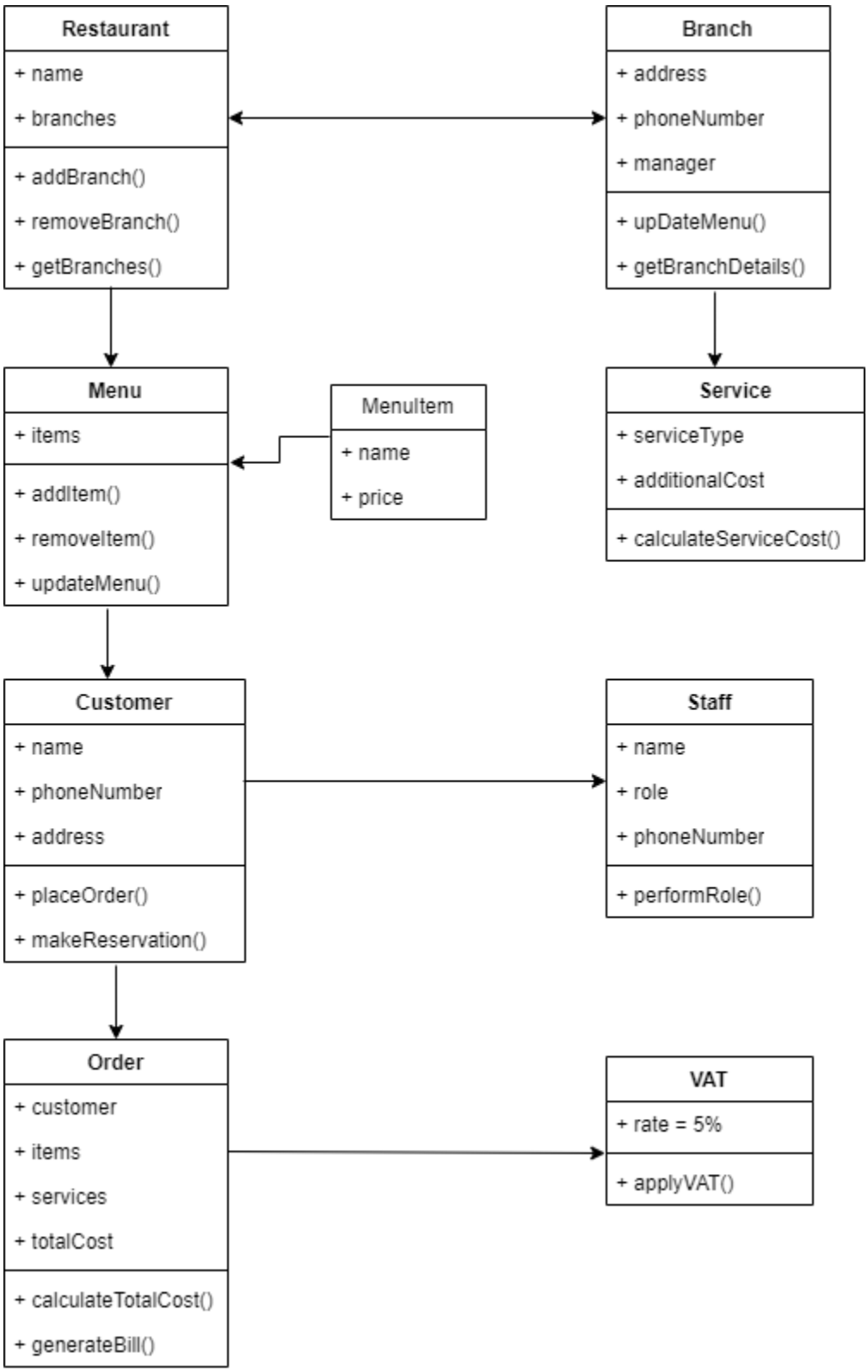
UML DIAGRAM


Restaurant
+ name
+ branches
+ addBranch()
+ removeBranch()
+ getBranches()
Branch
+ address
+ phoneNumber
+ manager
+ upDateMenu()
+ getBranchDetails()
Menu
+ items
+ addItem()
+ removeItem()
+ updateMenu()
MenuItem
+ name
+ price
Service
+ serviceType
+ additionalCost
+ calculateServiceCost()
Customer
+ name
+ phoneNumber
+ address
+ placeOrder()
+ makeReservation()
Staff
+ name
+ role
+ phoneNumber
+ performRole()
Order
+ customer
+ items
+ services
+ totalCost
+ calculateTotalCost()
+ generateBill()
VAT
+ rate = 5%
+ applyVAT()

Description

1. Relationships

The `Restaurant` class maintains a one-to-many relationship with the `Branch` class, signifying that a restaurant can operate multiple branches. Each branch is associated with one restaurant.

