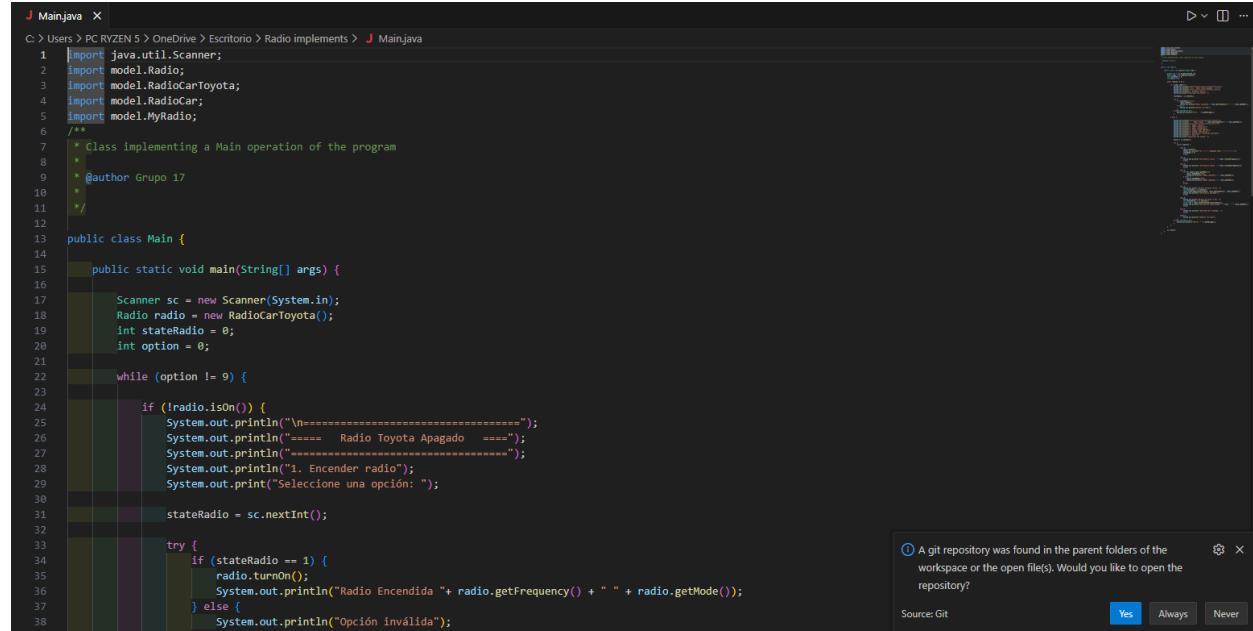


Funcionamiento del programa. Usted adjunta también screenshots o videos para mostrar que el programa funciona

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  * Class implementing a Main operation of the program
8  *
9  * @author Grupo 17
10 *
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=====\n      Radio Toyota Apagado      \n=====");
26                 System.out.println("=====1. Encender radio\n");
27                 System.out.print("Seleccione una opción: ");
28
29                 stateRadio = sc.nextInt();
30
31                 try {
32                     if (stateRadio == 1) {
33                         radio.turnOn();
34                         System.out.println("Radio Encendida " + radio.getFrequency() + " " + radio.getMode());
35                     } else {
36                         System.out.println("Opción inválida");
37                     }
38                 } catch (Exception e) {
39                     System.out.println("Error: " + e.getMessage());
40                 }
41             }
42         }
43     }
44 }
```

```
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 ===== Seleccione una opción: 2
=====
1. Encender radio                               Frecuencia actual: 88.1 FM
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: |
=====
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

Seleccione una opción: 4

Modo cambiado a AM

===== 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: |

===== 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.

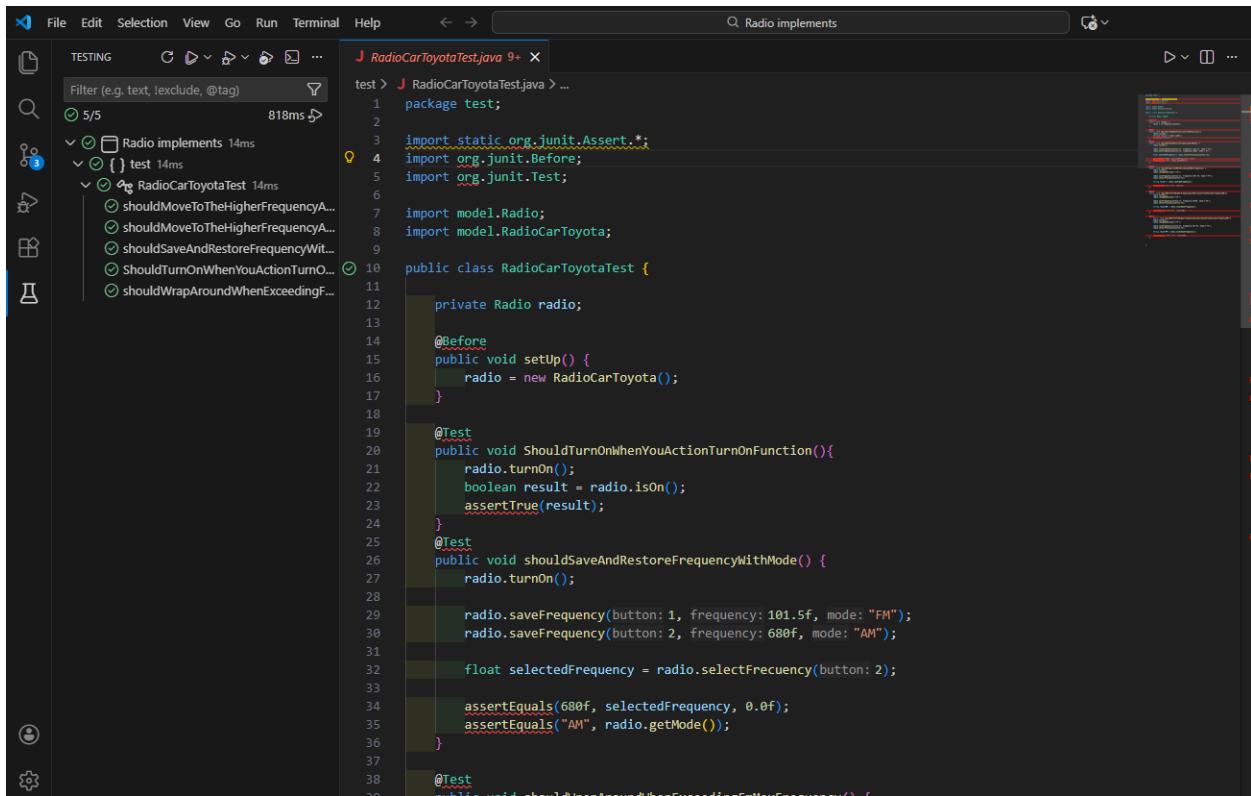
```
PS C:\Users\PC RYZEN 5\OneDrive\Escritorio\Radio implements> javac Main3.java
PS C:\Users\PC RYZEN 5\OneDrive\Escritorio\Radio implements> java Main3

=====
===== 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



