GRASP

1. getAppointmentDate:

* Location: Appointment
* Principle/Pattern: Information Expert
  + Responsible for holding information for other classes; holds the time of the appointment for the notes and the billing information for the patient.

1. getDescription / getName:

* Location: Diagnosis
* Principle/Pattern: Information Expert
  + Responsible for holding information about diagnoses, these methods relay the information the class knows about the ailment of the patient, its name and description.

1. createPrescription / orderTest:

* Location: Doctor
* Principle/Pattern: Creator
  + Responsible for using the input of the doctor in order to create test and prescription objects

1. updateHealthSheet

* Location: Doctor
* Principle/Pattern: Controller
  + Responsible for using the information passed to the Doctor class to edit the patient’s health sheet, which is outside of the doctor class.

1. sendBillToRecipients:

* Location: Health Care Provider
* Principle/Pattern: Controller
  + This operation is responsible for gathering information from the appointment and providing it to the necessary parties.

1. editGraphProperties:

* Location: Health Care Provider
* Principle/Pattern: Information Expert
  + This operation is responsible for changing the graph to represent the way it is to be displayed, which it knows.

1. getBillInfo:

* Location: Patient
* Principle/Pattern: Information Expert
  + Responsible for knowing the information regarding the bill the patient must pay, and potentially making that information available to the patient.

1. getPrescription:

* Location: Prescription
* Principle/Pattern: Information Expert
  + Responsible for knowing the information regarding the medication, number of refills, patient ID, and doctor ID along with the dosage. This information can be accessed by the doctor and patient.

1. storeDiagnosis / storePrescriptions:

* Location: Patient Diagnosis
* Principle/Pattern: Controller
  + Responsible for assigning an existing Diagnosis or Prescription to a Patient based on the input of the Doctor.

1. createAppointment / sendTestOrder:

* Location: Staff
* Principle/Pattern: Creator
  + Responsible for creating Appointments and Tests

1. getTestName / getInsuranceCode:

* Location: Test
* Principle/Pattern: Information Expert
  + Responsible for storing data necessary to be able to perform a test on a Patient in a lab.

1. sendTestOrder / confirmTest:

* Location: Test Order
* Principle/Pattern: Controller
  + Responsible for assigning an existing Test to a Patient based on input from a Staff member.

1. getTestResult:

* Location: Test Result
* Principle/Pattern: Controller
  + Responsible for taking input from the Test which was performed and relaying the information to the Patient.

1. viewSchedule:

* Location: User
* Principle/Pattern: Low Coupling, High Cohesion, Polymorphism, Information Expert
  + Useful for polymorphism, allows different types of users to view their own personalized schedule, being either Doctor, Patient, or Staff

1. SQLConnection():

* Location: SQLConnection
* Principle/Pattern: Low Coupling, Controller
  + Receives information from a database and initializes data in other

classes