**Universidad ORT Uruguay**

**Facultad de Ingeniería**

**Escuela de Tecnología**

**OBLIGATORIO 1**

**ALGORITMOS Y ESTRUCTURAS DE DATOS**



**Pablo Larnaudie – 340181**

****

**Natalia Rebella – 327283**

**Grupo: N3C**

**Docente: Hugo Cepeda**

**Analista en Tecnologías de la Información**

**Fecha de entrega del documento (12-05-2025)**

**Índice**

[1. Representación gráfica del sistema de gestión a resolver 3](#_heading=h.w5ymfd9ip7rv)

[2. Juego de pruebas, testeo y resultados 4](#_heading=h.zdbcupbiswbw)

[1.1. TESTEO REQUERIMIENTO 1.1 4](#_heading=h.ytzstbyz075e)

[1.2. TESTEO REQUERIMIENTO 1.2 – OK 4](#_heading=h.3veffk72nccu)

[1.3. TESTEO REQUERIMIENTO 1.2 – ERROR 1 5](#_heading=h.msbrlplqu66n)

[1.4. TESTEO REQUERIMIENTO 1.2 – ERROR 2 6](#_heading=h.de3tmuhj3wa6)

[1.5. TESTEO REQUERIMIENTO 1.3 – OK 7](#_heading=h.l6qfk8lede5i)

[1.6. TESTEO REQUERIMIENTO 1.3 – ERROR 1 8](#_heading=h.hgsvn9dse223)

[1.7. TESTEO REQUERIMIENTO 1.4 - OK 9](#_heading=h.y4ki7xj6plg)

[1.8. TESTEO REQUERIMIENTO 1.4 – ERROR 1 10](#_heading=h.nrqh9pmzpz6l)

[1.9. TESTEO REQUERIMIENTO 1.4 – ERROR 2 12](#_heading=h.n1wlrcg83cv2)

[1.10. TESTEO REQUERIMIENTO 1.4 –ERROR 3 12](#_heading=h.nxaveteb2bzi)

[1.11. TESTEO REQUERIMIENTO 1.5 – OK 14](#_heading=h.wl3il4xtbm8j)

[1.12. TESTEO REQUERIMIENTO 1.5 – ERROR 1 15](#_heading=h.vorada238n47)

[1.13. TESTEO REQUERIMIENTO 1.5 – ERROR 2 17](#_heading=h.pb8l62fw6qkt)

[1.14. TESTEO REQUERIMIENTO 2.1 18](#_heading=h.9prs54bnuky7)

[1.15. TESTEO REQUERIMIENTO 2.2 19](#_heading=h.64l9rexewzt1)

[1.16. TESTEO REQUERIMIENTO 2.3 20](#_heading=h.sjmugbk6hh36)

[1.17. TESTEO REQUERIMIENTO 2.4 21](#_heading=h.ld18vkfe464s)

# Representación gráfica del sistema de gestión a resolver

# Juego de pruebas, testeo y resultados

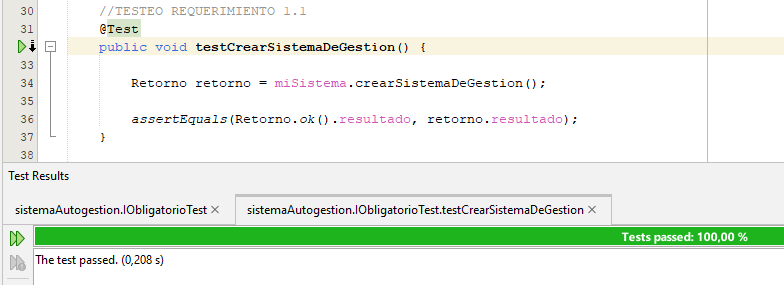
## TESTEO REQUERIMIENTO 1.1

@Test

public void testCrearSistemaDeGestion() {

Retorno retorno = miSistema.crearSistemaDeGestion();

assertEquals(Retorno.ok().resultado, retorno.resultado);