The `Branch` class has a one-to-one relationship with the `Menu` class, representing that each branch has its menu. Any modifications to a branch's menu are reflected in the corresponding instance of the `Menu` class.

The `Service` class establishes a one-to-many relationship with the `Order` class, indicating that multiple services can be associated with a single order. Conversely, a service can be linked to various orders.

The `Staff` class is in a many-to-one relationship with the `Branch` class, depicting that multiple staff members can work in a single branch. The `Staff` class has a reference to the `Branch` class, indicating the branch where each staff member operates.

The `Order` class has a one-to-one relationship with the `VAT` class, illustrating that each order is associated with a specific instance of the `VAT` class. This connection implies that each order includes a corresponding VAT.

2. Assumptions

It is assumed that any updates made to a branch's menu are automatically synchronized across all branches to maintain uniformity in the menu offerings.

The additional costs for services (Dine-In, Take Away, Delivery) are considered fixed and are AED 3, AED 4, and AED 5, respectively.

The total cost of an order is assumed to include the costs of the selected food items, additional service costs, and the application of a fixed 5% VAT.

The VAT rate is assumed to be fixed at 5%, and the application of VAT is straightforward, with the tax added to the total cost of the order.

PYTHON CODE

```
class Restaurant:
    """
    Represents a restaurant with multiple branches.

    Attributes:
        name (str): The name of the restaurant.
        branches (list): A list of Branch objects representing
different branches of the restaurant.
    """
```

```
    def __init__(self, name):
        """
        Initializes a new Restaurant object.

        Parameters:
            name (str): The name of the restaurant.
        """
        self.name = name
        self.branches = []

    def add_branch(self, branch):
        """
        Adds a branch to the list of branches.

        Parameters:
            branch (Branch): The Branch object to be added.
        """
        self.branches.append(branch)


class Branch:
    """
    Represents a branch of a restaurant.

    Attributes:
        address (str): The address of the branch.
        phone_number (str): The phone number of the branch.
        manager (str): The manager of the branch.
        menu (Menu): The menu associated with the branch.
        services (list): A list of Service objects offered by the
branch.
        staff (list): A list of Staff objects working at the branch.
        customers (list): A list of Customer objects who have visited
the branch.
    """

    def __init__(self, address, phone_number, manager):
        """
        Initializes a new Branch object.

        Parameters:
            address (str): The address of the branch.
            phone_number (str): The phone number of the branch.
            manager (str): The manager of the branch.
        """
        self.address = address
        self.phone_number = phone_number
        self.manager = manager
        self.menu = Menu()
        self.services = []
        self.staff = []
        self.customers = []
```

```
    def add_service(self, service):
        """
        Adds a service to the list of services offered by the branch.

        Parameters:
            service (Service): The Service object to be added.
        """
        self.services.append(service)

    def add_staff(self, staff):
        """
        Adds a staff member to the list of staff working at the
branch.

        Parameters:
            staff (Staff): The Staff object to be added.
        """
        self.staff.append(staff)

    def add_customer(self, customer):
        """
        Adds a customer to the list of customers who have visited the
branch.

        Parameters:
            customer (Customer): The Customer object to be added.
        """
        self.customers.append(customer)


class Menu:
    """
    Represents a menu containing various items.

    Attributes:
        items (list): A list of MenuItem objects representing items on
the menu.
    """

    def __init__(self):
        """
        Initializes a new Menu object.
        """
        self.items = []

    def add_item(self, item):
        """
        Adds an item to the list of items on the menu.

        Parameters:
            item (MenuItem): The MenuItem object to be added.
```

```
        """
        self.items.append(item)


class MenuItem:
    """
    Represents an item on a menu.

    Attributes:
        name (str): The name of the menu item.
        cost (float): The cost of the menu item.
    """

    def __init__(self, name, cost):
        """
        Initializes a new MenuItem object.

        Parameters:
            name (str): The name of the menu item.
            cost (float): The cost of the menu item.
        """
        self.name = name
        self.cost = cost


class Service:
    """
    Represents an additional service offered by a restaurant.

    Attributes:
        service_type (str): The type of service.
        additional_cost (float): The additional cost associated with
the service.
    """

    def __init__(self, service_type, additional_cost):
        """
        Initializes a new Service object.

