Refactorización y Pruebas Unitarias

Hugo Lozano Gallardo
Jorge Ramos González

MEDAC
Instituto Oficial de Formación Profesional

ÍNDICE

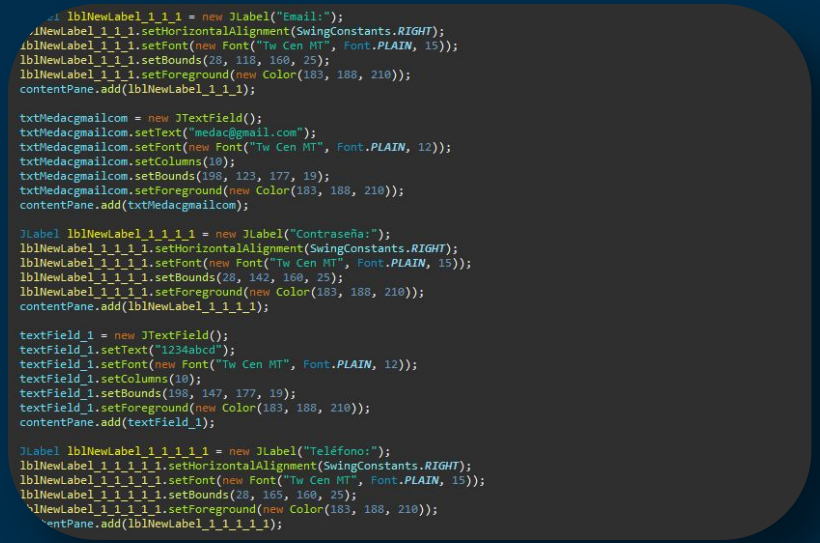
MEDAC

Refactorización el código

1.1 Primeros pasos

En la aplicación GYMFIT nos dimos cuenta de que teníamos un problema, los nombres de las variables no eran lo suficientemente descriptivos como para saber a qué se refería cada valor.

Teníamos JLabels llamados **lblNewLabel_1_1_1_1_1**, esto debido a que cuando lo creamos no cambiábamos el nombre, como este ejemplo muchos más...



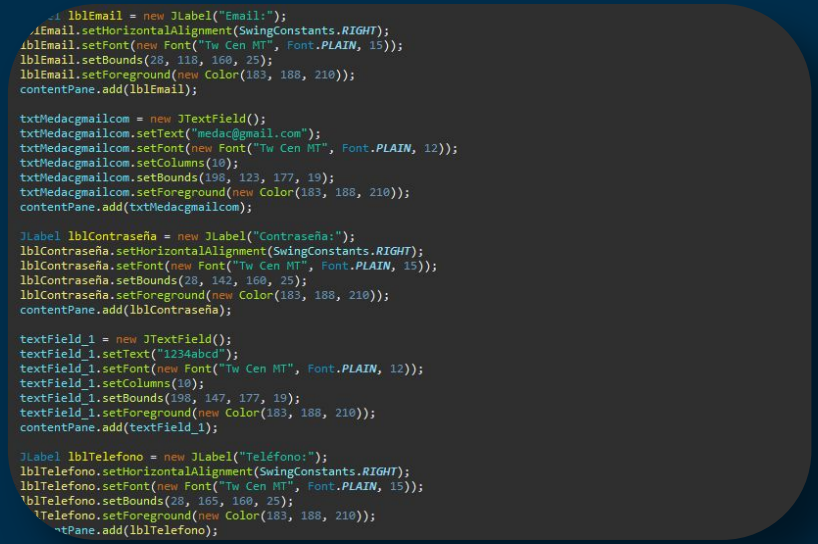
```
lblNewLabel_1_1_1 = new JLabel("Email:");
lblNewLabel_1_1_1.setHorizontalAlignment(SwingConstants.RIGHT);
lblNewLabel_1_1_1.setFont(new Font("Tw Cen MT", Font.PLAIN, 15));
lblNewLabel_1_1_1.setBounds(28, 118, 160, 25);
lblNewLabel_1_1_1.setForeground(new Color(183, 188, 210));
contentPane.add(lblNewLabel_1_1_1);

txtMedacgmailcom = new JTextField();
txtMedacgmailcom.setText("medac@gmail.com");
txtMedacgmailcom.setFont(new Font("Tw Cen MT", Font.PLAIN, 12));
txtMedacgmailcom.setColumns(10);
txtMedacgmailcom.setBounds(198, 123, 177, 19);
txtMedacgmailcom.setForeground(new Color(183, 188, 210));
contentPane.add(txtMedacgmailcom);

JLabel lblNewLabel_1_1_1_1 = new JLabel("Contraseña:");
lblNewLabel_1_1_1_1.setHorizontalAlignment(SwingConstants.RIGHT);
lblNewLabel_1_1_1_1.setFont(new Font("Tw Cen MT", Font.PLAIN, 15));
lblNewLabel_1_1_1_1.setBounds(28, 142, 160, 25);
lblNewLabel_1_1_1_1.setForeground(new Color(183, 188, 210));
contentPane.add(lblNewLabel_1_1_1_1);

textField_1 = new JTextField();
textField_1.setText("1234abcd");
textField_1.setFont(new Font("Tw Cen MT", Font.PLAIN, 12));
textField_1.setColumns(10);
textField_1.setBounds(198, 147, 177, 19);
textField_1.setForeground(new Color(183, 188, 210));
contentPane.add(textField_1);

JLabel lblNewLabel_1_1_1_1_1 = new JLabel("Teléfono:");
lblNewLabel_1_1_1_1_1.setHorizontalAlignment(SwingConstants.RIGHT);
lblNewLabel_1_1_1_1_1.setFont(new Font("Tw Cen MT", Font.PLAIN, 15));
lblNewLabel_1_1_1_1_1.setBounds(28, 165, 160, 25);
lblNewLabel_1_1_1_1_1.setForeground(new Color(183, 188, 210));
contentPane.add(lblNewLabel_1_1_1_1_1);
```

