DELIVERY SYSTEM

Data Structures Project – Bakery

ABSTRACT

Report of a Delivery System program for a Bakery, implemented with Data Structures concepts using Java language

Sandi Zein El Zein

Amani Khaled Merhi

CMPS347

Table of Contents

➢ Introduction:

This project report presents the design and implementation of a robust food delivery system using java data structures. The system provides the customer with a user-friendly list of options to ensure ordering our baked goods and receiving the order smoothly. In addition, the system also grants the staff a variety of options. For example, the admin can decide whenever to ship the orders, while the driver ensures that the order arrives at the customer's doorstep.

➢ Problem Statement:

Our order processing and delivery system aims to address the challenge of delivering orders to different regions efficiently by implementing advanced data structures concepts. Our vans deliver customer orders from the nearest region to the furthest one using the priority queue concept.

The system also deals with serving customers' orders smoothly without losing track of order, the customer can track whether his order has been shipped or not, to eliminate any customer inconvenience. Furthermore, the admin can see whether the order has been delivered or yet to be delivered, ensuring that all customers' orders are served successfully.

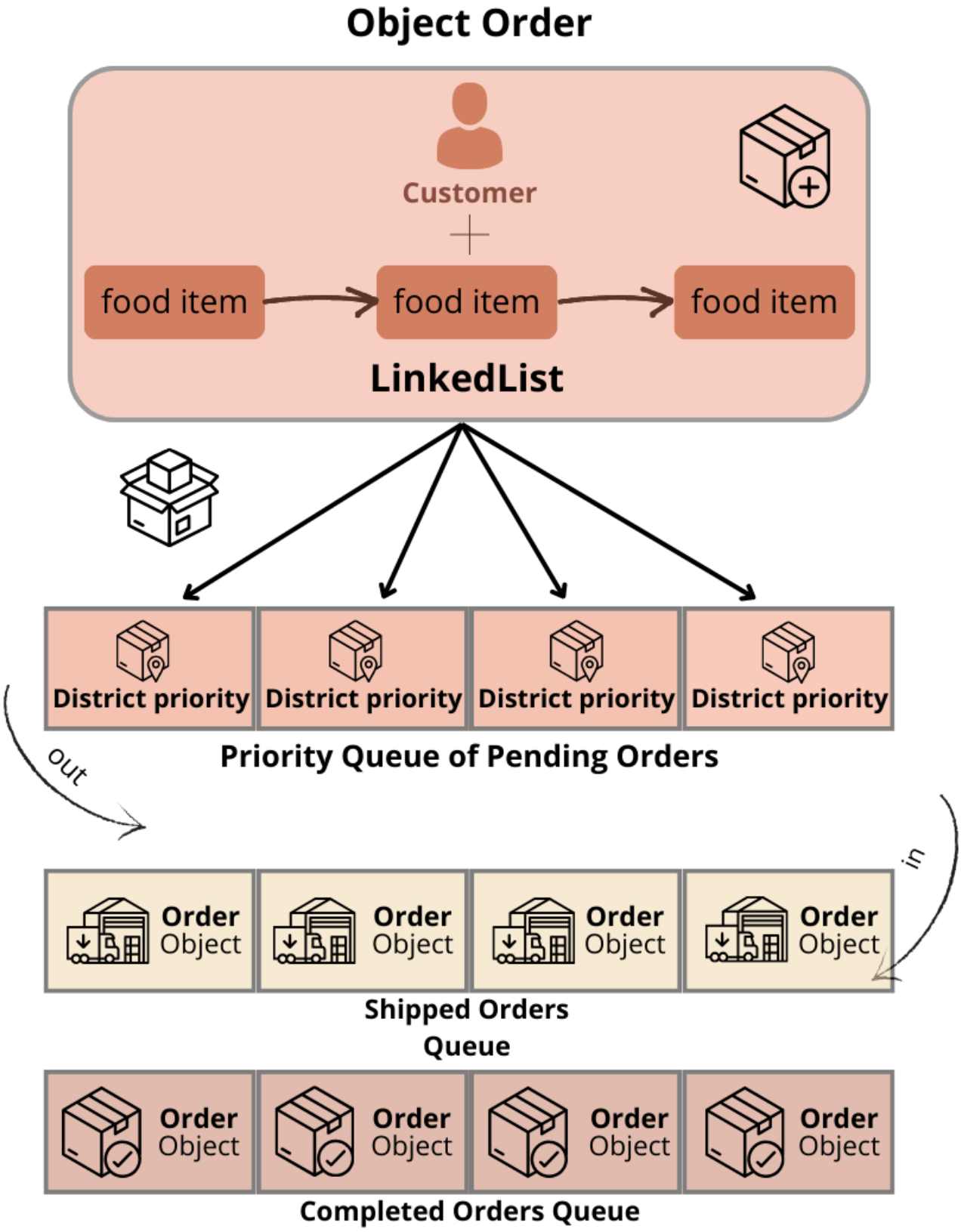
➢ Representation of Data Structures:

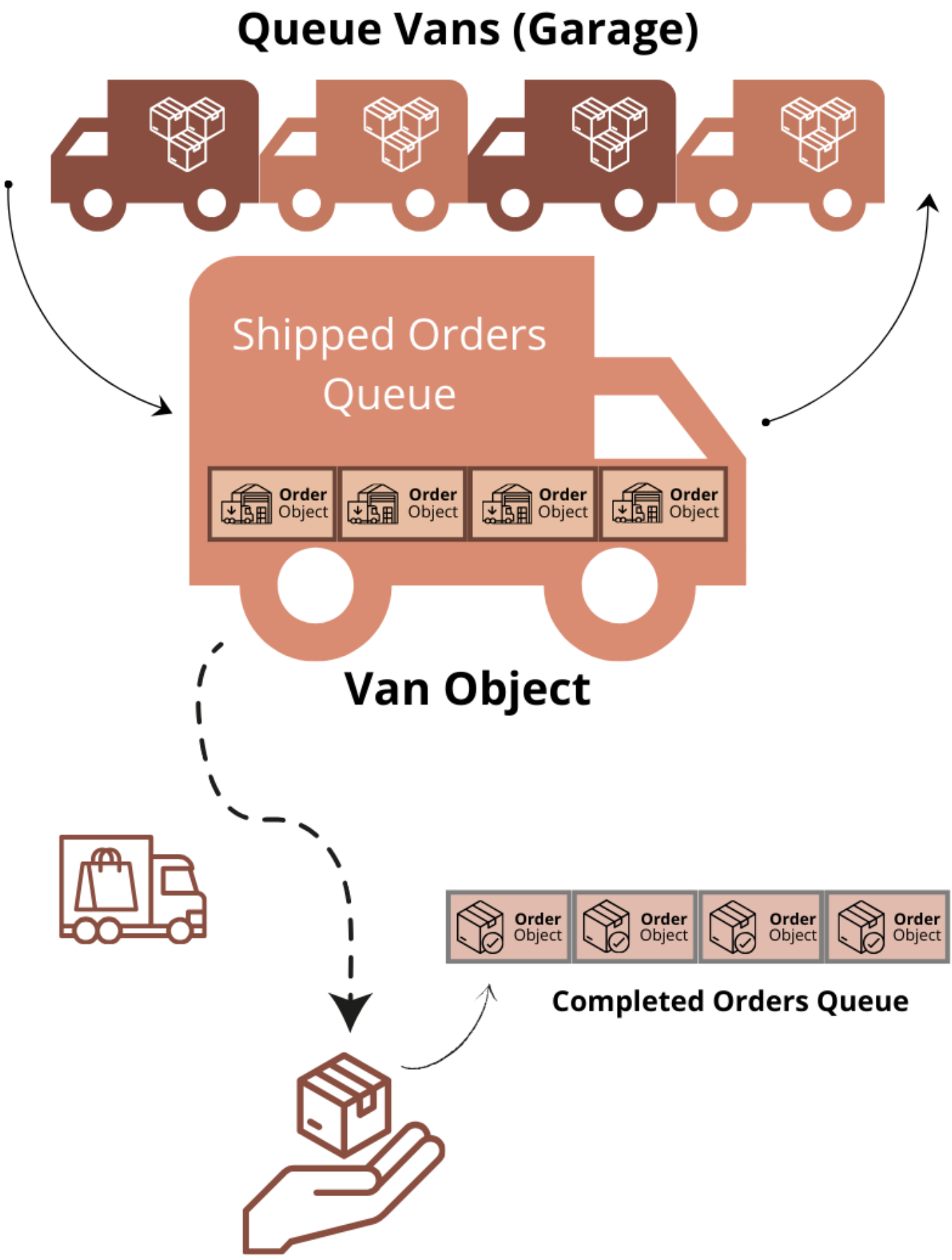
Linked List: We chose the Linked List concept for representing food items in each order chosen by the customer, since he is not obliged to choose a limited quantity of items.

Priority Queue: We chose the priority queue concept to add orders in a specific order, based on the customer's address distance from the bakery shop, to ensure an efficient shipping for the orders. Facilitating the driver' trip through a one-way path while saving time and fuel.

