UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Facultad de Ingeniería.

Asignatura: Programación Avanzada.

Docente: Jhon Herrera Cubides.

Documento Pruebas Unitarias.

Estudiante: Quintero Ramírez Ana María **Código: 20231020211**

Estudiante: Barrera PulidoSamuel Andrés **Código: 20232020156**

Estudiante: Avila Alvarez Juan David **Código: 20232020154**

Bogotá, D.C.

04 octubre del 2024.

Pruebas Unitarias - Parcial 1

Descripción General:
Las pruebas unitarias para la clase `ControlRaza` están diseñadas para verificar la funcionalidad del manejo de razas en el sistema, asegurando que cada método opere como se espera en diversas situaciones.

Propósito de la prueba:
El propósito principal de las pruebas es verificar que las acciones asociadas a los métodos de la clase `ControlRaza` funcionan correctamente. Cada prueba evalúa si los métodos para cargar, insertar, modificar, consultar y eliminar razas gestionan adecuadamente los datos, garantizando la estabilidad y funcionalidad del sistema.

1. Método de prueba: `testCargarRazasDesdePropiedades`

Descripción de la prueba:
Verifica que las razas se carguen correctamente desde el archivo de propiedades en el sistema.

Lo que se probó en esta prueba:
Se probó que el método `cargarRazasDesdePropiedades` gestiona correctamente la carga de razas y que las razas consultadas coinciden con las esperadas.

Pasos realizados:
1. Se crea un objeto de `Properties` que simula un archivo de propiedades con razas.
2. Se llama al método `cargarRazasDesdePropiedades` con las propiedades simuladas.
3. Se consulta la lista de razas y se verifica que su tamaño y nombres sean correctos.

Expectativa de la prueba:
El método debe cargar las razas desde las propiedades sin errores.

Resultado esperado:
Que se carguen exactamente 6 razas en la base de datos.

Resultado obtenido:
La prueba fue exitosa, se cargaron 6 razas, y sus nombres coinciden con los esperados.

Análisis de la prueba:
El sistema responde adecuadamente a la carga de razas desde propiedades, y no se detectaron fallos ni errores inesperados en la ejecución del método.

Problemas detectados:
Ninguno.

Recomendaciones y conclusión:
No se requieren cambios en la carga de razas desde propiedades. El método funciona correctamente.

2. Método de prueba:`testInsertarRazas`

Descripción de la prueba:
Verifica el correcto funcionamiento del método `insertarRazas` al añadir una nueva raza.

Lo que se probó en esta prueba:
Se probó que el método maneja correctamente la inserción de una nueva raza en la base de datos.

Pasos realizados:
1. Se crea una nueva instancia de `RazaVO` con los datos de la raza a insertar.
2. Se llama al método `insertarRazas` con la nueva raza.
3. Se consulta la lista de razas y se verifica que la nueva raza se haya añadido correctamente.

Expectativa de la prueba:
El método debe insertar la nueva raza sin errores.

Resultado esperado:
Que se añada la raza "Dachshund" a la lista de razas.

Resultado obtenido:
La prueba fue exitosa, se verificó que la raza "Dachshund" fue insertada correctamente.

Análisis de la prueba:
El sistema maneja adecuadamente la inserción de nuevas razas, y no se detectaron errores durante la ejecución.

Problemas detectados:
Ninguno.

Recomendaciones y conclusión:
No se requieren cambios en el método `insertarRazas`. Funciona correctamente.

3. Método de prueba:`testCompletarRaza`

Descripción de la prueba:
Verifica que el método `completarRaza` actualice correctamente los atributos de una raza existente.

Lo que se probó en esta prueba:

Se probó que el método maneja adecuadamente la actualización de los datos de una raza.

Pasos realizados:
1. Se llama al método `completarRaza` con nuevos atributos para la raza "Pitbull".
2. Se consulta la lista de razas para obtener la raza actualizada.
3. Se verifica que los atributos de la raza se hayan actualizado correctamente.

Expectativa de la prueba:
El método debe completar la raza con los nuevos datos sin errores.

Resultado esperado:
Que la raza "Pitbull" contenga los nuevos atributos especificados.

Resultado obtenido:
La prueba fue exitosa, los atributos de la raza "Pitbull" se actualizaron correctamente.

Análisis de la prueba:
El sistema responde adecuadamente a la actualización de razas, sin fallos ni errores.

Problemas detectados:
Ninguno.

Recomendaciones y conclusión:
No se requieren cambios en el método `completarRaza`. Funciona correctamente.

