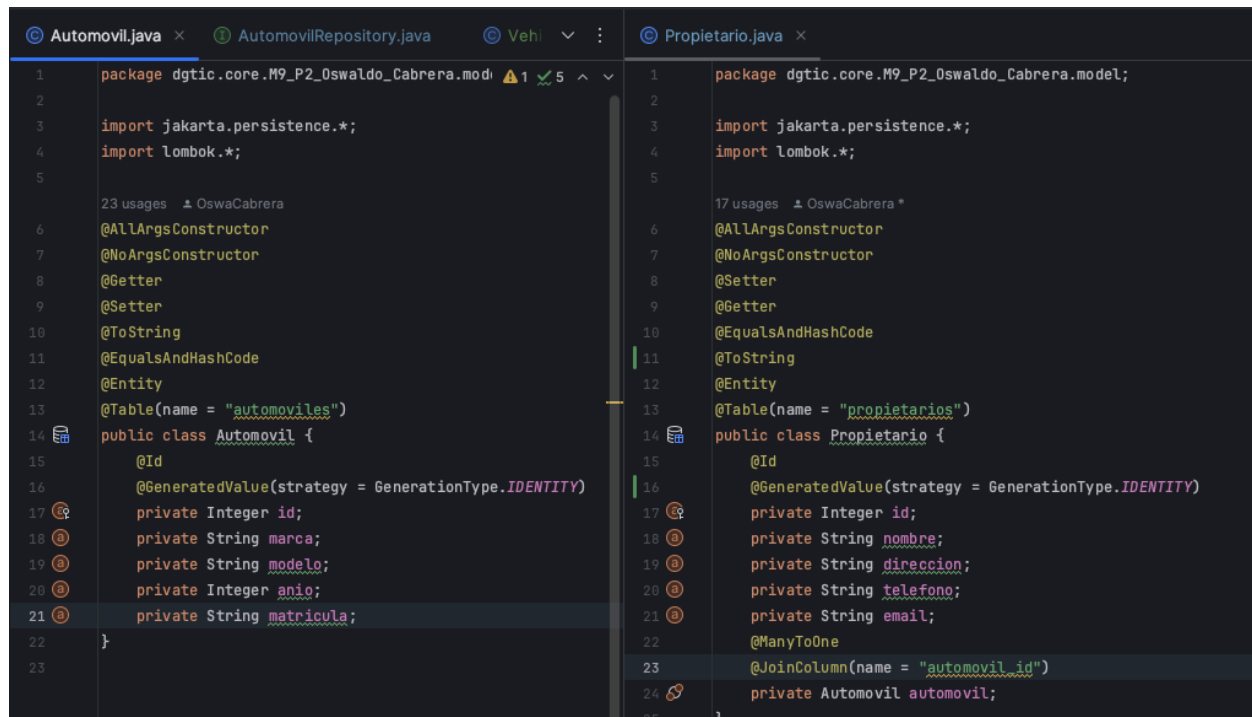


Oswaldo Cabrera Pérez  
Práctica dos

Empezamos por crear nuestros modelos Automovil y Propietario, nos apoyamos de Lombok para crear los métodos comunes.



```
1 package dgtic.core.M9_P2_Oswaldo_Cabrera.model;
2
3 import jakarta.persistence.*;
4 import lombok.*;
5
6 @AllArgsConstructor
7 @NoArgsConstructor
8 @Getter
9 @Setter
10 @ToString
11 @EqualsAndHashCode
12 @Entity
13 @Table(name = "automoviles")
14 public class Automovil {
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private Integer id;
18     private String marca;
19     private String modelo;
20     private Integer anio;
21     private String matricula;
22 }
23
```

```
1 package dgtic.core.M9_P2_Oswaldo_Cabrera.model;
2
3 import jakarta.persistence.*;
4 import lombok.*;
5
6 @AllArgsConstructor
7 @NoArgsConstructor
8 @Setter
9 @Getter
10 @EqualsAndHashCode
11 @ToString
12 @Entity
13 @Table(name = "propietarios")
14 public class Propietario {
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private Integer id;
18     private String nombre;
19     private String direccion;
20     private String telefono;
21     private String email;
22     @ManyToOne
23     @JoinColumn(name = "automovil_id")
24     private Automovil automovil;
25 }
```