Queue: We found the Queue concept suitable for representing the garage of vans, when shipping orders, the first van in the garage leaves, delivers orders and then returns back to the queue.

➢ Data Structures Diagram:

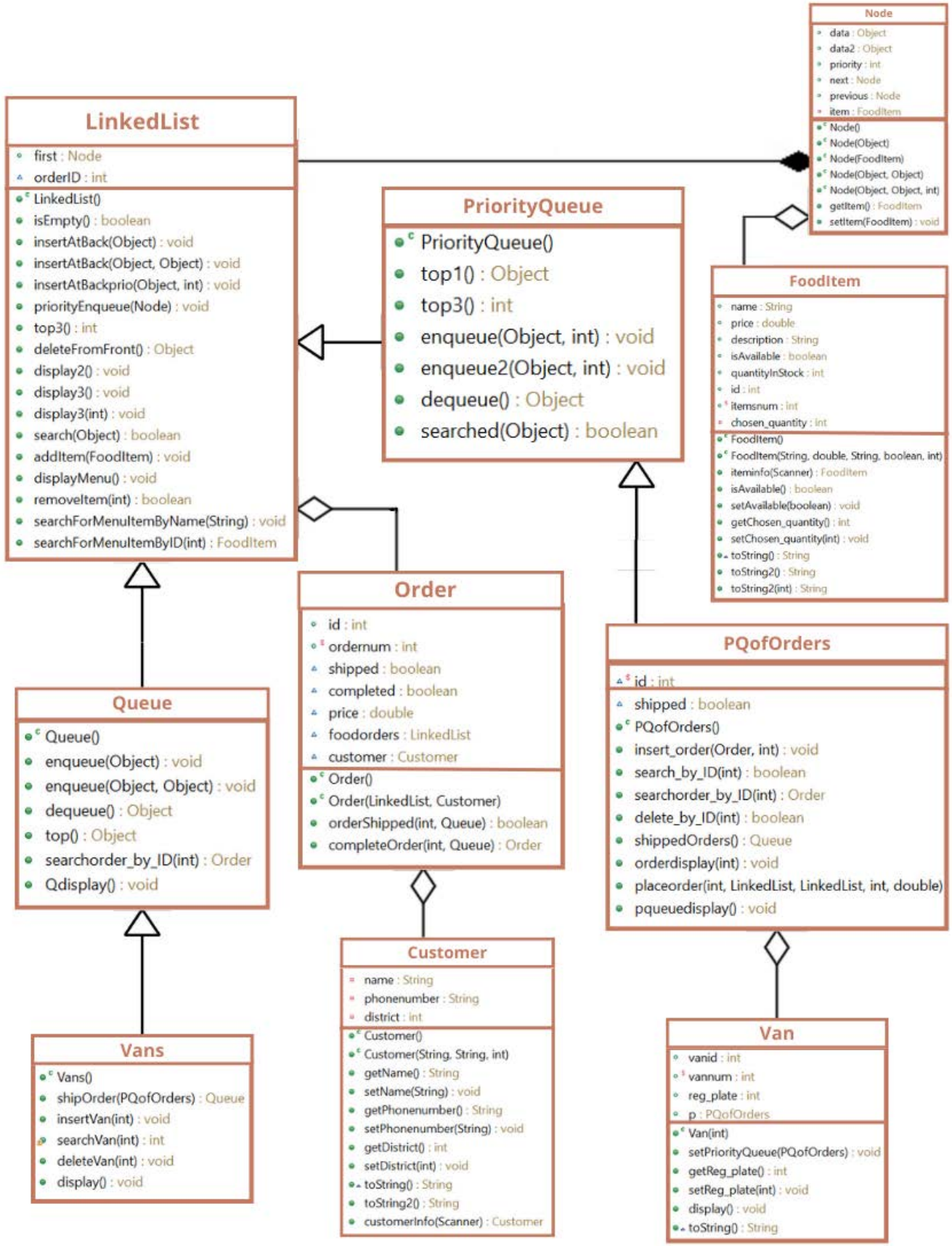



Queue Vans (Garage)
Shipped Orders Queue
Order Object
Order Object
Order Object
Order Object
Van Object
Order Object
Order Object
Order Object
Order Object
Completed Orders Queue

➢ Data Structures Relationship explanation:

In our project, object Order consists of the object customer and his food order that is expressed as a LinkedList of food items. This order is an object in the Priority Queue of orders, which has customer's district as a priority, the order is delivered to the nearest customer first. The orders present in the priority queue are afterwards transferred to a queue of shipped orders when moved to an object Van that is present in a queue of vans. The van then leaves the queue of vans to ship the orders to the customer's doorstep. Once done, the delivered order is transferred to a queue of completed orders that can be displayed in the admin's section.

➢ UML Diagram:


Node
data : Object
data2 : Object
priority : int
next : Node
previous : Node
item : FoodItem
Node()
Node(Object)
Node(FoodItem)
Node(Object, Object)
Node(Object, Object, int)
getItem() : FoodItem
setItem(FoodItem) : void
LinkedList
first : Node
orderID : int
LinkedList()
isEmpty() : boolean
insertAtBack(Object) : void
insertAtBack(Object, Object) : void
insertAtBackprio(Object, int) : void
priorityEnqueue(Node) : void
top3() : int
deleteFromFront() : Object
display2() : void
display3() : void
display3(int) : void
search(Object) : boolean
addItem(FoodItem) : void
displayMenu() : void
removeItem(int) : boolean
searchForMenuItemByName(String) : void
searchForMenuItemByID(int) : FoodItem
PriorityQueue
PriorityQueue()
top1() : Object
top3() : int
enqueue(Object, int) : void
enqueue2(Object, int) : void
dequeue() : Object
searched(Object) : boolean
FoodItem
name : String
price : double
description : String
isAvailable : boolean
quantityInStock : int
id : int
itemsnum : int
chosen_quantity : int
FoodItem()
FoodItem(String, double, String, boolean, int)
iteminfo(Scanner) : FoodItem
isAvailable() : boolean
setAvailable(boolean) : void
getChosen_quantity() : int
setChosen_quantity(int) : void
toString() : String
toString2() : String
toString2(int) : String
Order
id : int
ordernum : int
shipped : boolean
completed : boolean
price : double
foodorders : LinkedList
customer : Customer
Order()
Order(LinkedList, Customer)
orderShipped(int, Queue) : boolean
completeOrder(int, Queue) : Order
PQofOrders
id : int
shipped : boolean
PQofOrders()
insert_order(Order, int) : void
search_by_ID(int) : boolean
searchorder_by_ID(int) : Order
delete_by_ID(int) : boolean
shippedOrders() : Queue
orderdisplay(int) : void
placeorder(int, LinkedList, LinkedList, int, double)
pqueuedisplay() : void
Queue
Queue()
enqueue(Object) : void
enqueue(Object, Object) : void
dequeue() : Object
top() : Object
searchorder_by_ID(int) : Order
Qdisplay() : void
Customer
name : String
phonenumber : String
district : int
Customer()
Customer(String, String, int)
getName() : String
setName(String) : void
getPhonenumber() : String
setPhonenumber(String) : void
getDistrict() : int
setDistrict(int) : void
toString() : String
toString2() : String
customerInfo(Scanner) : Customer
Vans
Vans()
shipOrder(PQofOrders) : Queue
insertVan(int) : void
searchVan(int) : int
deleteVan(int) : void
display() : void
Van
vanid : int
vannum : int
reg_plate : int
p : PQofOrders
Van(int)
setPriorityQueue(PQofOrders) : void
getReg_plate() : int
setReg_plate(int) : void
display() : void
toString() : String

➢ Methods Explanation:

❖ Customer class:

- **CustomerInfo():** Takes informations from user such as name, phone number, and district and store it in object customer then returnd it.

❖ FoodItem:

- **iteminfo():** Used for inserting an item in the menu, takes the item's name, price, description, quantity and returns a FoodItem object.
- **isAvailable():** Checks if there is still quantity of a specific item in stock.
- **setAvailable():** Updates an item's availability based on its quantity in stock.

❖ Order:

- **orderShipped():** Checks if the orders is shipped based on its id.
- **completeOrder():** Searches for the order in the queue of shipped orders based on its id and marks the order as completed.

❖ PQofOrders:

- **insert_order():** Inserts the customer order (which is an object that contains the object customer and a linked list of food items that he ordered).
  It enqueues the customer order based on distance priority, the order that was placed from the nearset district to the bakery shop gets inserted at front.
- **search_by_ID():** Searches for the order in the priority queue of orders based on its id.
- **delete_by_ID():** Deletes an order from the priority queue of orders based on its id.
- **shippedOrders():** Removes orders from priority queue of orders and inserts them in the same prioritized way (based on distance priority) in the queue Shipped.
- **orderdisplay():** Displays all orders of a specific region.
- **placeorder():** Searches for a food item in the menu based on its id, checks if the wanted quantity is available in stock, then adds the item into the customer's linked list of foodorders, sums up the total price and returns it.
- **pqueuedisplay():** Displays Orders of all regions present in the priority queue

❖ Vans:

- **shipOrder():** Inserts the queue of shipped orders in the van and sends the van to deliver the orders
- **insertVan():** Inserts a van object into the Vans queue.
- **searchVan():** Searches for a van in the queue based on its plate number and returns the van's id

- **deleteVan():** Deletes a van in the queue based on its plate number.
- **display():** Displays all vans in the queue.