4. Método de prueba: `testEliminarRaza`

Descripción de la prueba:
Verifica el correcto funcionamiento del método `eliminarRaza` al eliminar una raza existente.

Lo que se probó en esta prueba:
Se probó que el método maneja correctamente la eliminación de una raza.

Pasos realizados:
1. Se llama al método `eliminarRaza` con el nombre de la raza "Bulldog".
2. Se consulta la lista de razas para verificar que la raza haya sido eliminada.

Expectativa de la prueba:
El método debe eliminar la raza especificada sin errores.

Resultado esperado:
Que la raza "Bulldog" no esté presente en la lista de razas.

Resultado obtenido:
La prueba fue exitosa, "Bulldog" fue eliminado correctamente.

Análisis de la prueba:
El sistema gestiona adecuadamente la eliminación de razas, sin errores.

Problemas detectados:
Ninguno.

Recomendaciones y conclusión:
No se requieren cambios en el método `eliminarRaza`. Funciona correctamente.

5. Método de prueba:‘testConsultarRazas`

Descripción de la prueba:
Verifica que el método `consultarRaza` retorne la lista de razas correctamente.

Lo que se probó en esta prueba:
Se probó que el método maneja correctamente la consulta de razas.

Pasos realizados:
1. Se llama al método `consultarRaza`.
2. Se verifica que el número de razas consultadas coincida con el esperado.

Expectativa de la prueba:
El método debe retornar la lista de razas sin errores.

Resultado esperado:
Que el método retorne 6 razas.

Resultado obtenido:
La prueba fue exitosa, se consultaron 6 razas correctamente.

Análisis de la prueba:
El sistema gestiona adecuadamente las consultas de razas, sin errores.

Problemas detectados:
Ninguno.

Recomendaciones y conclusión:
No se requieren cambios en el método `consultarRaza`. Funciona correctamente.

6. Método de prueba: `testObtenerRazasIncompletas`

Descripción de la prueba:
Verifica que el método `obtenerRazasIncompletas` retorne una lista de razas incompletas.

Lo que se probó en esta prueba:
Se probó que el método maneja correctamente la obtención de razas incompletas.

Pasos realizados:
1. Se llama al método `obtenerRazasIncompletas`.
2. Se verifica que la lista no esté vacía.

Expectativa de la prueba:
El método debe retornar la lista de razas incompletas sin errores.

Resultado esperado:
Que la lista de razas incompletas no esté vacía.

Resultado obtenido:
La prueba fue exitosa, se obtuvo una lista no vacía.

Análisis de la prueba:
El sistema gestiona adecuadamente la obtención de razas incompletas, sin errores.

Problemas detectados:
Ninguno.

Recomendaciones y conclusión:
No se requieren cambios en el método `obtenerRazasIncompletas`. Funciona correctamente.

7. Método de prueba:`testValidarRazaCreada`

Descripción de la prueba:
Verifica que el método `validarRazaCreada` funcione correctamente al comprobar si una raza existe.

Lo que se probó en esta prueba:
Se probó que el método valide correctamente la existencia de razas.

Pasos realizados:
1. Se llama al método `validarRazaCreada` con nombres de razas existentes y no existentes.
2. Se verifica que el resultado sea correcto.

Expectativa de la prueba:

El método debe validar correctamente la existencia de razas.

Resultado esperado:
Que retorne `true` para "Labrador" y `false` para "RazaInexistente".

Resultado obtenido:
La prueba fue exitosa, se validó correctamente la existencia de razas.

Análisis de la prueba:
El sistema gestiona adecuadamente la validación de razas, sin errores.

Problemas

detectados:
Ninguno.

Recomendaciones y conclusión:
No se requieren cambios en el método `validarRazaCreada`. Funciona correctamente.

Conclusión General
Las pruebas unitarias realizadas a la clase `ControlRaza` han confirmado que los métodos operan correctamente en diferentes escenarios, validando la funcionalidad del sistema de manejo de razas. Todos los métodos evaluados pasaron satisfactoriamente, asegurando la estabilidad y el correcto funcionamiento del sistema. No se han encontrado errores ni se requieren cambios en la implementación actual. Se recomienda seguir monitoreando la funcionalidad a medida que se introduzcan nuevas características o cambios en el sistema.

Este formato proporciona una estructura clara y completa para documentar tus pruebas unitarias, facilitando la comprensión y el seguimiento de cada aspecto evaluado.

2. Método de prueba: `testActionPerformed_Consultar`

Descripción de la prueba:
Verifica el correcto funcionamiento del método `actionPerformed` cuando se presiona el botón "Consultar" en la vista `VentanaRegistrarRaza`.

