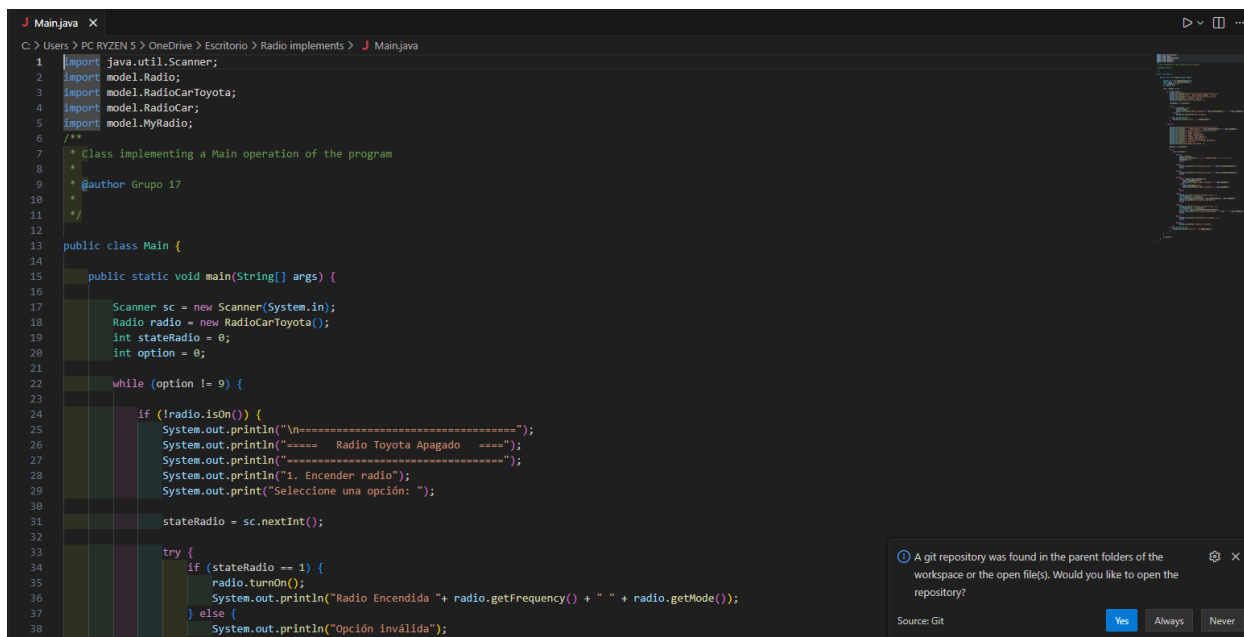


Funcionamiento del programa. Usted adjunta también screenshots o videos para mostrar que el programa funciona

Main – Main para el Objeto RadioCarToyota que es de nuestra creación



```
1 import java.util.Scanner;
2 import model.Radio;
3 import model.RadioCarToyota;
4 import model.RadioCar;
5 import model.MyRadio;
6
7 /**
8  * Class implementing a Main operation of the program
9  *
10  * @author Grupo 17
11  */
12
13 public class Main {
14
15     public static void main(String[] args) {
16
17         Scanner sc = new Scanner(System.in);
18         Radio radio = new RadioCarToyota();
19         int stateRadio = 0;
20         int option = 0;
21
22         while (option != 9) {
23
24             if (!radio.isOn()) {
25                 System.out.println("\n=====");
26                 System.out.println("==== Radio Toyota Apagado ===");
27                 System.out.println("=====");
28                 System.out.println("1. Encender radio");
29                 System.out.print("Seleccione una opción: ");
30
31                 stateRadio = sc.nextInt();
32
33                 try {
34                     if (stateRadio == 1) {
35                         radio.turnOn();
36                         System.out.println("Radio Encendida " + radio.getFrequency() + " " + radio.getMode());
37                     } else {
38                         System.out.println("Opción inválida");
39                     }
40                 } catch (Exception e) {
41                     System.out.println("Error: " + e.getMessage());
42                 }
43             }
44         }
45     }
46 }
```

```
PS C:\Users\PC RYZEN 5\OneDrive\Escritorio\Radio implements> javac Main.java
PS C:\Users\PC RYZEN 5\OneDrive\Escritorio\Radio implements> java Main
```

```
=====
==== Radio Toyota Apagado ====
=====
1. Encender radio
Seleccione una opción: |
```

```
=====
==== Radio Toyota Apagado ====
=====
1. Encender radio
Seleccione una opción: 1
Radio Encendida 87.9 FM

=====
Radio Toyota 87.9 FM
=====
1. Apagar radio
2. Subir frecuencia
3. Bajar frecuencia
4. Cambiar modo (AM/FM)
5. Guardar frecuencia
6. Seleccionar frecuencia guardada
7. Salir
Seleccione una opción: |
```

```
Seleccione una opción: 2
Frecuencia actual: 88.1 FM

=====
Radio Toyota 88.1 FM
=====
1. Apagar radio
2. Subir frecuencia
3. Bajar frecuencia
4. Cambiar modo (AM/FM)
5. Guardar frecuencia
6. Seleccionar frecuencia guardada
7. Salir
Seleccione una opción: |
```

