# RGR en Tic-Tac-Toe

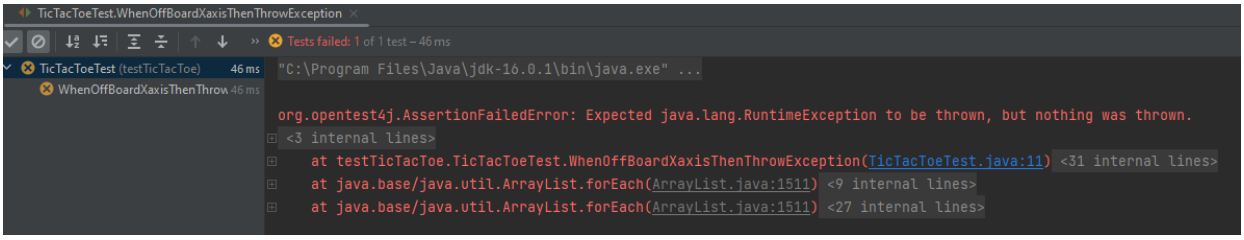
## Sin Refactorizacion:

Veamos el proceso RGR para el método de prueba **WhenOffBoardXaxisThenThrowException**

**Obs: No se ha puesto mucho comentario, ya que se puede entender fácilmente que es lo que hace cada método, sin embargo hay algunos métodos donde si se ha puesto algunos comentarios**

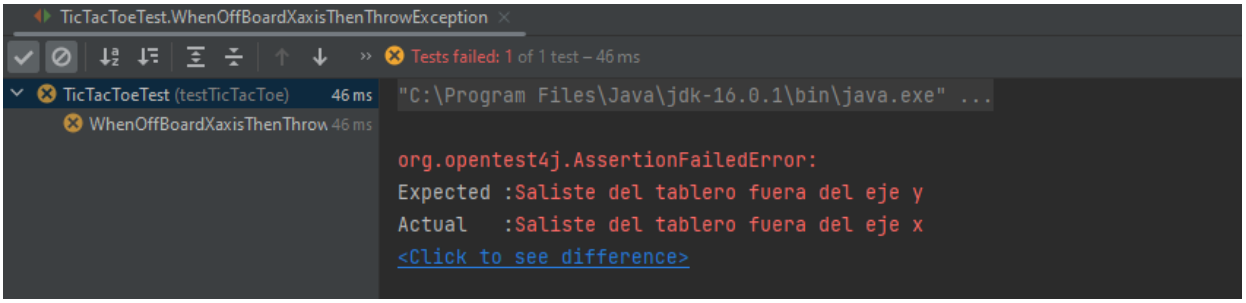
**Primero escribimos la prueba sin implementar el método jugar.**

```
@Test
public void WhenOffBoardXaxisThenThrowException() {
    TicTacToe tictactoe = new TicTacToe();
    Throwable exception = assertThrows(RuntimeException.class, () -> {
        tictactoe.jugar( row: 5,  column: 0);
    });
    assertEquals( expected: "Saliste del tablero fuera del eje x", exception.getMessage());
}
```