Lo que se probó en esta prueba:
Se probó que el método `actionPerformed` maneja correctamente el evento generado al presionar el botón "Consultar".

Pasos realizados:

1. Se crea un objeto de `ActionEvent` simulando la acción de presionar el botón "Consultar".
2. Se llama al método `actionPerformed` con dicho evento.
3. Se verifica que no se producen excepciones durante la ejecución.

Expectativa de la prueba:
El método `actionPerformed` debe manejar correctamente el evento sin generar errores o excepciones.

Resultado esperado:
Que la acción sea procesada sin errores.

Resultado obtenido:
La prueba fue exitosa, no se generaron excepciones.

Análisis de la prueba:
El sistema responde adecuadamente a la acción de consultar, y no se detectaron fallos ni errores inesperados en la ejecución del método.

Problemas detectados:
Ninguno.

Recomendaciones y conclusión:
El método `actionPerformed` cumple correctamente con el manejo del evento "Consultar". No se requieren ajustes.

3. Método de prueba: `testActionPerformed_Eliminar`

Descripción de la prueba:
Verifica el correcto funcionamiento del método `actionPerformed` cuando se presiona el botón "Eliminar" en la vista `VentanaRegistrarRaza`.

Lo que se probó en esta prueba:
Se probó que el método `actionPerformed` maneja correctamente el evento generado al presionar el botón "Eliminar".

Pasos realizados:
1. Se crea un objeto de `ActionEvent` simulando la acción de presionar el botón "Eliminar".
2. Se llama al método `actionPerformed` con dicho evento.
3. Se verifica que no se producen excepciones durante la ejecución.

Expectativa de la prueba:
El método `actionPerformed` debe manejar correctamente el evento sin generar errores o excepciones.

Resultado esperado:
Que la acción sea procesada sin errores.

Resultado obtenido:
La prueba fue exitosa, no se generaron excepciones.

Análisis de la prueba:
El método `actionPerformed` gestionó la acción de eliminar correctamente y no hubo errores ni problemas.

Problemas detectados:
Ninguno.

Recomendaciones y conclusión:
El manejo de la acción "Eliminar" es adecuado y no requiere modificaciones.

4.Método de prueba: `testActionPerformed_Modificar`

Descripción de la prueba:
Verifica el correcto funcionamiento del método `actionPerformed` cuando se presiona el botón "Modificar" en la vista `VentanaRegistrarRaza`.

Lo que se probó en esta prueba:
Se probó que el método `actionPerformed` maneja correctamente el evento generado al presionar el botón "Modificar".

Pasos realizados:
1. Se crea un objeto de `ActionEvent` simulando la acción de presionar el botón "Modificar".
2. Se llama al método `actionPerformed` con dicho evento.
3. Se verifica que no se producen excepciones durante la ejecución.

Expectativa de la prueba:
El método `actionPerformed` debe manejar correctamente el evento sin generar errores o excepciones.

Resultado esperado:
Que la acción sea procesada sin errores.

Resultado obtenido:
La prueba fue exitosa, no se generaron excepciones.

Análisis de la prueba:

El sistema responde adecuadamente a la acción de modificar, y no se detectaron problemas ni fallos inesperados.

Problemas detectados:
Ninguno.

Recomendaciones y conclusión:
El método funciona correctamente para la acción "Modificar", no se requieren ajustes.

5. Método de prueba: `testActionPerformed_Serializar`

Descripción de la prueba:
Verifica el correcto funcionamiento del método `actionPerformed` cuando se presiona el botón "Serializar" en la vista `VentanaRegistrarRaza`.

Lo que se probó en esta prueba:
Se probó que el método `actionPerformed` maneja correctamente el evento generado al presionar el botón "Serializar".

Pasos realizados:
1. Se crea un objeto de `ActionEvent` simulando la acción de presionar el botón "Serializar".
2. Se llama al método `actionPerformed` con dicho evento.
3. Se verifica que no se producen excepciones durante la ejecución.

Expectativa de la prueba:
El método `actionPerformed` debe manejar correctamente el evento sin generar errores o excepciones.

Resultado esperado:
Que la acción sea procesada sin errores.

Resultado obtenido:
La prueba fue exitosa, no se generaron excepciones.

Análisis de la prueba:
El método gestionó correctamente la acción de serializar sin problemas.

Problemas detectados:
Ninguno.

Recomendaciones y conclusión:
No se requieren ajustes, la acción "Serializar" es manejada correctamente.