}

## TESTEO REQUERIMIENTO 1.2 – OK

@Test

public void testRegistrarSalaOK() {

Retorno retorno = miSistema.registrarSala("Sala Aire", 80);

assertEquals(Retorno.ok().resultado, retorno.resultado);

retorno = miSistema.registrarSala("Sala Fuego", 150);

assertEquals(Retorno.ok().resultado, retorno.resultado);

retorno = miSistema.registrarSala("Sala Agua", 200);

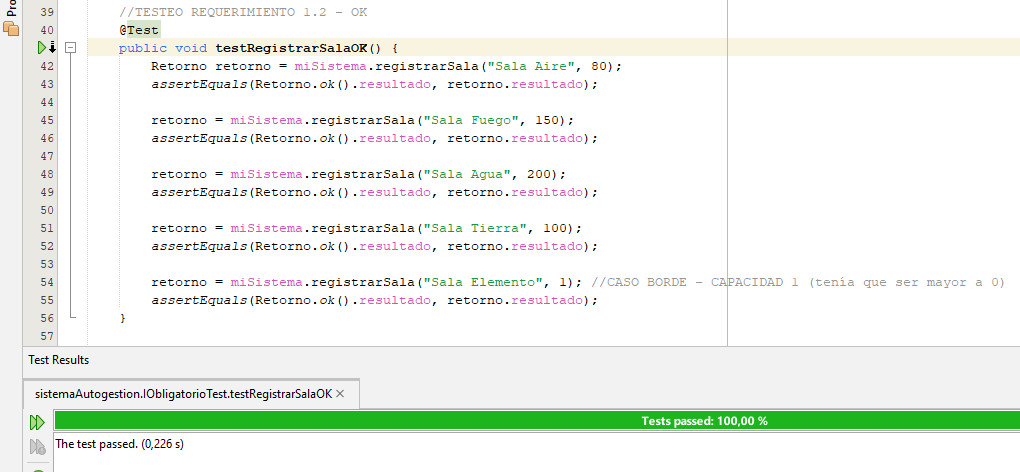
assertEquals(Retorno.ok().resultado, retorno.resultado);

retorno = miSistema.registrarSala("Sala Tierra", 100);

assertEquals(Retorno.ok().resultado, retorno.resultado);

retorno = miSistema.registrarSala("Sala Elemento", 1); //CASO BORDE - CAPACIDAD 1 (tenía que ser mayor a 0)

assertEquals(Retorno.ok().resultado, retorno.resultado);

}

## TESTEO REQUERIMIENTO 1.2 – ERROR 1

@Test

public void testRegistrarSalaERROR1() {

miSistema.registrarSala("Sala Aire", 100);

Retorno retorno = miSistema.registrarSala("Sala Aire", 80); //ERROR POR IGUAL NOMBRE

assertEquals(Retorno.error1().resultado, retorno.resultado);

retorno = miSistema.registrarSala("sala aire", 200);

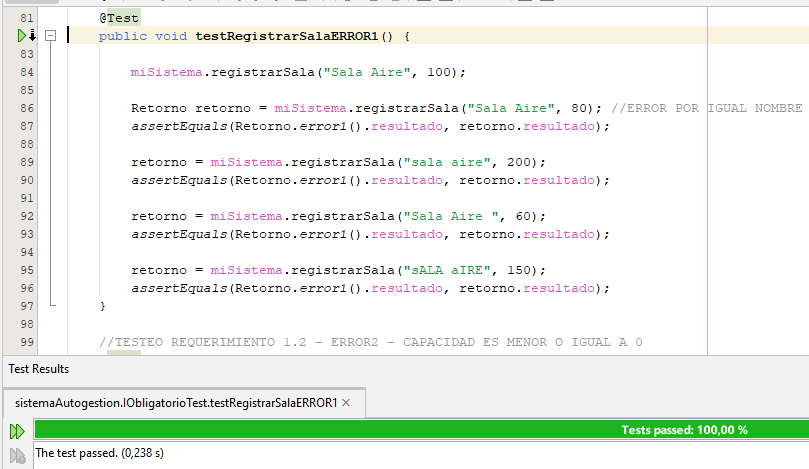
assertEquals(Retorno.error1().resultado, retorno.resultado);

retorno = miSistema.registrarSala("Sala Aire ", 60);

assertEquals(Retorno.error1().resultado, retorno.resultado);

retorno = miSistema.registrarSala("sALA aIRE", 150);

assertEquals(Retorno.error1().resultado, retorno.resultado);

