Prácticas Entorno

ÍNDICE

FASE 1: COMUNICACIÓN

1.Resumen básico

Mi proyecto consiste en el programa de gestión de coches de un concesionario, permitiendo agregar nuevos coches al catálogo, ver los coches que hay de base más los coches añadidos posteriormente, un apartado VIP donde hay una sección aparte de coches y la opción de poder salir del programa de manera sencilla.

2.Herramientas utilizadas

-He realizado el programa en lenguaje Java
-Para los test he utilizado el “JUnit”

3.IDE

He realizado el trabajo en el entorno de trabajo Visual Studio Code y luego he subido todo el proyecto a un repositorio de GitHub, a través del entorno de desarrollo he realizado los commits y push para sincronizarlos.

4.Clases empleadas

-Catálogo
-Concesionario

FASE 2: DISEÑO

Diagrama de clase:

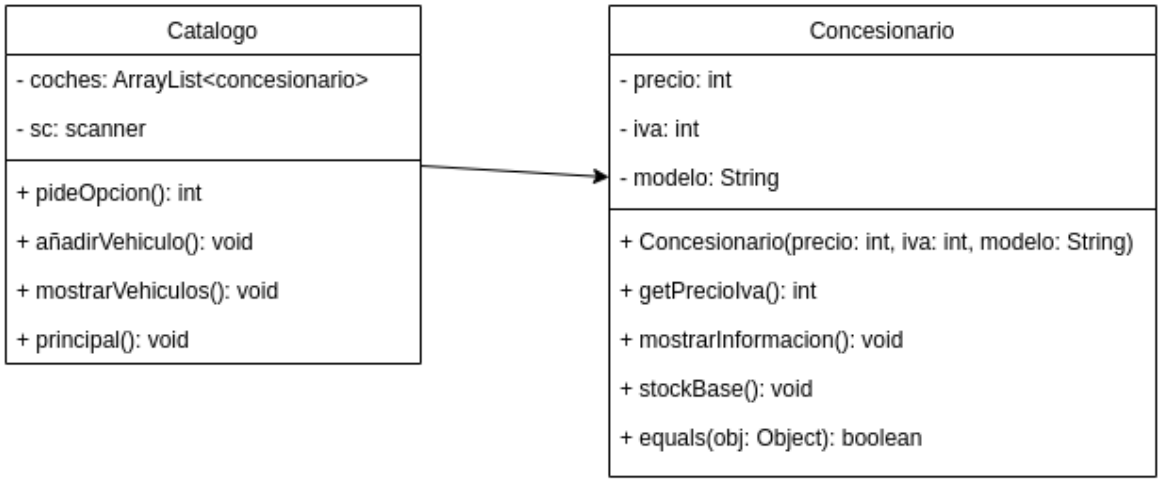
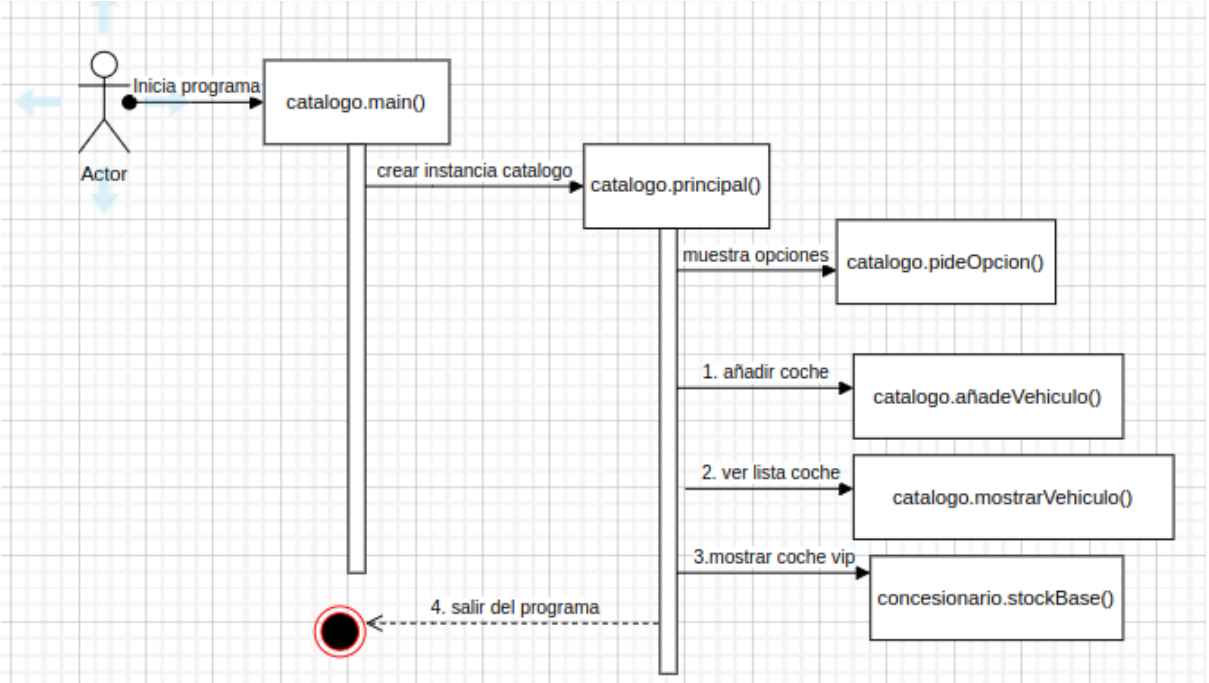


