

# Del Proyecto SOS

Usare el Sprint 3 para aplicar el proceso RGR :

## Para la clase TestFinishSimpleGame

Veamos el proceso RGR para el método de prueba **SOS\_WinBlue**

**Recordemos que primero escribimos las pruebas y después escribir el código fuente que pase la prueba satisfactoriamente**

**Obs:** La matriz info guarda la información del juego, tras cada movimiento en el tablero almacena 1 si se colocó "S" y 0 si se colocó "O". En un principio cuando el tablero esta vacío, es decir cuando ningún jugador hizo un movimiento se le asigna -1. Por ello en el bucle que vemos en la imagen hacemos que `info[i][j] = -1`.

**Obs:** Por defecto el blue player juega primero, por ello en la imagen `info[2][2] = 1` significa que el blue player coloco la "S" en la posición (2,2), luego el red player coloco la "O" en la posición (1,1) y finalmente el blue player coloco la "S" en la posición (0,0) del tablero.

**Obs:** Slots en una matriz de botones y slot es simplemente un botón, por ello `Slots[i][j] = slot`

**Obs:** el método SOS retorna la cantidad de SOS que hay en el tablero y lo pinta del color según el jugador que formo el SOS.

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

```
@Test
public void SOS_WinBlue(){
    int[][] info = new int[3][3];
    JButton[][] Slots = new JButton[3][3];

    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            info[i][j]= -1;
            JButton slot = new JButton();
            Slots[i][j] = slot;
        }
    }
    info[2][2]=1;
    info[1][1]=0;
    info[0][0]=1;
    assertEquals( message: "BLUE IS THE WINNER", expected: 1,game.SOS( letter: "S", i: 0, j: 0, info, Slots, jugador: 0));
}
```

Build: Build Output x

SOSgame: build failed At 15/05/2023 17:08 with 1 error 2 sec, 471 ms

GuiSOS.java src\proyecto 1 error

missing return statement :260

C:\Users\mavsf\OneDrive\Desktop\MiCarpeta\otoCiclo\DESARROLLO SOFTWARE\CC-3S2\Practica1-C3S2\sprint3\SOSgame\src\proyecto\GuiSOS.java:260:5

java: missing return statement

Implementamos el método SOS, sin embargo, hacemos que el valor esperado no coincide con el valor actual

```
@Test
public void SOS_WinBlue(){
    int[][] info = new int[3][3];
    JButton[][] Slots = new JButton[3][3];

    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            info[i][j]= -1;
            JButton slot = new JButton();
            Slots[i][j] = slot;
        }
    }
    info[2][2]=1;
    info[1][1]=0;
    info[0][0]=1;
    assertEquals( message: "BLUE IS THE WINNER", expected: 0, game.SOS( letter: "S", i: 0, j: 0, info, Slots, jugador: 0));
}
```

✓ Tests failed: 1 of 1 test – 463 ms

TestFinishSimpleGame (project: 463 ms) C:\Users\mavsf\.jdk\corretto-20.0.1\bin\java.exe ...

SOS\_WinBlue 463 ms

java.lang.AssertionError: BLUE IS THE WINNER  
Expected :0  
Actual :1  
[Click to see difference](#)

Implementamos el método SOS, pero ahora hacemos coincidir el valor esperado con el valor actual