}

## TESTEO REQUERIMIENTO 1.2 – ERROR 2

@Test

public void testRegistrarSalaERROR2() {

Retorno retorno = miSistema.registrarSala("Sala Aire", 0);

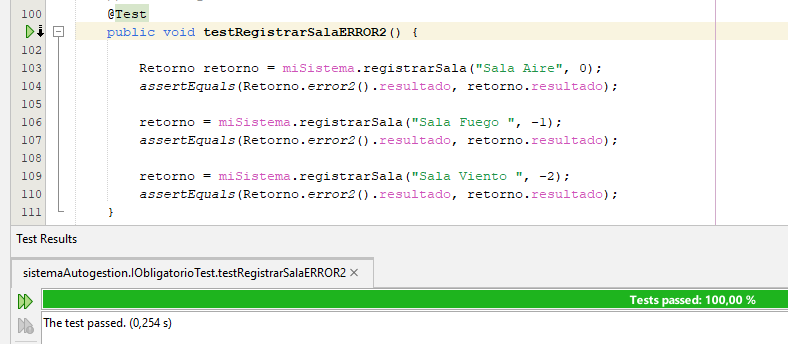
assertEquals(Retorno.error2().resultado, retorno.resultado);

retorno = miSistema.registrarSala("Sala Fuego ", -1);

assertEquals(Retorno.error2().resultado, retorno.resultado);

retorno = miSistema.registrarSala("Sala Viento ", -2);

assertEquals(Retorno.error2().resultado, retorno.resultado);

}

## TESTEO REQUERIMIENTO 1.3 – OK

@Test

public void testEliminarSalaOK() {

//Completar para primera entrega

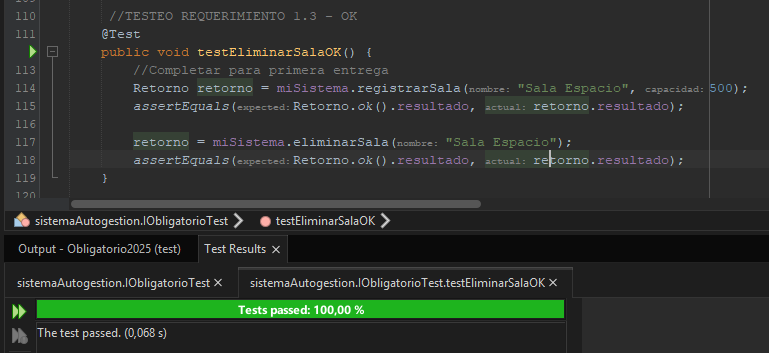
Retorno retorno = miSistema.registrarSala("Sala Espacio", 500);

assertEquals(Retorno.ok().resultado, retorno.resultado);

retorno = miSistema.eliminarSala("Sala Espacio");

assertEquals(Retorno.ok().resultado, retorno.resultado);

}



## TESTEO REQUERIMIENTO 1.3 – ERROR 1

@Test

public void testEliminarSalaERROR1() {

//Completar para primera entrega

Retorno retorno = miSistema.registrarSala("Sala Zitarrosa", 500);

assertEquals(Retorno.ok().resultado, retorno.resultado);

retorno = miSistema.eliminarSala("Sala Espacio");

assertEquals(Retorno.error1().resultado, retorno.resultado);

}

## TESTEO REQUERIMIENTO 1.4 - OK

@Test

public void testRegistrarEventoOK() {

miSistema.registrarSala("Sala Viento", 100);