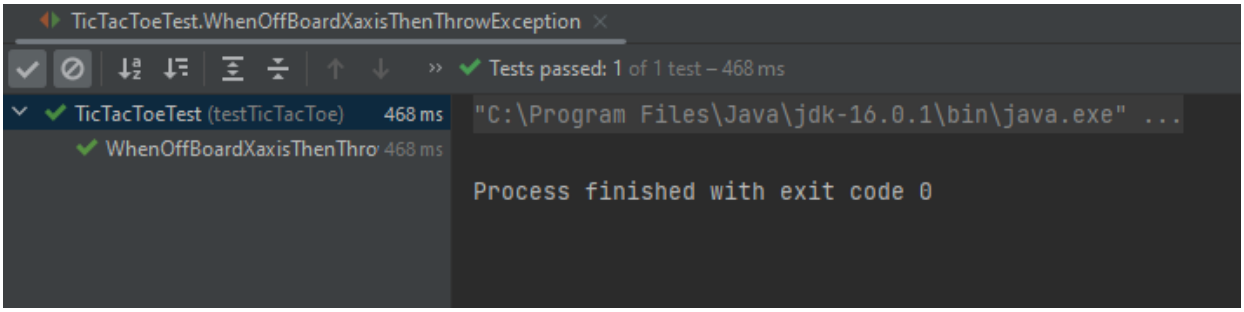

**Luego, implementamos el método jugar, sin embargo, hacemos que el valor esperado no coincide con el valor actual**

```
@Test
public void WhenOffBoardXaxisThenThrowException() {
    TicTacToe tictactoe = new TicTacToe();
    Throwable exception = assertThrows(RuntimeException.class, () -> {
        tictactoe.jugar( row: 5,  column: 0);
    });
    assertEquals( expected: "Saliste del tablero fuera del eje y", exception.getMessage());
}
```




**Ahora hacemos coincidir el valor esperado con el valor actual para que pase la prueba**

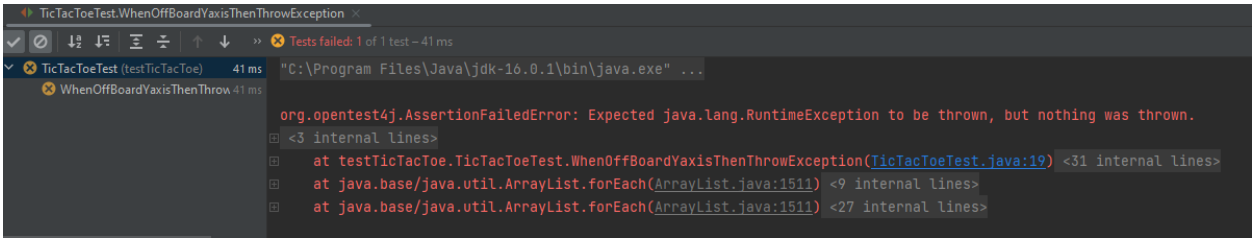
```
@Test
public void WhenOffBoardXaxisThenThrowException() {
    TicTacToe tictactoe = new TicTacToe();
    Throwable exception = assertThrows(RuntimeException.class, () -> {
        tictactoe.jugar( row: 5,  column: 0);
    });
    assertEquals( expected: "Saliste del tablero fuera del eje x", exception.getMessage());
}
```




Veamos el proceso RGR para el método de prueba **WhenOffBoardYaxisThenThrowException**

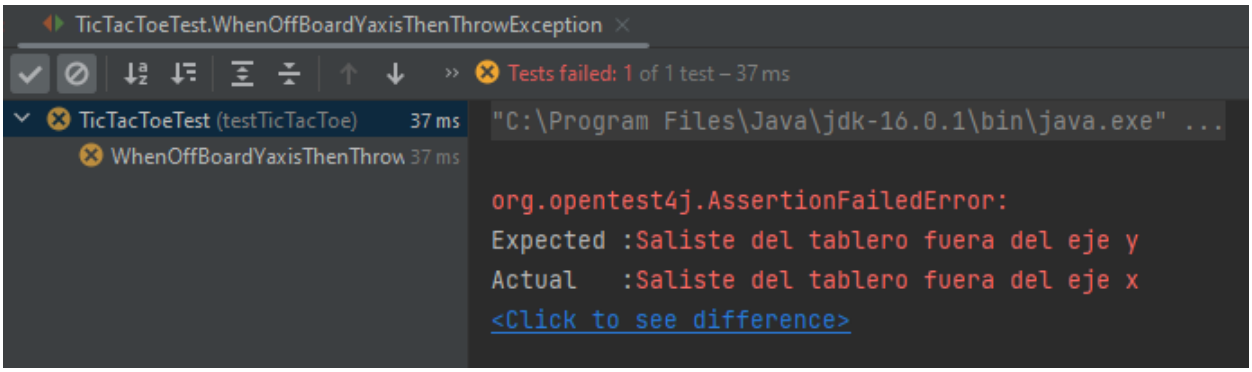
**Primero escribimos la prueba sin implementar el método jugar.**

```
@Test
public void WhenOffBoardYaxisThenThrowException() {
    TicTacToe tictactoe = new TicTacToe();
    Throwable exception = assertThrows(RuntimeException.class, () -> {
        tictactoe.jugar( row: 0,  column: 5);
    });
    assertEquals( expected: "Saliste del tablero fuera del eje y", exception.getMessage());
}
```



