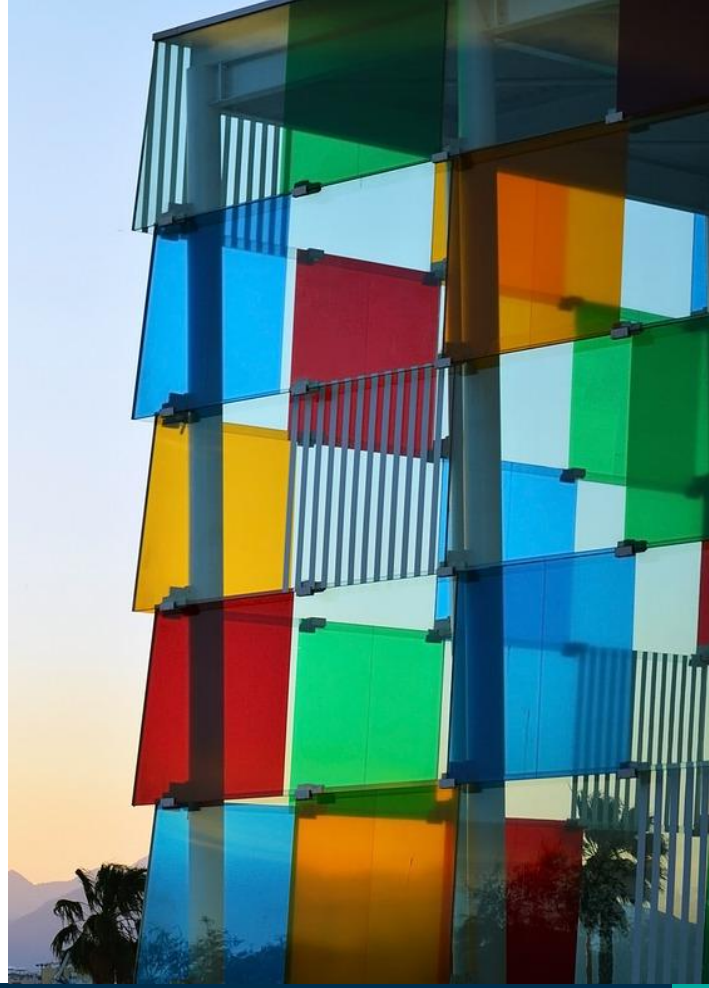
Refactorización y pruebas unitarias

Pablo Segura González
Gonzalo Benitez Diaz

MEDAC
Instituto Oficial de Formación Profesional



MEDAC

Refactorización del código

Cambio del nombre en variables

Antes de refactorizar

```
private  String contraseña;  4 usages
```

Después de refactorizar

Refactorización del código

Cambio del nombre en variables

Antes de refactorizar

```
private JPasswordField textContraseña;  4 usages
```

Después de refactorizar

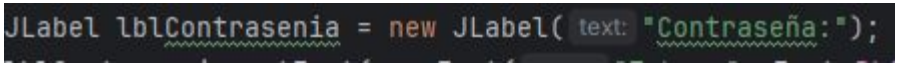
```
private JPasswordField textContrasenia;
```

