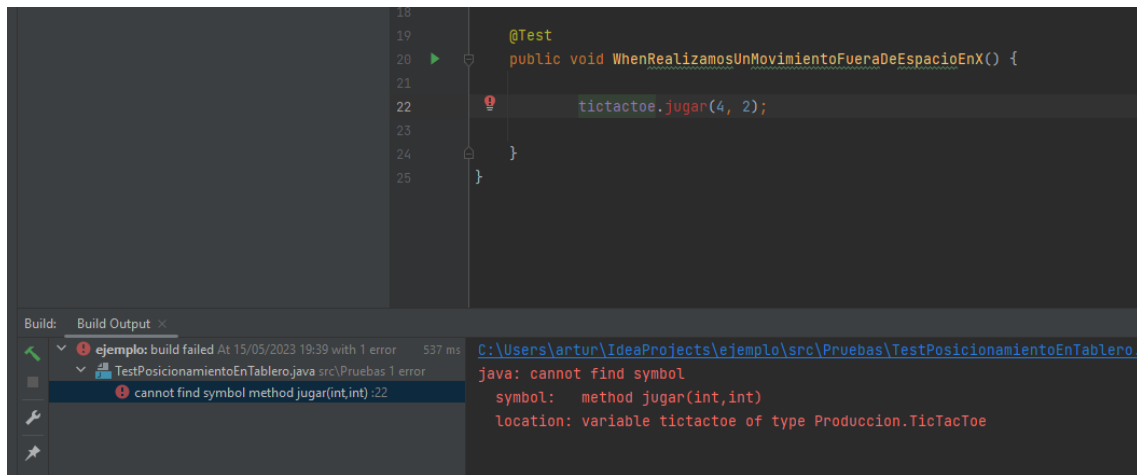


Vemos que en primera instancia aparece que la prueba falla ya que aún no se ha implementado el método jugar.



```
18
19
20 @Test
21 public void WhenRealizamosUnMovimientoFueraDeEspacioEnX() {
22     tictactoe.jugar(4, 2);
23
24 }
25 }
```

Build: Build Output < x

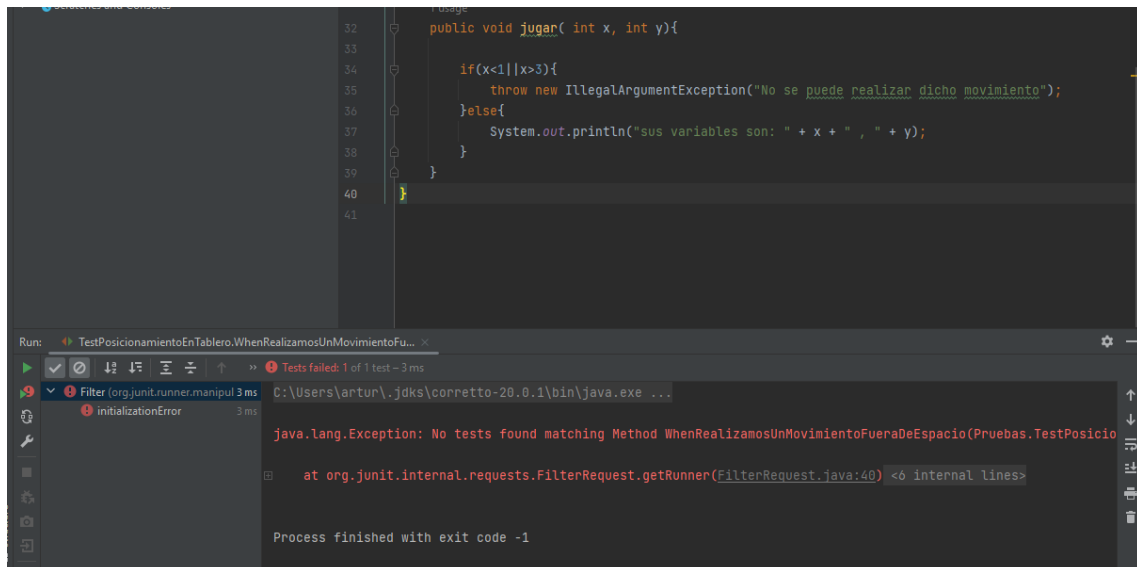
ejemplo: build failed At 15/05/2023 19:39 with 1 error 537 ms

TestPosicionamientoEnTablero.java src\Pruebas 1 error

cannot find symbol method jugar(int,int) :22

C:\Users\artur\IdeaProjects\ejemplo\src\Pruebas\TestPosicionamientoEnTablero.java:22: error: cannot find symbol  
symbol: method jugar(int,int)  
location: variable tictactoe of type Produccion.TicTacToe

Ahora, si bien se ha implementado el método jugar, sigue apareciendo que el error falla, esto debido a que no se a realizado en el Test el RuntimeException.



```
32 public void jugar( int x, int y){
33
34     if(x<1||x>3){
35         throw new IllegalArgumentException("No se puede realizar dicho movimiento");
36     }else{
37         System.out.println("sus variables son: " + x + " , " + y);
38     }
39 }
40
41 }
```

Run: TestPosicionamientoEnTablero.WhenRealizamosUnMovimientoFu... < x

Tests failed: 1 of 1 test - 3 ms

Filter (org.junit.runner.manipul 3 ms

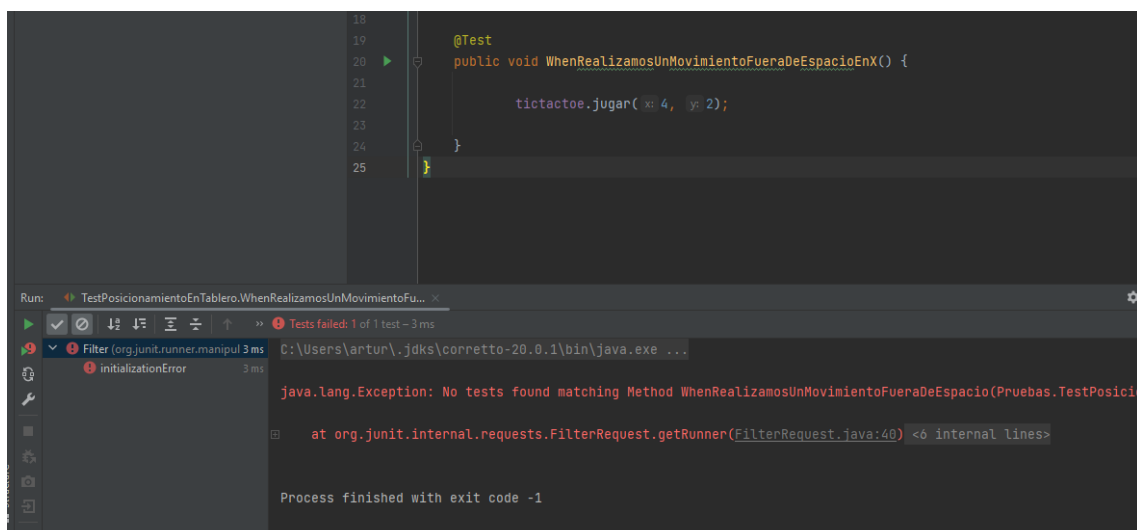
initializationError 3 ms

C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...

java.lang.Exception: No tests found matching Method WhenRealizamosUnMovimientoFueraDeEspacio(Pruebas.TestPosicio

at org.junit.internal.requests.FilterRequest.getRunner(FilterRequest.java:40) <6 internal lines>

Process finished with exit code -1



```
18
19
20 @Test
21 public void WhenRealizamosUnMovimientoFueraDeEspacioEnX() {
22     tictactoe.jugar( x: 4, y: 2);
23
24 }
25 }
```

