Code for classes in python:

```python
from datetime import datetime

class Address:
  def __init__(self, city, postal_code, state, street):
    self.city = city
    self.postal_code = postal_code
    self.state = state
    self.street = street

class Person:
  def __init__(self, name, cnic, address, gender, age):
    self.name = name
    self.cnic = cnic
    self.address = address
    self.gender = gender
    self.age = age

class Patient(Person):
  def __init__(self, patientId, phoneNumber, email, name, cnic, address, gender, age):
    super().__init__(name, cnic, address, gender, age)
    self.patientId = patientId
    self.phoneNumber = phoneNumber
    self.email = email

class Services:
  def __init__(self, serviceId, serviceName, serviceCost):
    self.serviceId = serviceId
    self.serviceName = serviceName
    self.serviceCost = serviceCost

class Employee(Person):
  def __init__(self, employeeId, dateOfJoin, dateOfLeave, salary, designation, name, cnic, address, gender,
age):
    super().__init__(name, cnic, address, gender, age)
    self.employeeId = employeeId
    self.dateOfJoin = dateOfJoin
    self.dateOfLeave = dateOfLeave
```

```
        self.salary = salary
        self.designation = designation

class Appointment:
    def __init__(self, appointmentId, date, time, patient, dentist, branch, service):
        self.appointmentId = appointmentId
        self.date = date
        self.time = time
        self.patient = patient
        self.dentist = dentist
        self.branch = branch
        self.service = service

class Branch:
    def __init__(self, branchId, address, phoneNumber, manager, receptionist, hygienist, dentist, services,
patients, appointments):
        self.branchId = branchId
        self.address = address
        self.phoneNumber = phoneNumber
        self.manager = manager
        self.receptionist = receptionist
        self.hygienist = hygienist
        self.dentist = dentist
        self.services = services
        self.patients = patients
        self.appointments = appointments

    def addService(self, service):
        self.services.append(service)

    def addAppointment(self, appointment):
        self.appointments.append(appointment)

    def addPatient(self, patient):
        self.patients.append(patient)

    def addStaff(self, staff):
        if staff.designation == "Manager":
            self.manager = staff
    elif staff.designation == "Receptionist":
      self.receptionist = staff
    elif staff.designation == "Hygienist":
      self.hygienist = staff
    elif staff.designation == "Dentist":
      self.dentist = staff

  def checkout(self, appointment):
    total_cost = appointment.service.serviceCost
    vat = 0.05 * total_cost
    final_bill = total_cost + vat
    return f"Patient {appointment.patient.name} is charged {final_bill}$ for
{appointment.service.serviceName} service with a VAT of {vat}$."
```

Testing

For the testing of this implementation of classes in above python code I have devised some test cases in which I tried adding some patients, staff (i.e., dentist, hygienist, receptionist), branch then also simulated the whole cycle from adding the patients and appointments to checking out and generating the report on console. Following is the code for that.

Code for testing

```
# Create a new address instance for the branch
branch_address = Address(city="New York", postal_code="10001", state="NY", street="123 Main St")

# Create some services
cleaning_service = Services(serviceId=1, serviceName="Cleaning", serviceCost=100)
whitening_service = Services(serviceId=2, serviceName="Whitening", serviceCost=150)

# Create some staff
manager = Employee(employeeId=1, dateOfJoin=datetime.now(), dateOfLeave=None, salary=5000,
designation="Manager",
          name="John Doe", cnic="1234567890123", address=branch_address, gender="Male", age=35)
receptionist = Employee(employeeId=2, dateOfJoin=datetime.now(), dateOfLeave=None, salary=2500,
designation="Receptionist",
```

```python
                    name="Jane Smith", cnic="1234567890124", address=branch_address, gender="Female", age=25)
hygienist = Employee(employeeId=3, dateOfJoin=datetime.now(), dateOfLeave=None, salary=3500, designation="Hygienist",
                  name="Bob Johnson", cnic="1234567890125", address=branch_address, gender="Male", age=45)
dentist = Employee(employeeId=4, dateOfJoin=datetime.now(), dateOfLeave=None, salary=4500, designation="Dentist",
                name="Sarah Williams", cnic="1234567890126", address=branch_address, gender="Female", age=30)

# Create some patients
patient1 = Patient(patientId=1, phoneNumber="1234567890", email="patient1@example.com", name="Tom Smith",
                 cnic="1234567890127", address=branch_address, gender="Male", age=40)
patient2 = Patient(patientId=2, phoneNumber="2345678901", email="patient2@example.com", name="Lisa Johnson",
                 cnic="1234567890128", address=branch_address, gender="Female", age=35)

# Create a new branch and add staff, patients, and services
branch = Branch(branchId=1, address=branch_address, phoneNumber="555-555-5555", manager=manager,
             receptionist=receptionist, hygienist=hygienist, dentist=dentist, services=[cleaning_service],
             patients=[patient1,patient2], appointments=[])

# Add another service to the branch
branch.addService(whitening_service)

# Add a new patient to the branch
branch.addPatient(patient2)

# Schedule an appointment for the second patient
appointment = Appointment(appointmentId="A001", date=datetime(2023, 4, 15), time=datetime(2023, 4, 15, 10, 30),
                     patient=patient2, dentist=dentist, branch=branch, service=whitening_service)

# Add the appointment to the branch
branch.addAppointment(appointment)

# Checkout the appointment
result = branch.checkout(appointment)
```

```
# Print the result
print(result)
```