LocalDate fecha = LocalDate.of(2025, 5, 15);

Retorno retorno = miSistema.registrarEvento("AAQ23", "Noche de gala", 50, fecha);

assertEquals(Retorno.ok().resultado, retorno.resultado);

LocalDate fecha1 = LocalDate.of(2025, 4, 20);

retorno = miSistema.registrarEvento("ABC12", "Baile gitano", 1, fecha1);

assertEquals(Retorno.ok().resultado, retorno.resultado);

LocalDate fecha2 = LocalDate.of(2025, 3, 13);

retorno = miSistema.registrarEvento("XYZ34", "Bingo", 90, fecha2);

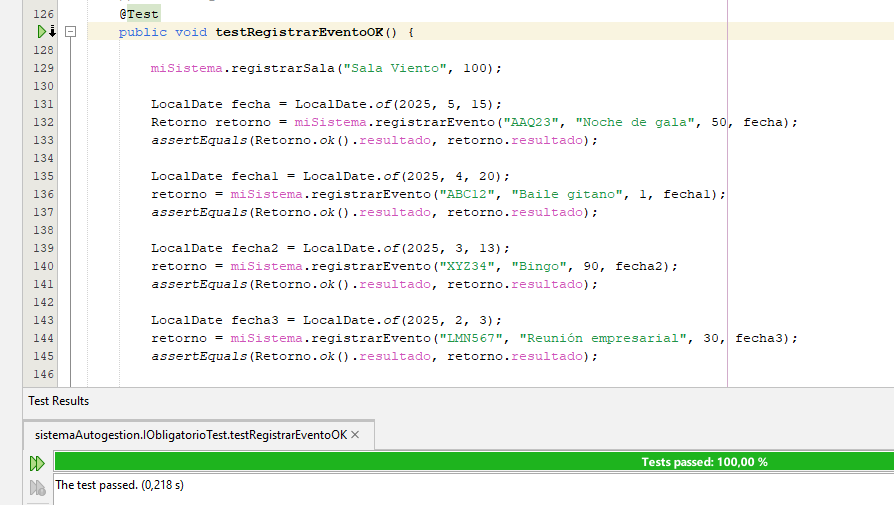
assertEquals(Retorno.ok().resultado, retorno.resultado);

LocalDate fecha3 = LocalDate.of(2025, 2, 3);

retorno = miSistema.registrarEvento("LMN567", "Reunión empresarial", 30, fecha3);

assertEquals(Retorno.ok().resultado, retorno.resultado);

}



## TESTEO REQUERIMIENTO 1.4 – ERROR 1

@Test

public void testRegistrarEventoERROR1() {

miSistema.registrarSala("Sala Baile", 200);

LocalDate fecha1 = LocalDate.of(2025, 6, 15);

LocalDate fecha2 = LocalDate.of(2026, 6, 15);

LocalDate fecha5 = LocalDate.of(2025, 11, 15);

LocalDate fecha6 = LocalDate.of(2026, 3, 15);

LocalDate fecha7 = LocalDate.of(2025, 2, 13);

miSistema.registrarEvento("AAQ23", "Bailamos", 90, fecha1);

miSistema.registrarEvento("ZZZ12", "Lectura compartida", 100, fecha2);

Retorno retorno = miSistema.registrarEvento("AAQ23", "Noche de gala", 50, fecha5);

assertEquals(Retorno.error1().resultado, retorno.resultado);

retorno = miSistema.registrarEvento("ZZZ12", "Noche de gala", 50, fecha6);

assertEquals(Retorno.error1().resultado, retorno.resultado);

retorno = miSistema.registrarEvento("ZZZ12", "Té", 150, fecha7);

assertEquals(Retorno.error1().resultado, retorno.resultado);

}

## TESTEO REQUERIMIENTO 1.4 – ERROR 2

@Test

public void testRegistrarEventoERROR2() {

miSistema.registrarSala("Sala Baile", 200);