Después creamos los repositorios los cuales son interfaces que extienden a CrudRepository para poder crear nuestro CRUD y consultas que necesitamos.

```
4 usages  OswaCabrera *  ⚠ 6 ✓ 20
8 public interface AutomovilRepository extends CrudRepository<Automovil, Integer> {
9     //create a new automovil
10     Automovil save(Automovil automovil);
11     1 usage new *
12     Automovil findAutomovilByMatricula(String matricula);
13
14     new *
15     List<Automovil> findAll();
16     1 usage new *
17     Automovil findAutomovilById(Integer id);
18     1 usage new *
19     List<Automovil> findAllById(Iterable<Integer> ids);
20
21     //Consultas derivadas
22     1 usage new *
23     List<Automovil> findAutomovilByAnio(Integer anio);
24
25     1 usage new *
26     List<Automovil> findAutomovilByMatriculaContainingAndAnio(String matricula, Integer anio);
27
28     1 usage new *
29     List<Automovil> findAutomovilByAnioNot(Integer anio);
30
31     1 usage new *
32     List<Automovil> findAllByOrderByAnioAsc();
33 }
```

```

public interface PropietarioRepository extends CrudRepository<Propietario, Integer> {
    //create a new propietario
    new *
    Propietario save(Propietario propietario);

    //Consultas derivadas
    1 usage new *
    List<Propietario> findPropietarioByNombreEndingWith(String nombre);

    1 usage new *
    Integer countPropietarioByNombreEndingWith(String nombre);

    1 usage new *
    List<Propietario> findPropietarioByIdBetween(Integer id1, Integer id2);

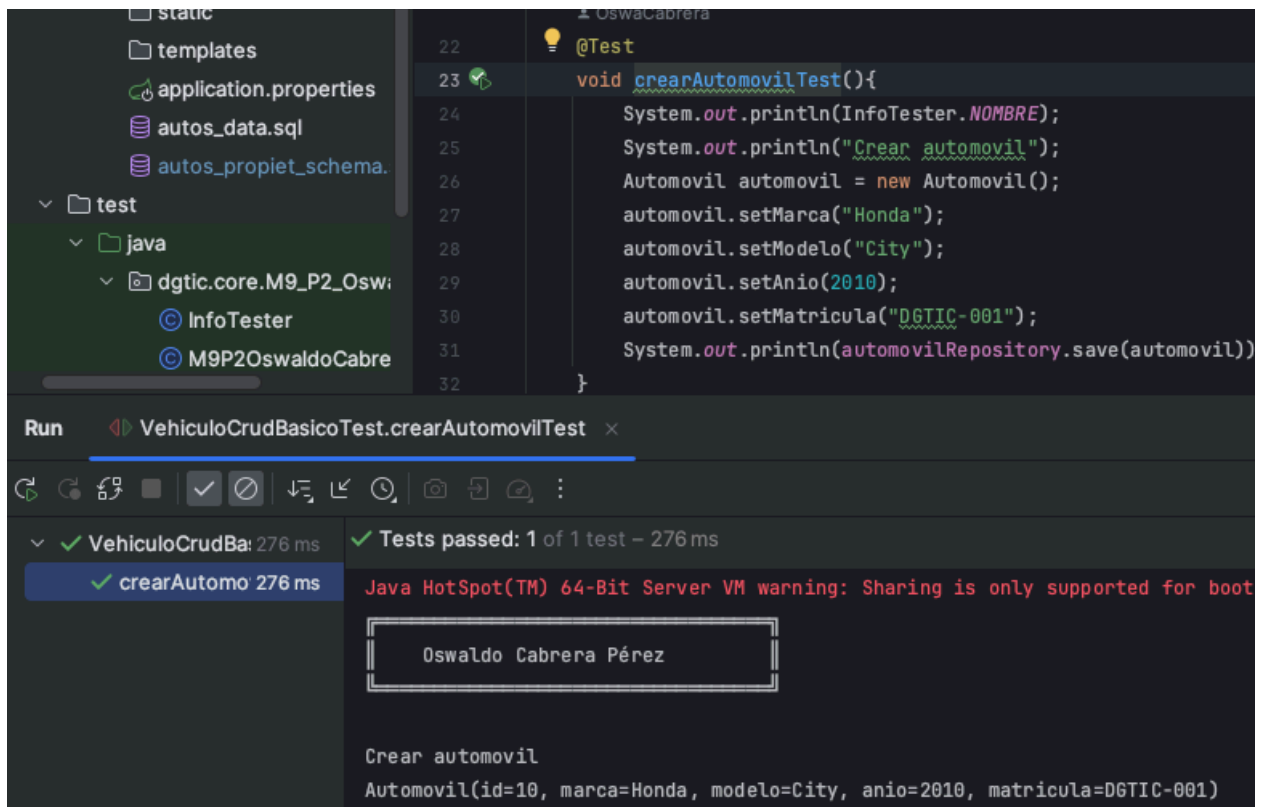
    1 usage new *
    List<Propietario> findPropietarioByTelefonoIsNull();

    1 usage new *
    List<Propietario> findPropietarioByTelefonoNotNull();
}

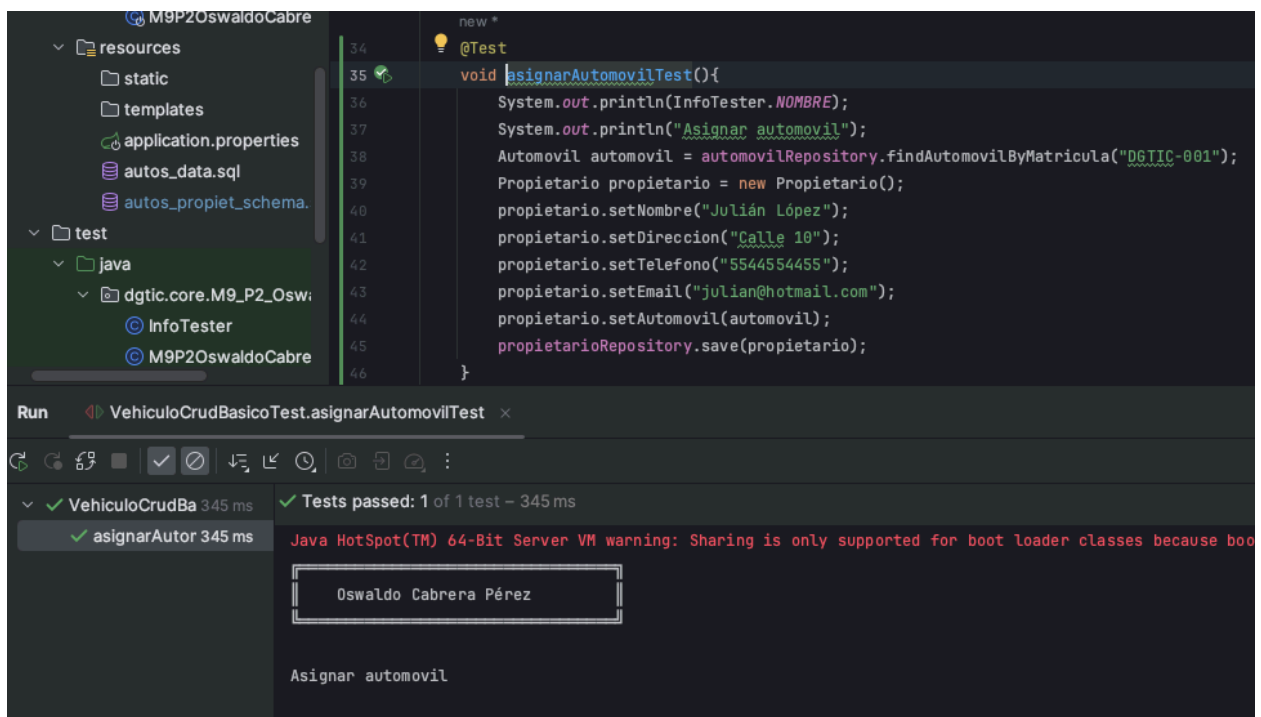
```