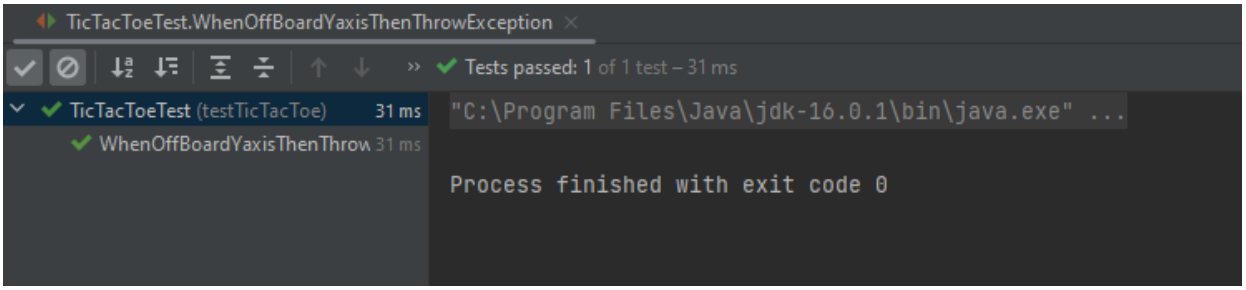

**Luego, implementamos el método jugar, sin embargo, hacemos que el valor esperado no coincide con el valor actual**

```
@Test
public void WhenOffBoardYaxisThenThrowException() {
    TicTacToe tictactoe = new TicTacToe();
    Throwable exception = assertThrows(RuntimeException.class, () -> {
        tictactoe.jugar( row: 5,  column: 0);
    });
    assertEquals( expected: "Saliste del tablero fuera del eje y", exception.getMessage());
}
```



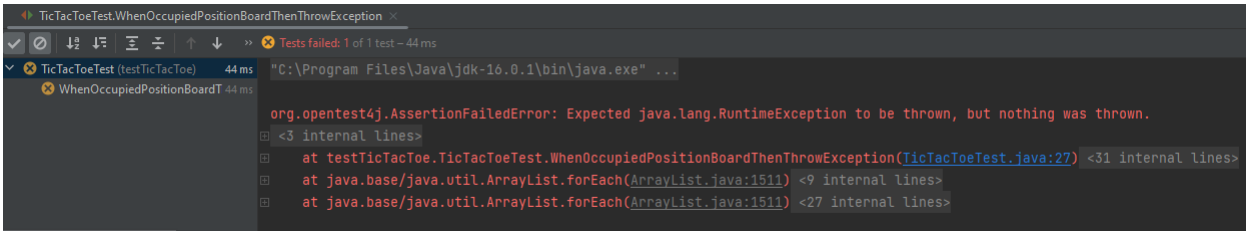

**Ahora hacemos coincidir el valor esperado con el valor actual para que pase la prueba**

```
@Test
public void WhenOffBoardYaxisThenThrowException() {
    TicTacToe tictactoe = new TicTacToe();
    Throwable exception = assertThrows(RuntimeException.class, () -> {
        tictactoe.jugar( row: 0,  column: 5);
    });
    assertEquals( expected: "Saliste del tablero fuera del eje y", exception.getMessage());
}
```