❖ LinkedList:

- **addItem():** Adds a food item to the menu linked list
- **displayMenu():** Displays all food items present in the menu.
- **removeItem():** Deletes a food item from the menu based on its id.
- **searchForMenuItemByName**(): Searches for a food item on the menu based on its name.
- **searchForMenuItemByID():** Searches for a food item on the menu based on its id and returns it.

❖ Queue:

- **searchorder_by_ID():** Searches for an order in a queue based on its id and returns it.
- **Qdisplay():** Displays Orders of all regions present in the queue.

➢ Run and test methods:

```
******** WELCOME ********
1. Re-Show list of options
2. View the Menu
3. Make An Order
4. Search for a menu item by name
5. Search for your order
6. Delete your order
7. Track order
8. Staff only section
0. Exit the program
choose an integer:
2
Menu Item 1:
Title: Baguette
Price: $2.0
Availabile in Stock
---------------------------
Menu Item 2:
Title: Cookie
Price: $1.0
Availabile in Stock
---------------------------
Menu Item 3:
Title: Croissant
Price: $3.0
Availabile in Stock
---------------------------
Menu Item 4:
Title: Cake
Price: $9.0
Availabile in Stock
---------------------------
Menu Item 5:
Title: Eclaire
Price: $5.0
```

```
Availabile in Stock
---------------------------
Menu Item 6:
Title: cinnamonrolls
Price: $4.0
Availabile in Stock
---------------------------
choose a number from the list above:
3
******** Customer's info ********
Name:
Dr.Ali
Phone number:
70870956
Enter your district number:
1. Beirut
2. Khalde
3. Debbieh
4. Zaarouriyeh
3
How many items you want to order:
2
Please enter the id of the item you want to order:
1
Enter the quantity of item:
2
Please enter the id of the item you want to order:
4
Enter the quantity of item:
3
| Order ID   | Quantity   | item ID    | Name            | Price      | Description
| 1          | 2          | 1          | Baguette        | 2.0        | French baguette

| Order ID   | Quantity   | item ID    | Name            | Price      | Description
| 1          | 3          | 4          | Cake            | 9.0        | Moist chocolate Cake
```

```
Customer [name=Dr.Ali, phonenumber=70870956, district=3]
Total price: 31.0$ + 3$ delivery fee = 34.0$
Thank you for visiting our bakery! ;)
choose a number from the list above:
3
******** Customer's info ********
Name:
Madona
Phone number:
70640899
Enter your district number:
1. Beirut
2. Khalde
3. Debbieh
4. Zaarouriyeh
1
How many items you want to order:
2
Please enter the id of the item you want to order:
5
Enter the quantity of item:
3
Please enter the id of the item you want to order:
3
Enter the quantity of item:
2
| Order ID   | Quantity   | item ID    | Name            | Price      | Description
| 2          | 3          | 5          | Eclaire         | 5.0        | French choux pastry

| Order ID   | Quantity   | item ID    | Name            | Price      | Description
| 2          | 2          | 3          | Croissant       | 3.0        | Classic croissant

Customer [name=Madona, phonenumber=70640899, district=1]
Total price: 21.0$ + 3$ delivery fee = 24.0$
Thank you for visiting our bakery! ;)
```

```
choose a number from the list above:
3
******** Customer's info ********
Name:
Amani
Phone number:
81937212
Enter your district number:
1. Beirut
2. Khalde
3. Debbieh
4. Zaarouriyeh
2
How many items you want to order:
2
Please enter the id of the item you want to order:
4
Enter the quantity of item:
10
Sorry we are out of stock of Cake
Please enter the id of the item you want to order:
4
Enter the quantity of item:
7
| Order ID   | Quantity   | item ID    | Name            | Price      | Description
| 3          | 7          | 4          | Cake            | 9.0        | Moist chocolate Cake

Customer [name=Amani, phonenumber=81937212, district=2]
Total price: 63.0$ + 3$ delivery fee = 66.0$
Thank you for visiting our bakery! ;)
choose a number from the list above:
2
Menu Item 1:
Title: Baguette
Price: $2.0

Availabile in Stock
---------------------------
Menu Item 2:
Title: Cookie
Price: $1.0
Availabile in Stock
---------------------------
Menu Item 3:
Title: Croissant
Price: $3.0
Availabile in Stock
---------------------------
Menu Item 4:
Title: Cake
Price: $9.0
Unavailable in Stock
---------------------------
Menu Item 5:
Title: Eclaire
Price: $5.0
Availabile in Stock
---------------------------
Menu Item 6:
Title: cinnamonrolls
Price: $4.0
Availabile in Stock
---------------------------
choose a number from the list above:
3
******** Customer's info ********
Name:
Sandy
Phone number:
71450749
Enter your district number:
```

```
1. Beirut
2. Khalde
3. Debbieh
4. Zaarouriyeh
4
How many items you want to order:
1
Please enter the id of the item you want to order:
2
Enter the quantity of item:
12
| Order ID   | Quantity   | item ID    | Name           | Price      | Description
| 4          | 12         | 2          | Cookie         | 1.0        | chocolate chips cookies

Customer [name=Sandy, phonenumber=71450749, district=4]
Total price: 12.0$ + 3$ delivery fee = 15.0$
Thank you for visiting our bakery! ;)
choose a number from the list above:
4
Enter the menu item you want to search for:
donut
donut is unfortunately not present on the menu!
choose a number from the list above:
cookie
Invalid input! Please enter a valid integer.
choose a number from the list above:
4
Enter the menu item you want to search for:
cookie
cookie is present on the menu!
---------------------------
Title: Cookie
Price: $1.0
Availabile in Stock
---------------------------

choose a number from the list above:
5
Enter your order's ID:
1
Your order is:
| Order ID   | Quantity   | item ID    | Name           | Price      | Description
| 1          | 2          | 1          | Baguette       | 2.0        | French baguette

| Order ID   | Quantity   | item ID    | Name           | Price      | Description
| 1          | 7          | 4          | Cake           | 9.0        | Moist chocolate Cake

choose a number from the list above:
6
Enter the ID of the order you want to delete:
4
Your order has been successfully deleted.
choose a number from the list above:
7
Enter the id of the item you want to track:
2
The order is still in the proccess!
choose a number from the list above:
8
Please enter password for staff operation:
1234
1. Admin
2. Driver
0. Exit program
choose an integer:
1
1. Re-show Admin list
2. View Pending Orders
3. Ship all orders
4. View completed orders
5. Display total bill of Pending Orders
```

```
6. Display total Earnings of Served Orders
7. Add a new item to the menu
8. Delete an item from the menu
9. Display orders of a specific region
0. Exit Admin list
choose an integer:
2
Pending Orders Display:
| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 5              | Eclaire               | 5.0        | French choux pastry             | true          | 17              |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 5              | Eclaire               | 5.0        | French choux pastry             | true          | 17              |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 5              | Eclaire               | 5.0        | French choux pastry             | true          | 17              |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 3              | Croissant             | 3.0        | Classic croissant               | true          | 22              |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 3              | Croissant             | 3.0        | Classic croissant               | true          | 22              |

Customer [name=Madona, phonenumber=70640899, district=1]

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 4              | Cake                  | 9.0        | Moist chocolate Cake            | false         | 0               |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 4              | Cake                  | 9.0        | Moist chocolate Cake            | false         | 0               |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 4              | Cake                  | 9.0        | Moist chocolate Cake            | false         | 0               |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |

| 4              | Cake                  | 9.0        | Moist chocolate Cake            | false         | 0               |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 4              | Cake                  | 9.0        | Moist chocolate Cake            | false         | 0               |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 4              | Cake                  | 9.0        | Moist chocolate Cake            | false         | 0               |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 4              | Cake                  | 9.0        | Moist chocolate Cake            | false         | 0               |

Customer [name=Amani, phonenumber=81937212, district=2]

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 1              | Baguette              | 2.0        | French baguette                 | true          | 10              |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 1              | Baguette              | 2.0        | French baguette                 | true          | 10              |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 4              | Cake                  | 9.0        | Moist chocolate Cake            | false         | 0               |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 4              | Cake                  | 9.0        | Moist chocolate Cake            | false         | 0               |

| item ID        | Name                  | Price      | Description                     | Available     | Stock Quantity  |
| 4              | Cake                  | 9.0        | Moist chocolate Cake            | false         | 0               |

Customer [name=Dr.Ali, phonenumber=70870956, district=3]

choose an integer:
5
Total bill of pending orders: 115.0
choose an integer:
6
```

```
Total earnings of served orders: 0.0
choose an integer:
7
Enter the item's name:
Donut
Enter the item's price:
6
Enter the item's description:
Glazed Donuts
Enter the quantity in stock:
30
The item Donut has been added successfully.
choose an integer:
8
Enter the id of the menu item you want to delete:
4
The item has been deleted successfully.
choose an integer:
0
1. Admin
2. Driver
0. Exit program
0
choose a number from the list above:
2
Menu Item 1:
Title: Baguette
Price: $2.0
Availabile in Stock
---------------------------
Menu Item 2:
Title: Cookie
Price: $1.0
Availabile in Stock
---------------------------

Menu Item 3:
Title: Croissant
Price: $3.0
Availabile in Stock
---------------------------
Menu Item 4:
Title: Eclaire
Price: $5.0
Availabile in Stock
---------------------------
Menu Item 5:
Title: cinnamonrolls
Price: $4.0
Availabile in Stock
---------------------------
Menu Item 6:
Title: Donut
Price: $6.0
Availabile in Stock
---------------------------
choose a number from the list above:
8
Please enter password for staff operation:
1234
1. Admin
2. Driver
0. Exit program
choose an integer:
1
1. Re-show Admin list
2. View Pending Orders
3. Ship all orders
4. View completed orders
5. Display total bill of Pending Orders
6. Display total Earnings of Served Orders
```

```
7. Add a new item to the menu
8. Delete an item from the menu
9. Display orders of a specific region
0. Exit Admin list
choose an integer:
9
Enter your district number:
1. Beirut
2. Khalde
3. Debbieh
4. Zaarouriyeh
3
All orders of Debbieh:
| item ID         | Name                | Price      | Description                  | Available    | Stock Quantity  |
| 1               | Baguette            | 2.0        | French baguette              | true         | 10              |

| item ID         | Name                | Price      | Description                  | Available    | Stock Quantity  |
| 1               | Baguette            | 2.0        | French baguette              | true         | 10              |

| item ID         | Name                | Price      | Description                  | Available    | Stock Quantity  |
| 4               | Cake                | 9.0        | Moist chocolate Cake         | false        | 0               |

| item ID         | Name                | Price      | Description                  | Available    | Stock Quantity  |
| 4               | Cake                | 9.0        | Moist chocolate Cake         | false        | 0               |

| item ID         | Name                | Price      | Description                  | Available    | Stock Quantity  |
| 4               | Cake                | 9.0        | Moist chocolate Cake         | false        | 0               |

Customer [name=Dr.Ali, phonenumber=70870956, district=3]
choose an integer:
3
The van 1 is in his way to ship the orders!
choose an integer:
0
1. Admin

2. Driver
0. Exit program
0
choose a number from the list above:
1
******** WELCOME ********
1. Re-Show list of options
2. View the Menu
3. Make An Order
4. Search for a menu item by name
5. Search for your order
6. Delete your order
7. Track order
8. Staff only section
0. Exit the program
choose a number from the list above:
7
Enter the id of the item you want to track:
1
The order is in his way to your door step!
choose a number from the list above:
8
Please enter password for staff operation:
1234
1. Admin
2. Driver
0. Exit program
choose an integer:
2
1. Re-show Driver list
2. View Shipped Orders
3. Add a new van
4. Delete an existing van
5. Search for a specific van
6. Display all vans
```

```
7. Complete an Order
0. Exit driver list
choose an integer:
2
Shipped orders Display:
| Quantity   | item ID    | Name           | Price      | Description
| 3          | 5          | Eclaire        | 5.0        | French choux pastry

| Quantity   | item ID    | Name           | Price      | Description
| 2          | 3          | Croissant      | 3.0        | Classic croissant

Customer [name=Madona, phonenumber=70640899]

| Quantity   | item ID    | Name           | Price      | Description
| 7          | 4          | Cake           | 9.0        | Moist chocolate Cake

Customer [name=Amani, phonenumber=81937212]

| Quantity   | item ID    | Name           | Price      | Description
| 2          | 1          | Baguette       | 2.0        | French baguette

| Quantity   | item ID    | Name           | Price      | Description
| 7          | 4          | Cake           | 9.0        | Moist chocolate Cake

Customer [name=Dr.Ali, phonenumber=70870956]

choose an integer:
7
Enter the ID of the order you want to mark as completed
1
The order 1 has been marked as completed.
choose an integer:
0
1. Admin
2. Driver
```

```
0. Exit program
1
1. Re-show Admin list
2. View Pending Orders
3. Ship all orders
4. View completed orders
5. Display total bill of Pending Orders
6. Display total Earnings of Served Orders
7. Add a new item to the menu
8. Delete an item from the menu
9. Display orders of a specific region
0. Exit Admin list
choose an integer:
4
| Quantity   | item ID    | Name           | Price      | Description
| 2          | 1          | Baguette       | 2.0        | French baguette

| Quantity   | item ID    | Name           | Price      | Description
| 7          | 4          | Cake           | 9.0        | Moist chocolate Cake

Customer [name=Dr.Ali, phonenumber=70870956]

choose an integer:
0
1. Admin
2. Driver
0. Exit program
2
1. Re-show Driver list
2. View Shipped Orders
3. Add a new van
4. Delete an existing van
5. Search for a specific van
6. Display all vans
7. Complete an Order
```

```
choose an integer:
7
Enter the ID of the order you want to mark as completed
1
The order 1 has been marked as completed.
choose an integer:
3
Enter the van plate number you want to insert:
4927
The van 4927 has been added successfully.
choose an integer:
3
Enter the van plate number you want to insert:
5102
The van 5102 has been added successfully.
choose an integer:
4
Enter the van plate number you want to delete:
4927
Van 5 was deleted!
choose an integer:
5
Enter the van plate number you want to search for:
9584
Van of plate number 9584 is not found!
choose an integer:
5
Enter the van plate number you want to search for:
5102
Van 6 was found!

choose an integer:
6
| van id     | reg plate       |
| 2          | 9876            |

| van id     | reg plate       |
| 3          | 4897            |

| van id     | reg plate       |
| 4          | 5236            |

| van id     | reg plate       |
| 1          | 1234            |

| van id     | reg plate       |
| 6          | 5102            |

choose an integer:
0
1. Admin
2. Driver
0. Exit program
0
choose a number from the list above:
0
```

