mutable through their getter and setter methods.
3. Truck – Trucks include an ID, a year of production, load, required licenses, make, and the model, all having their respective accessors and mutators.
4. Van- Vans are identical to trucks in terms of data fields and functionality.
5. Route – Each route has an ID, a distance in km, a starting point, a final destination, and a color code. These properties can be accessed and mutated through their getters and setters.
6. Delivery – Deliveries represent a completed product order. They consist of an ID, vectors of pointers to the delivered products, the covered routes, and received complaints regarding the order, a pointer to the driver carrying out the delivery, a title, a description, and a rating. Each data field can be accessed and altered through its accessor and mutator methods.

The second category contains employee types and comprises of the following ADTs:

1. Employee – This data type represents a basic employee with an ID, a salary, user permissions, defined by the polymorphic setPermissions method, a name, and an email. Together with the function determining the permissions and the getters and setters, this class contains a polymorphic function calculating the salary depending on the employee’s position.
2. Administrator – A class that completely inherits all data and functionality from Employee, overriding its setTotalSalary and setPermissions methods.
3. Driver – Identical to Administrator but sets a different salary and permissions.
4. Clerk – Inherits Administrator redefining the salary and permissions.
5. Manager – Inherits Administrator redefining the salary and permissions.
6. SuperAdmin – Inherits Administrator redefining the salary and permissions.
7. TruckDriver – Inherits Driver redefining the salary and permissions.
8. VanDriver – Inherits Driver redefining the salary and permissions.
9. Supervisor – Inherits Driver redefining the salary and permissions. Furthermore, this class includes an enumerator specifying whether the supervisor is a truck or a van driver, which in turn determines the user’s permissions.

The two main purposes of the application would be to allow searching by a keyword within a given list and sort the list by a certain property. The program will utilize the Boyer-Moore string pattern-matching algorithm for searching by a keyword and Mergesort for sorting the lists because it directly allows setting the sorting property and order instead of relying on a pivot, as is the case with quicksort. It also runs relatively quickly with a complexity of O (NlogN).