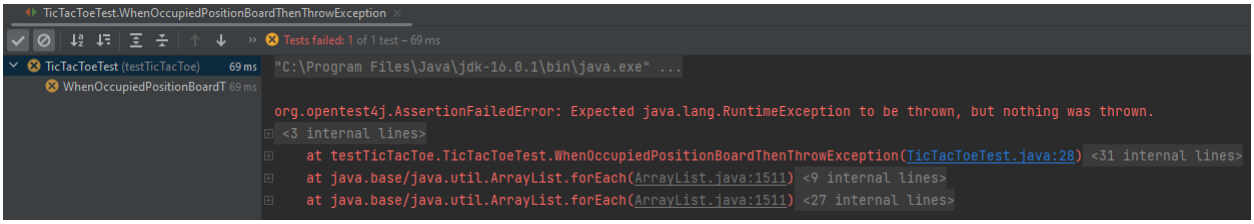

**Primero escribimos la prueba sin implementar el método jugar.**

```
@Test
public void WhenOccupiedPositionBoardThenThrowException() {
    TicTacToe tictactoe = new TicTacToe();
    Throwable exception = assertThrows(RuntimeException.class, () -> {
        tictactoe.jugar( row: 0,  column: 2);
        tictactoe.jugar( row: 0,  column: 2);
    });
    assertEquals( expected: "Esta posicion esta ocupada", exception.getMessage());
}
```



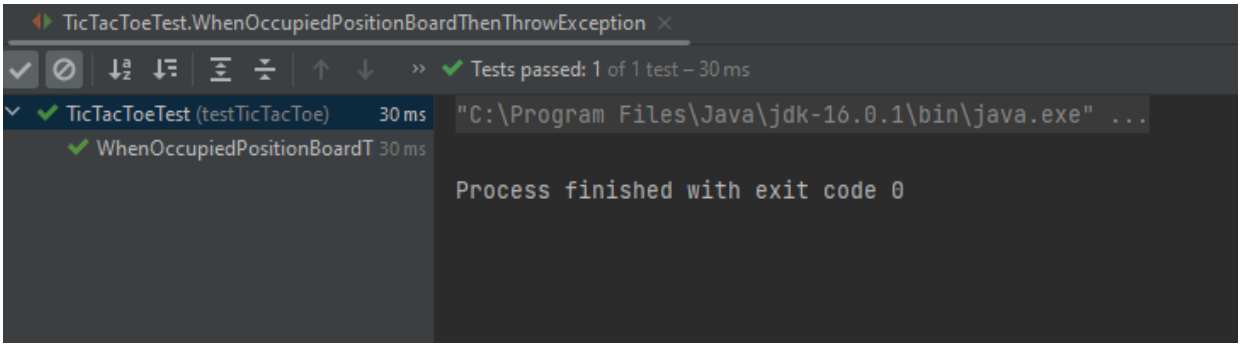

**Luego, implementamos el método jugar, sin embargo, hacemos que falle**

```
@Test
public void WhenOccupiedPositionBoardThenThrowException() {
    TicTacToe tictactoe = new TicTacToe();
    tictactoe.jugar( row: 1,  column: 2);
    Throwable exception = assertThrows(RuntimeException.class, () -> {
        tictactoe.jugar( row: 0,  column: 2);
    });
    assertEquals( expected: "Esta posicion esta ocupada", exception.getMessage());
}
```




**Ahora hacemos coincidir el valor esperado con el valor actual para que pase la prueba**

```java
@Test
public void WhenOccupiedPositionBoardThenThrowException() {
    TicTacToe tictactoe = new TicTacToe();
    tictactoe.jugar( row: 0,  column: 2);
    Throwable exception = assertThrows(RuntimeException.class, () -> {
        tictactoe.jugar( row: 0,  column: 2);
    });
    assertEquals( expected: "Esta posicion esta ocupada", exception.getMessage());
}
```