➢ Main explanation:

Within the user list, there's an option for staff operations, having both administrators and drivers. Admins possess the capability to view pending orders and access various functionalities, while drivers can observe shipped orders and execute operations related to vans.

➢ Insert Search Delete and Display:

❖ LinkedList (MenuList):

```
7. Add a new item to the menu
8. Delete an item from the menu
4. Search for a menu item by name
2. View the Menu
```

❖ Priority Queue of orders:

```
3. Make An Order
6. Delete your order
5. Search for your order
9. Display orders of a specific region
```

❖ Queue of Vans:

```
3. Add a new van
4. Delete an existing van
5. Search for a specific van
6. Display all vans
```

➢ Code Implementation:

❖ Customer class:

```
import java.util.Scanner;

public class Customer {
	private String name;
	private String phonenumber;
	private int district;

	public static final String ANSI_RESET = "\u001B[0m";
	public static final String ANSI_RED = "\u001B[91m";
	public static final String ANSI_BLUE = "\u001B[34m";
	public static final String ANSI_MAGENTA = "\u001B[95m";

	public Customer() {
		super();
	}

	public Customer(String name, String phonenumber, int district) {
		super();
		this.name = name;
		this.phonenumber = phonenumber;
		this.district = district;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getPhonenumber() {
		return phonenumber;
	}

	public void setPhonenumber(String phonenumber) {
		this.phonenumber = phonenumber;
	}

	public int getDistrict() {
		return district;
	}

	public void setDistrict(int district) {
		this.district = district;
	}

	public String toString() {
		return "Customer [name=" + name + ", phonenumber=" + phonenumber + ",
district=" + district + "]";
```

```
	}

	public String toString2() {
		return "Customer [name=" + name + ", phonenumber=" + phonenumber + "]";
	}

	public Customer customerInfo(Scanner in) {
		System.out.println(ANSI_MAGENTA + "******** Customer's info ********" +
ANSI_RESET);
		System.out.println("Name: ");
		String name = in.next();
		System.out.println("Phone number: ");
		String number = in.next();
		while (true) {
			try {
				long phoneNumber = Long.parseLong(number);
				if (phoneNumber >= 9999999) {
					break;
				} else {
					System.out.println(ANSI_RED + "Enter a valid Phone number:
" + ANSI_RESET);
					number = in.next();
				}
			} catch (NumberFormatException e) {
				System.out.println(ANSI_RED + "Enter a valid Phone number: " +
ANSI_RESET);
				number = in.next();
			}
		}
		System.out.println("Enter your district number: ");
		System.out.println(ANSI_BLUE + "1. Beirut \n2. Khalde \n3. Debbieh \n4.
Zaarouriyeh" + ANSI_RESET);
		int district = in.nextInt();
		if (district != 1 && district != 2 && district != 3 && district != 4) {
			System.out.println(ANSI_BLUE + "1. Beirut \n2. Khalde \n3. Debbieh \n4.
Zaarouriyeh" + ANSI_RESET);
			district = in.nextInt();
		}
		Customer c1 = new Customer(name, number, district);
		return c1;
	}
}
```

