



# UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

### DEPARTAMENTO DE CIENCIAS DE LA COMPUTA<mark>CIÓN</mark>

## **OBJECT ORIENTED PROGRAMMING**











**Student:** 

E-mail:

#Cell:

**Professor:** 

E-mail:

#Cell:

NRC:

Llumiquinga Moreno Jerson.

jsllumiquinga1@espe.edu.ec

0980460057

Edison Lascano.

jelascano@espe.edu.ec

0961195050

14539

ar\_name; = \$name\_info; lass(\$charid) { t = mysql\_query("select a = mysql\_fetch\_array)

**MAY24 - SEP24** 





### Handling null in birthDate:

**Problem:** If the calculateAge method receives a birthDate that is null, a NullPointerException will be thrown when trying to set the time to birthCalendar.

**Solution:** Add a check at the beginning of the method to handle this case.

#### Close the connection after each insert:

**Problem**: In the insertProfessor method, the connection to MongoDB is closed right after inserting a teacher. If insertProfessor is called multiple times, it will open and close the connection on each call, which could be inefficient.

**Solution:** Keep the connection open while the application is running and close it only when it is no longer needed, such as when ending the application.





#### Close the connection after each insert:

**Problem:** In the insertProfessor method, the connection to MongoDB is closed right after inserting a teacher. If insertProfessor is called multiple times, it will open and close the connection on each call, which could be inefficient.

**Solution:** Keep the connection open while the application is running and close it only when it is no longer needed, such as when ending the application.

```
package ec.edu.espe.academygradesystemfrm.controller;
import java.awt.Color;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
public class GradeCalculator {
    public static double calculateAverage (double firstTerm, double secondTerm, double thirdTerm) {
   public static String determineStatus(double average, JLabel statusLabel) {
       if (average >= 14) {
           statusLabel.setText("Aprobado");
          statusLabel.setForeground(Color.GREEN);
           return "Aprobado";
       } else {
           statusLabel.setText("Desaprobado");
           statusLabel.setForeground(Color.RED);
           return "Desaprobado";
    public static boolean validateGrades(JTextField... gradeFields) {
```





#### **Problem:**

Previously, there was duplicate code when validating data. Both in the lD entry of the teacher and the student.

#### **Solution:**

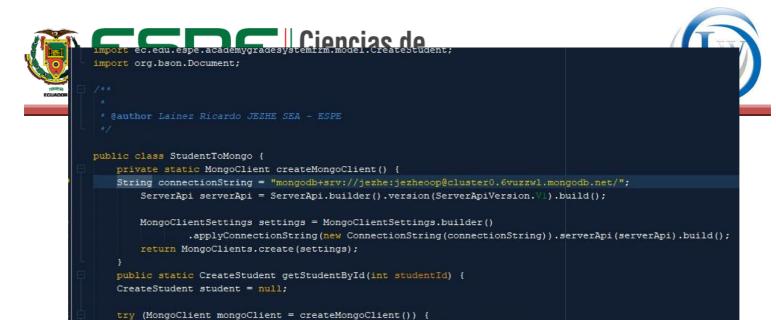
We can create a class that validates the fields, since both inputs require the same data types.

```
Source
        | History | 🛂 🍹 🐙 - | 🔼 🐉 🖶 🔡 | 👉 😓 | 🛂 🖳 🖳 🖳 🕌 📑
        package ec.edu.espe.academygradesystemfrm.controller;
        import java.awt.Color;
  4
5
6
7
8
9
        import javax.swing.JOptionPane;
        import javax.swing.JTextField;
        import javax.swing.JComboBox;
 11
12
13
        public class ValidateData {
              public static boolean validateIdLength(String idText, JTextField textField) {
 14
15
16
                if (idText.length() > 10) {
                    JOptionPane.showMessageDialog(null, "El ID no debe tener más de 10 digitos.", "Error de entrada", JO
                    textField.setForeground(Color.RED);
 17
18
19
                    textField.requestFocus();
                    return false;
 20
21
                return true;
 22
23
24
%
26
27
            public static boolean validateIdIsInteger(String idText, JTextField textField) {
                    Integer.parseInt(idText);
                     textField.setForeground(Color.BLACK);
 28
29
                 catch (NumberFormatException e) {
                    JOptionPane.showMessageDialog(null, "El ID debe ser un número entero.", "Error de entrada", JOptionP
                     textField.setForeground(Color.RED);
```

### **Repetitive Creation and Closing of MongoClient:**

**Problem:** The createMongoClient method is called repeatedly, creating a new MongoClient connection each time the database needs to be accessed. This can be inefficient and could lead to performance issues or connection limits.

**Solution:** Consider creating a single MongoClient instance and reusing it as long as the application is active.



MongoDatabase database = mongoClient.getDatabase("AcademyGradeRegister");
MongoCollection<Document> collection = database.getCollection("students");

Document query = new Document("id", studentId);

Document studentDocument = collection.find(query).first();

### Lack of Proper Exception Handling:

**Problem:** Using e.printStackTrace() is useful for debugging but is not suitable for production environments. Additionally, generic exceptions are being caught, which can make it difficult to identify specific problems.

**Solution:** Implement more detailed exception handling and use a logging framework like SLF4J to log errors.



```
public static void uploadSudentData(CreateStudent student) {
    try(MongoClient mongoClient = createMongoClient()){
        MongoDatabase database = mongoClient.getDatabase("AcademyGradeRegister");
        saveStudentToDatabase(student, database);
    }catch(Exception e) {
        e.printStackTrace();
private static void saveStudentToDatabase (CreateStudent student, MongoDatabase database) {
    MongoCollection<Document> collection = database.getCollection("students");
    Document studentDocument = new Document("id", student.getId())
            .append("nombre", student.getName())
            .append("grado", student.getDegree())
.append("edad", student.getAge());
        collection.insertOne(studentDocument);
        System.out.println("student guardado exitosamente!!");
    }catch (MongoException e) {
        e.printStackTrace();
```