LocalDate fecha5 = LocalDate.of(2025, 11, 15);

LocalDate fecha6 = LocalDate.of(2026, 3, 15);

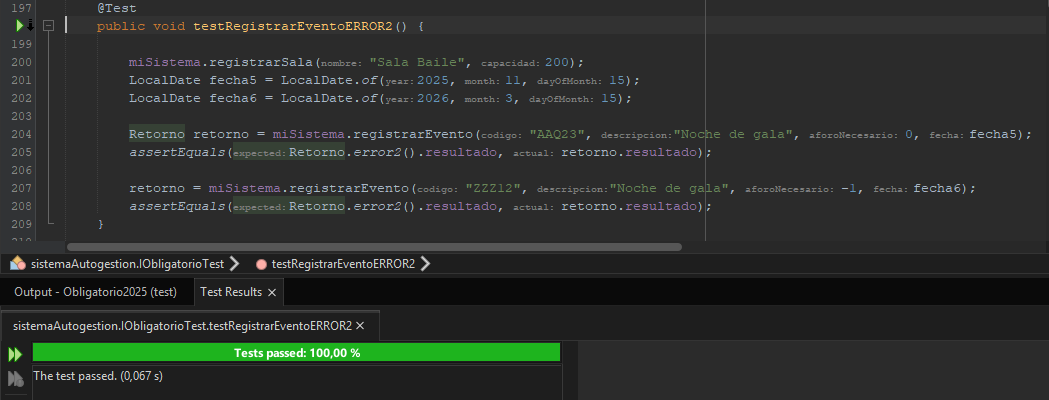
Retorno retorno = miSistema.registrarEvento("AAQ23", "Noche de gala", 0, fecha5);

assertEquals(Retorno.error2().resultado, retorno.resultado);

retorno = miSistema.registrarEvento("ZZZ12", "Noche de gala", -1, fecha6);

assertEquals(Retorno.error2().resultado, retorno.resultado);

}



## TESTEO REQUERIMIENTO 1.4 –ERROR 3

@Test

public void testRegistrarEventoERROR3() {

miSistema.registrarSala("Sala Eventos", 100);

LocalDate fecha1 = LocalDate.of(2025, 12, 15);

LocalDate fecha2 = LocalDate.of(2026, 12, 15);

Retorno retorno1 = miSistema.registrarEvento("AAQ23", "Noche de gala", 100, fecha1);

assertEquals(Retorno.ok().resultado, retorno1.resultado);

//misma fecha

Retorno retorno2 = miSistema.registrarEvento("AB7DG", "Evento", 90, fecha1);

assertEquals(Retorno.error3().resultado, retorno2.resultado);

//aforo insuficiente fecha disponible

Retorno retorno3 = miSistema.registrarEvento("A456SD", "Fiesta", 101, fecha2);

assertEquals(Retorno.error3().resultado, retorno3.resultado);

//aforo insuficiente y misma fecha

Retorno retorno4 = miSistema.registrarEvento("ALLL80", "Reunión", 100, fecha1);

assertEquals(Retorno.error3().resultado, retorno4.resultado);

}



## TESTEO REQUERIMIENTO 1.5 – OK

@Test

public void testRegistrarClienteOK() {

//Completar para primera entrega

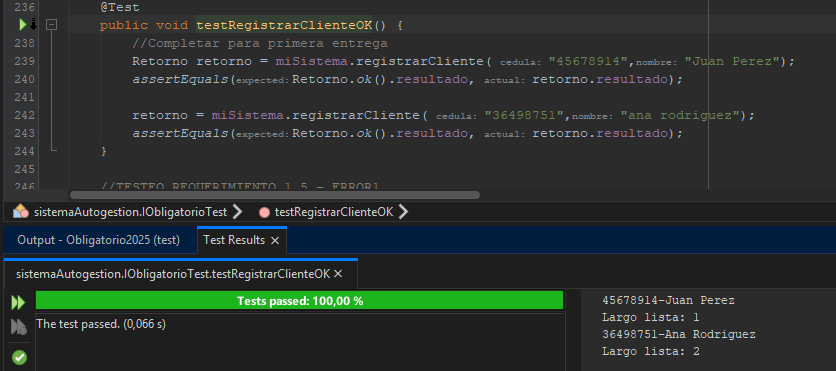
Retorno retorno = miSistema.registrarCliente( "45678914","Juan Perez");