❖ FoodItem class:

```
import java.util.Scanner;

public class FoodItem {
	public String name;
	public double price;
	public String description;
	public boolean isAvailable;
	public int quantityInStock;
	public int id = 0;
	public static int itemsnum = 0;
	private int chosen_quantity = 0;

	public FoodItem() {
		id = itemsnum + 1;
		itemsnum++;
	}

	public FoodItem(String name, double price, String description, int quantityInStock)
{
		super();
		this.name = name;
		this.price = price;
		this.description = description;
		this.quantityInStock = quantityInStock;
		this.isAvailable = isAvailable();
		id = itemsnum + 1;
		itemsnum++;
	}

	public FoodItem iteminfo(Scanner in) {
		System.out.println("Enter the item's name: ");
		String n = in.next();
		System.out.println("Enter the item's price: ");
		double p = in.nextDouble();
		System.out.println("Enter the item's description: ");
		String des = in.nextLine();
		in.nextLine();
		System.out.println("Enter the quantity in stock: ");
		int quantity = in.nextInt();
		FoodItem item = new FoodItem(n, p, des, quantity);
		return item;
	}

	public boolean isAvailable() {
		return quantityInStock > 0;
	}

	public void setAvailable(int quantityInStock) {
		if(isAvailable())
			isAvailable = true;
		else
			isAvailable = false;
	}

	public int getChosen_quantity() {
```

```
		return chosen_quantity;
	}

	public void setChosen_quantity(int chosen_quantity) {
		this.chosen_quantity = chosen_quantity;
	}

	public String toString() {
		String format = "| %-15s | %-20s | %-10s | %-30s | %-12s | %-15s |%n";
		StringBuilder sb = new StringBuilder();

		sb.append(String.format(format, "item ID", "Name", "Price", "Description",
"Available", "Stock Quantity"));

		sb.append(String.format(format, id, name, price, description, isAvailable,
quantityInStock));

		return sb.toString();
	}

	public String toString2() {
		String format = "| %-10s | %-10s | %-15s | %-10s | %-20s %n";
		StringBuilder sb = new StringBuilder();

		sb.append(String.format(format, "Quantity", "item ID", "Name", "Price",
"Description"));

		sb.append(String.format(format, getChosen_quantity(), id, name, price,
description));

		return sb.toString();
	}

	public String toString2(int OrderID) {
		String format = "| %-10s | %-10s | %-10s | %-15s | %-10s | %-20s %n";
		StringBuilder sb = new StringBuilder();

		sb.append(String.format(format, "Order ID", "Quantity", "item ID", "Name",
"Price", "Description"));

		sb.append(String.format(format, OrderID, getChosen_quantity(), id, name,
price, description));

		return sb.toString();
	}

}
```

❖ LinkedList class:

```
public class LinkedList {

	public static final String ANSI_RESET = "\u001B[0m";
	public static final String ANSI_RED = "\u001B[91m";
	public static final String ANSI_GREEN = "\u001B[32m";

	public Node first;
	int orderID = 0;

	public LinkedList() {
		first = null;
	}

	public boolean isEmpty() {
		return (first == null);
	}

	// pqoforders and q
	public void insertAtBack(Object d) {
		Node N = new Node(d);
		if (isEmpty())
			first = N;
		else {
			Node current = first;
			while (current.next != null) {
				current = current.next;
			}
			current.next = N;
		}
	}

	// Queue
	public void insertAtBack(Object d, Object d2) {
		Node N = new Node(d, d2);
		if (isEmpty())
			first = N;
		else {
			Node current = first;
			while (current.next != null) {
				current = current.next;
			}
			current.next = N;
		}
	}

	// pqoforders
	public void insertAtBackprio(Object d, int p) {
		Node n = new Node(d, p);
		if (isEmpty() || first.priority > p) {
			n.next = first;
			first = n;
		} else {
			Node current = first;
			while (current.next != null && current.next.priority <= p) {
				current = current.next;
```

```
            }
            n.next = current.next;
            current.next = n;
      }
}

public void priorityEnqueue(Node newNode) {
      if (isEmpty() || newNode.priority < first.priority) {
            newNode.next = first;
            first = newNode;
      } else {
            Node current = first;
            Node previous = null;

            while (current != null && current.priority <= newNode.priority) {
                  previous = current;
                  current = current.next;
            }

            if (current == null) {
                  // Insert at the end
                  previous.next = newNode;
            } else {
                  // Insert in the middle
                  previous.next = newNode;
                  newNode.next = current;
            }
      }

}

public int top3() {
      return first.priority;
}

public Object deleteFromFront() {
      Object temp = first.data;
      if (!isEmpty()) {
            temp = first.data;
            first = first.next;
      }
      return temp;
}

public void display2() {
      if (isEmpty()) {
            return;
      }
      Node current = first;
      while (current != null) {
            System.out.println(current.data);
            current = current.next;
      }
}

public void display3() {
      if (isEmpty()) {
            return;
      }
```

```
		Node current = first;
		FoodItem f = (FoodItem) current.data;
		System.out.println(f.toString2());
		while (current.next != null) {
			f = (FoodItem) current.data;
			current = current.next;
			if (!f.equals(current.data)) {
				FoodItem f2 = (FoodItem) current.data;
				System.out.println(f2.toString2());
			}
		}
	}

	public void display3(int id) {
		if (isEmpty()) {
			return;
		}
		Node current = first;
		FoodItem f = (FoodItem) current.data;
		System.out.println(f.toString2(id));
		while (current.next != null) {
			f = (FoodItem) current.data;
			current = current.next;
			if (!f.equals(current.data)) {
				FoodItem f2 = (FoodItem) current.data;
				System.out.println(f2.toString2(id));
			}
		}
	}

	public boolean search(Object O) {
		if (first == null) {
			return false;
		}
		Node current = first;
		while (current != null) {
			if (current.data.equals(O)) {
				return true;
			}
			current = current.next;
		}
		return false;
	}

	public void addItem(FoodItem item) { // insert at back
		Node newNode = new Node(item);
		if (first == null) {
			first = newNode;
		} else {
			Node current = first;
			while (current.next != null) {
				current = current.next;
			}
			current.next = (newNode);
		}
	}

	public void displayMenu() {
		Node current = first;
```

```
		int count = 1;
		while (current != null) {
			FoodItem item = current.getItem();
			System.out.println("Menu Item " + (count) + ": ");
			System.out.println("Title: " + item.name);
			System.out.println("Price: $" + item.price);
			if (item.isAvailable)
				System.out.println(ANSI_GREEN + "Availabile in Stock" +
ANSI_RESET);
			else
				System.out.println(ANSI_RED + "Unavailable in Stock" +
ANSI_RESET);
			System.out.println("----------------------------");
			current = current.next;
			count++;
		}
	}

	public boolean removeItem(int id) {
		if (first == null) {
			return false;
		}
		if (first.getItem().id == id) {
			first = first.next;
			return true;
		}
		Node current = first;
		while (current.next != null) {
			if (current.next.getItem().id == id) {
				current.next = (current.next.next);
				return true;
			}
			current = current.next;
		}
		return false;
	}

	public void searchForMenuItemByName(String name) {
		if (first == null) {
			return;
		}
		Node current = first;
		while (current != null) {
			if (current.getItem().name.equalsIgnoreCase(name)) {
				FoodItem item = current.getItem();
				System.out.println(ANSI_GREEN + name + " is present on the menu!"
+ ANSI_RESET);
				System.out.println("----------------------------");
				System.out.println("Title: " + item.name);
				System.out.println("Price: $" + item.price);
				if (item.isAvailable)
					System.out.println(ANSI_GREEN + "Available in Stock" +
ANSI_RESET);
				else
					System.out.println(ANSI_RED + "Unavailable in Stock" +
ANSI_RESET);
				System.out.println("----------------------------");
				return;
			}
```

```
                current = current.next;
            }
            if (name.equals("water"))
                System.out.println(ANSI_RED + "Akalet 7elo w baddak toshrab may kamen!"
+ ANSI_RESET);
            System.out.println(ANSI_RED + name + " is unfortunately not present on the
menu!" + ANSI_RESET);
      }

      public FoodItem searchForMenuItemByID(int id) {
            Node current = first;
            while (current != null) {
                  if (current.getItem().id == id) {
                        FoodItem item = current.getItem();
                        return item;
                  }
                  current = current.next;
            }
            return first.getItem();
      }

}
```

- Node class:

```
public class Node {
      public Object data;
      public Object data2;
      public int priority;
      public Node next;
      public Node previous;
      private FoodItem item;

      public Node() {
            next = null;
            previous = null;
      }

      public Node(Object d) {
            data = d;
            next = null;
            previous = null;
      }

      public Node(FoodItem item) {
            this.item = item;
            next = null;
      }

      public Node(Object d, Object d2) {
            data = d;
            data2 = d2;
            next = null;
            previous = null;
      }
```

```
	public Node(Object d, Object d2, int p) {
		data = d;
		data2 = d2;
		priority = p;
		next = null;
		previous = null;
	}

	public FoodItem getItem() {
		return item;
	}

	public void setItem(FoodItem item) {
		this.item = item;
	}

}
```