Seleccione una opción: 3

Frecuencia actual: 87.9 FM

=====

Radio Toyota 87.9 FM

=====

1. Apagar radio
2. Subir frecuencia
3. Bajar frecuencia
4. Cambiar modo (AM/FM)
5. Guardar frecuencia
6. Seleccionar frecuencia guardada
7. Salir

Seleccione una opción: |

Seleccione una opción: 4

Modo cambiado a AM

=====

Radio Toyota 530.0 AM

=====

1. Apagar radio
2. Subir frecuencia
3. Bajar frecuencia
4. Cambiar modo (AM/FM)
5. Guardar frecuencia
6. Seleccionar frecuencia guardada
7. Salir

Seleccione una opción: |

Seleccione una opción: 5

Número de botón (1-12): 3

Frecuencia guardada

=====

Radio Toyota 530.0 AM

=====

1. Apagar radio
2. Subir frecuencia
3. Bajar frecuencia
4. Cambiar modo (AM/FM)
5. Guardar frecuencia
6. Seleccionar frecuencia guardada
7. Salir

Seleccione una opción: |

Número de botón (1-12): 3

Frecuencia seleccionada: 530.0 AM

=====

Radio Toyota 530.0 AM

=====

1. Apagar radio
2. Subir frecuencia
3. Bajar frecuencia
4. Cambiar modo (AM/FM)
5. Guardar frecuencia
6. Seleccionar frecuencia guardada
7. Salir

Seleccione una opción: |

Seleccione una opción: 1

----- Apagando Radio -----

=====

==== Radio Toyota Apagado ====

=====

1. Encender radio

Seleccione una opción: |

Funcionamiento del programa usando la clase implementada por otros grupos. El auxiliar seleccionará estos grupos al azar, pero usted puede indicar con quienes ha probado que funciona su programa. Adjunte screenshots de funcionamiento de su programa funcionando con clases implementadas por otros grupos.

#### Main2 – Objeto RadioCar

```
PS C:\Users\PC RYZEN 5\OneDrive\Escritorio\Radio implements> javac Main2.java
PS C:\Users\PC RYZEN 5\OneDrive\Escritorio\Radio implements> java Main2

=====
====  Radio Toyota Apagado  ====
=====
1. Encender radio
Seleccione una opción: 1
Radio Encendida 87.9 FM

=====
          Radio Toyota  87.9 FM
=====
1. Apagar radio
2. Subir frecuencia
3. Bajar frecuencia
4. Cambiar modo (AM/FM)
5. Guardar frecuencia
6. Seleccionar frecuencia guardada
7. Salir
Seleccione una opción: |
```