1. Crear las siguientes clases de prueba(cada bullet es un. Método de prueba):
  - i. VehiculoCrudBasicotTest.java

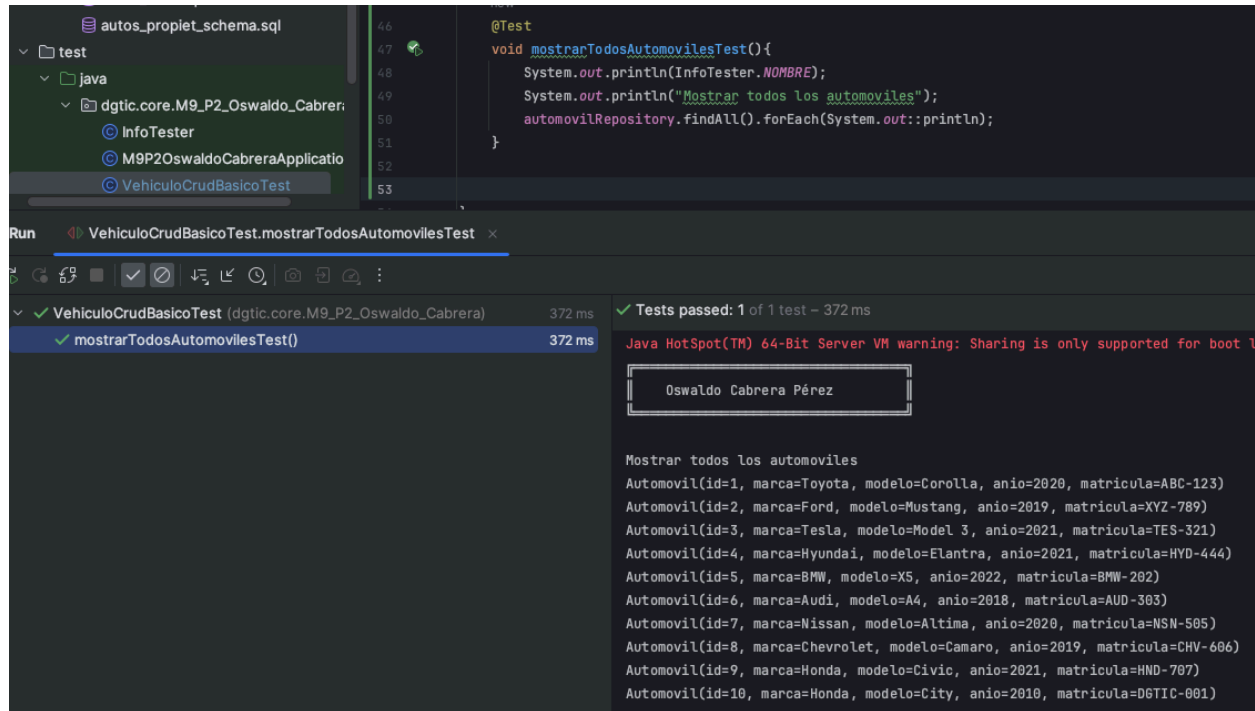
Crear un nuevo Vehículo con los siguientes datos: Honda, City, 2010,DGT-001.



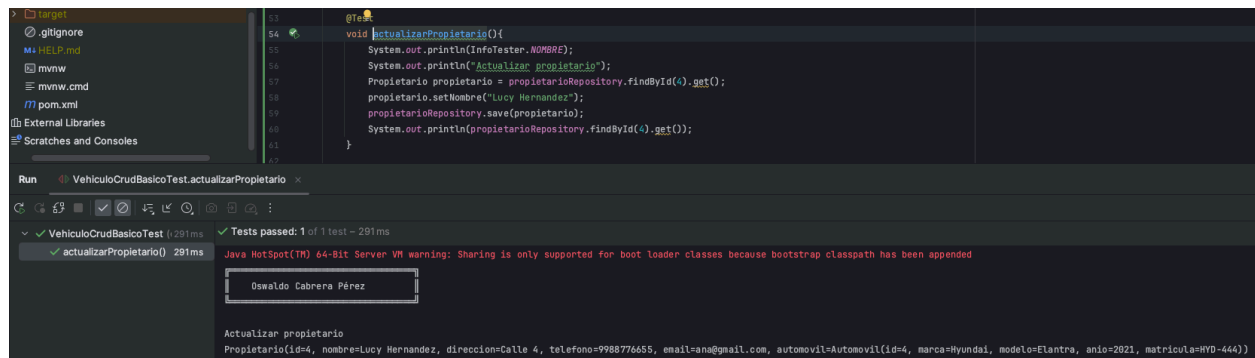
Asignarle ese vehículo al propietario: Julián López, Calle 10, <tel:5544554455>, [julian@hotmail.com](mailto:julian@hotmail.com).



Debe seleccionar y mostrar a la salida todos los automóviles.



Debe cambiar el nombre a Lucy Hernández del propietario con id 4.



Debe seleccionar el automóvil con id = 5(BMW)

The screenshot shows an IDE with a project structure on the left including `autos_propiet_schema.sql`, `test`, `java`, `dgctic.core.M9_P2_Oswaldo_Cabrera`, and `InfoTester`. The main editor displays a Java test method:

```
@Test
void encontrarAutomovilId(){
    System.out.println(InfoTester.NOMBRE);
    System.out.println("Encontrar automovil por id");
    System.out.println(automovilRepository.findAutomovilById(5));
}
```

The Run window shows the test `VehiculoCrudBasicoTest.encontrarAutomovilId` passed in 335 ms. The output includes a warning about Java HotSpot(TM) 64-Bit Server VM sharing support, the name `Oswaldo Cabrera Pérez` in a box, and the following text:

```
Encontrar automovil por id
Automovil(id=5, marca=BMW, modelo=X5, anio=2022, matricula=BMW-202)
```

Seleccionar varios los vehículos con los id 1,3 y 5 con el método `findAllById()` .