- PriorityQueue class:

```
public class PriorityQueue extends LinkedList {

	public PriorityQueue() {
		super();
	}

	public Object top1() {
		return first.data;
	}

	public int top3() {
		return first.priority;
	}

	public void enqueue(Object d, int pr) {
		Node newNode = new Node(d);
		newNode.priority = pr;
		priorityEnqueue(newNode);
	}

	public void enqueue2(Object d, int pr) {
		insertAtBackprio(d, pr);
	}

	public Object dequeue() {
		return deleteFromFront();
	}

	public boolean searched(Object O) {
		return super.search(O);
	}

}
```

❖ Order class:

```
public class Order {
	public int id = 0;
	public static int ordernum = 0;
	boolean shipped;
	boolean completed;
	double price;
	LinkedList foodorders;
	Customer customer;

	public Order() {
		id = ordernum + 1;
		ordernum++;
		shipped = false;
		completed = false;
	}

	public Order(LinkedList foodlist, Customer c1) {
		id = ordernum + 1;
		ordernum++;
		foodorders = foodlist;
		customer = c1;
	}

	public boolean orderShipped(int id, Queue shipped) {
		Order order = shipped.searchorder_by_ID(id);
		return order.shipped;
	}

	public Order completeOrder(int id, Queue shipped) {
		if (shipped.isEmpty()) {
			return null;
		}
		Order order = shipped.searchorder_by_ID(id);
		order.completed = true;
		return order;
	}

}
```

❖ PQofOrders class:

```
public class PQofOrders extends PriorityQueue {
	static int id = 0;
	boolean shipped;

	public static final String ANSI_RESET = "\u001B[0m";
	public static final String ANSI_RED = "\u001B[91m";

	public PQofOrders() {
		super();
		id++;
	}

	public void insert_order(Order customerorder, int district) {
		enqueue(customerorder, district);
	}

	public boolean search_by_ID(int orderID) {
		if (first == null) {
			System.out.println(ANSI_RED + "Couldn't find order!" + ANSI_RESET);
			return false;
		}
		Node current = first;
		while (current != null) {
			Order O = (Order) current.data;
			int id = O.id;
			if (id == orderID) {
				System.out.println("Your order is: ");
				LinkedList L = O.foodorders;
				L.display3(orderID);
				return true;
			}
			current = current.next;
		}
		System.out.println(ANSI_RED + "Couldn't find order!" + ANSI_RESET);
		return false;
	}

	public Order searchorder_by_ID(int orderID) {
		if (first == null) {
			System.out.println(ANSI_RED + "Couldn't find order!" + ANSI_RESET);
			return null;
		}
		Node current = first;
		while (current != null) {
			Order O = (Order) current.data;
			int id = O.id;
			if (id == orderID) {
				return O;
			}
			current = current.next;
		}
		System.out.println(ANSI_RED + "Couldn't find order!" + ANSI_RESET);
		return null;
	}
```

```
public boolean delete_by_ID(int orderID) {
	if (first == null) {
		System.out.println("Couldn't find order!");
		return false;
	}

	if (((Order) first.data).id == orderID) {
		first = first.next;
		return true;
	}

	Node previous = first;
	Node current = first.next;

	while (current != null) {
		Order O = (Order) current.data;
		if (O.id == orderID) {
			previous.next = current.next;
			return true;
		}
		previous = current;
		current = current.next;
	}

	System.out.println(ANSI_RED + "Couldn't find order!" + ANSI_RESET);
	return false;
}

public Queue shippedOrders() {
	Queue shipped = new Queue();

	while (!isEmpty()) {
		Order O = (Order) first.data;
		O.shipped = true;
		shipped.enqueue(O);
		dequeue();
	}

	return shipped;
}

public void orderdisplay(int district) {
	if (isEmpty()) {
		return;
	}
	PQofOrders temp = new PQofOrders();
	while (!isEmpty()) {
		Node current = first;

		Order O = (Order) current.data;
		Customer c = (Customer) O.customer;
		if (c.getDistrict() == district) {
			LinkedList L = (LinkedList) O.foodorders;
			L.display2();
			System.out.println(c + " "); // customer
		}
		temp.enqueue2(current.data, current.priority);
		dequeue();
	}
```

```
            while (!temp.isEmpty()) {
                  enqueue2(temp.top1(), temp.top3());
                  temp.dequeue();
            }
      }

      public double placeorder(int id, LinkedList menu, LinkedList foodorders, int
quantity, double total_price) {
            FoodItem f1 = menu.searchForMenuItemByID(id);
            if (quantity <= f1.quantityInStock) {
                  f1.quantityInStock = f1.quantityInStock - quantity;
                  f1.setChosen_quantity(quantity);
                  f1.setAvailable(f1.quantityInStock);
                  while (quantity-- > 0) {
                        total_price += f1.price;
                        foodorders.insertAtBack(f1);
                  }
            } else {
                  System.out.println(ANSI_RED + "Sorry we are out of stock of " + f1.name
+ ANSI_RESET);
                  return 0;
            }
            return total_price;
      }

      public void pqueuedisplay() {
            if (isEmpty()) {
                  return;
            }
            PQofOrders temp = new PQofOrders();
            while (!isEmpty()) {
                  Node current = first;
                  Order O = (Order) current.data;
                  Customer c = (Customer) O.customer;
                  LinkedList L = (LinkedList) O.foodorders;
                  L.display2();
                  System.out.println(c + " "); // customer
                  System.out.println("");
                  temp.enqueue2(O, current.priority);
                  dequeue();
            }
            while (!temp.isEmpty()) {
                  enqueue2(temp.top1(), temp.top3());
                  temp.dequeue();
            }
      }

}
```

❖ Queue class:

```
public class Queue extends LinkedList {

	public static final String ANSI_RESET = "\u001B[0m";
	public static final String ANSI_RED = "\u001B[91m";

	public Queue() {
		super();
	}

	public void enqueue(Object a) {
		insertAtBack(a);
	}

	public void enqueue(Object d, Object d2) {
		insertAtBack(d, d2);
	}

	public Object dequeue() {
		return deleteFromFront();
	}

	public Object top() {
		return first.data;
	}

	public Order searchorder_by_ID(int orderID) {
		if (first == null) {
			System.out.println(ANSI_RED + "Couldn't find order!" + ANSI_RESET);
			return null;
		}
		Node current = first;
		while (current != null) {
			Order O = (Order) current.data;
			int id = O.id;
			if (id == orderID) {
				return O;
			}
			current = current.next;
		}
		System.out.println(ANSI_RED + "Couldn't find order!" + ANSI_RESET);
		return null;
	}

	public void Qdisplay() {
		if (isEmpty()) {
			System.out.println("The queue is still empty!");
			return;
		}
		Queue temp = new Queue();
		while (!isEmpty()) {
			Node current = first;
			Order O = (Order) current.data;
			Customer c = (Customer) O.customer;
			LinkedList L = (LinkedList) O.foodorders;
			L.display3();
```

```
                System.out.println(c.toString2() + " "); // customer
                System.out.println("");
                temp.enqueue(0);
                ;
                dequeue();
            }
            while (!temp.isEmpty()) {
                enqueue(temp.top());
                temp.dequeue();
            }
        }
}
```

- Van class

```
public class Van {

    public int vanid = 0;
    public static int vannum = 0;
    public int reg_plate;
    public PQofOrders p;

    public Van(int plate) {
        reg_plate = plate;
        vanid = vannum + 1;
        vannum++;
    }

    public void setPriorityQueue(PQofOrders pq) {
        p = pq;
    }

    public int getReg_plate() {
        return reg_plate;
    }

    public void setReg_plate(int reg_plate) {
        this.reg_plate = reg_plate;
    }

    public void display() {
        p.pqueuedisplay();
    }

    public String toString() {
        String format = "| %-10s | %-15s |%n";
        StringBuilder sb = new StringBuilder();

        sb.append(String.format(format, "van id", "reg plate"));

        sb.append(String.format(format, vanid, reg_plate));

        return sb.toString();
    }

}
```

❖ Vans class:

```
public class Vans extends Queue {

	public static final String ANSI_RESET = "\u001B[0m";
	public static final String ANSI_RED = "\u001B[91m";
	public static final String ANSI_GREEN = "\u001B[32m";

	public Vans() {
		super();
	}

	public Queue shipOrder(PQofOrders orders) {
		Van v = (Van) dequeue();
		Queue shipped = orders.shippedOrders();
		System.out.println(ANSI_GREEN + "The van " + v.vanid + " is in his way to
ship the orders!" + ANSI_RESET);
		enqueue(v);
		return shipped;
	}

	public void insertVan(int platenum) {
		Van v = new Van(platenum);
		enqueue(v);

	}

	public int searchVan(int platenum) {
		int num = -1;
		if (isEmpty()) {
			return num;
		} else {
			Queue t = new Queue();
			while (!isEmpty()) {
				Van v = (Van) dequeue();
				if (v.reg_plate == platenum) {
					num = v.vanid;
				}
				t.enqueue(v);
			}
			while (!t.isEmpty()) {
				enqueue(t.dequeue());
			}
			return num;
		}
	}

	public void deleteVan(int platenum) {
		Queue t = new Queue();
		while (!isEmpty()) {
			Van v = (Van) dequeue();
			if (v.reg_plate != platenum) {
				t.enqueue(v);
			}
```

```
            }
            while (!t.isEmpty()) {
                  enqueue(t.dequeue());
            }
      }

      public void display() {
            Queue t = new Queue();
            while (!isEmpty()) {
                  Van v = (Van) dequeue();
                  t.enqueue(v);
                  System.out.println(v.toString());
            }
            while (!t.isEmpty()) {
                  enqueue(t.dequeue());
            }
      }
}
```