```
RadioCarToyotaTest.java 9+ x
test > J RadioCarToyotaTest.java > ...
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     private Radio radio;
12
13     @Before
14     public void setUp() {
15         radio = new RadioCarToyota();
16     }
17
18     @Test
19     public void ShouldTurnOnWhenYouActionTurnOnFunction(){
20         radio.turnOn();
21         boolean result = radio.isOn();
22         assertTrue(result);
23     }
24     @Test
25     public void shouldSaveAndRestoreFrequencyWithMode() {
26         radio.turnOn();
27
28         radio.saveFrequency(button: 1, frequency: 101.5f, mode: "FM");
29         radio.saveFrequency(button: 2, frequency: 680f, mode: "AM");
30
31         float selectedFrequency = radio.selectFrequency(button: 2);
32
33         assertEquals(680f, selectedFrequency, 0.0f);
34         assertEquals("AM", radio.getMode());
35     }
36
37
38     @Test
39     public void shouldWrapAroundWhenExceedingFrequency() {
40
41     }
42 }
```

Falla ya que nos retorna frecuencias que no esperamos

The screenshot shows an IDE interface with a left sidebar titled "TESTING" and a main code editor area. The code editor contains a Java file named "RadioCarToyotaTest.java". The file includes several test methods, many of which are marked as failing (indicated by red squiggly lines under the method names). The failing tests include:

- shouldMoveToTheHigherFrequency() - Fails with a message: "118 AM != 87.9 FM"
- shouldSaveAndRestoreFrequencyWithMode() - Fails with a message: "118 AM != 87.9 FM"
- shouldWrapAroundWhenExceedingFmMaxFrequency() - Fails with a message: "118 AM != 87.9 FM"
- shouldMoveToTheHigherFrequencyAsItDecreasesFromTheLowerFrequencyAM() - Fails with a message: "118 AM != 87.9 FM"

The code itself is related to testing a radio component, specifically its frequency selection and mode switching logic.

```
TESTING ⌂ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋ ... J RadioCarToyotaTest.java 9+, M X
Filter (e.g. text, !exclude, @tag) ⌈
3/6 628ms
Radio implements 24ms
{ } test 24ms
RadioCarToyotaTest 24ms
shouldMoveToTheHigherFrequency()
shouldSaveAndRestoreFrequencyWithMode()
shouldWrapAroundWhenExceedingFmMaxFrequency()
shouldMoveToTheHigherFrequencyAsItDecreasesFromTheLowerFrequencyAM()

public class RadioCarToyotaTest {
    public void shouldTurnOnWhenYouActionTurnOnFunction(){
        radio.turnOn();
        boolean result = radio.isOn();
        assertTrue(result);
    }
    //Falla
    @Test
    public void shouldSaveAndRestoreFrequencyWithMode() { ⌈ at test.RadioCarToyotaTest.shouldSaveAndRestoreFrequencyWithMode() ...
        radio.turnOn();

        radio.saveFrequency(button: 1, frequency: 101.5f, mode: "AM");
        radio.saveFrequency(button: 2, frequency: 680f, mode: "FM");

        float selectedFrequency = radio.selectFrequency(button: 2);

        assertEquals(680f, selectedFrequency, 0.0f);
        assertEquals("AM", radio.getMode());
    }

    //Falla
    @Test
    public void shouldWrapAroundWhenExceedingFmMaxFrequency() { ⌈ 118 AM != 87.9 FM
        radio.turnOn();
        radio.changeMode(mode: "FM");

        radio.saveFrequency(button: 1, frequency: 107.9f, mode: "AM");
        radio.selectFrequency(button: 1);

        String result = radio.levelUpFrequency();

        assertEquals("87.9 FM", result);
    }

    @Test
    public void shouldMoveToTheHigherFrequencyAsItDecreasesFromTheLowerFrequencyAM(){
        radio.turnOn();
        radio.changeMode(mode: "AM");
    }
}
```