The screenshot shows the same IDE with a different test method:

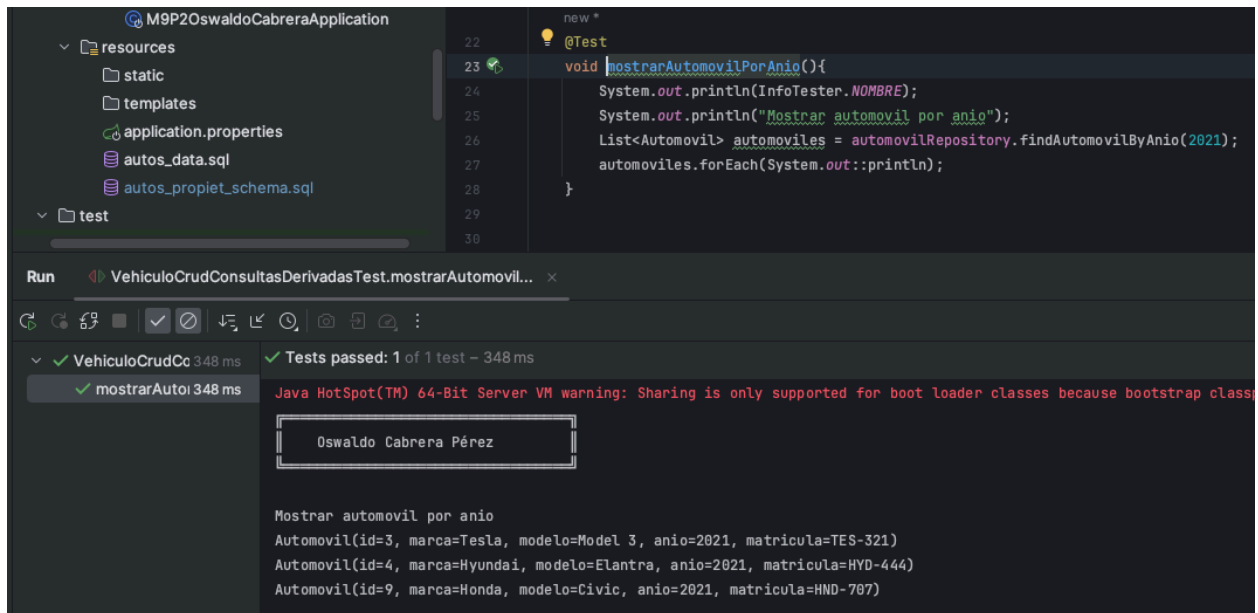
```
@Test
void encontrarPorIterable(){
    System.out.println(InfoTester.NOMBRE);
    System.out.println("Encontrar automovil por iterable");
    automovilRepository.findAllById(List.of(1, 3, 5)).forEach(System.out::println);
}
```

The Run window shows the test `VehiculoCrudBasicoTest.encontrarPorIterable` passed in 315 ms. The output includes the same warning and name box, followed by:

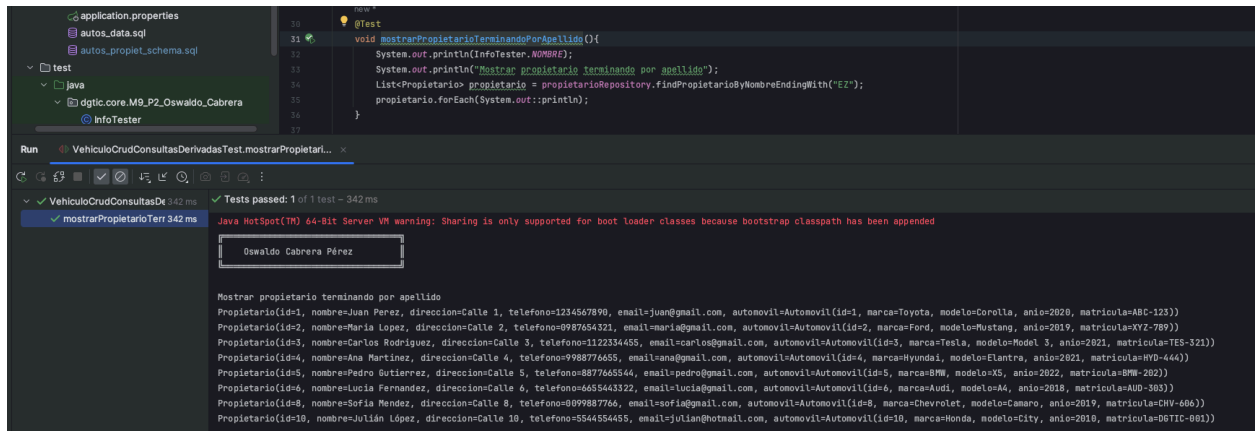
```
Encontrar automovil por iterable
Automovil(id=1, marca=Toyota, modelo=Corolla, anio=2020, matricula=ABC-123)
Automovil(id=3, marca=Tesla, modelo=Model 3, anio=2021, matricula=TES-321)
Automovil(id=5, marca=BMW, modelo=X5, anio=2022, matricula=BMW-202)
```