- Main class:

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class Main {
      public static final String ANSI_RESET = "\u001B[0m";
      public static final String ANSI_RED = "\u001B[91m";
      public static final String ANSI_GREEN = "\u001B[32m";
      public static final String ANSI_YELLOW = "\u001B[33m";
      public static final String ANSI_BLUE = "\u001B[34m";
      public static final String ANSI_MAGENTA = "\u001B[95m";

      public static void main(String[] args) {
            Scanner in = new Scanner(System.in);

            FoodItem f1 = new FoodItem("Baguette", 2, "French baguette",
12);
            FoodItem f2 = new FoodItem("Cookie", 1, "chocolate chips
cookies", 36);
```

```
        FoodItem f3 = new FoodItem("Croissant", 3, "Classic croissant",
24);
        FoodItem f4 = new FoodItem("Cake", 9, "Moist chocolate Cake",
10);
        FoodItem f5 = new FoodItem("Eclaire", 5, "French choux pastry",
20);
        FoodItem f6 = new FoodItem("cinnamonrolls", 4, "with
butter,sugar and cinnamon", 18);

        LinkedList menu = new LinkedList();
        menu.addItem(f1);
        menu.addItem(f2);
        menu.addItem(f3);
        menu.addItem(f4);
        menu.addItem(f5);
        menu.addItem(f6);

        Customer c1 = new Customer();
        LinkedList foodorders = new LinkedList();
        PQofOrders orders = new PQofOrders();
        Queue completed_orders = new Queue();
        Order customerorder = null;
        Queue shipped = new Queue();
        Vans vans = new Vans();
        vans.insertVan(1234);
        vans.insertVan(9876);
        vans.insertVan(4897);
        vans.insertVan(5236);
        double pending_bill = 0.0;
        double served_bill = 0.0;
        int quantity = 0;
```

```
          System.out.println(ANSI_MAGENTA + "******** WELCOME ********" +
ANSI_RESET);
          System.out.println(ANSI_BLUE + "1. Re-Show list of options\n" +
"2. View the Menu\n" + "3. Make An Order\n"
                    + "4. Search for a menu item by name\n" + "5. Search
for your order \n" + "6. Delete your order \n"
                    + "7. Track order \n" + "8. Staff only section \n" +
"0. Exit the program" + ANSI_RESET);
          int option = 0;
          boolean validInput = false;
          while (!validInput) {
               try {
                    System.out.println(ANSI_MAGENTA + "choose an integer:
" + ANSI_RESET);
                    option = in.nextInt();
                    validInput = true;
               } catch (InputMismatchException e) {
                    System.out.println(ANSI_RED + "Invalid input! Please
enter a valid integer." + ANSI_RESET);
                    in.next();
               }
          }
          while (option != 0) {
               switch (option) {
               case 1:
                    System.out.println(ANSI_MAGENTA + "******** WELCOME
********" + ANSI_RESET);
                    System.out.println(ANSI_BLUE + "1. Re-Show list of
options\n" + "2. View the Menu\n"
                              + "3. Make An Order\n" + "4. Search for a
menu item by name\n" + "5. Search for your order \n"
                              + "6. Delete your order \n" + "7. Track
order \n" + "8. Staff only section \n"
```

```
                    + "0. Exit the program" + ANSI_RESET);
                break;
            case 2:
                menu.displayMenu();
                break;
            case 3:
                c1 = new Customer();
                c1 = c1.customerInfo(in);

                foodorders = new LinkedList();

                System.out.println("How many items you want to order:
");
                int num = in.nextInt();
                double total_price = 0;
                while (num-- > 0) {
                    System.out.println("Please enter the id of the
item you want to order: ");
                    int chosenid = in.nextInt();
                    while (chosenid != 1 && chosenid != 2 && chosenid
!= 3 && chosenid != 4 && chosenid != 5
                            && chosenid != 6) {
                        System.out.println(ANSI_RED + "Re-enter the
id" + ANSI_RESET);
                        chosenid = in.nextInt();
                    }
                    System.out.println("Enter the quantity of item:
");
                    quantity = in.nextInt();

                    total_price = orders.placeorder(chosenid, menu,
foodorders, quantity, total_price);
```

```
                    }
                    customerorder = new Order(foodorders, c1);
                    customerorder.price = total_price;
                    foodorders.display3(customerorder.id);
                    System.out.println(c1.toString());

                    orders.insert_order(customerorder, c1.getDistrict());
                    Van v = (Van) vans.top();
                    v.setPriorityQueue(orders);
                    System.out.println(ANSI_YELLOW + "Total price: " +
total_price + "$ + 3$ delivery fee = "
                              + (total_price + 3) + "$ " + ANSI_RESET);
                    pending_bill += total_price;
                    System.out.println(ANSI_YELLOW + "Thank you for
visiting our bakery! ;)" + ANSI_RESET);
                    break;
               case 4:
                    System.out.println("Enter the menu item you want to
search for: ");
                    String name = in.next();
                    menu.searchForMenuItemByName(name);
                    break;
               case 5:
                    System.out.println("Enter your order's ID: ");
                    int idd = in.nextInt();
                    orders.search_by_ID(idd);
                    break;
               case 6:
                    System.out.println("Enter the ID of the order you want
to delete: ");
                    int id = in.nextInt();
```

```
                    Order o = orders.searchorder_by_ID(id);
                    if (orders.delete_by_ID(id)) {
                         System.out.println(ANSI_GREEN + "Your order has
been successfully deleted." + ANSI_RESET);
                         pending_bill -= o.price;
                         break;
                    } else
                         break;
               case 7:
                    System.out.println("Enter the id of the item you want
to track: ");
                    int orderid = in.nextInt();
                    if (shipped.isEmpty()) {
                         System.out.println(ANSI_RED + "The order is still
in the proccess!" + ANSI_RESET);
                         break;
                    }
                    Order orderr = shipped.searchorder_by_ID(orderid);
                    if (orderr.orderShipped(orderid, shipped))
                         System.out.println(ANSI_GREEN + "The order is in
his way to your door step!" + ANSI_RESET);
                    else
                         System.out.println(ANSI_RED + "The order is still
in the proccess!" + ANSI_RESET);
                    break;


               case 8:
                    System.out.println("Please enter password for staff
operation: ");
                    String pass = in.next();
                    while (!pass.equals("1234") &&
!pass.equals("madandan")) {
```

```
                    System.out.println(ANSI_RED
                              + "Incorrect Password! please reenter
the correct password.\nForget Password? daber rasak"
                              + ANSI_RESET);
                    pass = in.next();
               }
               //
               System.out.println(ANSI_BLUE + "1. Admin \n" + "2.
Driver \n" + "0. Exit program" + ANSI_RESET);
               int opt = 0;
               boolean valid = false;
               while (!valid) {
                    try {
                         System.out.println(ANSI_MAGENTA + "choose an
integer: " + ANSI_RESET);
                         opt = in.nextInt();
                         valid = true;
                    } catch (InputMismatchException e) {
                         System.out.println(ANSI_RED + "Invalid
input! Please enter a valid integer." + ANSI_RESET);
                         in.next();
                    }
               }
               while (opt != 0) {
                    switch (opt) {
                    case 1:
                         System.out.println(ANSI_BLUE + "1. Re-show
Admin list\n" + "2. View Pending Orders \n"
                                   + "3. Ship all orders \n" + "4.
View completed orders \n"
                                   + "5. Display total bill of
Pending Orders \n"
```

```
                                        + "6. Display total Earnings of
Served Orders \n" + "7. Add a new item to the menu \n"
                                        + "8. Delete an item from the menu
\n" + "9. Display orders of a specific region \n"
                                        + "0. Exit Admin list" +
ANSI_RESET);
                                int op = 0;
                                boolean validd = false;
                                while (!validd) {
                                     try {
                                          System.out.println(ANSI_MAGENTA +
"choose an integer: " + ANSI_RESET);
                                          op = in.nextInt();
                                          validd = true;
                                     } catch (InputMismatchException e) {
                                          System.out.println(
                                                    ANSI_RED + "Invalid
input! Please enter a valid integer." + ANSI_RESET);
                                          in.next();
                                     }
                                }
                                while (op != 0) {
                                     switch (op) {
                                     case 1:
                                          System.out.println(ANSI_BLUE + "1.
Re-show Admin list\n" + "2. View Pending Orders \n"
                                                    + "3. Ship all orders
\n" + "4. View completed orders \n"
                                                    + "5. Display total bill
of Pending Orders \n"
                                                    + "6. Display total
Earnings of Served Orders \n"
```