        Parameters:
            service_type (str): The type of service.
            additional_cost (float): The additional cost associated
with the service.
        """
        self.service_type = service_type
        self.additional_cost = additional_cost


class Staff:
    """
    Represents a staff member working at a restaurant.
```

```
    Attributes:
        name (str): The name of the staff member.
        role (str): The role or position of the staff member.
        phone_number (str): The phone number of the staff member.
    """

    def __init__(self, name, role, phone_number):
        """
        Initializes a new Staff object.

        Parameters:
            name (str): The name of the staff member.
            role (str): The role or position of the staff member.
            phone_number (str): The phone number of the staff member.
        """
        self.name = name
        self.role = role
        self.phone_number = phone_number


class Customer:
    """
    Represents a customer who visits a restaurant.

    Attributes:
        name (str): The name of the customer.
        phone_number (str): The phone number of the customer.
        address (str): The address of the customer.
        orders (list): A list of Order objects placed by the customer.
    """

    def __init__(self, name, phone_number, address):
        """
        Initializes a new Customer object.

        Parameters:
            name (str): The name of the customer.
            phone_number (str): The phone number of the customer.
            address (str): The address of the customer.
        """
        self.name = name
        self.phone_number = phone_number
        self.address = address
        self.orders = []

    def place_order(self, order):
        """
        Places an order for the customer.

        Parameters:
            order (Order): The Order object to be placed.
        """
```

```
        self.orders.append(order)


class Order:
    """
    Represents an order placed by a customer.

    Attributes:
        customer (Customer): The customer placing the order.
        items (list): A list of MenuItem objects included in the
order.
        services (list): A list of Service objects added to the order.
        total_cost (float): The total cost of the order.
    """

    def __init__(self, customer):
        """
        Initializes a new Order object.

        Parameters:
            customer (Customer): The customer placing the order.
        """
        self.customer = customer
        self.items = []
        self.services = []
        self.total_cost = 0

    def calculate_total_cost(self):
        """
        Calculates the total cost of the order, including items and
services.
        """
        for item in self.items:
            self.total_cost += item.cost
        for service in self.services:
            self.total_cost += service.additional_cost

    def generate_bill(self):
        """
        Generates a bill for the order, including total cost and VAT.

        Returns:
            str: The generated bill.
        """
        self.calculate_total_cost()
        vat = VAT()
        vat.apply_vat(self)
        return f"Total Bill: AED {self.total_cost}"


class VAT:
    """
```

```
    Represents Value Added Tax (VAT) applied to an order.

    Attributes:
        rate (float): The VAT rate.
    """

    def __init__(self):
        """
        Initializes a new VAT object with a default rate of 0.05.
        """
        self.rate = 0.05

    def apply_vat(self, order):
        """
        Applies VAT to the total cost of an order.

        Parameters:
            order (Order): The Order object to which VAT is applied.
        """
        order.total_cost += order.total_cost * self.rate

# Test Cases

# a. Addition of branches to the restaurant
restaurant = Restaurant("Happy Tummy")
branch1 = Branch("Address1", "123-456-7890", Staff("Manager1",
"Manager", "987-654-3210"))
branch2 = Branch("Address2", "987-654-3210", Staff("Manager2",
"Manager", "123-456-7890"))
restaurant.add_branch(branch1)
restaurant.add_branch(branch2)

# b. Addition of services, staff, and customers to a branch
service_dine_in = Service("Dine-In", 3)
service_take_away = Service("Take Away", 4)
service_delivery = Service("Delivery", 5)

staff_chef = Staff("Chef1", "Chef", "111-222-3333")
staff_waiter = Staff("Waiter1", "Waiter", "444-555-6666")

customer1 = Customer("Customer1", "777-888-9999", "Address3")
customer2 = Customer("Customer2", "111-222-3333", "Address4")

branch1.add_service(service_dine_in)
branch1.add_service(service_take_away)
branch1.add_service(service_delivery)

branch1.add_staff(staff_chef)
branch1.add_staff(staff_waiter)

branch1.add_customer(customer1)
branch1.add_customer(customer2)
```

```
# c. Addition of customers booking appointments
order_customer1 = Order(customer1)
order_customer1.items.append(MenuItem("Item1", 10))
order_customer1.services.append(service_dine_in)
order_customer1.services.append(service_delivery)

order_customer2 = Order(customer2)
order_customer2.items.append(MenuItem("Item2", 15))
order_customer2.services.append(service_take_away)

# d. Addition of a menu item
branch1.menu.add_item(MenuItem("Item1", 10))
branch1.menu.add_item(MenuItem("Item2", 15))

# e. Display of payment receipts for services
print(order_customer1.generate_bill())
print(order_customer2.generate_bill())
```