#### Main3 – Objeto MyRadio.

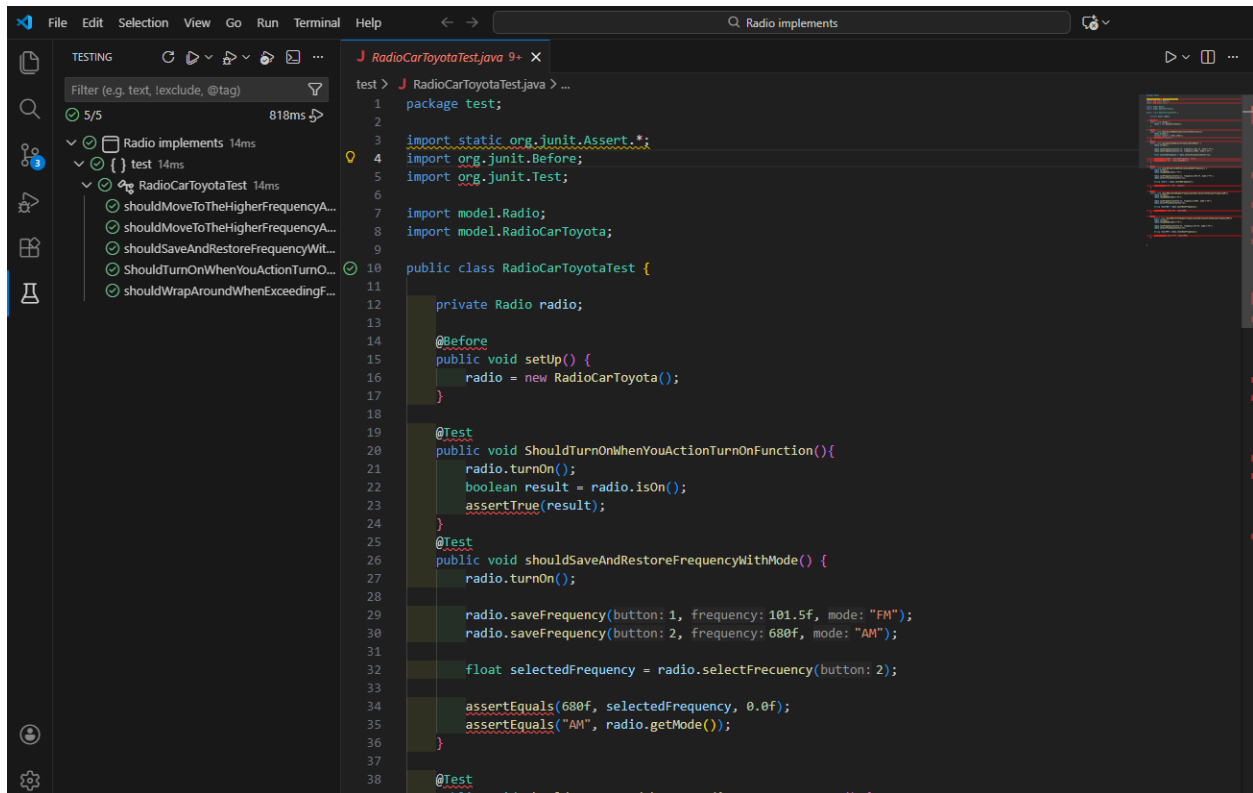
```
PS C:\Users\PC RYZEN 5\OneDrive\Escritorio\Radio implements> javac Main3.java
PS C:\Users\PC RYZEN 5\OneDrive\Escritorio\Radio implements> java Main3

=====
====  Radio Toyota Apagado  ====
=====
1. Encender radio
Seleccione una opción: 1
Radio Encendida 87.9 FM

=====
          Radio Toyota  87.9 FM
=====
1. Apagar radio
2. Subir frecuencia
3. Bajar frecuencia
4. Cambiar modo (AM/FM)
5. Guardar frecuencia
6. Seleccionar frecuencia guardada
7. Salir
Seleccione una opción: |
```

Utiliza JUnit para probar por lo menos tres de las operaciones del radio. Adjunte con su tarea las pruebas unitarias definidas. Adjunte screenshots de la ejecución de las pruebas. Debe existir por lo menos una muestra de que las pruebas fallan y una de que las pruebas si funcionan.