i. `VehiculoCrudConsultasDerivadasTest.java`

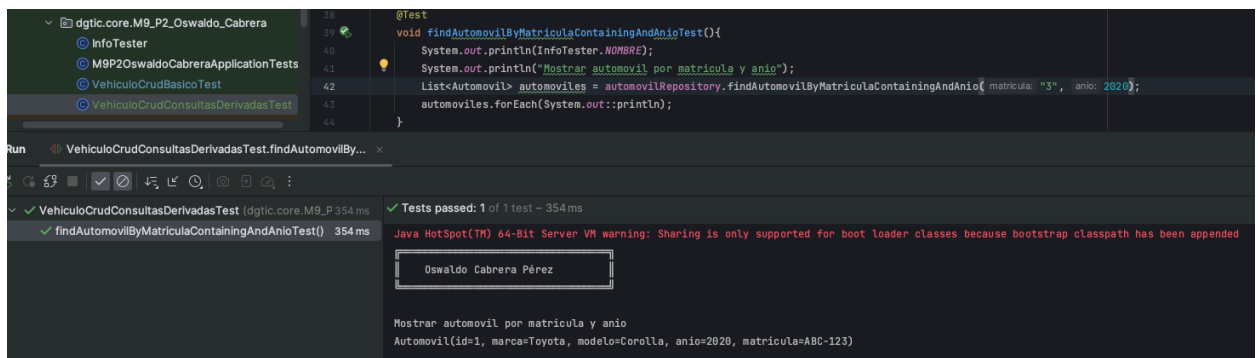
Debe mostrar todos los vehículos del año 2021.



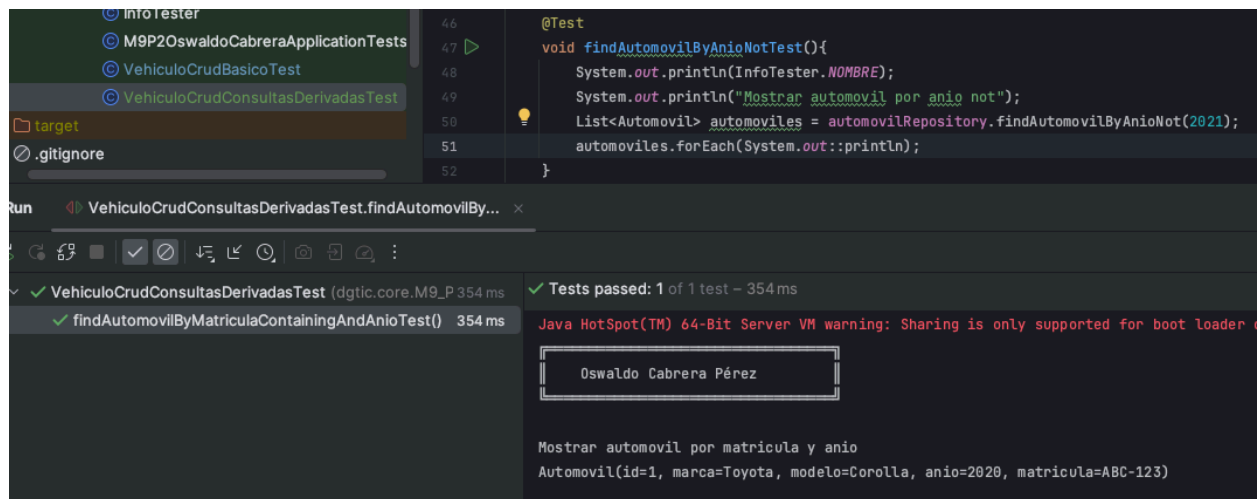
Debe mostrar todos los propietarios cuyos apellidos terminen en 'ez'



Debe mostrar todos los automóviles que la matricula contenga un numero 3 y del año 2020.

