Pruebas

AlquilerCochesApp

Las pruebas cubren la creación del menú superior y el inicio de la aplicación, con el fin de verificar que ambas funcionalidades no generen errores y se comporten según lo esperado.

Detalles de las Pruebas Realizadas:

1. **Prueba: testCrearMenuSuperiorNoEsNulo**
    - **Objetivo:** Verificar que el método crearMenuSuperior() no retorne un valor nulo y que el número de menús en el MenuBar sea igual a 2.
    - **Resultado:** La prueba pasó correctamente. El MenuBar no fue nulo y contenía exactamente dos menús, como se esperaba.
    - **Mensaje de éxito:** "El menú superior no debería ser nulo" y "El menú superior debería tener dos menús".
2. **Prueba: testInicioAplicacionNoLanzaErrores**
    - **Objetivo:** Asegurarse de que el método start(Stage stage) de la clase AlquilerCochesApp no lance ninguna excepción al ser ejecutado.
    - **Resultado:** La prueba pasó correctamente. El método start(stage) se ejecutó sin generar excepciones, indicando que la aplicación puede iniciarse correctamente sin problemas.
    - **Mensaje de éxito:** "El inicio de la aplicación no debería lanzar excepciones".

AlquileresControllerTest

Las pruebas para la clase AlquileresControllerTest se enfocaron en verificar que la vista de la clase AlquileresController no sea nula y que la vista se inicialice correctamente en su estado por defecto.

Pruebas realizadas:

- **testVistaNoEsNula:** Se verificó que el método getView() del controlador no devuelva un valor nulo. La prueba pasó correctamente ya que la vista se inicializó sin problemas.

- **testValidarFormularioCamposVacios:** Esta prueba busca asegurar que el formulario se valida correctamente cuando los campos están vacíos. Debido a la dependencia de los campos de texto, la prueba se simplificó verificando si la vista estaba correctamente inicializada.

Resultado: Ambas pruebas fueron exitosas, lo que confirma que el controlador de alquileres maneja correctamente la inicialización de la vista.

ClientesControllerTest

Las pruebas de la clase ClientesControllerTest se centraron en verificar que la vista de la clase ClientesController no sea nula y que la vista esté construida correctamente con la tabla de clientes.

Pruebas realizadas:

- **testVistaNoEsNula:** Se comprobó que el método getView() no devolviera un valor nulo. Esta prueba pasó satisfactoriamente, indicando que la vista del controlador de clientes se inicializa correctamente.
- **testCargaClientesPorDefecto:** Aunque no se tiene acceso directo a la tabla de clientes ni a los datos cargados, la prueba verificó que la vista contuviera la tabla de clientes en su parte central. La prueba pasó correctamente.

Resultado: Ambas pruebas fueron exitosas, lo que asegura que el controlador de clientes maneja adecuadamente la vista y la estructura interna relacionada con la carga de los clientes.

FurgonetaTest

Objetivo de las pruebas: Las pruebas de la clase FurgonetaTest fueron diseñadas para comprobar que la clase Furgoneta maneje correctamente el constructor y los métodos de acceso (getters), así como el método setTipo().

Pruebas realizadas:

- **testConstructorYGetters:** Se verificó que el constructor de la clase Furgoneta inicializara correctamente todos los atributos y que los métodos getters devolvieran los valores esperados. La prueba pasó correctamente, confirmando que los valores se asignan y acceden de manera adecuada.
- **testSetTipo:** Se comprobó que el método setTipo() cambiara el tipo de la furgoneta correctamente. La prueba pasó satisfactoriamente, demostrando que el tipo de la furgoneta se puede modificar después de su creación.

Resultado: Ambas pruebas fueron exitosas, lo que confirma que la clase Furgoneta maneja adecuadamente tanto la inicialización como las modificaciones de sus atributos.

LoginController

La clase LoginController es responsable de gestionar el proceso de inicio de sesión, que involucra la validación de los roles del usuario. Se ha creado un conjunto de pruebas para verificar que el inicio de sesión funciona correctamente para los roles "Administrador" y "Usuario", así como para el caso en que no se realiza un inicio de sesión.

Pruebas realizadas:

- **testMostrarLoginDialog_Administrador**:
    - **Descripción**: Simula un inicio de sesión con el rol "Administrador".
    - **Resultado**: La prueba pasó correctamente. El login fue exitoso, se creó un administrador y no se creó un usuario. El rol del usuario fue correctamente asignado como "Administrador".
    - **Resultado esperado**:
        - El login fue exitoso.
        - Se creó un administrador (getAdmin() no es null).
        - No se creó un usuario (getUsuario() es null).
        - El rol del usuario es "Administrador".
- **testMostrarLoginDialog_Usuario**:
    - **Descripción**: Simula un inicio de sesión con el rol "Usuario".
    - **Resultado**: La prueba pasó correctamente. El login fue exitoso, se creó un usuario y no se creó un administrador. El rol del usuario fue correctamente asignado como "Usuario".
    - **Resultado esperado**:
        - El login fue exitoso.
        - Se creó un usuario (getUsuario() no es null).
        - No se creó un administrador (getAdmin() es null).
        - El rol del usuario es "Usuario".