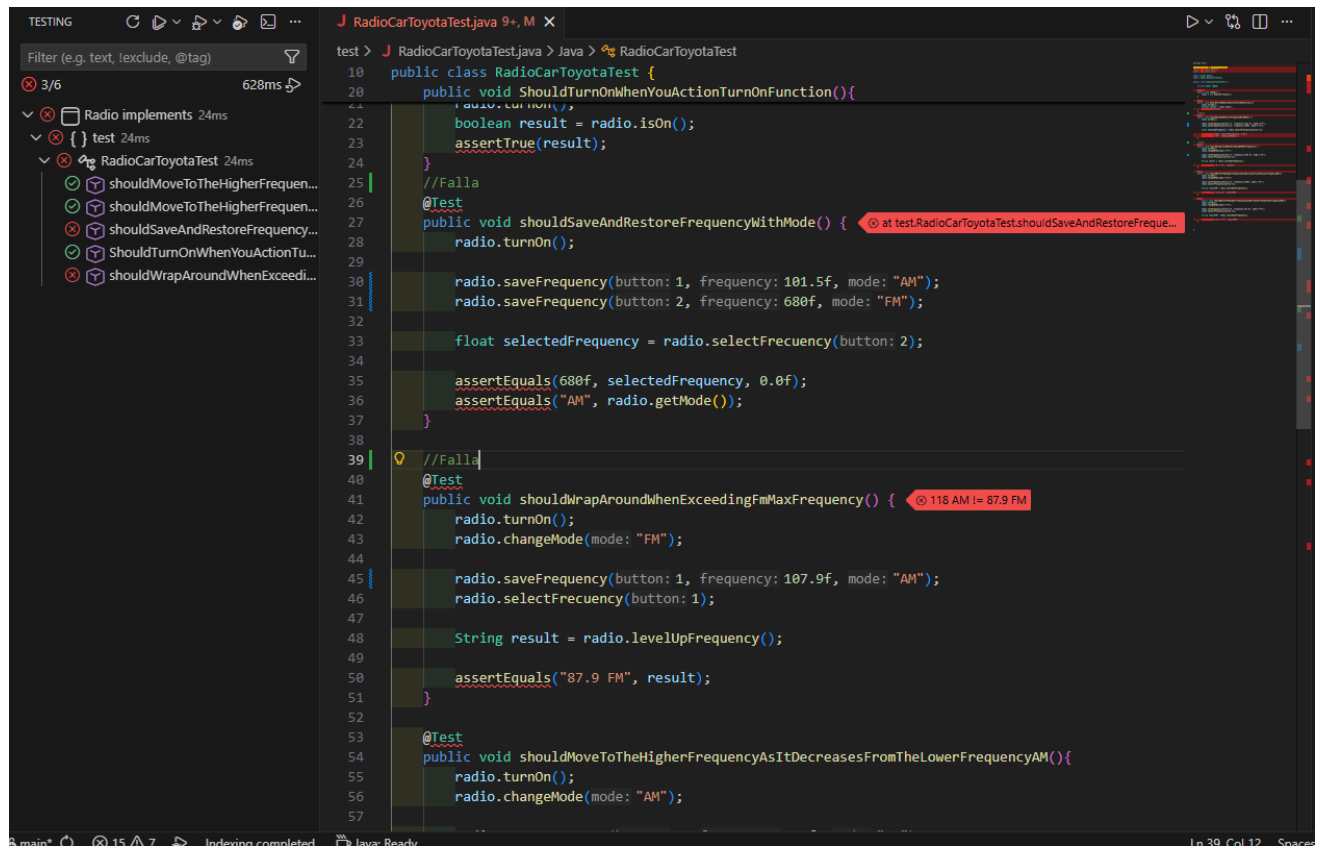
Exitos



The screenshot shows an IDE with a dark theme. On the left, the 'TESTING' sidebar displays a list of test results. The first test, 'Radio implements', passed in 14ms. Below it, 'test' also passed in 14ms. Further down, 'RadioCarToyotaTest' is listed with a duration of 14ms. Under 'RadioCarToyotaTest', several individual test methods are shown, all with green checkmarks indicating they passed. The main editor area displays the source code for 'RadioCarToyotaTest.java'. The code includes package declarations, imports for JUnit and the 'model' package, and the test class itself. The test class has a private 'Radio' instance, a '@Before' setup method, and several '@Test' methods. The visible test methods include 'ShouldTurnOnWhenYouActionTurnOnFunction()', 'shouldSaveAndRestoreFrequencyWithMode()', and 'shouldWrapAroundWhenExceedingFrequency()'. The code uses 'assertEquals' and 'assertTrue' for assertions.

```
1 package test;
2
3 import static org.junit.Assert.*;
4 import org.junit.Before;
5 import org.junit.Test;
6
7 import model.Radio;
8 import model.RadioCarToyota;
9
10 public class RadioCarToyotaTest {
11
12     private Radio radio;
13
14     @Before
15     public void setUp() {
16         radio = new RadioCarToyota();
17     }
18
19     @Test
20     public void ShouldTurnOnWhenYouActionTurnOnFunction(){
21         radio.turnOn();
22         boolean result = radio.isOn();
23         assertTrue(result);
24     }
25
26     @Test
27     public void shouldSaveAndRestoreFrequencyWithMode() {
28         radio.turnOn();
29
30         radio.saveFrequency(button: 1, frequency: 101.5f, mode: "FM");
31         radio.saveFrequency(button: 2, frequency: 680f, mode: "AM");
32
33         float selectedFrequency = radio.selectFrequency(button: 2);
34
35         assertEquals(680f, selectedFrequency, 0.0f);
