UNIVERSIDAD DE LAS FUERZAS ARMADAS

ESPE

Object-oriented programming

NRC: 9642

U3 Asigment N° 26

Topic: ReviewSolidPrinciples

ING. Jorge Edison Lascano.

1. CALERO ARGUELLO ADONNY MATEO 10/10

Single Responsibility Principle: In general, this code seems to comply with the SRP principle. Each method within the PatientController class performs a specific task related to patient management, such as adding, displaying, updating, deleting patients, and getting appointments. However, some methods are longer and could do a bit more damage to keep more unique liability clear.

```
public static void update(int patientId, Patient newPatient) {

    String connectionString = "mongodb+srv://RBenavides:RBenavides@cluster0.js2ve9m.mongodb.net/";

            ServerApi serverApi = ServerApi.builder()
            .version(ServerApiVersion.V1)
            .build();

    MongoClientSettings settings = MongoClientSettings.builder()
            .applyConnectionString(new ConnectionString(connectionString))
            .serverApi(serverApi)
            .build();

    try (MongoClient mongoClient = MongoClients.create(settings)) {
        try {
            MongoDatabase database = mongoClient.getDatabase("OdontoApp");
            Document filter = new Document("clinicalHistory.id", patientId);

            Gson json = new Gson();
            String patientData = json.toJson(newPatient);
            Document document = Document.parse(patientData);
            Document newDocument = new Document("$set", document);
```

```
public static void delete(int patientId){

    String connectionString = "mongodb+srv://RBenavides:RBenavides@cluster0.js2ve9m.mongodb.net/";

    ServerApi serverApi = ServerApi.builder()
    .version(ServerApiVersion.V1)
    .build();

    MongoClientSettings settings = MongoClientSettings.builder()
            .applyConnectionString(new ConnectionString(connectionString))
            .serverApi(serverApi)
            .build();

    try (MongoClient mongoClient = MongoClients.create(settings)) {
        try {
            MongoDatabase database = mongoClient.getDatabase("OdontoApp");
            Document filter = new Document("clinicalHistory.id", patientId);

            database.getCollection("Patients").deleteOne(filter);
        } catch (MongoException e) {
            e.printStackTrace();
        }
    }
```

Open/Closed Principle: The OCP principle is partly fulfilled. The code is closed in the sense that each method implements a specific operation. However, the design could be made more modular and extensible by introducing interfaces to separate database related operations, which would allow adding new operations or changing the data source without directly modifying this class.

```
public static void update(int patientId, Patient newPatient) {

    String connectionString = "mongodb+srv://RBenavides:RBenavides@cluster0.js2ve9m.mongodb.net/";

            ServerApi serverApi = ServerApi.builder()
            .version(ServerApiVersion.V1)
            .build();

    MongoClientSettings settings = MongoClientSettings.builder()
            .applyConnectionString(new ConnectionString(connectionString))
            .serverApi(serverApi)
            .build();

    try (MongoClient mongoClient = MongoClients.create(settings)) {
        try {
            MongoDatabase database = mongoClient.getDatabase("OdontoApp");
            Document filter = new Document("clinicalHistory.id", patientId);

            Gson json = new Gson();
            String patientData = json.toJson(newPatient);
```

Dependency Inversion Principle: The code does not fully follow the DIP principle. The PatientController class has a direct and strong dependency on the MongoDB database.

```
public static void update(int patientId, Patient newPatient) {

    String connectionString = "mongodb+srv://RBenavides:RBenavides@cluster0.js2ve9m.mongodb.net/";

            ServerApi serverApi = ServerApi.builder()
            .version(ServerApiVersion.V1)
            .build();

    MongoClientSettings settings = MongoClientSettings.builder()
            .applyConnectionString(new ConnectionString(connectionString))
            .serverApi(serverApi)
            .build();

    try (MongoClient mongoClient = MongoClients.create(settings)) {
        try {
            MongoDatabase database = mongoClient.getDatabase("OdontoApp");
            Document filter = new Document("clinicalHistory.id", patientId);

            Gson json = new Gson();
            String patientData = json.toJson(newPatient);
```