assertEquals(Retorno.ok().resultado, retorno.resultado);

retorno = miSistema.registrarCliente( "36498751","ana rodriguez");

assertEquals(Retorno.ok().resultado, retorno.resultado);

}



## TESTEO REQUERIMIENTO 1.5 – ERROR 1

@Test

public void testRegistrarClienteERROR1() {

//Completar para primera entrega

//Test con 9 digitos

Retorno retorno = miSistema.registrarCliente( "456789147","Julio Cesar");

assertEquals(Retorno.error1().resultado, retorno.resultado);

//Test con 7 digitos

retorno = miSistema.registrarCliente( "4567891","Marco Aurelio");

assertEquals(Retorno.error1().resultado, retorno.resultado);

//Test con 8 letras

retorno = miSistema.registrarCliente( "asdfghjk","Cleopatra Nilo");

assertEquals(Retorno.error1().resultado, retorno.resultado);

//Test con letras y numeros

retorno = miSistema.registrarCliente( "1234ghjk","Alejandro Magno");

assertEquals(Retorno.error1().resultado, retorno.resultado);

//Test con simbolos

retorno = miSistema.registrarCliente( "!¡?¿#@\_-", "Pompeyo Magno");

assertEquals(Retorno.error1().resultado, retorno.resultado);

//Test con simbolos y letras

retorno = miSistema.registrarCliente( "!@#abcde", "Trajano Romano");

assertEquals(Retorno.error1().resultado, retorno.resultado);

//Test con simbolos y numeros

retorno = miSistema.registrarCliente( "!1234?86", "Marcus Ulpius Traianus");

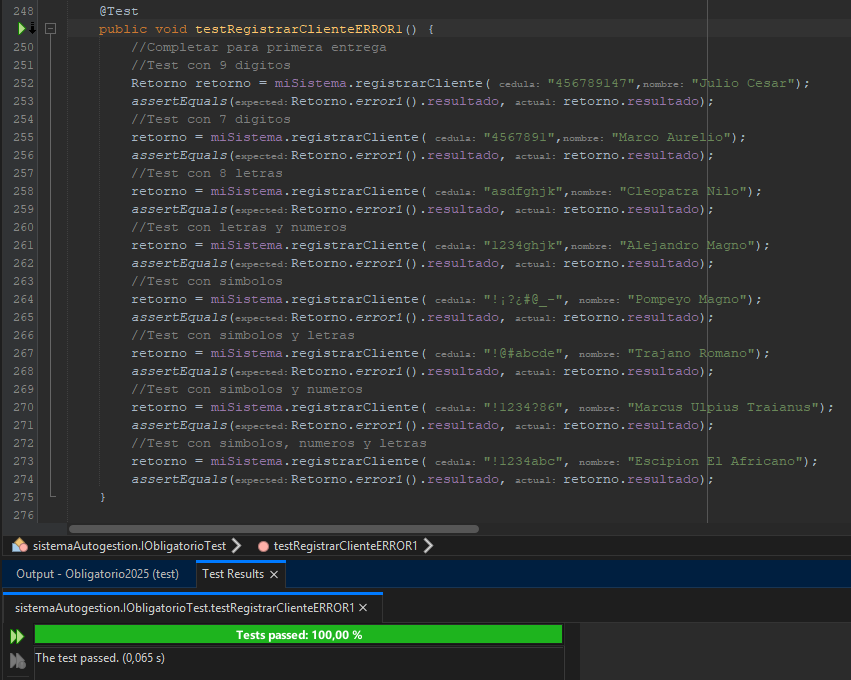
assertEquals(Retorno.error1().resultado, retorno.resultado);

