```python
#import ENUM
from enum import Enum
# define an enum for the names of different services
class ServiceName(Enum):
    CLEANING = 1
    IMPLANTS = 2
    CROWNS = 3
    FILLINGS = 4
    MORE = 5

# define an enum for the job titles
class job_title(Enum):
    Manager = 1
    Receptionist = 2
    Hygienist = 3
    Dentist = 4

#Aggregation relationship between the dental company and its branches
class DentalCompany:
    # Constructor method to initialize attributes
    def __init__(self, companyName, owner, email, website):
        self.companyName = companyName
        self.owner = owner
        self.email = email
        self.website = website
        self.branches = []  # Empty list to store branches

    # Method to add a new branch to the company's list
    def add_branch(self, branch):
        self.branches.append(branch)

class Branch:
    # Constructor to initialize attributes
    def __init__(self, address, phoneNumber, manager):
        self.address = address
        self.phoneNumber = phoneNumber
        self.manager = manager
        # initialize an empty list for the branch's services
        self.services = []  # list of Service objects

    # method to add a new service to the branch's list
    def add_service(self, service):
        self.services.append(service)

    # method to add a new appointment to the branch's list
    def add_appointment(self, appointment):
        self.appointments.append(appointment)

class Service:
    # Constructor to initialize attributes
    def __init__(self, name, cost):
        self.name = name
        self.cost = cost

# define a class for a patient
class Patient:
    # Initialize common attributes
    def __init__(self, first_name, last_name, age, address, phone_number):
        # Initialize common attributes for all people
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
        self.address = address
        self.phone_number = phone_number




# inheritance
class Person:
    # Constructor method to initialize attributes
    def __init__(self, first_name, last_name, age, address, phone_number):
        # Initialize common attributes for all people
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
        self.address = address
        self.phone_number = phone_number

# inherit from Person class
class Staff(Person):
    # Constructor method to initialize attributes
    def __init__(self, first_name, last_name, age, address, phone_number,
job_title, staff_id):
        # Call the constructor of the parent class (Person) to initialize
common attributes
        # using super function
        super().__init__(first_name, last_name, age, address, phone_number)
        # Initialize attributes specific to staff members
        self.job_title = job_title
        self.staff_id = staff_id

# define a class for a dental appointment
class Appointment:
    # Constructor to initialize attributes
    def __init__(self, patient_name, date, time,patient, staff):
        self.patient_name = patient_name
        self.date = date
        self.time = time
        # The patient object associated with the appointment
        self.patient = patient
        # The staff member associated with the appointment
        self.staff = staff

class PaymentReceipt:
    # Initialize instance variables for the given arguments
    def __init__(self, patient, service):
        self.patient = patient
        self.service = service

    def get_total_cost(self):
        total_cost = 0
        # Loop through the list of services
        for sr in self.service:
            # Calculate the total cost by adding the cost of each service
            total_cost += sr.service.cost
        return total_cost

    def get_vat(self):
        # Calculate the VAT (Value Added Tax) by multiplying the total cost
by 0.05
        return self.get_total_cost() * 0.05

    def get_total_cost_with_vat(self):
        # Calculate the total cost including VAT
        return self.get_total_cost() + self.get_vat()
```