Run: TestPosicionamientoEnTablero.WhenRealizamosUnMovimientoFu... < x

Tests failed: 1 of 1 test - 3 ms

Filter (org.junit.runner.manipul 3 ms

initializationError 3 ms

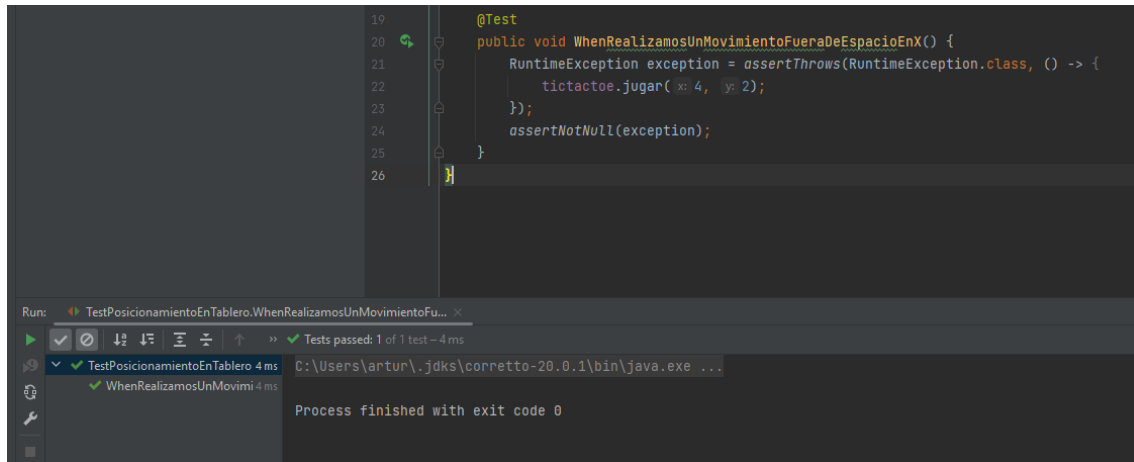
C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...

java.lang.Exception: No tests found matching Method WhenRealizamosUnMovimientoFueraDeEspacio(Pruebas.TestPosicio

at org.junit.internal.requests.FilterRequest.getRunner(FilterRequest.java:40) <6 internal lines>

Process finished with exit code -1

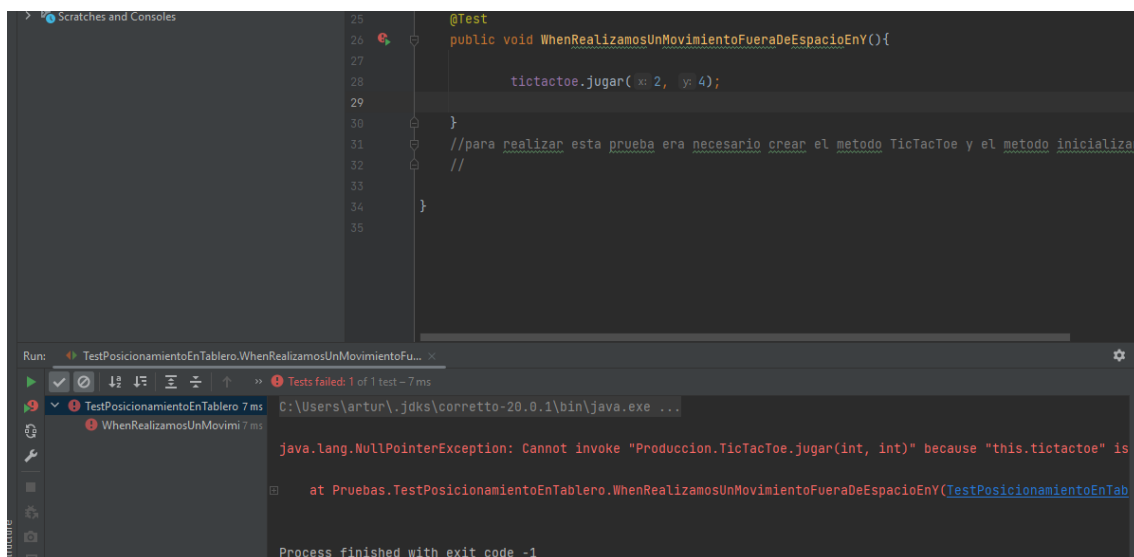
Ahora vemos que la prueba no falla ya que se implementó la excepción.



```
19
20 @Test
21 public void WhenRealizamosUnMovimientoFueraDeEspacioEnX() {
22     RuntimeException exception = assertThrows(RuntimeException.class, () -> {
23         tictactoe.jugar( 4,  2);
24     });
25     assertNotNull(exception);
26 }
```

Run: TestPosicionamientoEnTablero.WhenRealizamosUnMovimientoFu...  
Tests passed: 1 of 1 test - 4 ms  
C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...  
Process finished with exit code 0

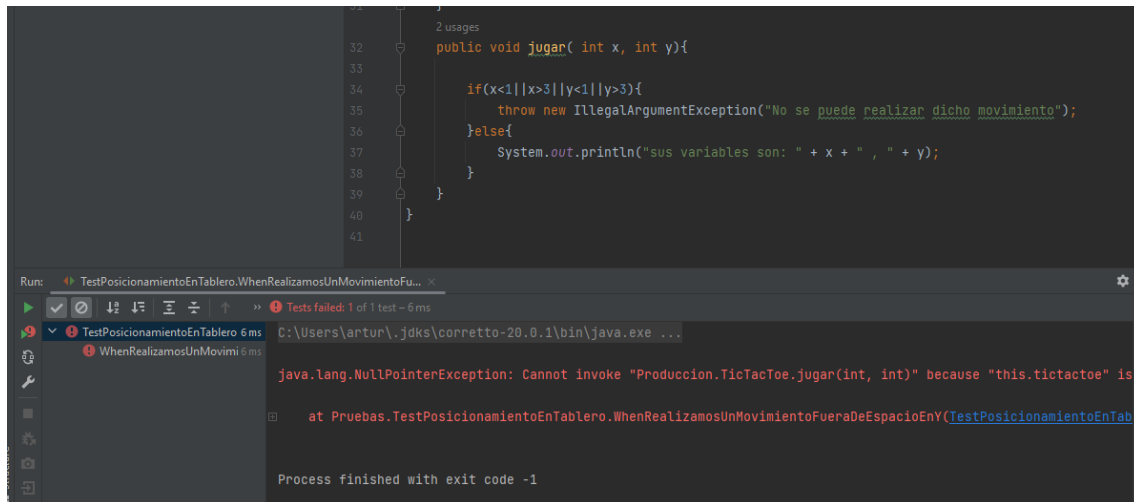
Posteriormente ahora implementamos para Y, similarmente como el ejercicio anterior iniciamos primero sin la implementación de Y. El cual sale como prueba no fallida, lo cual es una falsa prueba positivo debido a que aun no se ha implementado restricciones para Y.



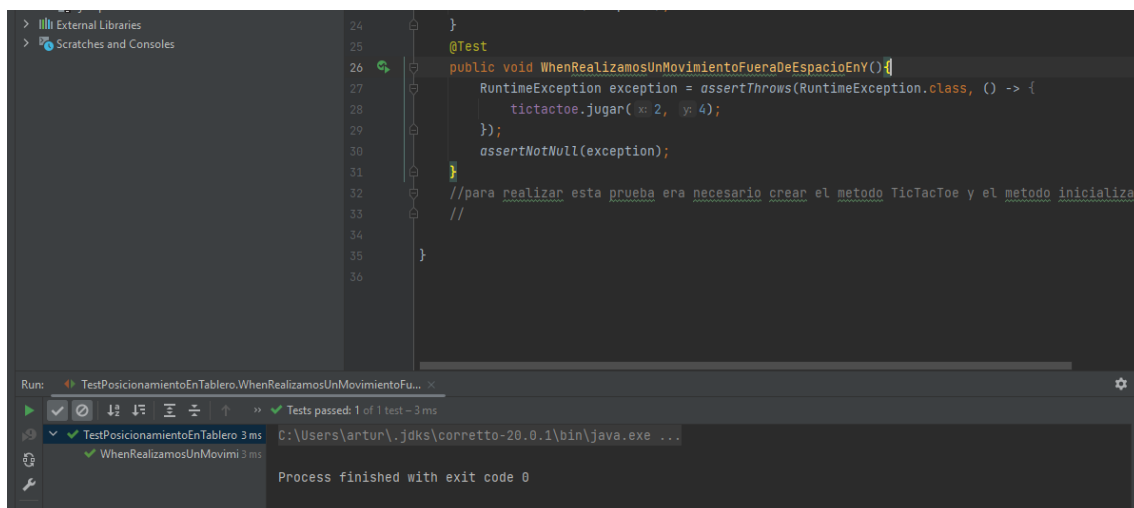
```
25
26 @Test
27 public void WhenRealizamosUnMovimientoFueraDeEspacioEnY(){
28     tictactoe.jugar( 2,  4);
29 }
30
31 //para realizar esta prueba era necesario crear el metodo TicTacToe y el metodo inicializar
32 //
33
34
35 }
```

