Assignment due Saturday, April 15, 2023 by 11:00pm

ICS 220 – Programming Fundamentals

Assignment 2

Scenario:

Consider the following problem statement:

A dental company, “Bright Smiles”, has multiple branches (i.e., dental clinics) and would like you to create a software application to help manage the business processes and dental services. Each dental branch has an address, phone number, and a manager. A dental branch offers dental services to patients. Examples of services include cleaning, implants, crowns, fillings, and more. Each of the services has a cost. The clinic keeps track of its patients and staff. The staff includes managers, receptionists, hygienists, and dentists. The patient needs to book an appointment before coming to the clinic. Upon checkout, the clinic charges the patient depending on the services she/he has received. Also, a 5% value-added tax (VAT) is added to the final bill.

Requirements

1. Design a UML class diagram representing the concepts and relationships in the scenario. Ensure using the different types of association and inheritance relationships where necessary. You may make assumptions about attributes (with access specifications) and concepts not explicitly mentioned in the problem statement. A clear description of the relationships and assumptions must be included.
2. Write Python code to implement your UML diagram. Ensure that you define test cases to showcase the program features. Examples of test cases include

a. the addition of branches to the dental company.

b. the addition of dental services, staff, and patients to a branch.

c. the addition of patients booking appointments

d. the display of payment receipts for patient services (one or more) upon checking out. The final bill should be presented to the patient on completion of service.

3. Ensure that your UML diagram and the Python code are well-documented and structured.

Notes:
7 classes

- Person
- Address
- Services
- Patient
- Employee
- Appointment
- Branches

Person Class

This class has properties to store information about a person, including their name, CNIC, address (an object of the Address class), gender, and age. Which will be then used as inherited class for further use in other classes.

Address Class

This class has properties to store the address of a person or branch, including the city, postal code, state, and street.

Services Class

This class has properties to store information about a service provided by a branch, including a service ID, service name, and service cost.

Patient Class

Patient class inherits from Person class and adds additional properties to represent a patient, including a patient ID, phone number, and email.

Employee Class

Employee class also inherits from Person and adds additional properties to represent an employee, including an employee ID, date of join, date of leave, salary, and designation.

Appointment Class

This class has properties to store information about a patient's appointment, including an appointment ID, date, time, patient (an object of the Patient class), dentist (an object of the Employee class), branch (an object of the Branch class), and service (an object of the Services class).

Branch Class

This class has properties to store information about a dental branch, including a branch ID, address (an object of the Address class), phone number, manager (an object of the Employee class), receptionist (object of the Employee class), hygienist (object of the Employee class), dentist (an object of the Employee class), services (list of objects of the Services class), patients (a list of objects of the Patient class), and appointments (a list of objects of the Appointment class). This class also has methods to add new services, appointments, patients, and staff (employees) to the branch, and to calculate the final bill for an appointment.