## Con Refactorizacion:

```
public void jugar(int row, int column) {

    if (row >= 0 && row < TOTALROWS && column >= 0 && column < TOTALCOLUMNS ) {
        if(grid[row][column] == Cell.EMPTY) {
            grid[row][column] = (turn == 'X') ? Cell.CROSS : Cell.NOUGHT;
            updateGameState(turn, row, column);
            turn = (turn == 'X') ? 'O' : 'X';
        }
        else{
            throw new RuntimeException("Esta posicion esta ocupada");
        }
    }
    else{
        if(row >= TOTALROWS || row<0) {
            throw new RuntimeException("Saliste del tablero fuera del eje x");
        }
        if(column >= TOTALCOLUMNS || column<0){

            throw new RuntimeException("Saliste del tablero fuera del eje y");
        }
    }

}
```

**Si bien el código que hemos hecho hasta ahora cumple con los requisitos establecidos por las pruebas, parece un poco confuso. Si alguien lo leyera, no quedaría claro qué hace el método jugar. Refactoriza moviendo el código a métodos separados.**

**Por ello hacemos los siguiente cambios:**

```java
public void jugar(int row, int column) {
    if (row >= 0 && row < TOTALROWS && column >= 0 && column < TOTALCOLUMNS ) {
        posicion_OcupadoDesacupada(row,column);
    }
    else{
        posicion_fuerdaDelTablero(row, column);
    }
}
```

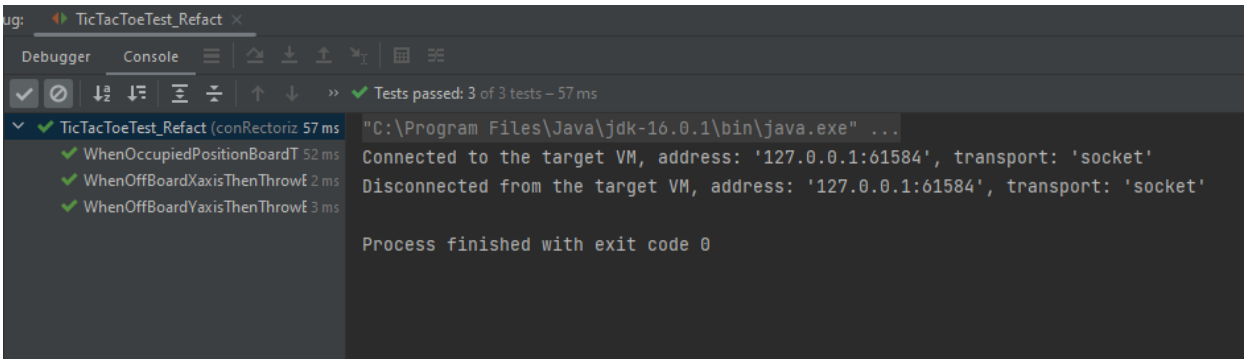
```java
private void posicion_OcupadoDesacupada(int row, int column){
    if(grid[row][column] == TicTacToe.Cell.EMPTY) {
        grid[row][column] = (turn == 'X') ? TicTacToe.Cell.CROSS : TicTacToe.Cell.NOUGHT;
        updateGameState(turn, row, column);
        turn = (turn == 'X') ? 'O' : 'X';
    }
    else{
        throw new RuntimeException("Esta posicion esta ocupada");
    }
}
```