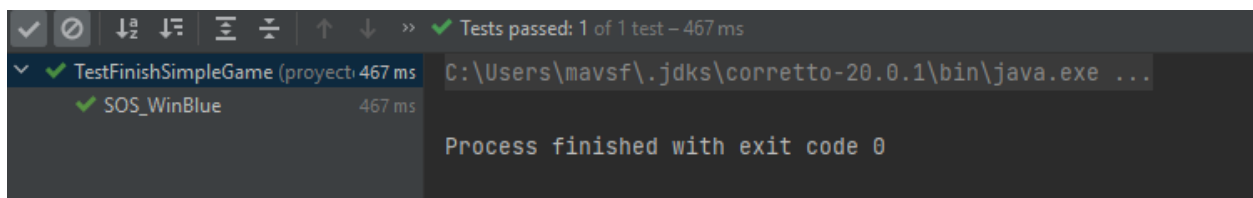
```

@Test
public void SOS_WinBlue(){
    int[][] info = new int[3][3];
    JButton[][] Slots = new JButton[3][3];

    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            info[i][j]= -1;
            JButton slot = new JButton();
            Slots[i][j] = slot;
        }
    }

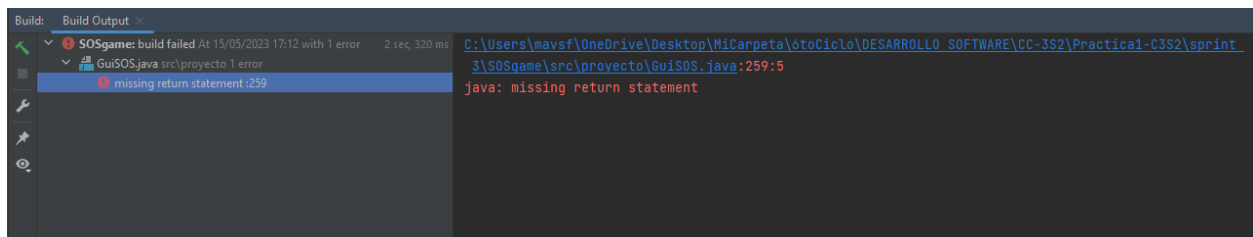
    info[2][2]=1;
    info[1][1]=0;
    info[0][0]=1;
    assertEquals( message: "BLUE IS THE WINNER", expected: 1,game.SOS( letter: "S", i: 0, j: 0, info, Slots, jugador: 0));
}

```



En general ejecutaremos todos los métodos de prueba dentro de la clase TestFinishSimpleGame

Primero escribimos las pruebas sin implementar el método SOS.



Implementamos el método SOS, sin embargo, hacemos que el valor esperado no coincide con el valor actual

```
TestFinishSimpleGame x
Tests failed: 3 of 3 tests - 541 ms
TestFinishSimpleGame (project: 541 ms)
  SOS_Draw 511 ms java.lang.AssertionError: DRAW
  SOS_WinRed 17 ms Expected :1
  SOS_WinBlue 13 ms Actual :0
  <Click to see difference>
```

```
TestFinishSimpleGame x
Tests failed: 3 of 3 tests - 541 ms
TestFinishSimpleGame (project: 541 ms)
  SOS_Draw 511 ms java.lang.AssertionError: RED IS THE WINNER
  SOS_WinRed 17 ms Expected :0
  SOS_WinBlue 13 ms Actual :1
  <Click to see difference>
```

```
TestFinishSimpleGame x
Tests failed: 3 of 3 tests - 541 ms
TestFinishSimpleGame (project: 541 ms)
  SOS_Draw 511 ms java.lang.AssertionError: BLUE IS THE WINNER
  SOS_WinRed 17 ms Expected :0
  SOS_WinBlue 13 ms Actual :1
  <Click to see difference>
```

Implementamos el método SOS, pero ahora hacemos coincidir el valor esperado con el valor actual

```
✓ TestFinishSimpleGame (project: 483 ms) C:\Users\mavsf\.jdk\corretto-20.0.1\bin\java.exe ...
  ✓ SOS_Draw 463 ms
  ✓ SOS_WinRed 9 ms
  ✓ SOS_WinBlue 11 ms
  Process finished with exit code 0
```

## Para la clase TestGameMode

Veamos el proceso RGR para el método de prueba **SOS\_GameModeSimpleGame** y **SOS\_GameModeGeneralGame**

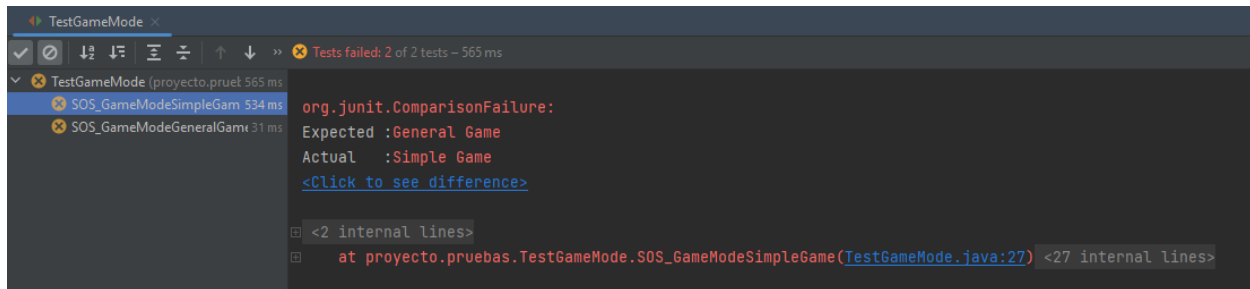
Primero escribimos la prueba sin implementar el **getModeGame**

```
@Test
public void SOS_GameModeSimpleGame(){
    game.NuevoTableroSimple();
    assertEquals("Simple Game",game.getModeGame());
}
```

```
@Test
public void SOS_GameModeGeneralGame(){
    game.NuevoTableroGeneral();
    assertEquals("General Game",game.getModeGame());
}
```

```
Build Output
SOSgame: build failed At 15/05/2023 17:21 with 1 error 2 sec, 656 ms
GuiSOS.java src\proyecto 1 error
missing return statement: 134
C:\Users\mavsf\OneDrive\Desktop\MiCarpeta\6toCiclo\DESARROLLO SOFTWARE\CC-3S2\Practical1-C3S2\sprint_3\SOSgame\src\proyecto\GuiSOS.java:134:5
java: missing return statement
```