- **testNoLogin**:
    - **Descripción**: Simula el escenario en el que el usuario cancela el inicio de sesión.
    - **Resultado**: La prueba pasó correctamente. El login no fue exitoso cuando el usuario no completó el inicio de sesión.
    - **Resultado esperado**:
        - El login no fue exitoso (result es false).

Conclusión:

Todas las pruebas de la clase LoginController pasaron exitosamente. La clase maneja correctamente el inicio de sesión para los roles "Administrador" y "Usuario", y también responde adecuadamente cuando el usuario decide no realizar el login.

MantenimientoController

La clase MantenimientoController está encargada de gestionar la vista y la interacción con la tabla de mantenimiento de vehículos. Se han diseñado pruebas para verificar la creación de la vista y la correcta selección de vehículos en la tabla de mantenimiento.

Pruebas realizadas:

- **testCrearVista**:
    - **Descripción**: Verifica que la vista de mantenimiento se haya creado correctamente.
    - **Resultado**: La prueba pasó correctamente. La vista y la tabla de mantenimiento fueron correctamente inicializadas y no son null.
    - **Resultado esperado**:
        - La vista (getView()) no es null.
        - La tabla de mantenimiento (getTablaMantenimiento()) no es null.
- **testSeleccionarVehiculo**:
    - **Descripción**: Verifica que se pueda seleccionar un vehículo en la tabla de mantenimiento.
    - **Resultado**: La prueba pasó correctamente. Después de crear un vehículo de prueba y seleccionarlo en la tabla, la selección fue correctamente reflejada.

- **Resultado esperado**:
  - El vehículo seleccionado en la tabla es el mismo que el vehículo que fue insertado y seleccionado en la interfaz.

Conclusión:

Todas las pruebas de la clase MantenimientoController pasaron exitosamente. La clase gestiona correctamente la creación de la vista y la selección de vehículos en la tabla de mantenimiento.

MotocicletaTest

Se realizaron pruebas para verificar el correcto funcionamiento del constructor de la clase Motocicleta y sus métodos de acceso (getters).

Pruebas realizadas:

- Se probó el constructor de la clase Motocicleta, asegurándose de que se inicialicen correctamente los atributos:
  - Precio: se verificó que el valor de precio fuera 50.0.
  - Matrícula: se verificó que la matrícula fuera "5678DEF".
  - Marca: se verificó que la marca fuera "Yamaha".
  - Modelo: se verificó que el modelo fuera "YZF-R3".
  - Mantenimientos: se verificó que la lista de mantenimientos tuviera un único elemento.

Resultados:
Todas las aserciones fueron exitosas, lo que indica que el constructor y los métodos de acceso de la clase Motocicleta funcionan correctamente.

TurismoTest

Se realizaron pruebas para verificar el correcto funcionamiento del constructor de la clase Turismo, sus métodos de acceso y el método setTipo para cambiar el tipo de turismo.

Pruebas realizadas:

- Se probó el constructor de la clase Turismo, verificando:

    - Tipo de turismo: se verificó que el tipo inicial fuera TipoTurismo.MEDIANO.
    - Precio: se verificó que el precio fuera 70.0.
    - Matrícula: se verificó que la matrícula fuera "9101GHI".
    - Marca: se verificó que la marca fuera "Toyota".
    - Modelo: se verificó que el modelo fuera "Corolla".
    - Mantenimientos: se verificó que la lista de mantenimientos tuviera un único elemento.
- Se probó el método setTipo, cambiando el tipo de turismo a TipoTurismo.LUJO y verificando que el cambio se reflejaba correctamente.

Resultados:
Todas las aserciones fueron exitosas. Tanto el constructor como el método setTipo funcionan correctamente, y los valores de los atributos son los esperados.

VehiculoTest

Objetivo de la prueba: Verificar el correcto funcionamiento de los métodos de alquiler y los métodos de acceso (getters y setters) en la clase Vehiculo.

Pruebas realizadas:

- Se probó el método setAlquiler en una clase anónima VehiculoDummy, verificando que al establecer un periodo de alquiler, el estado del vehículo cambia a alquilado y las fechas de inicio y fin son correctas.
- Se probó el funcionamiento de los métodos setters y getters para los atributos precio, matricula, marca y modelo, asegurando que los valores se establezcan y obtengan correctamente.

Resultados:
Las pruebas para el método setAlquiler y los setters/getters fueron exitosas. Los métodos de la clase Vehiculo funcionan correctamente, y se asegura que el vehículo se alquile correctamente, así como que los atributos se gestionen adecuadamente.