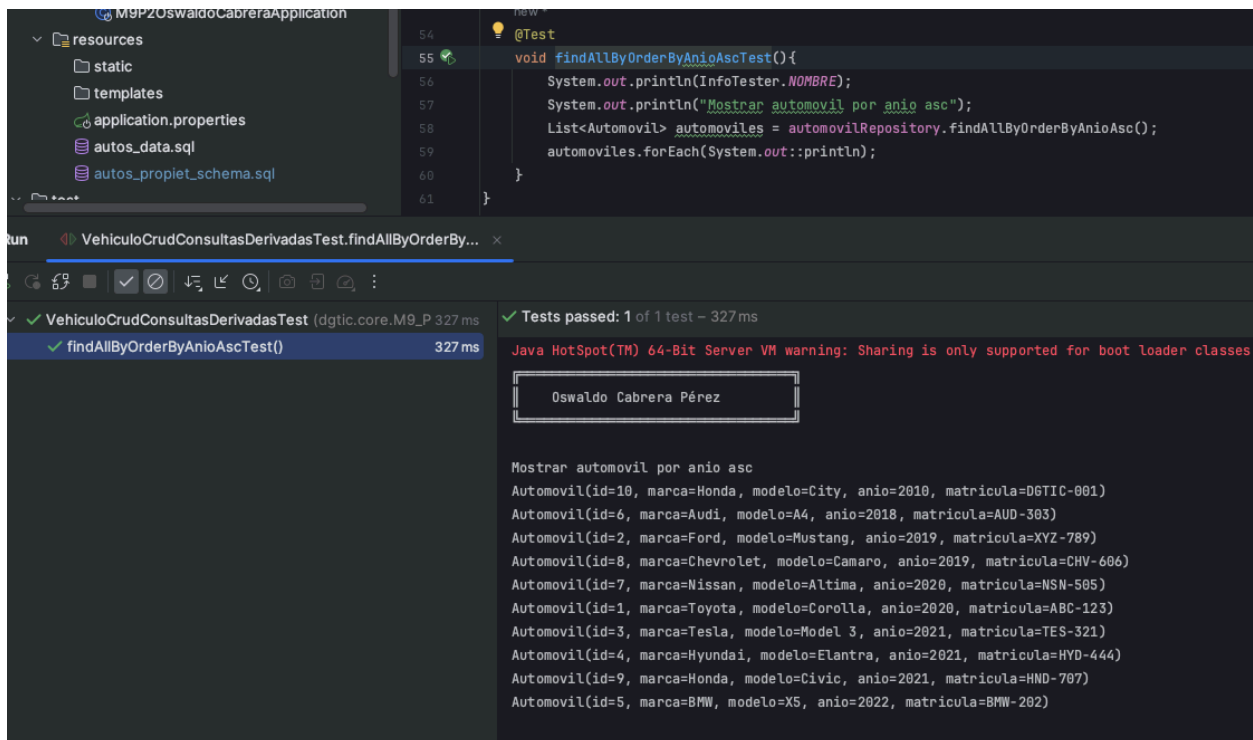


Mostrar todos los automóviles excepto los del año 2021(Not).