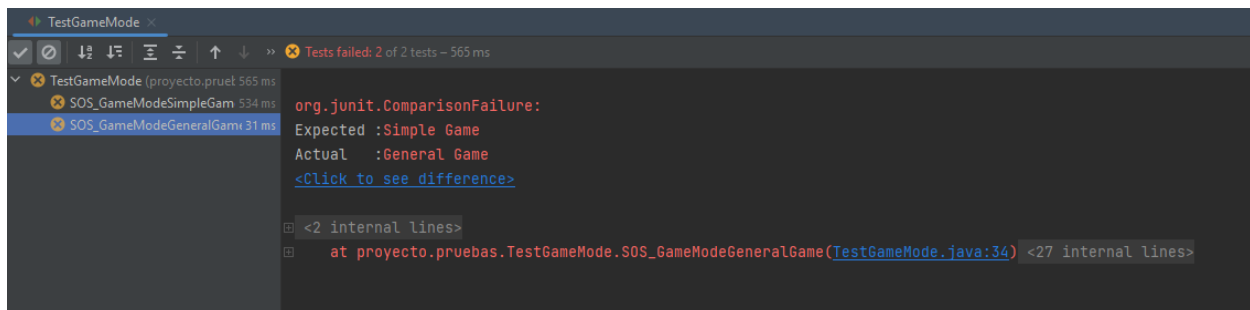
Implementamos el método `getModeGame`, sin embargo, hacemos que el valor esperado no coincide con el valor actual



```
TestGameMode (proyecto.pruebas) 565 ms
  SOS_GameModeSimpleGame 534 ms
  SOS_GameModeGeneralGame 31 ms

org.junit.ComparisonFailure:
Expected :General Game
Actual   :Simple Game
<Click to see difference>

<2 internal lines>
at proyecto.pruebas.TestGameMode.SOS_GameModeSimpleGame(TestGameMode.java:27) <27 internal lines>
```

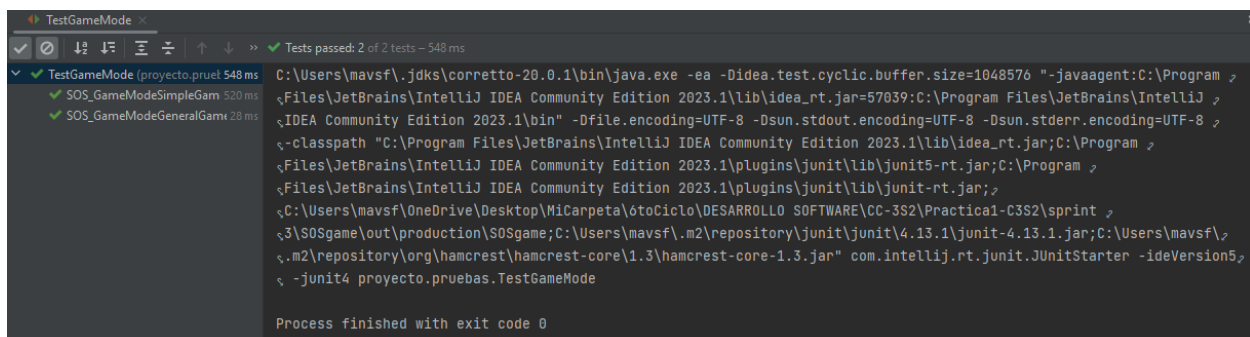


```
TestGameMode (proyecto.pruebas) 565 ms
  SOS_GameModeSimpleGame 534 ms
  SOS_GameModeGeneralGame 31 ms

org.junit.ComparisonFailure:
Expected :Simple Game
Actual   :General Game
<Click to see difference>

<2 internal lines>
at proyecto.pruebas.TestGameMode.SOS_GameModeGeneralGame(TestGameMode.java:34) <27 internal lines>
```