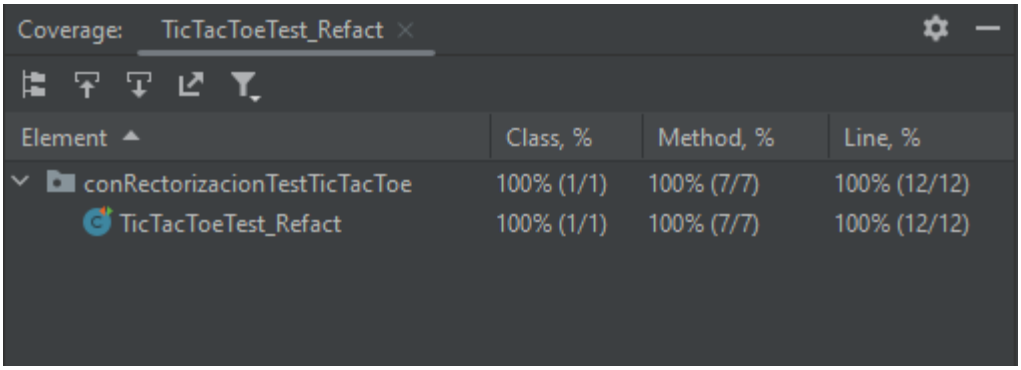
```java
private void posicion_fuerdaDelTablero(int row, int column){
    if(row >= TOTALROWS || row<0) {
        throw new RuntimeException("Saliste del tablero fuera del eje x");
    }
    if(column >= TOTALCOLUMNS || column<0){
        throw new RuntimeException("Saliste del tablero fuera del eje y");
    }
}
```

**De la misma manera mejoramos el codigo de prueba de la siguiente manera:**

```java
package conRectorizacionTestTicTacToe;

import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import conRefactorizacionTicTacToe.TicTacToe_Refact;

import static org.junit.jupiter.api.Assertions.*;
public class TicTacToeTest_Refact {
    2 usages
    private static TicTacToe_Refact tictactoe;

    @BeforeAll
    public static void init(){
        tictactoe = new TicTacToe_Refact();
    }
```

```java
@Test
public void WhenOffBoardXaxisThenThrowException() {

    Throwable exception = assertThrows(RuntimeException.class, () -> {
        tictactoe.jugar( row: 5,  column: 0);
    });
    assertEquals( expected: "Saliste del tablero fuera del eje x", exception.getMessage());
}
@Test
public void WhenOffBoardYaxisThenThrowException() {
    Throwable exception = assertThrows(RuntimeException.class, () -> {
        tictactoe.jugar( row: 0,  column: 5);
    });
    assertEquals( expected: "Saliste del tablero fuera del eje y", exception.getMessage());
}
@Test
public void WhenOccupiedPositionBoardThenThrowException() {
    tictactoe.jugar( row: 0,  column: 2);
    Throwable exception = assertThrows(RuntimeException.class, () -> {
        tictactoe.jugar( row: 0,  column: 2);
    });
    assertEquals( expected: "Esta posicion esta ocupada", exception.getMessage());
}
```




## Requisito 2 :

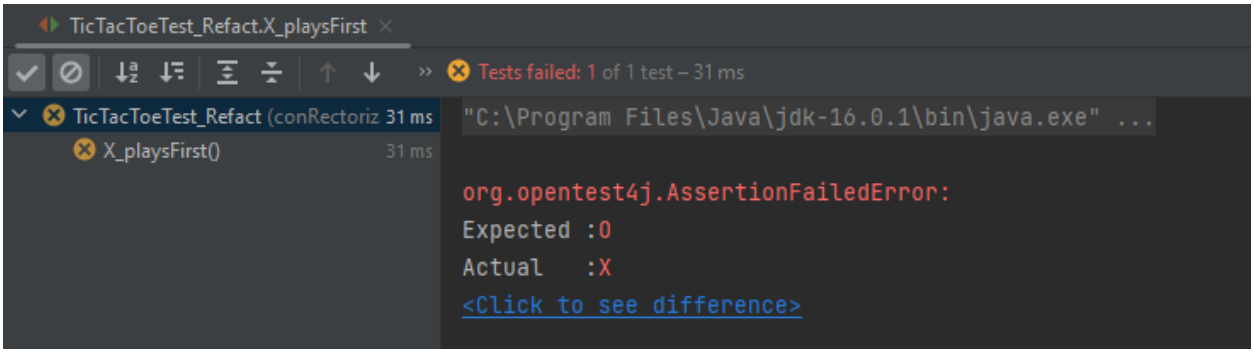
Veamos el proceso RGR para el método de prueba **X_playsFirst**

