Solution: Linked list behind the scenes

Add customer already implemented as given first step check if the list is empty (if *head* is null). If it is, the new customer becomes the head. Otherwise, find the last node and add the new customer there.

Similarly, to solve this challenge, follow the instructions below:

1. Create a New Node:
   1. First, create a new node to store the VIP customer’s information.
   2. This new node will hold the customer's name and details.
2. Link the New Node to the Current Head:
   1. Next, you’ll link the new node to the current list.
   2. Since the new node is a VIP, it should be placed at the beginning of the list.
   3. To do this, set the new node’s *next* reference to point to the current head of the list. This step connects the new VIP customer to the existing list, effectively inserting them at the start.
3. Update the Head to the New Node:
   1. Finally, you need to update the *head* of the list to this new node.
   2. This step makes the new VIP customer the first node in the linked list.

—-------------------------------------------------------------------

public class Main {

public static void main(String[] args) {

LinkedList waitlist = new LinkedList();

// Adding regular customers

waitlist.addCustomer("Alice", "Party of 2");

waitlist.addCustomer("Bob", "Party of 3");

// Adding VIP customers

waitlist.addVIPCustomer("VIP Charlie", "Party of 1");

waitlist.addVIPCustomer("VIP Dave", "Party of 4");

// Adding another regular customer

waitlist.addCustomer("Eve", "Party of 2");

// Displaying the final waitlist

waitlist.printList();

}

}

Private static class Node {

String name;

String details;

Node next;

Node(String name, String details) {

this.name = name;

this.details = details;

this.next = null;

}

}

class LinkedList {

Node head;

LinkedList() {

this.head = null;

}

// Method to add a regular customer to the end of the list

void addCustomer(String name, String details) {

Node newNode = new Node(name, details);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

// Method to add a VIP customer at the beginning of the list

void addVIPCustomer(String name, String details) {

//Step 1: Create the new node the *head*

Node newNode = new Node(name, details);

//Step 2: Point it to the *current head* of the list.

newNode.next = head;

// Step 3: Update the Head to the New Node

head = newNode;

}

// Method to print the linked list

void printList() {

Node current = head;

while (current != null) {

System.out.print(current.name + " (" + current.details + ")");

current = current.next;

if (current != null) {

System.out.print(" -> ");

}

}

System.out.println(" -> null");

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Method to update a customer's details

void updateCustomer(String customerName, String newDetails) {

    Node current = head;

    while (current != null) {

        if (current.name.equals(customerName)) {

            current.details = newDetails; // Update the customer’s details

            return;

        }

        current = current.next;

    }

}

/////////////////////////////////////////////////////////////

// Method to remove a specific customer from the waitlist

void removeCustomer(String customerName) {

    if (head == null) return; // If the waitlist is empty, there's nothing to remove

    // If the customer to remove is at the head of the list

    if (head.name.equals(customerName)) {

        head = head.next; // Move the head to the next customer

        return;

    }

    Node current = head;

    while (current.next != null && !current.next.name.equals(customerName)) {

        current = current.next;

    }

    if (current.next != null) {

        current.next = current.next.next; // Bypass the node to remove the customer

    }

}