Implementamos el método `getModeGame`, pero ahora hacemos coincidir el valor esperado con el valor actual



```
TestGameMode (proyecto.pruebas) 548 ms
  SOS_GameModeSimpleGame 520 ms
  SOS_GameModeGeneralGame 28 ms

C:\Users\mavsf\.jdk\corretto-20.0.1\bin\java.exe -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1\lib\idea_rt.jar=57039:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath "C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1\lib\idea_rt.jar;C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1\plugins\junit\lib\junit5-rt.jar;C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1\plugins\junit\lib\junit-rt.jar;C:\Users\mavsf\OneDrive\Desktop\MiCarpeta\6toCiclo\DESARROLLO SOFTWARE\CC-3S2\Practica1-C3S2\sprint 3\SOSgame\out\production\SOSgame;C:\Users\mavsf\.m2\repository\junit\junit\4.13.1\junit-4.13.1.jar;C:\Users\mavsf\.m2\repository\org\hamcrest\hamcrest-core\1.3\hamcrest-core-1.3.jar" com.intellij.rt.junit.JUnit4Starter -ideVersion5, -junit4 proyecto.pruebas.TestGameMode

Process finished with exit code 0
```

## **Para la clase TestMoveGeneralGame**

Veamos el proceso RGR para el método de prueba **SOS\_MoveGeneral**

**Primero escribimos la prueba sin implementar el método getTurn**

```

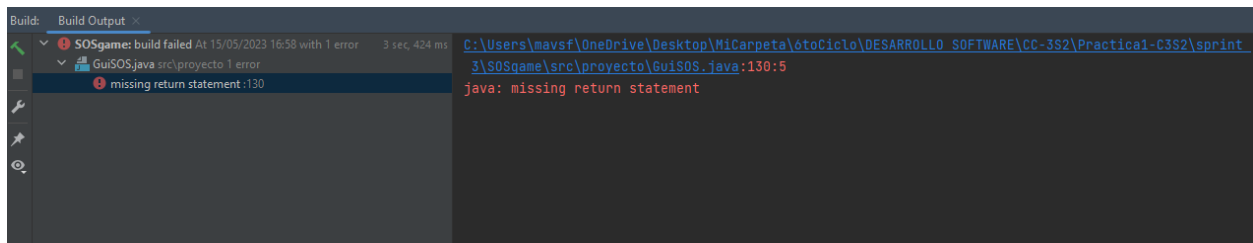
@Test
public void SOS_MoveGeneral(){

    int[][] info = new int[3][3];
    JButton[][] Slots = new JButton[3][3];

    for(int i=0;i< info.length;i++){
        for(int j=0;j< info.length;j++){
            info[i][j]= -1;
            JButton slot = new JButton();
            Slots[i][j] = slot;
        }
    }
    game.bluePlayer();
    info[0][0]=1;
    game.redPlayer();
    info[0][1]=0;
    game.bluePlayer();
    info[0][2]=1;
    game.bluePlayer();
    info[1][2]=1;

    assertEquals( expected: "Blue turn",game.getTurn());
}

```



Implementamos el método getTurn , sin embargo, hacemos que el valor esperado no coincide con el valor actual



```

@Test
public void SOS_MoveGeneral(){

    int[][] info = new int[3][3];
    JButton[][] Slots = new JButton[3][3];

    for(int i=0;i< info.length;i++){
        for(int j=0;j< info.length;j++){
            info[i][j]= -1;
            JButton slot = new JButton();
            Slots[i][j] = slot;
        }
    }
    game.bluePlayer();
    info[0][0]=1;
    game.redPlayer();
    info[0][1]=0;
    game.bluePlayer();
    info[0][2]=1;
    game.bluePlayer();
    info[1][2]=1;

    assertEquals( expected: "Red turn",game.getTurn());
}