Primero escribimos la prueba sin implementar el método **proximoJugador.**

Luego, implementamos el método para hacer que la prueba pase

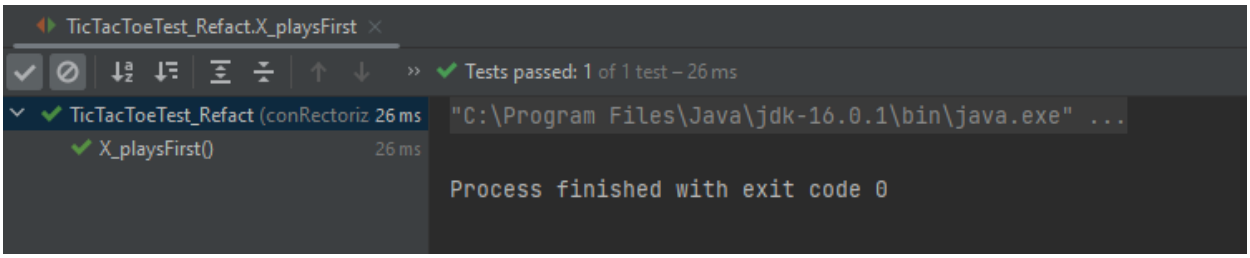
**Obs: Se ha puesto comentarios en el código para que sea más entendible**

```
@Test
public void X_playsFirst(){
    assertEquals( expected: 'O', tictactoe.proximoJugador());
}
```




```java
@Test
public void X_playsFirst(){
    //Por defecto X empieza el juego
    assertEquals( expected: 'X', tictactoe.proximoJugador());
}
```