//Test con simbolos, numeros y letras

retorno = miSistema.registrarCliente( "!1234abc", "Escipion El Africano");

assertEquals(Retorno.error1().resultado, retorno.resultado);

}



## TESTEO REQUERIMIENTO 1.5 – ERROR 2

@Test

public void testRegistrarClienteERROR2() {

//Completar para primera entrega

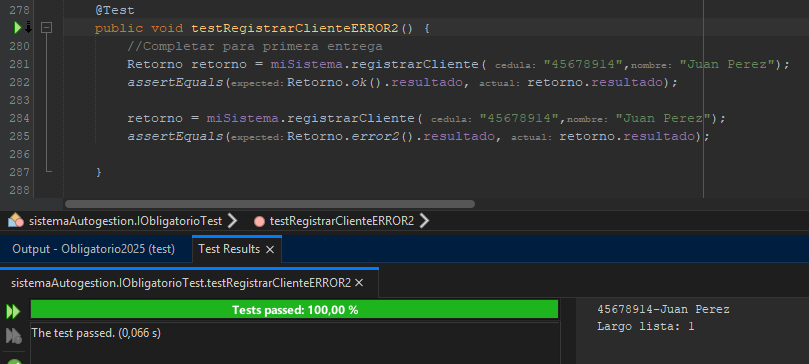
Retorno retorno = miSistema.registrarCliente( "45678914","Juan Perez");

assertEquals(Retorno.ok().resultado, retorno.resultado);

retorno = miSistema.registrarCliente( "45678914","Juan Perez");

assertEquals(Retorno.error2().resultado, retorno.resultado);

}



## TESTEO REQUERIMIENTO 2.1

@Test

public void testListarSalas() {

//Completar para primera entrega

miSistema.registrarSala("Sala Aire", 80);

miSistema.registrarSala("Sala Fuego", 150);

miSistema.registrarSala("Sala Agua", 200);

miSistema.registrarSala("Sala Tierra", 100);

miSistema.registrarSala("Sala Elemento", 1); //CASO BORDE - CAPACIDAD 1 (tenía que ser mayor a 0)

Retorno retorno = miSistema.listarSalas();

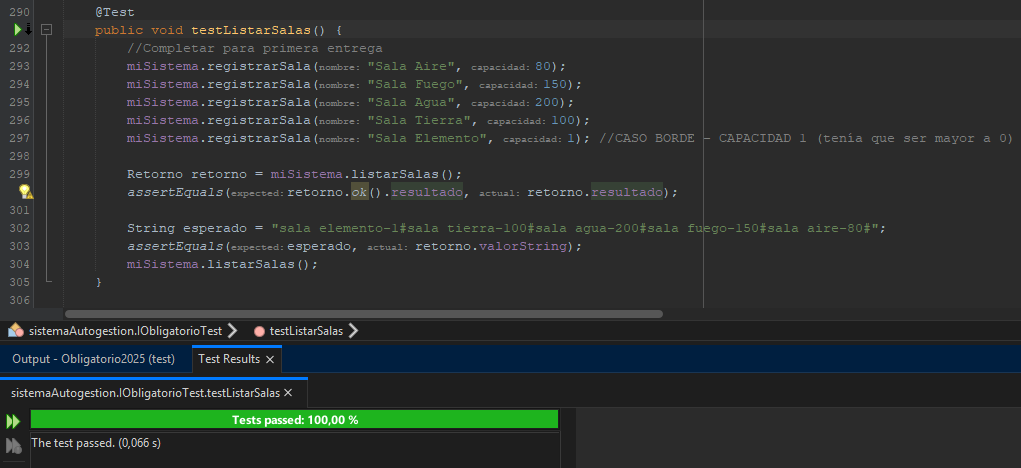
assertEquals(retorno.ok().resultado, retorno.resultado);