```

```

TestMoveGeneralGame.SOS_MoveGeneral
Tests failed: 1 of 1 test - 636 ms
TestMoveGeneralGame (proyecto 636 ms)
  SOS_MoveGeneral 636 ms
    org.junit.ComparisonFailure:
      Expected :Red turn
      Actual   :Blue turn
      <Click to see difference>
    <2 internal lines>
    at proyecto.pruebas.TestMoveGeneralGame.SOS_MoveGeneral(TestMoveGeneralGame.java:47) <27 internal lines>

    Process finished with exit code -1

```

Implementamos el método getTurn, pero ahora hacemos coincidir el valor esperado con el valor actual

```

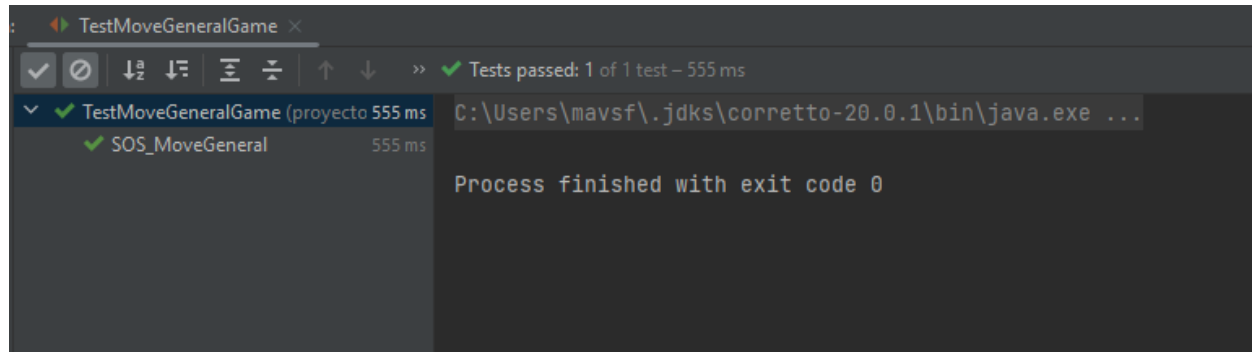
@Test
public void SOS_MoveGeneral(){

    int[][] info = new int[3][3];
    JButton[][] Slots = new JButton[3][3];

    for(int i=0;i< info.length;i++){
        for(int j=0;j< info.length;j++){
            info[i][j]= -1;
            JButton slot = new JButton();
            Slots[i][j] = slot;
        }
    }
    game.bluePlayer();
    info[0][0]=1;
    game.redPlayer();
    info[0][1]=0;
    game.bluePlayer();
    info[0][2]=1;
    game.bluePlayer();
    info[1][2]=1;

    assertEquals( expected: "Blue turn",game.getTurn());
}

```



## Para la clase TestMoveSimpleGame

Veamos el proceso RGR para el método de prueba `SOS_MoveSimple`

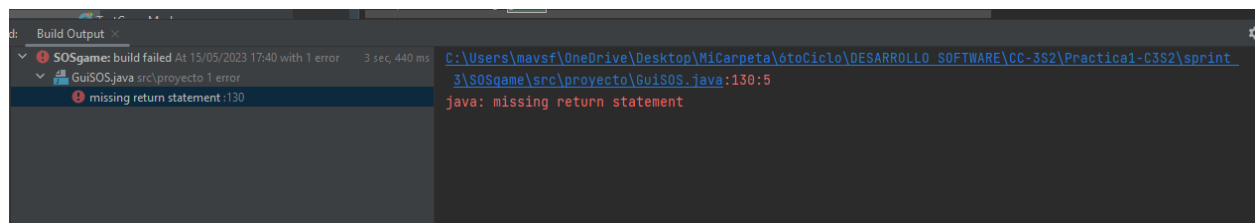
Primero escribimos la prueba sin implementar el método `getTurn`

```
@Test
public void SOS_MoveSimple(){

    int[][] info = new int[3][3];
    JButton[][] Slots = new JButton[3][3];

    for(int i=0;i< info.length;i++){
        for(int j=0;j< info.length;j++){
            info[i][j]= -1;
            JButton slot = new JButton();
            Slots[i][j] = slot;
        }
    }
    game.bluePlayer();
    info[0][0]=0;
    game.redPlayer();
    info[0][1]=1;

    assertEquals( expected: "Red turn",game.getTurn());
}
```