Run: TestPosicionamientoEnTablero.WhenRealizamosUnMovimientoFu...  
Tests failed: 1 of 1 test - 7 ms  
C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...  
java.lang.NullPointerException: Cannot invoke "Produccion.TicTacToe.jugar(int, int)" because "this.tictactoe" is null  
at Pruebas.TestPosicionamientoEnTablero.WhenRealizamosUnMovimientoFueraDeEspacioEnY(TestPosicionamientoEnTab...  
Process finished with exit code -1

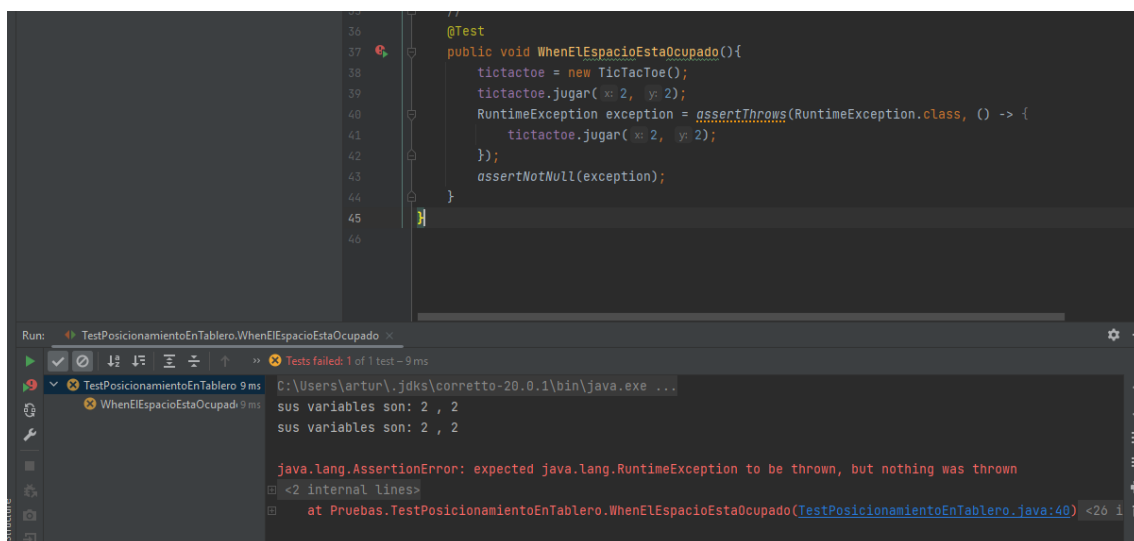
Similar al paso anterior, y sin modificar la prueba vemos que nos lanza una prueba fallida, esto debido a que falta agregar RunTimeException



Una vez implementado, entonces la prueba aparece en verde



Para el tercer requisito, se va primero realizar la prueba para ver si es que el espacio está ocupado, posteriormente se va a implementar su funcionalidad.



Implementamos para una prueba verde

```
32 public void jugar( int x, int y){
33
34     if(x<1||x>3||y<1||y>3||tablero[x][y]!='v'){
35         throw new IllegalArgumentException("No se puede realizar dicho movimiento");
36     }else{
37         tablero[x][y]='x';
38         System.out.println("sus variables son: " + x + " , " + y);
39     }
40 }
41 }
```

Run: TestPosicionamientoEnTablero.WhenElEspacioEstaOcupado

Tests passed: 1 of 1 test - 7 ms

TestPosicionamientoEnTablero 7 ms C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...

WhenElEspacioEstaOcupado 7 ms

sus variables son: 2 , 2

Process finished with exit code 0

```
36 //
37 @Test
38 public void WhenElEspacioEstaOcupado(){
39     tictactoe = new TicTacToe();
40     tictactoe.jugar( x: 2, y: 2);
41     RuntimeException exception = assertThrows(RuntimeException.class, () -> {
42         tictactoe.jugar( x: 2, y: 2);
43     });
44     assertNotNull(exception);
45 }
46 }
```

Run: TestPosicionamientoEnTablero.WhenElEspacioEstaOcupado

Tests passed: 1 of 1 test - 7 ms

TestPosicionamientoEnTablero 7 ms C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...

WhenElEspacioEstaOcupado 7 ms

sus variables son: 2 , 2

Process finished with exit code 0

Ahora Agregaremos el jugar para dos jugadores

Agregamos la prueba cuando X juega primero

```
45 }
46 @Test
47 public void WhenXJuegaPrimero(){
48     tictactoe = new TicTacToe();
49     assertEquals( expected: 'x', tictactoe.proximoJugador());
50 }
51 }
52 }
```

Build: Build Output

parcial2: build failed At 15/05/2023 20:2535 ms

TestPosicionamientoEnTablero.java src\Prue

cannot find symbol method proximoJug

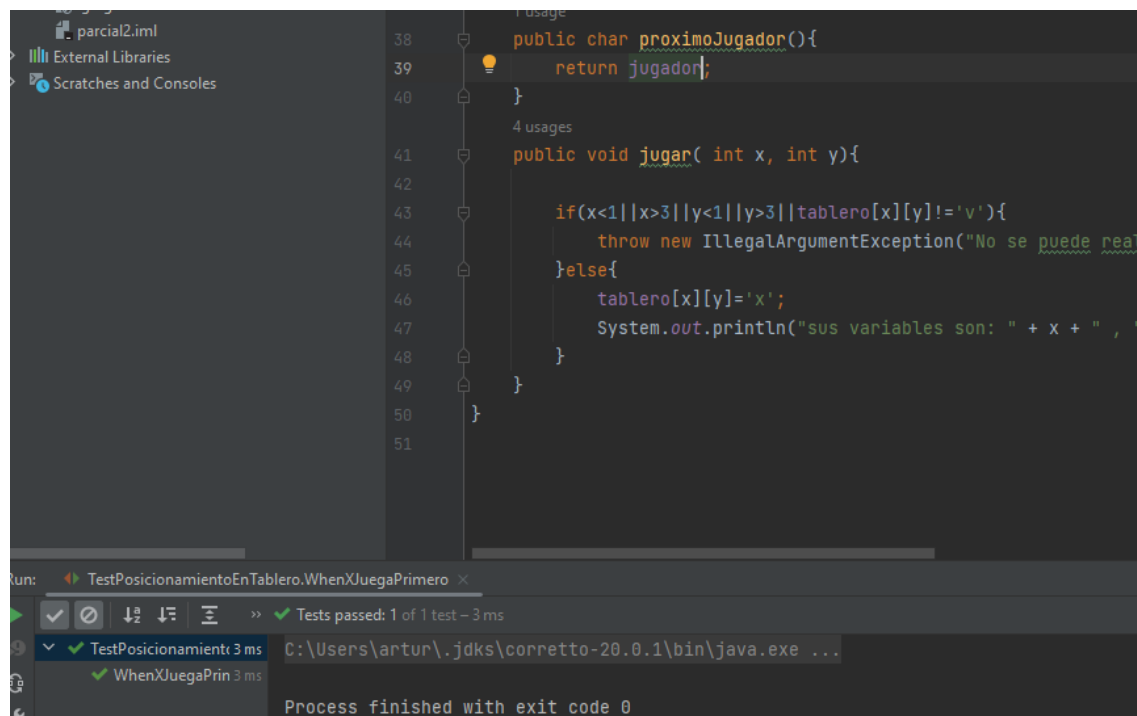
C:\Users\artur\OneDrive\Escritorio\examen\_parcial\_desarrollo\parcial2