MEDAC

Refactorización el código

1.2 Resolución

Decidimos refactorizar el nombre de estos elementos para que fuesen más fácilmente identificables, de manera que la comprensión del código al ser leído sea muchísimo mayor.



```
    lblEmail = new JLabel("Email:");
lEmail.setHorizontalAlignment(SwingConstants.RIGHT);
blEmail.setFont(new Font("Tw Cen MT", Font.PLAIN, 15));
lblEmail.setBounds(28, 118, 160, 25);
lblEmail.setForeground(new Color(183, 188, 210));
contentPane.add(lblEmail);

txtMedacgmailcom = new JTextField();
txtMedacgmailcom.setText("medac@gmail.com");
txtMedacgmailcom.setFont(new Font("Tw Cen MT", Font.PLAIN, 12));
txtMedacgmailcom.setColumns(10);
txtMedacgmailcom.setBounds(198, 123, 177, 19);
txtMedacgmailcom.setForeground(new Color(183, 188, 210));
contentPane.add(txtMedacgmailcom);

JLabel lblContraseña = new JLabel("Contraseña:");
lblContraseña.setHorizontalAlignment(SwingConstants.RIGHT);
lblContraseña.setFont(new Font("Tw Cen MT", Font.PLAIN, 15));
lblContraseña.setBounds(28, 142, 160, 25);
lblContraseña.setForeground(new Color(183, 188, 210));
contentPane.add(lblContraseña);

textField_1 = new JTextField();
textField_1.setText("1234abcd");
textField_1.setFont(new Font("Tw Cen MT", Font.PLAIN, 12));
textField_1.setColumns(10);
textField_1.setBounds(198, 147, 177, 19);
textField_1.setForeground(new Color(183, 188, 210));
contentPane.add(textField_1);

JLabel lblTelefono = new JLabel("Teléfono:");
lblTelefono.setHorizontalAlignment(SwingConstants.RIGHT);
lblTelefono.setFont(new Font("Tw Cen MT", Font.PLAIN, 15));
lblTelefono.setBounds(28, 165, 160, 25);
lTelefono.setForeground(new Color(183, 188, 210));
tPane.add(lblTelefono);
```

MEDAC

Pruebas Unitarias JUnit5

2.1 Problemas

Para hacer las pruebas unitarias era necesario extraer el código de los métodos a probar de SQL a Java, de otra manera, tendríamos que tener Maven.