```
                                                  + "7. Add a new item to
the menu \n" + "8. Delete an item from the menu \n"
                                                  + "9. Display orders of
a specific region \n" + "0. Exit Admin list"
                                                  + ANSI_RESET);
                                        break;
                                   case 2:
                                        System.out.println("Pending Orders
Display: ");
                                        orders.pqueuedisplay();
                                        break;
                                   case 3:
                                        shipped = vans.shipOrder(orders);
                                        break;
                                   case 4:
                                        completed_orders.Qdisplay();
                                        break;
                                   case 5:
                                        System.out.println(
                                                  ANSI_GREEN + "Total bill
of pending orders: " + pending_bill + ANSI_RESET);
                                        break;
                                   case 6:
                                        System.out.println(
                                                  ANSI_GREEN + "Total
earnings of served orders: " + served_bill + ANSI_RESET);
                                        break;
                                   case 7:
                                        FoodItem fooditem = new
FoodItem();
                                        fooditem = fooditem.iteminfo(in);
                                        menu.addItem(fooditem);
```

```
                                        System.out.println(ANSI_GREEN +
"The item " + fooditem.name
                                                + " has been added
successfully." + ANSI_RESET);
                                        break;
                                    case 8:
                                        System.out.println("Enter the id
of the menu item you want to delete: ");
                                        int menuid = in.nextInt();
                                        if (menu.removeItem(menuid))
                                            System.out.println(
                                                    ANSI_GREEN + "The
item has been deleted successfully." + ANSI_RESET);
                                        else
                                            System.out.println(ANSI_RED +
"The item was not found." + ANSI_RESET);
                                        break;
                                    case 9:
                                        System.out.println("Enter your
district number: ");
                                        System.out.println(
                                                ANSI_BLUE + "1. Beirut
\n2. Khalde \n3. Debbieh \n4. Zaarouriyeh" + ANSI_RESET);
                                        int dist_search = in.nextInt();
                                        if (dist_search == 1) {
                                            System.out.println("All
orders of Beirut: ");
                                        } else if (dist_search == 2) {
                                            System.out.println("All
orders of Khalde: ");
                                        } else if (dist_search == 3) {
                                            System.out.println("All
orders of Debbieh: ");
```

```
                                        } else if (dist_search == 4) {
                                             System.out.println("All
orders of Zaarouriyeh: ");
                                        }
                                        orders.orderdisplay(dist_search);
                                        break;
                                   default:
                                        System.out.println(ANSI_RED +
"Invalid character" + ANSI_RESET);
                                        break;
                                   }
                                   op = 0;
                                   validd = false;
                                   while (!validd) {
                                        try {

     System.out.println(ANSI_MAGENTA + "choose an integer: " +
ANSI_RESET);
                                             op = in.nextInt();
                                             validd = true;
                                        } catch (InputMismatchException e)
{
                                             System.out.println(
                                                       ANSI_RED + "Invalid
input! Please enter a valid integer." + ANSI_RESET);
                                             in.next();
                                        }
                                   }
                              }

                              break;
                         case 2:
```

```
                                System.out.println(ANSI_BLUE + "1. Re-show
Driver list\n" + "2. View Shipped Orders \n"
                                        + "3. Add a new van \n" + "4.
Delete an existing van \n"
                                        + "5. Search for a specific van
\n" + "6. Display all vans \n"
                                        + "7. Complete an Order \n" + "0.
Exit driver list" + ANSI_RESET);


                                int opp = 0;
                                boolean validdd = false;
                                while (!validdd) {
                                     try {
                                          System.out.println(ANSI_MAGENTA +
"choose an integer: " + ANSI_RESET);
                                          opp = in.nextInt();
                                          validdd = true;
                                     } catch (InputMismatchException e) {
                                          System.out.println(
                                                    ANSI_RED + "Invalid
input! Please enter a valid integer." + ANSI_RESET);
                                          in.next();
                                     }
                                }
                                while (opp != 0) {
                                     switch (opp) {
                                     case 1:
                                          System.out.println(ANSI_BLUE + "1.
Re-show Driver list\n" + "2. View Shipped Orders \n"
                                                  + "3. Add a new van \n"
+ "4. Delete an existing van \n"
                                                  + "5. Search for a
specific van \n" + "6. Display all vans \n"
```

```
                                                    + "7. Complete an Order
\n" + "0. Exit driver list" + ANSI_RESET);
                                        break;
                                    case 2:
                                        System.out.println("Shipped orders
Display:");
                                        shipped.Qdisplay();
                                        break;
                                    case 3:
                                        System.out.println("Enter the van
plate number you want to insert:");
                                        int plate_number = in.nextInt();
                                        vans.insertVan(plate_number);
                                        System.out.println(ANSI_GREEN +
"The van " + plate_number
                                                + " has been added
successfully." + ANSI_RESET);
                                        break;
                                    case 4:
                                        System.out.println("Enter the van
plate number you want to delete:");
                                        int platenumber = in.nextInt();
                                        int vanidd =
vans.searchVan(platenumber);
                                        if (vanidd != -1)
                                            System.out.println(ANSI_GREEN
+ "Van " + vanidd + " was deleted!" + ANSI_RESET);
                                        else
                                            System.out.println(ANSI_RED +
"Van " + vanidd + " is not found!" + ANSI_RESET);
                                        vans.deleteVan(platenumber);

                                        break;
```

```
                                    case 5:
                                        System.out.println("Enter the van
plate number you want to search for:");
                                        int platenum = in.nextInt();
                                        int vanid =
vans.searchVan(platenum);
                                        if (vanid != -1)
                                            System.out.println(ANSI_GREEN
+ "Van " + vanid + " was found!" + ANSI_RESET);
                                        else
                                            System.out.println(ANSI_RED +
"Van of plate number " + platenum + " is not found!"
                                                    + ANSI_RESET);
                                        break;
                                    case 6:
                                        vans.display();
                                        break;
                                    case 7:
                                        System.out.println("Enter the ID
of the order you want to mark as completed");
                                        int orderID = in.nextInt();
                                        Order orderrr =
shipped.searchorder_by_ID(orderID);

                                        if (orderrr != null) {
                                            Order completedOrder =
customerorder.completeOrder(orderID, shipped);
                                            served_bill += orderrr.price;
                                            pending_bill -=
orderrr.price;
                                            if (completedOrder != null) {

    completed_orders.enqueue(completedOrder);
```

```
     System.out.println(ANSI_GREEN + "The order " + orderID
                                                       + " has been
marked as completed." + ANSI_RESET);
                                             } else {
                                                  System.out.println(
                                                       ANSI_RED +
"Unable to mark the order as completed." + ANSI_RESET);
                                                  }
                                             } else {
                                                  System.out.println(ANSI_RED +
"Order with ID " + orderID
                                                            + " not found in
the shipped queue." + ANSI_RESET);
                                             }
                                             break;
                                        default:
                                             System.out.println(ANSI_RED +
"Invalid character" + ANSI_RESET);
                                             break;
                                        }
                                        opp = 0;
                                        validdd = false;
                                        while (!validdd) {
                                             try {

     System.out.println(ANSI_MAGENTA + "choose an integer: " +
ANSI_RESET);
                                                  opp = in.nextInt();
                                                  validdd = true;
                                             } catch (InputMismatchException e)
{
                                                  System.out.println(
```

```
                                                  ANSI_RED + "Invalid
input! Please enter a valid integer." + ANSI_RESET);
                                                  in.next();
                                             }
                                        }
                                   } // while driver
                                   break;
                              default: // default admin driver
                                   System.out.println(ANSI_RED + "Invalid
character" + ANSI_RESET);
                                   break;
                              }
                              opt = 0;
                              valid = false;
                              while (!valid) {
                                   try {
                                        System.out.println(
                                                  ANSI_BLUE + "1. Admin \n" +
"2. Driver \n" + "0. Exit program" + ANSI_RESET);
                                        opt = in.nextInt();
                                        valid = true;
                                   } catch (InputMismatchException e) {
                                        System.out.println(ANSI_RED + "Invalid
input! Please enter a valid integer." + ANSI_RESET);
                                        in.next();
                                   }
                              }
                         }
                         break;
                    default:
```

```
                    System.out.println(ANSI_RED + "Invalid character" +
ANSI_RESET);
                    break;
               } // user switch
               option = 0;
               validInput = false;
               while (!validInput) {
                    try {
                         System.out.println(ANSI_MAGENTA + "choose a
number from the list above: " + ANSI_RESET);
                         option = in.nextInt();
                         validInput = true;
                    } catch (InputMismatchException e) {
                         System.out.println(ANSI_RED + "Invalid input!
Please enter a valid integer." + ANSI_RESET);
                         in.next();
                    }
               }
          } // user while


     }


}
```