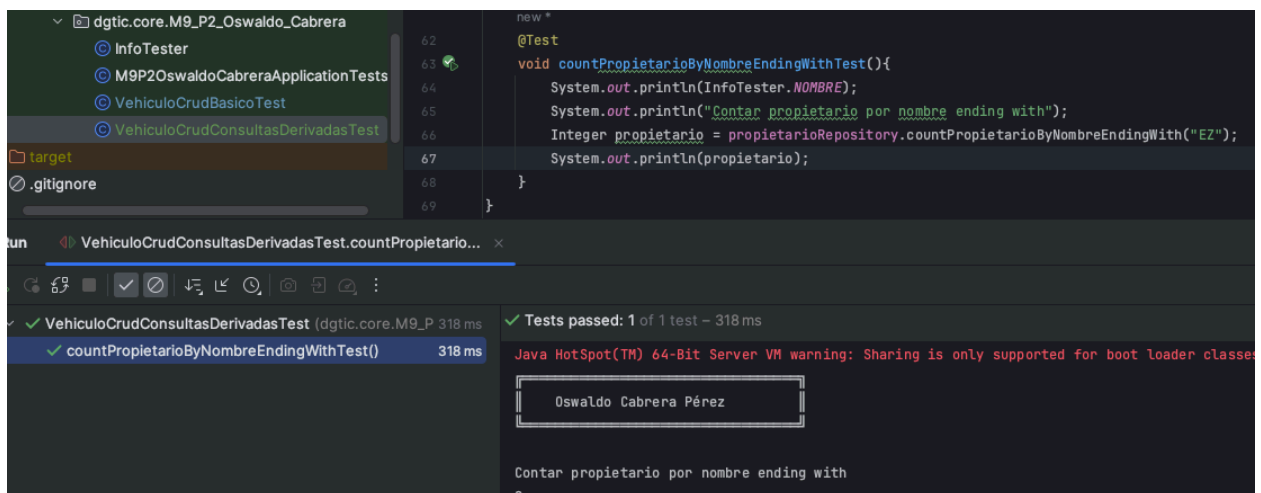


Mostrar los automóviles ordenados por año

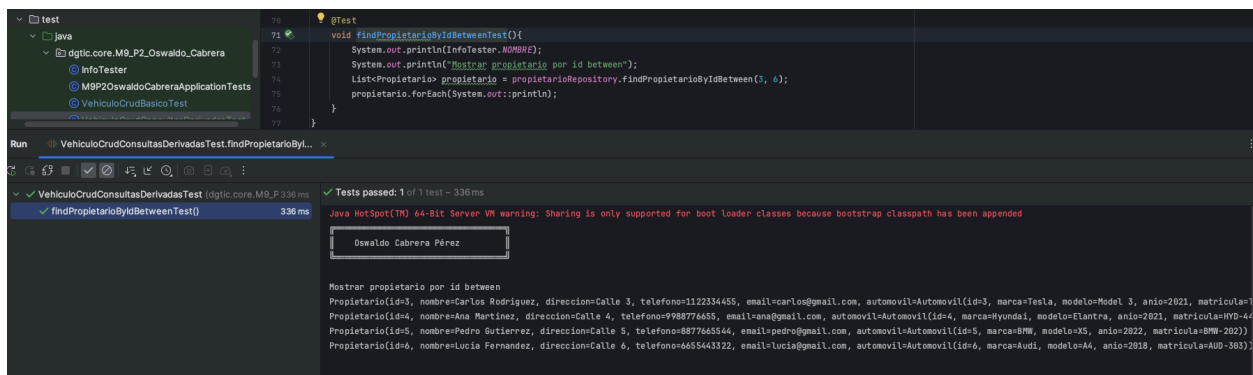




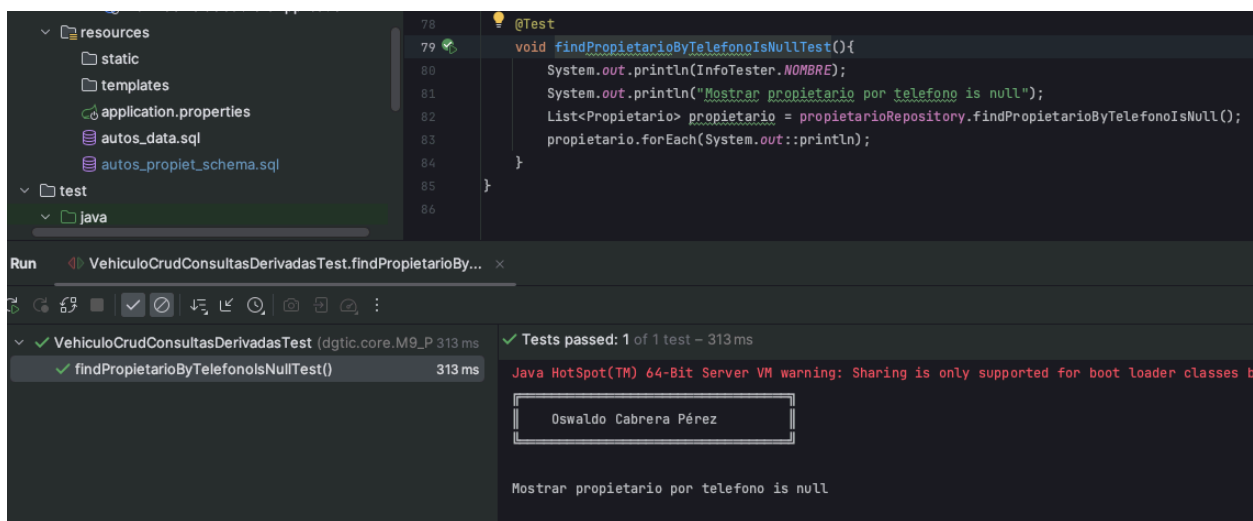
Contar cuantos propietarios su nombre termina en 'ez'.



Mostrar los propietarios cuyos id enten entre 3 y 6.



Mostrar los propietarios cuyo teléfono sea NULL.



Mostrar los propietarios cuyo teléfono sea NotNULL.

