Turnitin Originality Report

Processed on: 17-Jan-2024 21:43 +08
ID: 2272514700
Word Count: 893
Submitted: 2

TechTurtles Project Report By Azhar Yazid

Similarity Index	Similarity by Source	
0%	Internet Sources:	0%
	Publications:	0%
	Student Papers:	0%

Objective In this project, Courier Service System aims to: To utilize a stack and queue operations to manage packages, the administrator can add new packages with information like the tracking number, address, sender, recipient, category, and delivery status, among other tasks, using the system. The system also uses stack and queue operations to delete a package. Synopsis First and foremost, the system allows the administrator to add a new parcel using two methods; using stack or queue operations. When the user is using stack operations to add a new parcel, the new parcel will be "pushed" at the top of the list. Then, when the user is using queue operations, the new parcel will be added at the back of the list. Secondly, administrators can delete existing parcels using either stack or queue operations. When the user is using stack operations to delete an existing parcel, the parcel will be "popped" from the top of the list. Then, when the user is using queue operations, the parcel will be removed from the front of the list. Design Pseudocode Start The system reads the file data from ParcelDataProject.txt. If there is an error in the opening file, the system will terminate. The system will prompt the user to input a choice number between 1 to 3. Case 1: If user selects case 1, the user will be implementing stack operations to enter a new parcel using push function, or remove an item using pop function. If case 1, the parcel will be added at the top of the stack. Else if case 2, the parcel will be removed from the top of the stack. Else if case 3, the user will be returned to the main menu. Else, the choice made by the user is invalid. Case 2 : If user enters case 2, the user will be implementing queue operations to enter a new parcel using enQueue function, or remove an item using deQueue function. If case 1, the parcel will be added at the back of the queue. Else if case 2, the parcel will be removed from the front of the queue. Else if case 3, the user will be returned to the main menu. Else, the choice made by the user is invalid. Case 3 : If the user enters choice 3, the system will be terminated. Else the system will display "Invalid choice. Please enter a number between 1 and 3". End Description of how to implement data structure operations: stack and queue. In this program, we apply stack and queue operations and utilize a class named ParcelList. Then, the ParcelList class uses stack and queue templates in its class. The implementation of stack and queue operations here are used for adding a parcel at the top of the stack or at the back of the queue, or deleting a parcel from the list, either by deleting a parcel from the top of the stack or the front of the queue. The two main classes Parcel and ParcelList are used in the provided C++ code to construct a courier service system. The technology makes it possible to handle packages using a queue or a stack. The ParcelList class has functions for adding parcels to the stack (addNode), showing all parcels in the stack (displayAllNodes), and removing the most recent parcel added (deleteNode) among other stack

operations. The std::stack data structure is used by the stack implementation. The ParcelList class, on the other hand, has functions for handling queue activities, such as enqueueing a parcel, dequeuing the first parcel added, and displayAllNodesQueue, which shows every parcel in the queue. The queue data structure is used by the queue implementation. Using a stack or a queue, these operations let users add, remove, and inspect parcel details in order to interact with the system. The primary menu of the programme allows you to choose between stack and queue operations, giving you freedom in how you manage the Courier Service System. For OOP principles used in our code, we applied encapsulation, in which the tracking number, address, sender/receiver names, category, status, and remark, along with any associated methods, are all encapsulated within the Parcel class. Public getter functions (getTrackingNumber(), getAddress(), etc.) regulate access to the data members, while member functions (addNoteToParcel(), displayDetails(), etc.) make it easier to modify the data. Next, we used abstraction by shielding the user from the implementation specifics, the Parcel and ParcelList classes offer a high-level abstraction. Through well specified interfaces (member functions), users can interact with parcels and parcel lists without having to comprehend the underlying technology. Then, inheritance is an essential OOP concept, even though it isn't shown clearly in the code that is provided. Inheritance could be used to build derived classes that inherit the characteristics and actions of the base classes in the event that the system evolved to include additional kinds of parcels or specialized lists. Function overloading is the means by which polymorphism is accomplished. For instance, the Parcel class's displayShippingCategory() and displayStatusDelivery() methods are overloaded to offer various behaviors depending on the category and status of the parcel, respectively. When instances of the Parcel class are included as members of the ParcelList class, it illustrates composition. The ParcelList class uses the features offered by the Parcel class to manage parcels. It is made up of a stack (parcelStack) and a queue (parcelQueue).