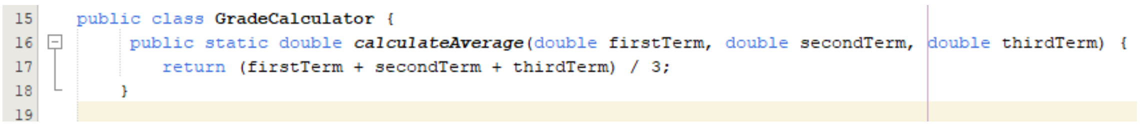
HW 23 SOLID PRINCIPLES

SIGLE RESPONSABILITY

- **Explanation:** The GradeCalculator class adheres to SRP by having each method focused on a single responsibility. For example, the calculateAverage method only calculates the average of three grades, and the determineStatus method determines the status based on the average. This separation of concerns ensures that each method does only one thing, making the class easier to maintain and

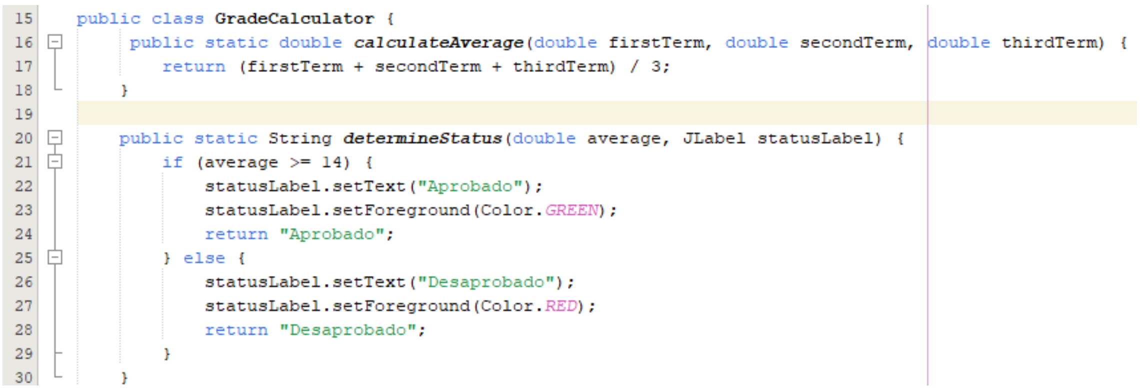


```java
public class GradeCalculator {
     public static double calculateAverage(double firstTerm, double secondTerm, double thirdTerm) {
        return (firstTerm + secondTerm + thirdTerm) / 3;
    }

```

understand.

Open/Closed Principle (OCP):

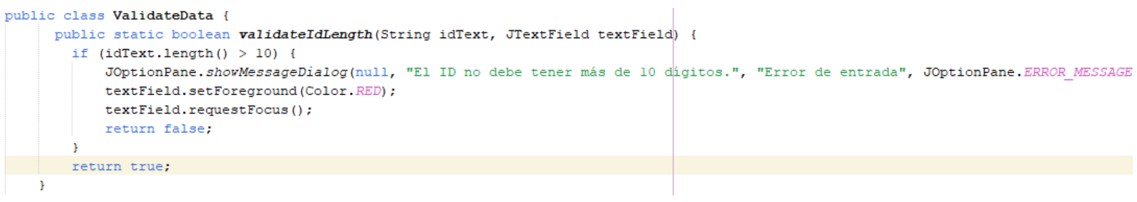
- **Explanation:** The `determineStatus` method is an example of OCP. It is open for extension but closed for modification. If you wanted to add more statuses (e.g., "Excellent" for averages over 18), you could extend the logic without modifying the existing method by perhaps adding another condition or creating a new method that overrides the existing behavior. This approach avoids changing the existing code and reduces the risk of introducing bugs.



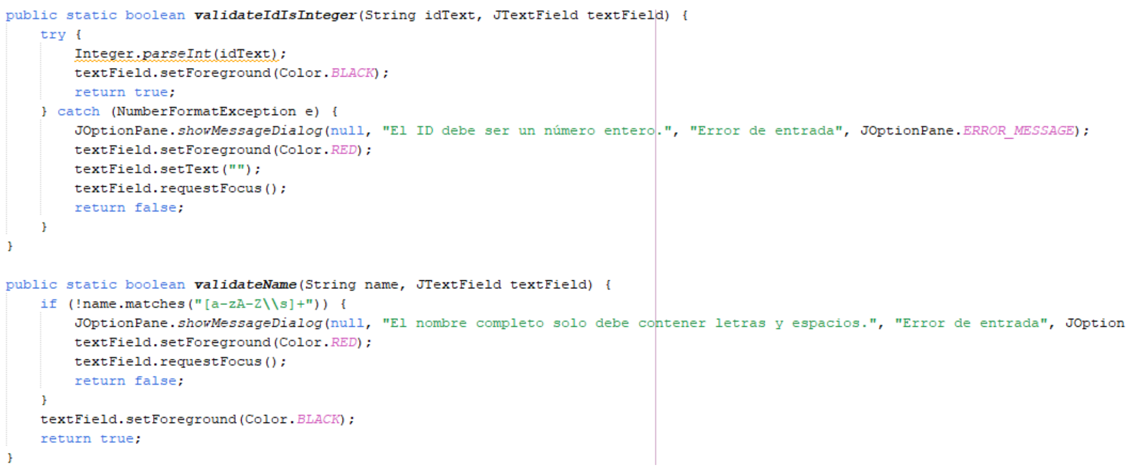
```java
public class GradeCalculator {
     public static double calculateAverage(double firstTerm, double secondTerm, double thirdTerm) {
        return (firstTerm + secondTerm + thirdTerm) / 3;
    }

    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";
        }
    }
```

Open/Closed Principle (OCP):

- **Explicación**: La clase `ValidateData` está abierta para la extensión pero cerrada para la modificación. Si en el futuro se necesita agregar una nueva validación, se puede crear un nuevo método sin necesidad de modificar los ya existentes, lo cual respeta este principio.



```java
public class ValidateData {
    public static boolean validateIdLength(String idText, JTextField textField) {
        if (idText.length() > 10) {
            JOptionPane.showMessageDialog(null, "El ID no debe tener más de 10 dígitos.", "Error de entrada", JOptionPane.ERROR_MESSAGE
            textField.setForeground(Color.RED);
            textField.requestFocus();
            return false;
        }
        return true;
    }
```



```java
public static boolean validateIdIsInteger(String idText, JTextField textField) {
    try {
        Integer.parseInt(idText);
        textField.setForeground(Color.BLACK);
        return true;
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "El ID debe ser un número entero.", "Error de entrada", JOptionPane.ERROR_MESSAGE);
        textField.setForeground(Color.RED);
        textField.setText("");
        textField.requestFocus();
        return false;
    }
}

public static boolean validateName(String name, JTextField textField) {
    if (!name.matches("[a-zA-Z\\s]+")) {
        JOptionPane.showMessageDialog(null, "El nombre completo solo debe contener letras y espacios.", "Error de entrada", JOption
        textField.setForeground(Color.RED);
        textField.requestFocus();
        return false;
    }
    textField.setForeground(Color.BLACK);
    return true;
}
```