Implementamos el método `getTurn` , sin embargo, hacemos que el valor esperado no coincide con el valor actual

```

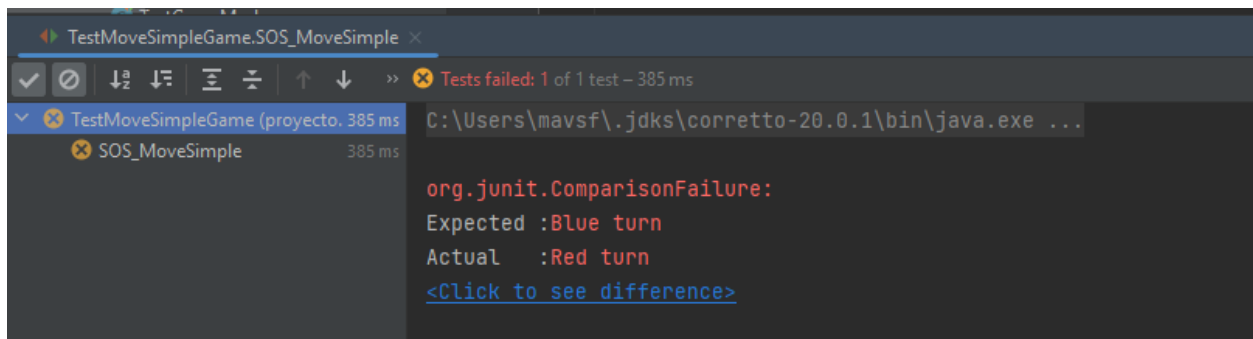
@Test
public void SOS_MoveSimple(){

    int[][] info = new int[3][3];
    JButton[][] Slots = new JButton[3][3];

    for(int i=0;i< info.length;i++){
        for(int j=0;j< info.length;j++){
            info[i][j]= -1;
            JButton slot = new JButton();
            Slots[i][j] = slot;
        }
    }
    game.bluePlayer();
    info[0][0]=0;
    game.redPlayer();
    info[0][1]=1;

    assertEquals( expected: "Blue turn",game.getTurn());
}

```



Implementamos el método getTurn, pero ahora hacemos coincidir el valor esperado con el valor actual

```

@Test
public void SOS_MoveSimple(){

    int[][] info = new int[3][3];
    JButton[][] Slots = new JButton[3][3];

    for(int i=0;i< info.length;i++){
        for(int j=0;j< info.length;j++){
            info[i][j]= -1;
            JButton slot = new JButton();
            Slots[i][j] = slot;
        }
    }
    game.bluePlayer();
    info[0][0]=0;
    game.redPlayer();
    info[0][1]=1;

    assertEquals( expected: "Red turn",game.getTurn());
}

```

TestMoveSimpleGame.SOS\_MoveSimple ×

✓ Tests passed: 1 of 1 test – 484 ms

✓ TestMoveSimpleGame (proyecto, 484 ms) C:\Users\mavsf\.jdk\corretto-20.0.1\bin\java.exe ...

✓ SOS\_MoveSimple 484 ms

Process finished with exit code 0

## Para la clase TestMoveSimpleGame

Veamos el proceso RGR para el método de prueba `SOS_MoveSimple`

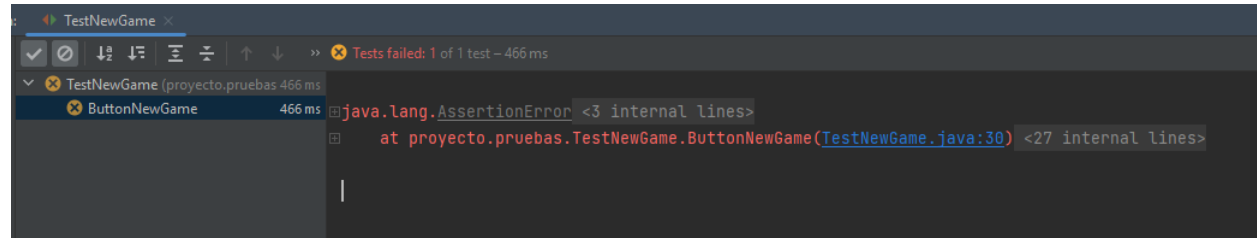
Primero escribimos la prueba sin implementar el método `ini_Tablero` ni `getIniGame`

```
@Test
public void ButtonNewGame(){
    game.init_Tablero();
    assertTrue(game.getIniGame());
}
```