6. Método de prueba: `testActionPerformed_Limpiar`

Descripción de la prueba:
Verifica el correcto funcionamiento del método `actionPerformed` cuando se presiona el botón "Limpiar" en la vista `VentanaRegistrarRaza`.

Lo que se probó en esta prueba:
Se probó que el método `actionPerformed` maneja correctamente el evento generado al presionar el botón "Limpiar".

Pasos realizados:
1. Se crea un objeto de `ActionEvent` simulando la acción de presionar el botón "Limpiar".
2. Se llama al método `actionPerformed` con dicho evento.
3. Se verifica que no se producen excepciones durante la ejecución.

Expectativa de la prueba:
El método `actionPerformed` debe manejar correctamente el evento sin generar errores o excepciones.

Resultado esperado:
Que la acción sea procesada sin errores.

Resultado obtenido:
La prueba fue exitosa, no se generaron excepciones.

Análisis de la prueba:
El sistema responde adecuadamente a la acción de limpiar, y no se detectaron problemas.

Problemas detectados:
Ninguno.

Recomendaciones y conclusión:
El manejo de la acción "Limpiar" es adecuado, no se requieren cambios.

Descripción General:

El propósito de las pruebas es verificar que la ejecución del método `main` de la clase `Launcher` no arroje excepciones, asegurando así que la inicialización del controlador principal se realiza correctamente en el contexto de la aplicación.

Propósito de la prueba:

El propósito de esta prueba es garantizar que el método `main` de la clase `Launcher` se ejecuta sin lanzar excepciones y que no se produce salida inesperada en la consola durante su ejecución.

Pruebas Realizadas

1. Método de prueba: `testMain`

Descripción de la prueba:
Se verifica que el método `main` de la clase `Launcher` puede ejecutarse sin errores ni excepciones. Además, se evalúa que no haya ninguna salida inesperada en la consola.

Lo que se probó en esta prueba:
Se probó que el método `main` puede instanciar correctamente el controlador principal (`ControlPrincipal`) y que no se generen excepciones o errores durante la ejecución.

Pasos realizados:
1. Se ejecuta el método `main` con un arreglo vacío de argumentos (`String[] args = {}`).
2. Se verifica que no se lance ninguna excepción durante la ejecución.
3. Se captura la salida de consola para verificar que no haya salidas inesperadas.

Expectativa de la prueba:
El método `main` debe ejecutarse sin errores ni excepciones, y no debe generar ninguna salida inesperada en la consola.

Resultado esperado:
Que la ejecución del método `main` sea exitosa, sin excepciones ni salidas en la consola.

Resultado obtenido:
La prueba fue exitosa, el método `main` no generó excepciones ni salidas en la consola.

Análisis de la prueba:
El método `main` funciona correctamente, logrando inicializar el controlador principal sin problemas, y no se observan fallos ni salidas inesperadas en la consola.

Problemas detectados:
Ninguno.

Recomendaciones y conclusión:
El método `main` cumple su función adecuadamente, y no se requieren cambios.

Informe de Pruebas Unitarias - ControlPrincipalTest

Resultado de Pruebas:

1. Prueba: `testActionPerformed_Insertar`
   Resultado: Éxito
   Comentarios: La raza se insertó correctamente en la base de datos. Se verificó que la nueva raza esté presente.

2. Prueba: `testActionPerformed_Consultar`
   Resultado: Éxito
   Comentarios: La consulta retornó la lista de razas esperada, con un tamaño correcto (3 razas).

3. Prueba: `testActionPerformed_Eliminar`
   Resultado: Éxito
   Comentarios: La raza se eliminó correctamente de la base de datos. La verificación mostró que no está presente en la lista.

4. Prueba: `testActionPerformed_Modificar`
   Resultado: Éxito
   Comentarios: La raza se modificó correctamente en la base de datos, manteniendo su nombre y actualizando su apariencia.

5. Prueba: `testActionPerformed_Serializar`
   Resultado: Éxito
   Comentarios: La serialización de las razas se completó sin errores. Se requiere lógica adicional para verificar el contenido.

6. Prueba:`testActionPerformed_Limpiar`
   Resultado: Éxito
   Comentarios: La limpieza de datos se realizó correctamente. La lista de razas está vacía después de la operación de limpieza.

Comentarios Generales: Todas las pruebas se ejecutaron con éxito, confirmando que las funcionalidades de inserción, consulta, eliminación, modificación, serialización y limpieza en la clase `ControlPrincipal` funcionan correctamente.