36         assertEquals("AM", radio.getMode());
37     }
38
39     @Test
40     public void shouldWrapAroundWhenExceedingFrequency() {
```

Falla ya que nos retorna frecuencias que no esperamos



```
10 public class RadioCarToyotaTest {
20     public void ShouldTurnOnWhenYouActionTurnOnFunction(){
21         radio.turnOn();
22         boolean result = radio.isOn();
23         assertTrue(result);
24     }
25     //Falla
26     @Test
27     public void shouldSaveAndRestoreFrequencyWithMode() {
28         radio.turnOn();
29
30         radio.saveFrequency(button: 1, frequency: 101.5f, mode: "AM");
31         radio.saveFrequency(button: 2, frequency: 680f, mode: "FM");
32
33         float selectedFrequency = radio.selectFrequency(button: 2);
34
35         assertEquals(680f, selectedFrequency, 0.0f);
36         assertEquals("AM", radio.getMode());
37     }
38
39     //Falla
40     @Test
41     public void shouldWrapAroundWhenExceedingFmMaxFrequency() {
42         radio.turnOn();
43         radio.changeMode(mode: "FM");
44
45         radio.saveFrequency(button: 1, frequency: 107.9f, mode: "AM");
46         radio.selectFrequency(button: 1);
47
48         String result = radio.levelUpFrequency();
49
50         assertEquals("87.9 FM", result);
51     }
52
53     @Test
54     public void shouldMoveToTheHigherFrequencyAsItDecreasesFromTheLowerFrequencyAM(){
55         radio.turnOn();
56         radio.changeMode(mode: "AM");
57     }
```

TESTING

Filter (e.g. text, !exclude, @tag)

3/6 628ms

- Radio implements 24ms
- test 24ms
- RadioCarToyotaTest 24ms
  - shouldMoveToTheHigherFrequen...
  - shouldMoveToTheHigherFrequen...
  - shouldSaveAndRestoreFrequency...
  - ShouldTurnOnWhenYouActionTu...
  - shouldWrapAroundWhenExceedi...

at test.RadioCarToyotaTest.shouldSaveAndRestoreFrequencyWithMode() (RadioCarToyotaTest.java:36)

118 AM != 87.9 FM

main\* 15 7 Indexing completed Java Ready Ln 39, Col 12