**Finalmente, logramos pasar la prueba a verde**



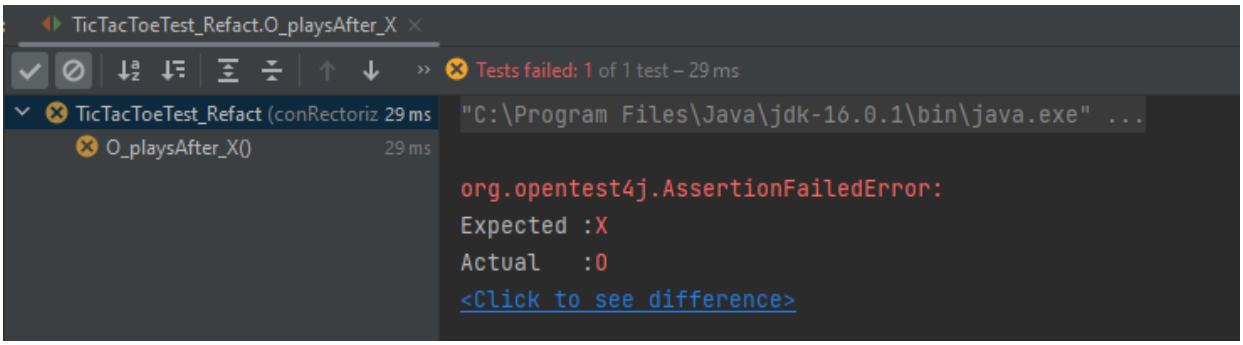

Veamos el proceso RGR para el método de prueba **O_playsAfter_X**

Primero escribimos la prueba sin implementar el método **proximoJugador.**

Luego, implementamos el método para hacer que la prueba pase

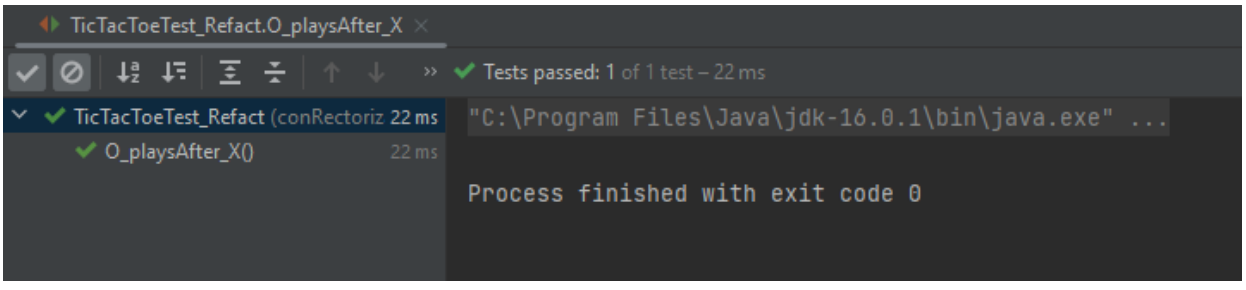
**Obs: Se ha puesto comentarios en el código para que sea más entendible**

```java
@Test
public void O_playsAfter_X(){
    tictactoe.jugar( row: 0,  column: 2);//X se coloca en la psocion (0,2)
    //El siguiente movimiento le corresponde a O
    assertEquals( expected: 'X', tictactoe.proximoJugador());
}
```




```java
@Test
public void O_playsAfter_X(){
    tictactoe.jugar( row: 0,  column: 2);//X se coloca en la psocion (0,2)
    //El siguiente movimiento le corresponde a O
    assertEquals( expected: 'O', tictactoe.proximoJugador());
}
```

**Finalmente, logramos pasar la prueba a verde**



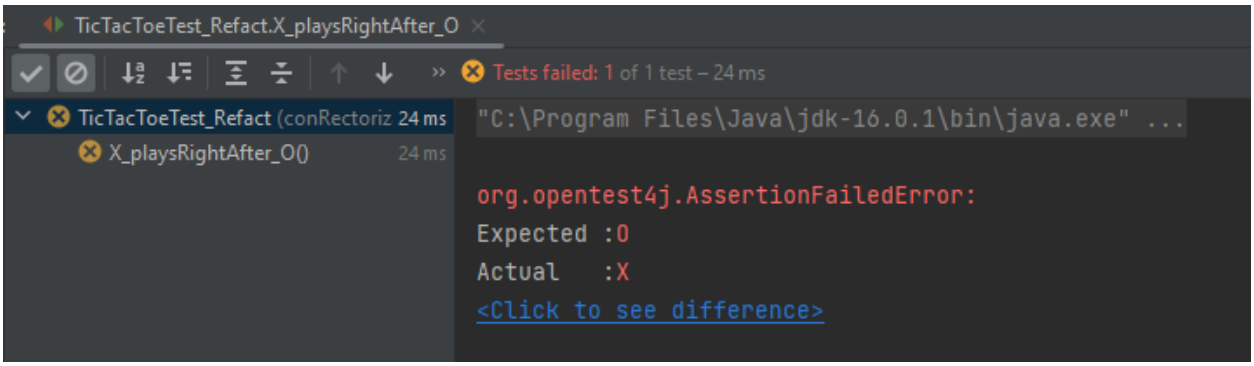

Veamos el proceso RGR para el método de prueba **X_playsRightAfter_O**

Primero escribimos la prueba sin implementar el método **proximoJugador.**

Luego, implementamos el método para hacer que la prueba pase

**Obs: Se ha puesto comentarios en el código para que sea más entendible**

```java
@Test
public void X_playsRightAfter_O(){
    tictactoe.jugar( row: 0,  column: 2);//Primero juega X
    tictactoe.jugar( row: 1,  column: 1);// Luego juega  O
    //Entonces el siguiente movimiento le toca a X
    assertEquals( expected: 'O', tictactoe.proximoJugador());
}
```




```java
@Test
public void X_playsRightAfter_O(){
    tictactoe.jugar( row: 0,  column: 2);//Primero juega X
    tictactoe.jugar( row: 1,  column: 1);// Luego juega  O
    //Entonces el siguiente movimiento le toca a X
    assertEquals( expected: 'X', tictactoe.proximoJugador());
}
```

**Finalmente, logramos pasar la prueba a verde**