Por eso se ha hecho una representación simple de esos métodos para poder hacer las pruebas unitarias sin que falle en el intento, permitiéndonos verificar la funcionalidad de los métodos correctamente.

```java
public class MetodosTest {

    public static String Aforo(int aforo) {
        if (aforo < 50) {
            return (aforo * 100) / 50 + "%";
        } else {
            return "El Gimnasio está lleno";
        }
    }

    public static int setEdad(int edad) {
        if (edad <=0) {
            return 0;
        } else {
            return 1;
        }
    }

    public static int setAltura(int Altura) {
```

MEDAC

Pruebas Unitarias JUnit5

2.2 Tests

Hemos hecho pruebas unitarias de los siguientes métodos:

- Aforo
- setEdad
- setAltura
- setTelefono
- setPeso

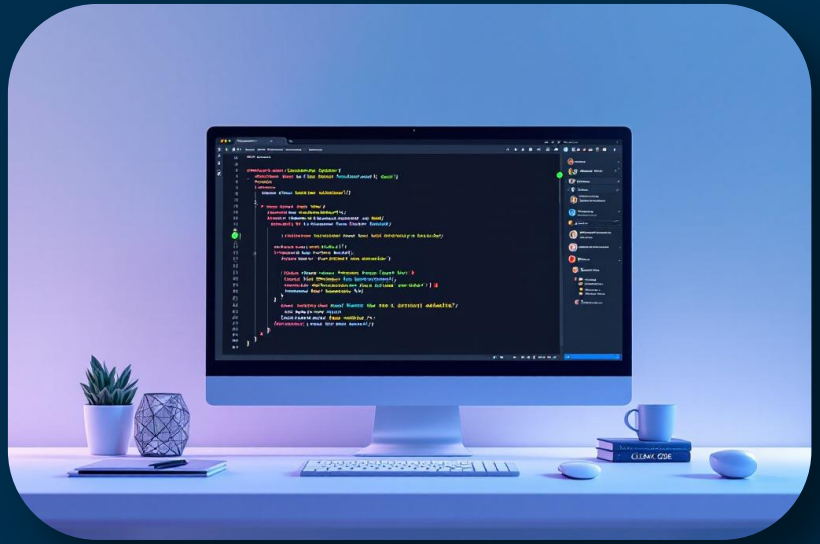
```
@Test
void testAforo() {
    String resultado = MetodosTest.Aforo(25);
    assertEquals("50%", resultado, "El aforo debería ser 50% cuando hay 25 personas");

    resultado = MetodosTest.Aforo(49);
    assertEquals("98%", resultado, "El aforo debería ser 98% cuando hay 49 personas");

    resultado = MetodosTest.Aforo(50);
    assertEquals("El Gimnasio está lleno", resultado, "El aforo debería estar lleno cuando hay 50 personas");
}

@Test
void testEdad() {
    int resultado = MetodosTest.setEdad(25);
    assertEquals(1, resultado, "Debe devolver 1 porque edad es positiva.");

    resultado = MetodosTest.setEdad(14);
    assertEquals(1, resultado, "Debe devolver 1 porque edad es positiva.");

    resultado = MetodosTest.setEdad(-4);
    assertEquals(0, resultado, "Debe devolver 0 porque edad es negativa");
}

@Test
void testAltura() {
    int resultado = MetodosTest.setAltura(-40);
    assertEquals(0, resultado, "Debe devolver 0 porque la altura es posible es negativa.");

    resultado = MetodosTest.setAltura(170);
    assertEquals(1, resultado, "Debe devolver 1 porque la altura es posible.");

    resultado = MetodosTest.setAltura(400);
    assertEquals(0, resultado, "Debe devolver 0 porque la altura es demasiado para ser real");
```

MEDAC

Conclusión

Tanto la refactorización del código como las pruebas unitarias son esenciales en el desarrollo del proyecto, gracias a esto tanto nosotros mismos como otros programadores pueden leer y comprender el código correctamente sin perder tiempo extra en el entendimiento de lo que guardan las variables o lo que hacen los métodos.

Por otra parte, las pruebas unitarias nos permiten tener la seguridad de que nuestros métodos van a funcionar correctamente incluso cuando los ponemos al límite.



MEDAC

MEDAC

Instituto Oficial de Formación Profesional