java: cannot find symbol

symbol: method proximoJugador()

location: variable tictactoe of type Produccion.TicTacToe

## Implementacion con prueba en verde



The screenshot shows an IDE with a Java class containing two methods: `proximoJugador()` and `jugar()`. The `jugar()` method includes a validation check for board boundaries and a `throw new IllegalArgumentException` if the move is invalid. Below the code, the test runner shows a single test, `TestPosicionamientoEnTablero.WhenXJuegaPrimero`, which passed successfully. The output window displays the command executed: `C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...` and the message: `Process finished with exit code 0`.

```
38 public char proximoJugador(){
39     return jugador;
40 }
41
42 4 usages
43 public void jugar( int x, int y){
44     if(x<1||x>3||y<1||y>3||tablero[x][y]!='v'){
45         throw new IllegalArgumentException("No se puede real
46     }else{
47         tablero[x][y]='x';
48         System.out.println("sus variables son: " + x + " , "
49     }
50 }
51 }
```

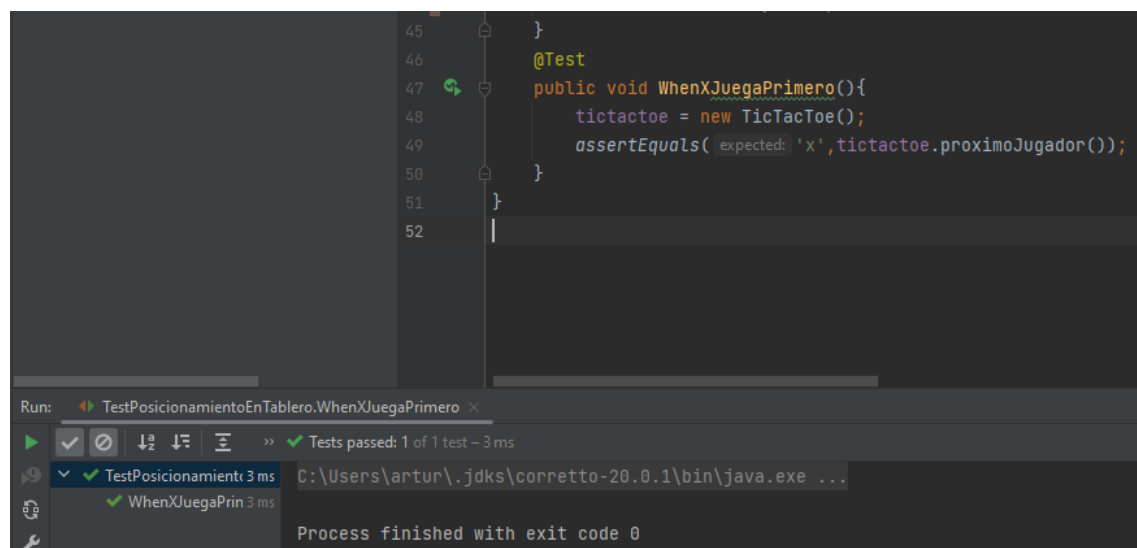
Run: TestPosicionamientoEnTablero.WhenXJuegaPrimero

Tests passed: 1 of 1 test - 3 ms

TestPosicionamientoEnTablero.WhenXJuegaPrimero 3 ms

C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...

Process finished with exit code 0



The screenshot shows the same IDE with a JUnit test method, `WhenXJuegaPrimero()`, added to the class. The test creates a `TicTacToe` object and asserts that `proximoJugador()` returns 'x'. The test runner shows the test passed. The output window shows the command: `C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...` and the message: `Process finished with exit code 0`.

```
45 }
46
47 @Test
48 public void WhenXJuegaPrimero(){
49     tictactoe = new TicTacToe();
50     assertEquals( expected: 'x', tictactoe.proximoJugador());
51 }
52 }
```

Run: TestPosicionamientoEnTablero.WhenXJuegaPrimero

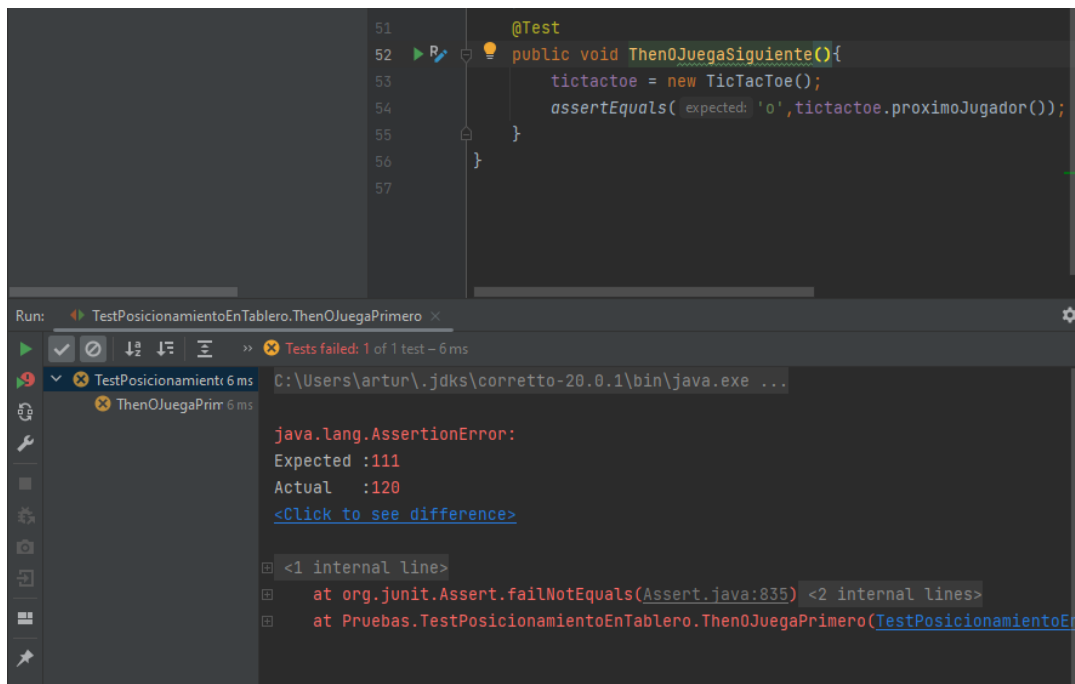
Tests passed: 1 of 1 test - 3 ms

TestPosicionamientoEnTablero.WhenXJuegaPrimero 3 ms

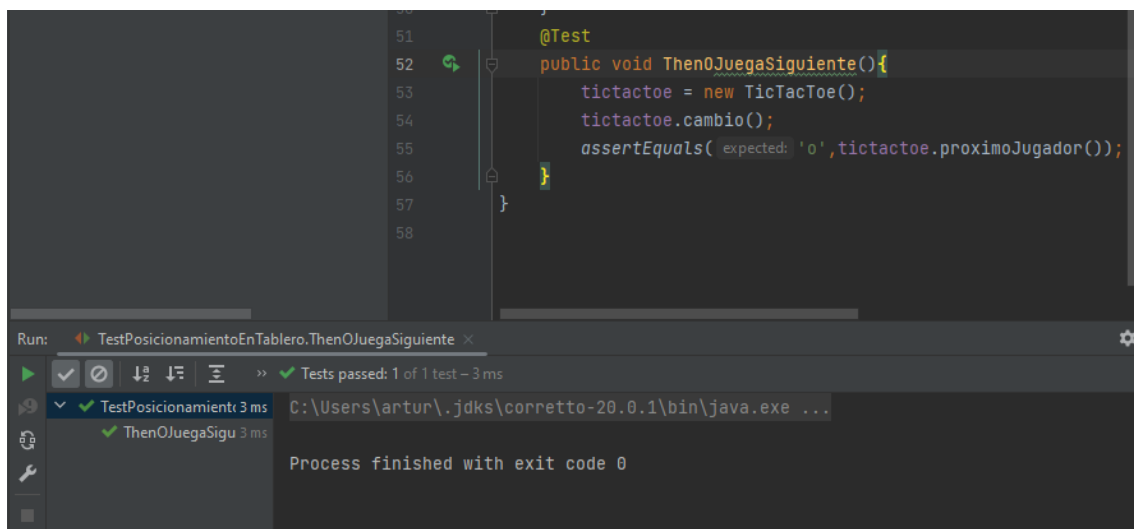
C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...