Refactorización del código

Cambio del nombre en variables

Antes de refactorizar

Después de refactorizar



Refactorización del código

Cambio del nombre en variables

Antes de refactorizar

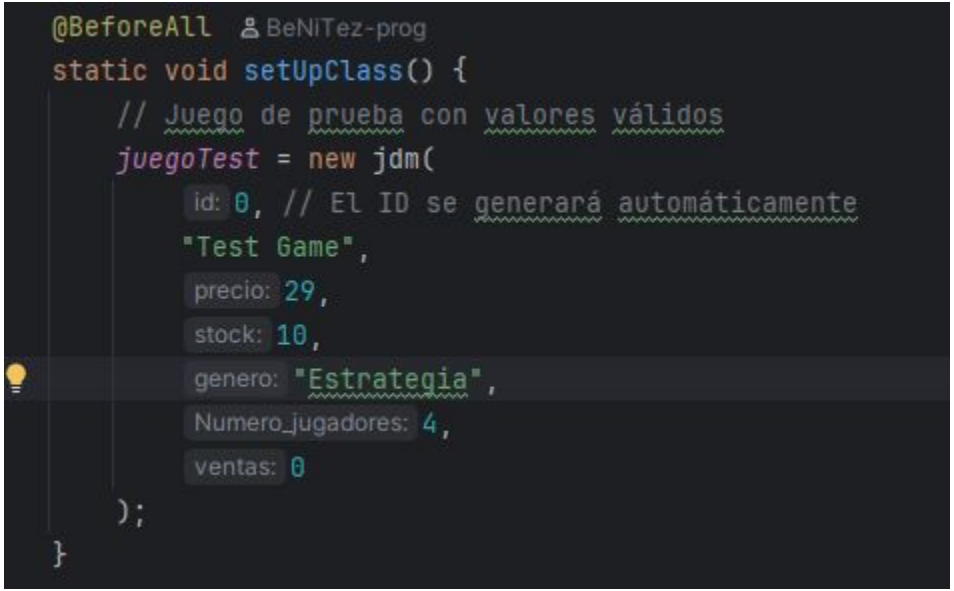
```
JPasswordField contraseña)
```

Después de refactorizar

```
JPasswordField contrasenia)
```

Pruebas Unitarias con JUnit 5

Para las pruebas unitarias primero hemos hecho que antes de comenzar a ejecutarse cree un objeto de tipo jdm para que así tengan unos valores con los que comprobar el funcionamiento de la clase jdmDAO



Pruebas Unitarias con JUnit 5

La primera prueba verifica que se puede acceder a la tabla de juegos de mesa sin problema, verificando que la ejecución no lanza excepciones y que el resultado no es nulo

```
@Test  BeNiTez-prog
void obtenerTodosLosJuegos() {
    assertDoesNotThrow(() -> {
        List<jdm> juegos = jdmDAO.obtenerTodosLosJuegos();
        assertNotNull(juegos, message: "La lista de juegos no debería ser null");
    }, message: "Debería obtener la lista de juegos sin lanzar excepciones");
}
```

Pruebas Unitarias con JUnit 5

Este caso de prueba comprueba que se pueda buscar un juego de mesa por si nombre, verificando que al realizar el select el resultado no es nulo y que tanto el nombre como el género coinciden.

```
@Test  BeNiTez-prog
void obtenerJuegoPorNombre() {
    assertDoesNotThrow(() -> {
        // Primero insertamos el juego de prueba
        jdmDAO.insertarJuego(juegoTest);

        // Luego intentamos recuperarlo
        jdm juego = jdmDAO.obtenerJuegoPorNombre("Test Game");
        assertNotNull(juego, message: "El juego recuperado no deberia ser null");
        assertEquals( expected: "Test Game", juego.getNombre());
        assertEquals( expected: "Estrategia", juego.getGenero());

        // Limpieza
        jdmDAO.eliminarJuego( nombre: "Test Game");
    }, message: "Deberia obtener el juego por nombre sin lanzar excepciones");
}
```

Pruebas Unitarias con JUnit 5

Luego comprobaremos que los juegos se eliminan correctamente de la base de datos eliminandolo y luego verificando por nombre si el resultado es null, en el caso de no ser null significa que el juego no se ha eliminado

```
@Test  BeNiTez-prog
void insertarJuego() {
    assertDoesNotThrow(() -> {
        boolean resultado = jdmDAO.insertarJuego(juegoTest);
        assertTrue(resultado, message: "La inserción debería ser exitosa");

        // Verificar que el juego se insertó correctamente
        jdm juegoInsertado = jdmDAO.obtenerJuegoPorNombre("Test Game");
        assertNotNull(juegoInsertado);
        assertEquals(juegoTest.getNombre(), juegoInsertado.getNombre());
        assertEquals(juegoTest.getPrecio(), juegoInsertado.getPrecio());

        // Limpieza
        jdmDAO.eliminarJuego( nombre: "Test Game");
    }, message: "Debería insertar el juego sin lanzar excepciones");
}
```

Pruebas Unitarias con JUnit 5

Este test comprueba si el metodo BuscarJuegoPorNumJugadores realmente filtra los juegos por el numero de jugadores permitidos.

```
@Test  BeNiTez-prog
void buscarJuegosPorGenero() {
    assertDoesNotThrow(() -> {
        // Insertamos el juego de prueba
        jdmDAO.insertarJuego(juegoTest);

        List<jdm> juegos = jdmDAO.buscarJuegosPorGenero("Estrategia");
        assertNotNull(juegos, message: "La lista de juegos no deberia ser null");
        assertFalse(juegos.isEmpty(), message: "Deberia encontrar al menos un juego");
        assertTrue(juegos.stream().allMatch( jdm j -> j.getGenero().equalsIgnoreCase( anotherString: "Estrategia")),
                message: "Todos los juegos deberian ser del género Estrategia");

        // Limpieza
        jdmDAO.eliminarJuego( nombre: "Test Game");
    }, message: "Deberia buscar juegos por género sin lanzar excepciones");
}
```

MEDAC

Instituto Oficial de Formación Profesional