Diagrama de secuencia:



FASE 3: CODIFICACIÓN

En este apartado voy a explicar un poco lo que hace cada uno de los métodos de mi código.

En el archivo Catalogo.java tenemos los siguientes metodos:

1. main
    - Es el punto de entrada del programa.
    - Crea una instancia de la clase Catalogo y llama al método principal() para iniciar el flujo del programa.
2. pideOpcion()
    - Muestra un menú de opciones al usuario.
    - Solicita al usuario que seleccione una opción y devuelve el número ingresado.
    - Utiliza la instancia de Scanner para leer la entrada del usuario.
3. añadirVehiculo()
    - Solicita al usuario que introduzca los datos de un coche: precio, IVA y modelo.
    - Crea una instancia de la clase Concesionario con los datos proporcionados.
    - Añade el coche creado a la lista coches.
4. mostrarVehiculos()

    - Itera sobre la lista coches y llama al método mostrarInformacion() de cada objeto Concesionario.
    - Muestra la información de todos los coches almacenados en la lista.
5. principal()
    - Controla el flujo principal del programa.
    - Muestra el menú de opciones en un bucle hasta que el usuario elige salir.
    - Según la opción seleccionada, llama a los métodos correspondientes:
        - añadirVehiculo() para añadir un coche.
        - mostrarVehiculos() para mostrar la lista de coches.
        - Concesionario.stockBase() para mostrar coches VIP.
        - Finaliza el programa si el usuario selecciona la opción de salir.

Y en el archivo Concesionario.java tenemos los siguientes:

1. constructor de concesionario
    - Inicializa los atributos precio, iva y modelo con los valores proporcionados.
2. getPrecioIva()
    - Calcula el precio del coche con IVA (21%).
    - Devuelve el precio con IVA.
3. mostrarInformacion()
    - Muestra la información del coche, incluyendo:
        - Modelo.
        - Precio.
        - IVA.
        - Precio con IVA (calculado llamando a getPrecioIva()).

4. stockBase()
    - Método estático que muestra información de coches VIP predefinidos (Ferrari, Lamborghini, Bugatti).
    - No requiere una instancia de Concesionario para ser llamado.

FASE 4: IMPLANTACIÓN

Primera prueba testPideOpcion:
Propósito: Verificar que el método pide Opción() de la clase Catalogo devuelve un valor válido dentro del rango esperado.
Funcionamiento: Simula la entrada del usuario utilizando un InputStream. En este caso, la entrada simulada es "1\n", que representa que el usuario selecciona la opción 1.
Redirige la entrada estándar del sistema (System.in) al flujo de entrada simulado. Crea una instancia de la clase Catalogo y llama al método pideOpcion().
Verifica que el valor devuelto por pideOpcion() está dentro del rango válido entre 1 y 3.
Resultado esperado: La prueba pasa si el valor devuelto está dentro del rango especificado.

Segunda prueba equals (incluida en el archivo Concesionario.java)
Propósito: Sobrescribir el método equals para comparar dos objetos de la clase Concesionario.
Funcionamiento: Comprueba si el objeto actual es igual al objeto pasado como parámetro. Si el objeto es null o no pertenece a la clase Concesionario, devuelve false. Compara los modelos de ambos objetos para determinar si son iguales.
Resultado esperado: Devuelve true si los modelos son iguales, de lo contrario, devuelve false.

```
public class CatalogoTest {
    @Test
    public void testPideOpcion() {
        String input= "1\n";
        InputStream in = new ByteArrayInputStream(input.getBytes());
        System.setIn(in);
        Catalogo catalogo = new Catalogo();
        int opcion = catalogo.pideOpcion();
        assertTrue(opcion >= 1 && opcion <= 3);
    }
 @Test
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Concesionario that = (Concesionario) obj;
        return modelo.equals(that.modelo);
    }
}
```