Process finished with exit code 0

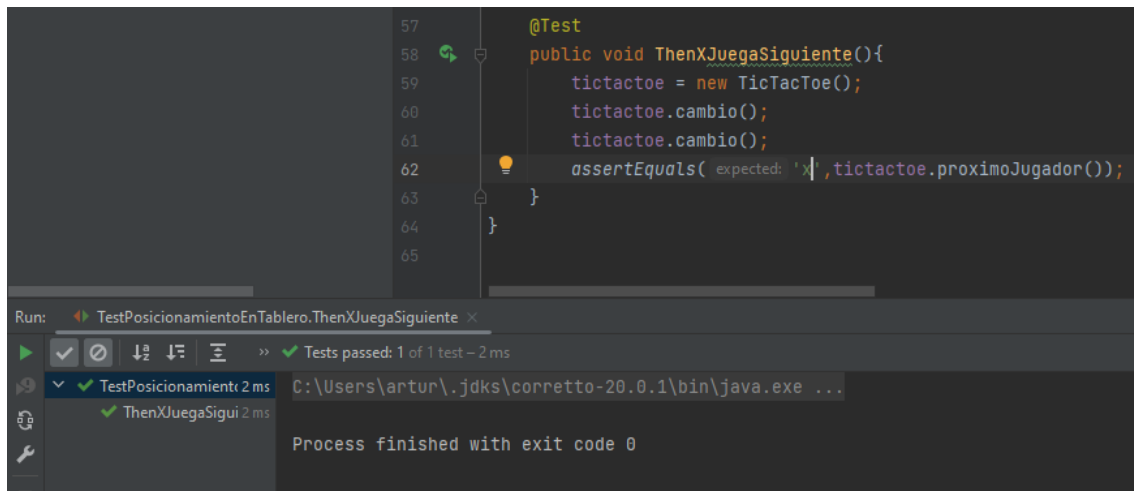
Prueba cuando o juega justo después de x



Sale en rojo debido a que no se agrego el cambio, vamos a implementarlo



De modo similar realizaremos la prueba para X juega justo después de O agregando otro `tictactoe.cambio()`. El cual resultará en una prueba verde.



```
57
58
59
60
61
62
63
64
65
```

```
@Test
public void ThenXJuegaSiguiente(){
    tictactoe = new TicTacToe();
    tictactoe.cambio();
    tictactoe.cambio();
    assertEquals( expected: 'X', tictactoe.proximoJugador());
}
```

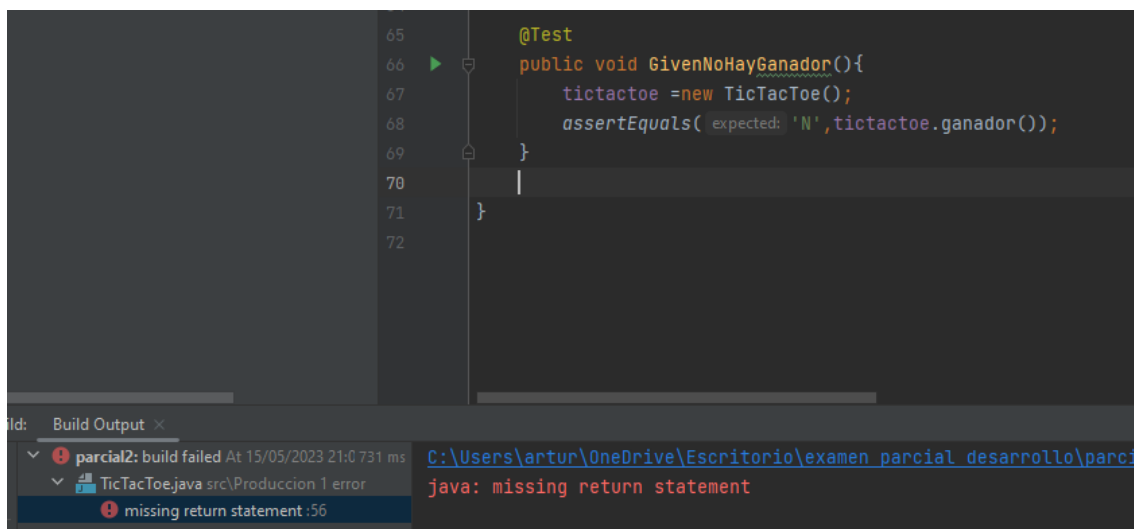
Run: TestPosicionamientoEnTablero.ThenXJuegaSiguiente

Tests passed: 1 of 1 test - 2 ms

TestPosicionamientoEnTablero.ThenXJuegaSiguiente 2 ms

Process finished with exit code 0

Cuando por defecto no hay ganador



```
65
66
67
68
69
70
71
72
```

```
@Test
public void GivenNoHayGanador(){
    tictactoe = new TicTacToe();
    assertEquals( expected: 'N', tictactoe.ganador());
}
```

Build Output

parcial2: build failed At 15/05/2023 21:07:31 ms

TicTacToe.java src\Produccion 1 error

missing return statement :56

java: missing return statement

El método ganador aun no esta definido por lo que se realizara según las siguientes pruebas

## Ganador en horizontal

```
69     }
70     @Test
71     public void WhenGanadorHorizontal(){
72         tictactoe = new TicTacToe();
73         tictactoe.movimiento( x: 2, y: 0);
74         tictactoe.movimiento( x: 1, y: 1);
75         tictactoe.movimiento( x: 2, y: 1);
76         tictactoe.movimiento( x: 0, y: 2);
77         tictactoe.movimiento( x: 2, y: 2);
78         assertEquals( expected: 'x', tictactoe.ganador());
79     }
80
81 }
82
```

Build: Build Output x

parcial2: build failed At 15/05/2023 21:16 with 567 ms

TicTacToe.java src\Produccion 1 error

missing return statement :56

C:\Users\artur\OneDrive\Escritorio\examen parcial desarrollo\parcial2\src\Produccion\TicTacToe.java:56: error: missing return statement

Implementando la función ganador, vemos que si devuelve el valor de N para la prueba anterior a esta.

