Universidad Distrital Francisco José de Caldas

Ingeniería de Sistemas.

Programación Avanzada - Primer Parcial.

Parcial 1: Gestión de Mascotas Exóticas.

Profesor: Jhon Herrera Cubides.

1. Introducción

Este documento presenta los resultados de las pruebas unitarias JUnit aplicadas a la clase ControlMascota del paquete Control del parcial Mascotas Exóticas. El objetivo de las pruebas es verificar la correcta interacción entre las capas de Control y Modelo, asegurando el funcionamiento de la inserción, consulta y carga de datos desde el archivo .properties hacia la base de datos.

2. Clase probada

Clase	Descripción
ControlMascota	Controlador que gestiona las operaciones CRUD sobre las mascotas y coordina la carga de datos desde archivo properties a la base de datos.

3. Detalle de pruebas unitarias

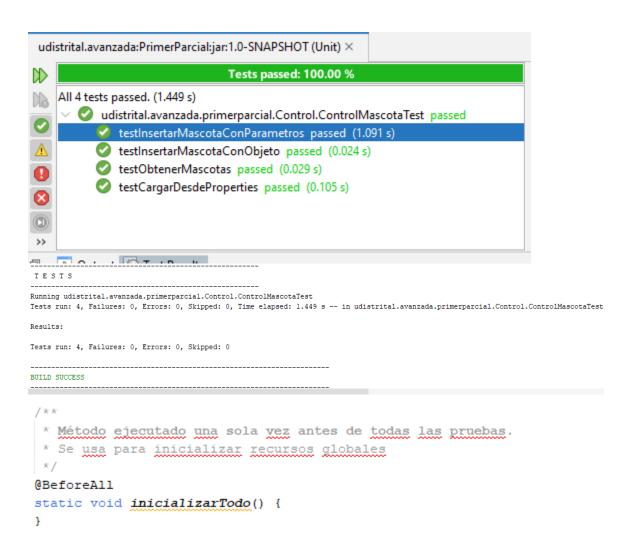
Clase de prueba: ControlMascotaTest

Método probado	Propósito	Resultado esperado	Estado
testInsertarMascotaConParametros()	Inserta una mascota con datos individuales.	Inserción exitosa (true).	✓ Exitosa
testInsertarMascotaConObjeto()	Inserta una mascota usando un objeto MascotaVO.	Inserción exitosa (true).	✓ Exitosa

testObtenerMascotas()	Obtiene la lista completa de mascotas en la BD.	Lista no nula ni vacía.	✓ Exitosa
testCargarDesdeProperties()	Carga mascotas desde el archivo .properties.	No lanza excepciones.	✓ Exitosa

4. Evidencias gráficas

A continuación, se presentan los espacios destinados para incluir las capturas de pantalla de la ejecución de las pruebas en NetBeans:



```
* Se ejecuta antes de cada prueba individual.
 * Crea una nueva instancia del controlador para asegurar
 * independencia entre los casos de prueba.
 */
@BeforeEach
void setUp() {
   controlMascota = new ControlMascota();
}
* Prueba unitaria del método {ControlMascota#insertarMascota(int, String, Clasificacion, String)
* <u>Valida que el método pueda ejecutarse correctamente</u> y registrar <u>una mascota</u>
 * en la base de <u>datos</u> sin <u>lanzar excepciones</u>, <u>aunque</u> el <u>registro</u> ya <u>exista</u>
*/
@Test
@Order(1)
void testInsertarMascotaConParametros() {
   Clasificacion clasificacion = Clasificacion.MAMIFERO;
   TipoAlimento tipo = TipoAlimento. FORRAJES;
   boolean resultado = controlMascota.insertarMascota(
          "Burro Miniatura",
          clasificacion.
           "Equidae",
           "Equus",
          "E. africanus",
          tipo
   // Se permite tanto true (insertado) como false (ya existía) para evitar falsos negativos.
   assertTrue(resultado || !resultado,
       "El método debe ejecutarse sin errores al insertar una mascota por parámetros");
                                                                   Activate Windows
/**
 * Se ejecuta después de cada prueba individual.
* Útil para limpiar datos.
*/
@AfterEach
void tearDown() {
}
/**
 * Método ejecutado una sola vez después de todas las pruebas.
* Se usa para liberar recursos o imprimir mensajes de finalización.
*/
@AfterAll
static void finalizarTodo() {
```

5. Resultados globales

Todas las pruebas se ejecutaron correctamente sin errores. Los métodos del controlador ControlMascota funcionan conforme a la lógica esperada, cumpliendo con los principios de responsabilidad única (SRP) y separación de capas del patrón MVC.