Implementamos el método `init_Tablero` y `getIniGame`, sin embargo, eliminamos la primera instrucción dentro del cuerpo del método de prueba, de tal modo que la prueba falle

```
@Test
public void ButtonNewGame(){
    /* game.init_Tablero(); Esta instruccion nos indica que ha empezado el juego, entonces
    si eliminamos esta instruccion entonces no se presiona el
    Boton New Game, por ello getIniGame retorna false
    */
    assertTrue(game.getIniGame());
}
```



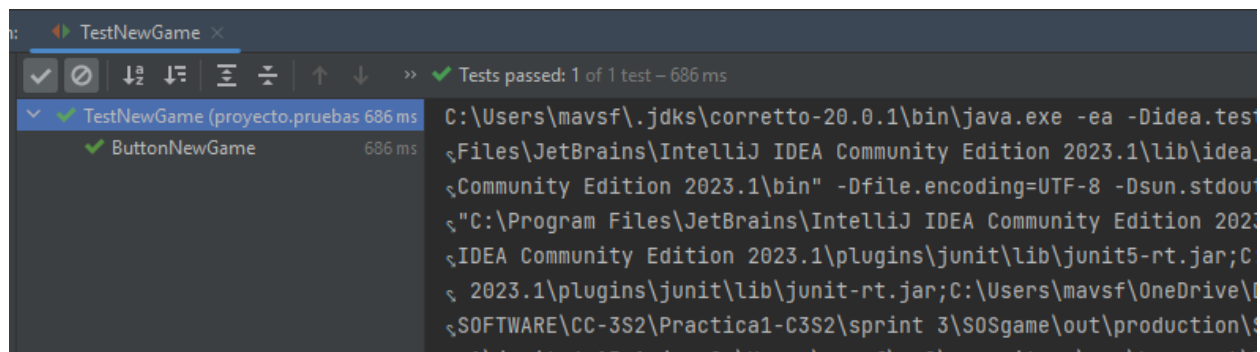
Implementamos el método `init_Tablero` y `getIniGame`, pero ahora hacemos que la prueba pase

```

@Test
public void ButtonNewGame(){
    game.init_Tablero();

    assertTrue(game.getIniGame());
}

```



## Para la clase TestSizeBoard

Esta vez mostrare solo las pruebas en verde y para ello explicare el método SizeN

```

public int SizeN(int n){
    if (n < 3 || n > 16) {
        JOptionPane.showMessageDialog( parentComponent: null, message: "La dimension debe estar entre 3 y 16",
            title: "Error", JOptionPane.ERROR_MESSAGE);
        SizeOfGameN.setEnabled(false);
        return n+1; //Solo es para que se cumpla la condicion dentro de init_tablero
    }else{
        return n;
    }
}

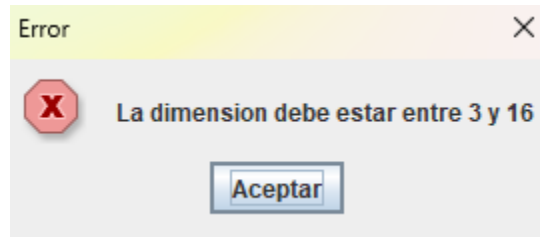
```

```

public void init_Tablero() {
    iniGame = true;
    SBluePlayer.setEnabled(true);
    OBluePlayer.setEnabled(true);
    SRedPlayer.setEnabled(true);
    ORedPlayer.setEnabled(true);
    Tablero.removeAll();
    int n = ((Integer) SizeOfGameN.getValue());
    if(SizeN(n)==n){

```

El comentario en el código es claro, para que empiece con la creación del tablero del juego SOS necesito que entre dentro del condicional if que se encuentra en el método init\_Tablero, para eso necesito que SizeN(n) retorne n que es la dimensión del tablero que establece el jugador, ahora bien, si se sale del rango permitido pues hacemos que retorne n+1 y esto hace que no entre dentro del condicional if y no se cree ningún tablero, además muestra la siguiente ventana.



```
@Test
public void CorrectN() { assertEquals( expected: 3, game.SizeN( n: 3)); }
```

```
@Test
public void WrongN1() { assertEquals( expected: 3, game.SizeN( n: 2)); }
```

```
@Test
public void WrongN2(){
    Throwable exception = assertThrows(IllegalArgumentException.class, () -> {
        if(game.SizeN( n: -1)!=-1){
            throw new IllegalArgumentException("La dimension debe estar entre 3 y 16");
        };
    });
    assertEquals( expected: "La dimension debe estar entre 3 y 16", exception.getMessage());
}
```