La implementación es la siguiente

```
public char ganador(){
    //horizontal
    for(int i=0;i<3;i++){
        if(tablero[i][0]==jugador && tablero[i][1]==jugador && tablero[i][2]==jugador ){
            terminar=true;
            System.out.println("El jugador " + jugador + " es el ganador");
            return jugador;
        }
    }
    return 'N';
}
```

```
65     @Test
66     public void GivenNoHayGanador(){
67         tictactoe =new TicTacToe();
68         assertEquals( expected: 'N', tictactoe.ganador());
69     }
70     @Test
71     public void WhenGanadorHorizontal(){
72         tictactoe = new TicTacToe();
73         tictactoe.movimiento( x: 0, y: 0);
74         tictactoe.movimiento( x: 2, y: 1);
75         tictactoe.movimiento( x: 1, y: 0);

```

Run: TestPosicionamientoEnTablero.GivenNoHayGanador x

Tests passed: 1 of 1 test - 3 ms

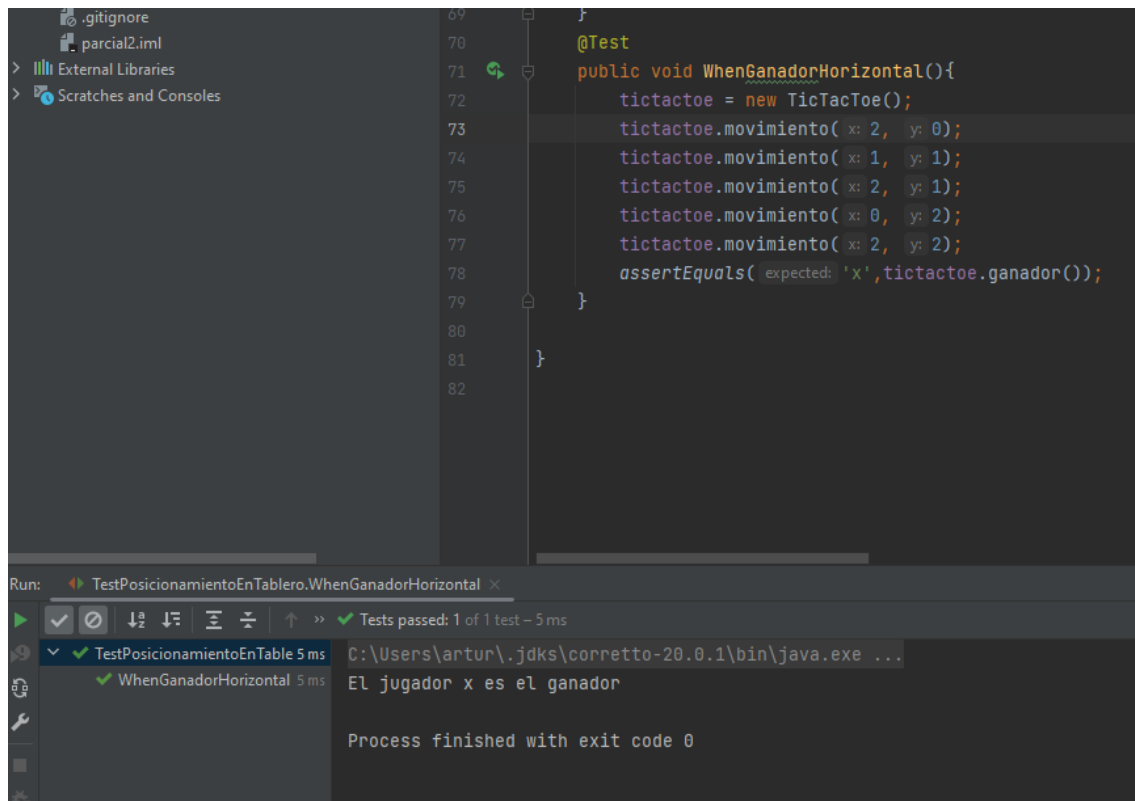
TestPosicionamientoEnTablero.GivenNoHayGanador 3 ms

C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...

Process finished with exit code 0



Para la prueba actual en la que trabajamos tenemos una prueba en verde



```
69 }
70
71 @Test
72 public void WhenGanadorHorizontal(){
73     tictactoe = new TicTacToe();
74     tictactoe.movimiento( x: 2, y: 0);
75     tictactoe.movimiento( x: 1, y: 1);
76     tictactoe.movimiento( x: 2, y: 1);
77     tictactoe.movimiento( x: 0, y: 2);
78     tictactoe.movimiento( x: 2, y: 2);
79     assertEquals( expected: 'X', tictactoe.ganador());
80 }
81
82 }
```

Run: TestPosicionamientoEnTablero.WhenGanadorHorizontal x

Tests passed: 1 of 1 test - 5 ms

TestPosicionamientoEnTable 5 ms

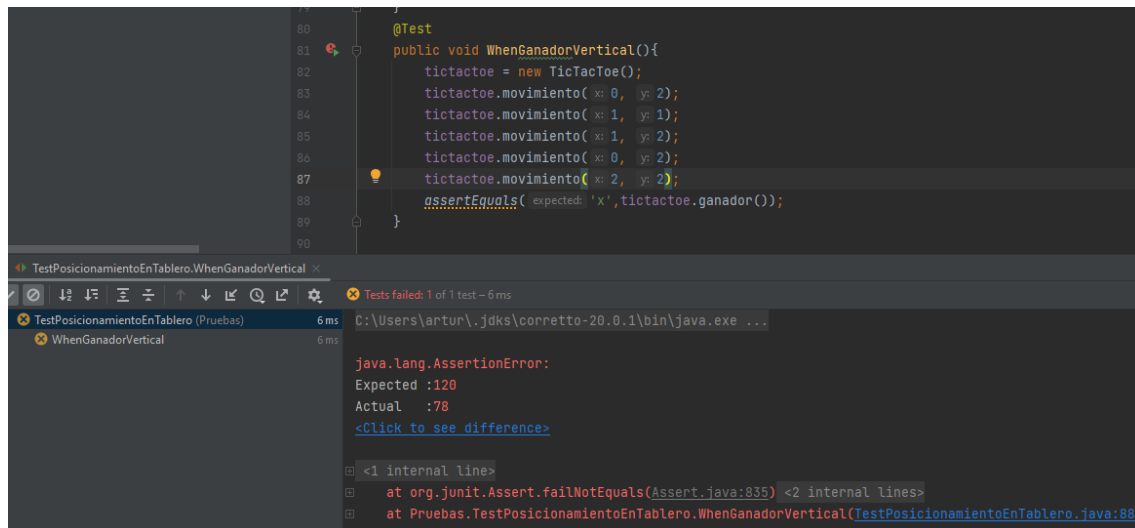
WhenGanadorHorizontal 5 ms

C:\Users\artur\jdk\corretto-20.0.1\bin\java.exe ...

El jugador x es el ganador

Process finished with exit code 0

Realizaremos las pruebas para una ganador en vertical



```
80 }
81
82 @Test
83 public void WhenGanadorVertical(){
84     tictactoe = new TicTacToe();
85     tictactoe.movimiento( x: 0, y: 2);
86     tictactoe.movimiento( x: 1, y: 1);
87     tictactoe.movimiento( x: 1, y: 2);
88     tictactoe.movimiento( x: 0, y: 2);
89     tictactoe.movimiento( x: 2, y: 2);
90     assertEquals( expected: 'X', tictactoe.ganador());
91 }
```

TestPosicionamientoEnTablero.WhenGanadorVertical x

Tests failed: 1 of 1 test - 6 ms

TestPosicionamientoEnTablero (Pruebas) 6 ms

WhenGanadorVertical 6 ms

C:\Users\artur\jdk\corretto-20.0.1\bin\java.exe ...

java.lang.AssertionError:

Expected :120

Actual :78

<Click to see difference>

<1 internal line>

at org.junit.Assert.failNotEquals(Assert.java:835) <2 internal lines>

at Pruebas.TestPosicionamientoEnTablero.WhenGanadorVertical(TestPosicionamientoEnTablero.java:88)

Ahora realizaremos la implementación

```
//vertical
for(int i=0;i<3;i++){
    if(tablero[0][i]==jugador && tablero[1][i]==jugador && tablero[2][i]==jugador ){
        terminar=true;
        System.out.println("El jugador " + jugador + " es el ganador");
        return jugador;
    }
}
return 'N';
```

Vemos que la prueba ahora se realiza de manera satisfactoria

```
80
81  @Test
82  public void WhenGanadorVertical(){
83      tictactoe = new TicTacToe();
84      tictactoe.movimiento( x: 0, y: 2);
85      tictactoe.movimiento( x: 1, y: 1);
86      tictactoe.movimiento( x: 1, y: 2);
87      tictactoe.movimiento( x: 0, y: 2);
88      tictactoe.movimiento( x: 2, y: 2);
89      assertEquals( expected: 'X', tictactoe.ganador());
90  }
```

TestPosicionamientoEnTablero.WhenGanadorVertical

Tests passed: 1 of 1 test - 5 ms

TestPosicionamientoEnTablero (Pruebas) 5 ms C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...

WhenGanadorVertical 5 ms El jugador x es el ganador

Process finished with exit code 0

Prueba de para ganador con diagonal superior izquierda a inferior derecha

```
91  @Test
92  public void WhenGanadorDiagonalDeSuperiorIzquierdaAInferiorDerecha(){
93      tictactoe = new TicTacToe();
94      tictactoe.movimiento( x: 0, y: 0);
95      tictactoe.movimiento( x: 1, y: 0);
96      tictactoe.movimiento( x: 1, y: 1);
97      tictactoe.movimiento( x: 0, y: 2);
98      tictactoe.movimiento( x: 2, y: 2);
99      assertEquals( expected: 'X', tictactoe.ganador());
100  }
```

Run: TestPosicionamientoEnTablero.WhenGanadorDiagonalDeSuperiorIzquierdaAInferiorDerecha

Tests failed: 1 of 1 test - 6 ms

TestPosicionamientoEnTablero (Pruebas) 6 ms C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...

WhenGanadorDiagonalDeSuperiorIzquierdaAInferiorDerecha 6 ms

java.lang.AssertionError:  
Expected :120  
Actual :78  
<Click to see difference>

<1 internal line>  
at org.junit.Assert.failNotEquals(Assert.java:835) <2 internal lines>  
at Pruebas.TestPosicionamientoEnTablero.WhenGanadorDiagonalDeSuperiorIzquierdaAInferiorDerecha()

Una vez realizada la implementación se tiene una prueba green.

```
//DiagonalDeSuperiorIzquierdaAInferiorDerecha
if (tablero[0][0] == jugador && tablero[1][1] == jugador && tablero[2][2] == jugador) {
    terminar = true;
    System.out.println("El jugador " + jugador + " es el ganador");
    return jugador;
}

return 'N';
```

```
90 @Test
91 public void WhenGanadorDiagonalDeSuperiorIzquierdaAInferiorDerecha(){
92     tictactoe = new TicTacToe();
93     tictactoe.movimiento(x: 0, y: 0);
94     tictactoe.movimiento(x: 1, y: 0);
95     tictactoe.movimiento(x: 1, y: 1);
96     tictactoe.movimiento(x: 0, y: 2);
97     tictactoe.movimiento(x: 2, y: 2);
98     assertEquals( expected: 'X', tictactoe.ganador());
99 }
100
```

TestPosicionamientoEnTablero.WhenGanadorDiagonalDeSuperiorIzquierdaAInferiorDerecha

Tests passed: 1 of 1 test - 5 ms

TestPosicionamientoEnTablero (Pruebas) 5 ms C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...

WhenGanadorDiagonalDeSuperiorIzquierdaAInferiorDerecha 5 ms El jugador x es el ganador

Process finished with exit code 0

Ahora para cuando el jugador gana cuando toda la línea diagonal desde la parte inferior izquierda hasta la parte superior derecha

```
100 }
101 @Test
102 public void WhenGanadorDiagonalDeInferiorIzquierdaASuperiorDerecha(){
103     tictactoe = new TicTacToe();
104     tictactoe.movimiento(x: 0, y: 2);
105     tictactoe.movimiento(x: 1, y: 0);
106     tictactoe.movimiento(x: 1, y: 1);
107     tictactoe.movimiento(x: 2, y: 2);
108     tictactoe.movimiento(x: 2, y: 0);
109     assertEquals( expected: 'X', tictactoe.ganador());
110 }
111 }
112
```

Run: TestPosicionamientoEnTablero.WhenGanadorDiagonalDeInferiorIzquierdaASuperiorDerecha

Tests failed: 1 of 1 test - 8 ms

TestPosicionamientoEnTablero (Pruebas) 8 ms C:\Users\artur\.jdk\corretto-20.0.1\bin\java.exe ...

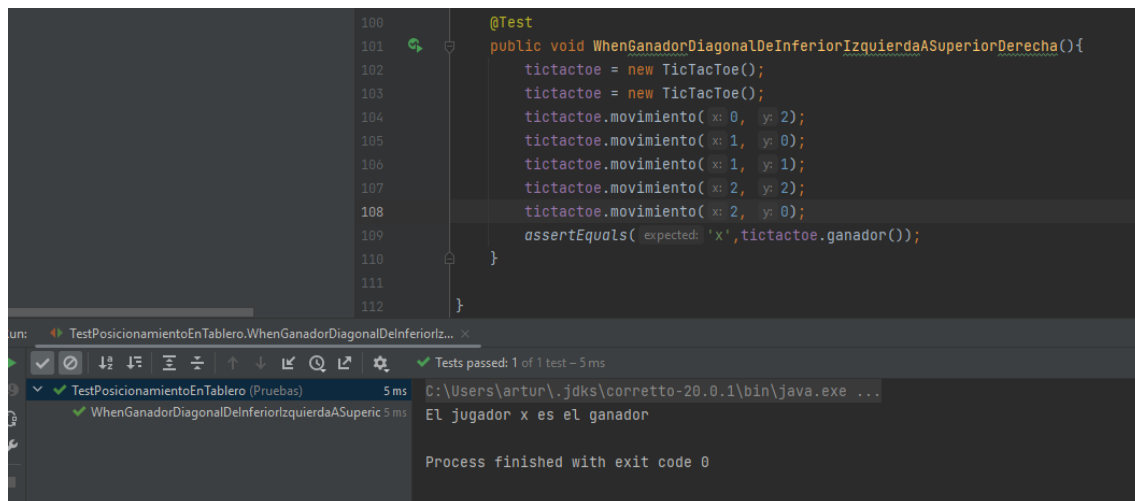
WhenGanadorDiagonalDeInferiorIzquierdaASuperiorDerecha 8 ms

java.lang.AssertionError:  
Expected :120  
Actual :78  
<Click to see difference>

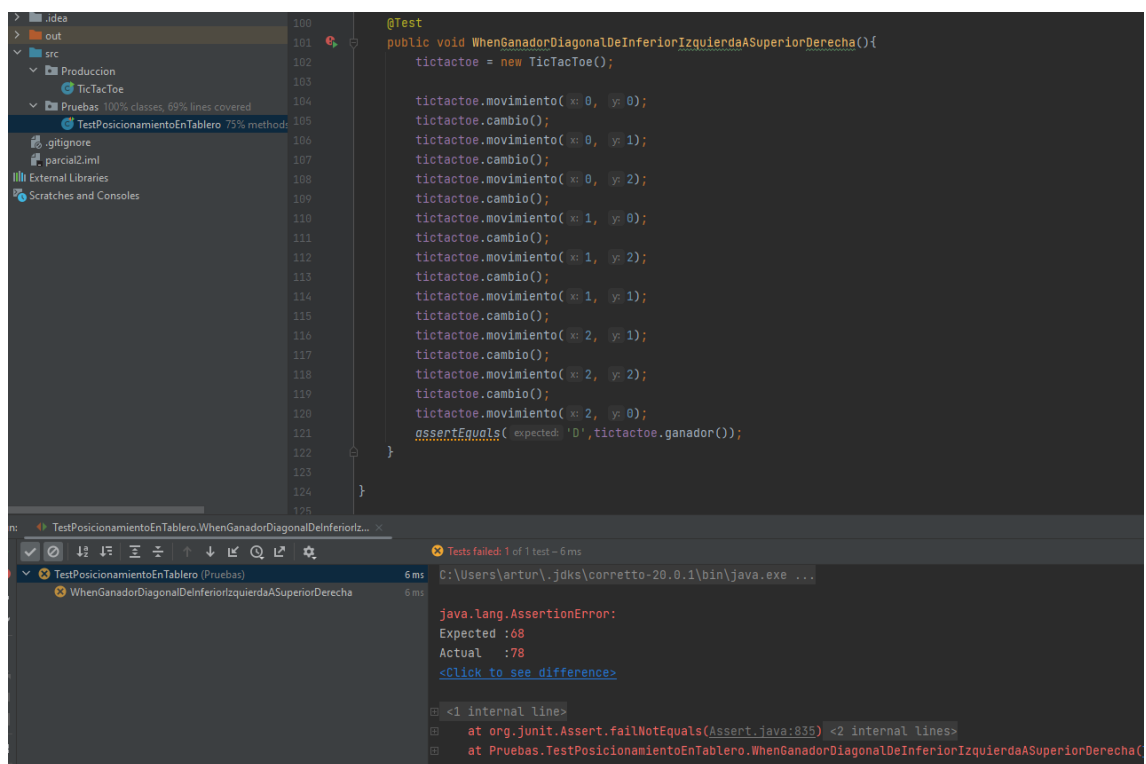
<1 internal line>  
at org.junit.Assert.failNotEquals(Assert.java:835) <2 internal lines>  
at Pruebas.TestPosicionamientoEnTablero.WhenGanadorDiagonalDeInferiorIzquierdaASuperiorDerecha(Te

Realizando la implementación se tiene una prueba en verde.

```
//DiagonalDeInferiorIzquierdaASuperiorDerecha
if (tablero[0][2] == jugador && tablero[1][1] == jugador && tablero[2][0] == jugador) {
    terminar = true;
    System.out.println("El jugador " + jugador + " es el ganador");
    return jugador;
}
return 'N';
}
```



Para cuando se realiza un empate vamos a realizar una prueba.



Ahora procedemos con su implementación

```

        boolean empate=true;
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
                if(tablero[i][j]=='v'){
                    empate=false;
                    break;
                }
            }
        }

        if(!empate){
            break;
        }

        if (empate){
            terminar=true;
            System.out.println("Empate");
            return 'D';
        }

        return 'N';
    }
}

```

Nos da una prueba en verde

The screenshot shows an IDE with a project structure on the left, a Java test file in the center, and a run console at the bottom.

**Project Structure:**

- out
- src
  - Produccion
  - TicTacToe
  - Pruebas (100% classes, 69% lines covered)
    - TestPosicionamientoEnTablero (75% methods covered)
  - .gitignore
  - parcial2.iml
  - External Libraries
  - Scratches and Consoles

**Test Code (TestPosicionamientoEnTablero.java):**

```

100 @rest
101 public void WhenGanadorDiagonalDeInferiorIzquierdaASuperiorDerecha(){
102     tictactoe = new TicTacToe();
103
104     tictactoe.movimiento(x: 0, y: 0);
105     tictactoe.cambio();
106     tictactoe.movimiento(x: 0, y: 1);
107     tictactoe.cambio();
108     tictactoe.movimiento(x: 0, y: 2);
109     tictactoe.cambio();
110     tictactoe.movimiento(x: 1, y: 0);
111     tictactoe.cambio();
112     tictactoe.movimiento(x: 1, y: 2);
113     tictactoe.cambio();
114     tictactoe.movimiento(x: 1, y: 1);
115     tictactoe.cambio();
116     tictactoe.movimiento(x: 2, y: 1);
117     tictactoe.cambio();
118     tictactoe.movimiento(x: 2, y: 2);
119     tictactoe.cambio();
120     tictactoe.movimiento(x: 2, y: 0);
121     assertEquals( expected: 'D', tictactoe.ganador());
122 }
123
124
125

```

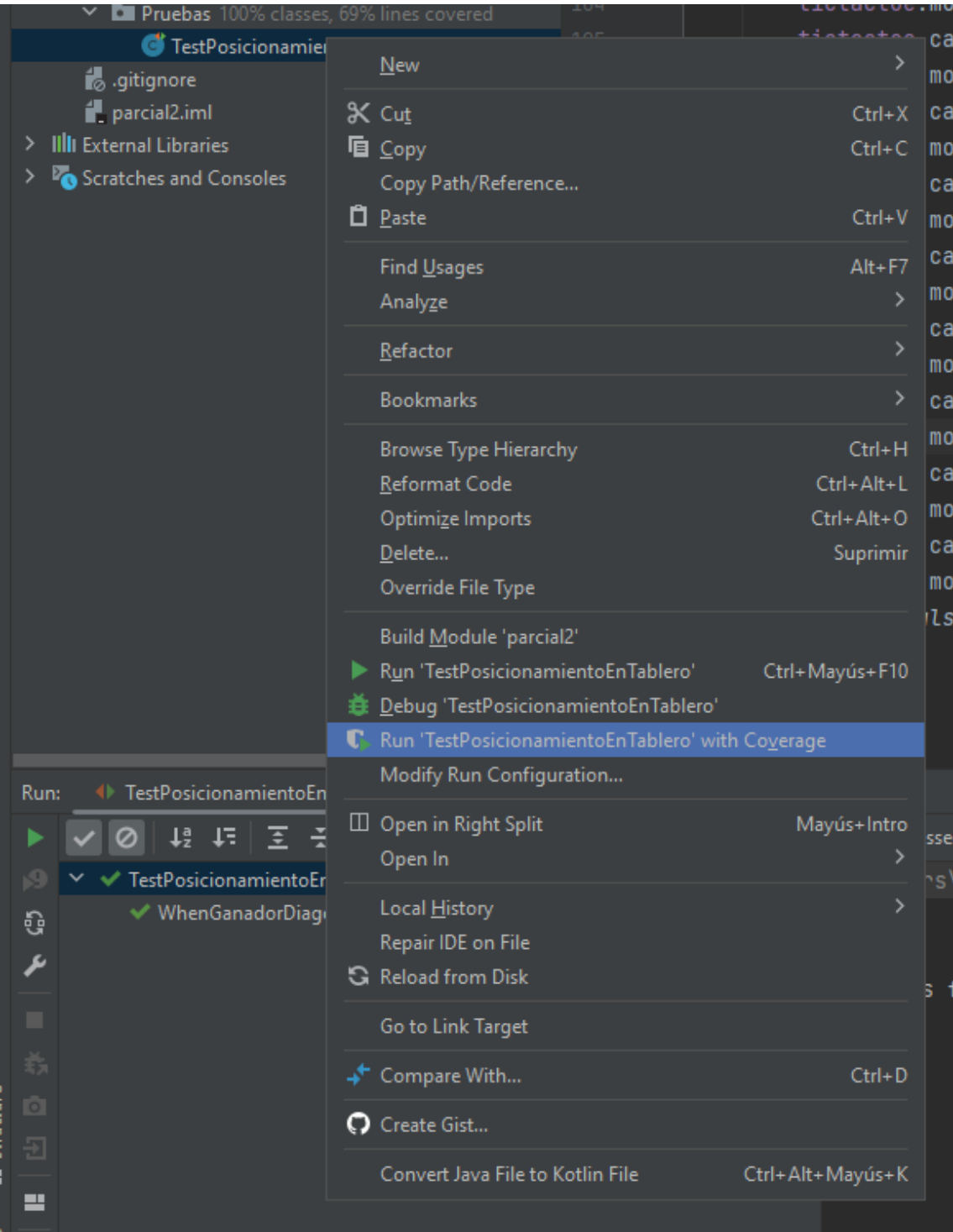
**Run Console:**

```

Run: TestPosicionamientoEnTablero.WhenGanadorDiagonalDeInferiorIzquierdaASuperiorDerecha...
Tests passed: 1 of 1 test - 2 ms
TestPosicionamientoEn Tablero (Pruebas) 2 ms
  WhenGanadorDiagonalDeInferiorIzquierdaASuperiorDerecha 2 ms
    Empate
Process finished with exit code 0

```

Para ver en mayor medida el porcentaje de utilización de nuestras pruebas podemos utilizar la siguiente opción Run ---- with Coverage, hacemos clic y visualizamos



Con lo cual obtenemos los siguientes parámetros

Coverage: Pruebas in parcial2			
Element	Class, %	Method, %	Line, %
Pruebas	100% (1/1)	88% (16/18)	94% (64/68)
TestF	100% (1/1)	88% (16/18)	94% (64/68)