String esperado = "sala elemento-1#sala tierra-100#sala agua-200#sala fuego-150#sala aire-80#";

assertEquals(esperado, retorno.valorString);

miSistema.listarSalas();

}



## TESTEO REQUERIMIENTO 2.2

@Test

public void testListarEventos() {

miSistema.registrarSala("Sala Aire", 220);

LocalDate fecha1 = LocalDate.of(2025, 12, 15);

miSistema.registrarEvento("AAQ23", "Noche de gala", 100, fecha1);

LocalDate fecha2 = LocalDate.of(2024, 12, 15);

miSistema.registrarEvento("AZD456", "Tango feliz", 150, fecha2);

LocalDate fecha3 = LocalDate.of(2023, 12, 15);

miSistema.registrarEvento("AZK765", "Lectura", 200, fecha3);

Retorno retorno = miSistema.listarEventos();

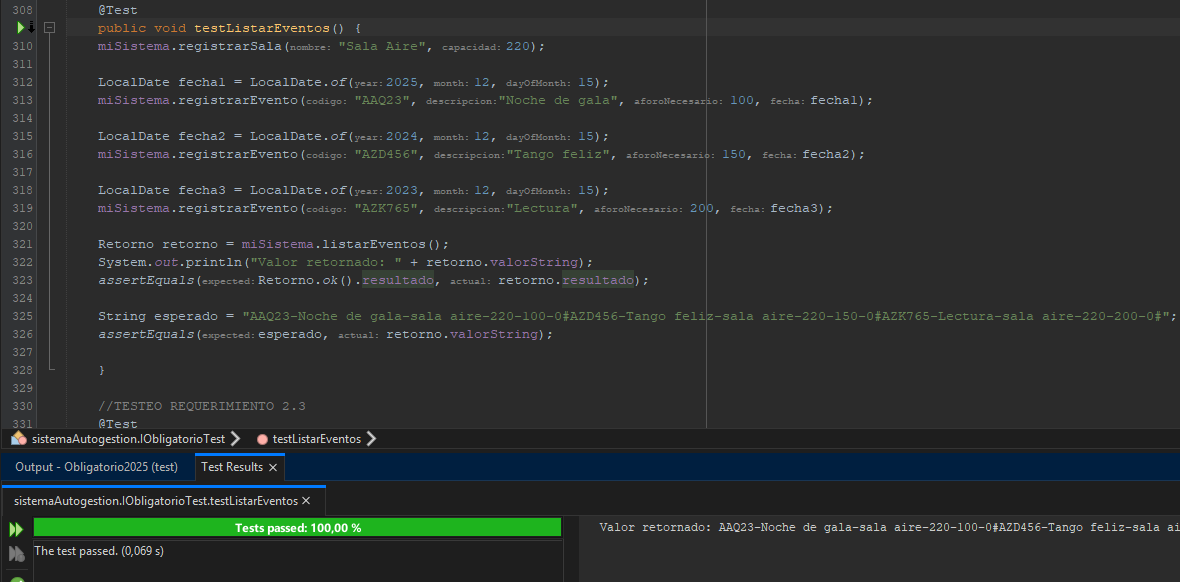
System.out.println("Valor retornado: " + retorno.valorString);

assertEquals(Retorno.ok().resultado, retorno.resultado);

String esperado = "AAQ23-Noche de gala-sala aire-220-100-0#AZD456-Tango feliz-sala aire-220-150-0#AZK765-Lectura-sala aire-220-200-0#";

assertEquals(esperado, retorno.valorString);

}



## TESTEO REQUERIMIENTO 2.3

@Test

public void testListarClientes() {

//Completar para primera entrega

miSistema.registrarCliente( "45678913","Julio Cesar");

miSistema.registrarCliente( "35678982","Marco Aurelio");

miSistema.registrarCliente( "15678965","Alejandro Magno");

miSistema.registrarCliente( "55678956", "Pompeyo Magno");

miSistema.registrarCliente( "49267892", "Escipion El Africano");

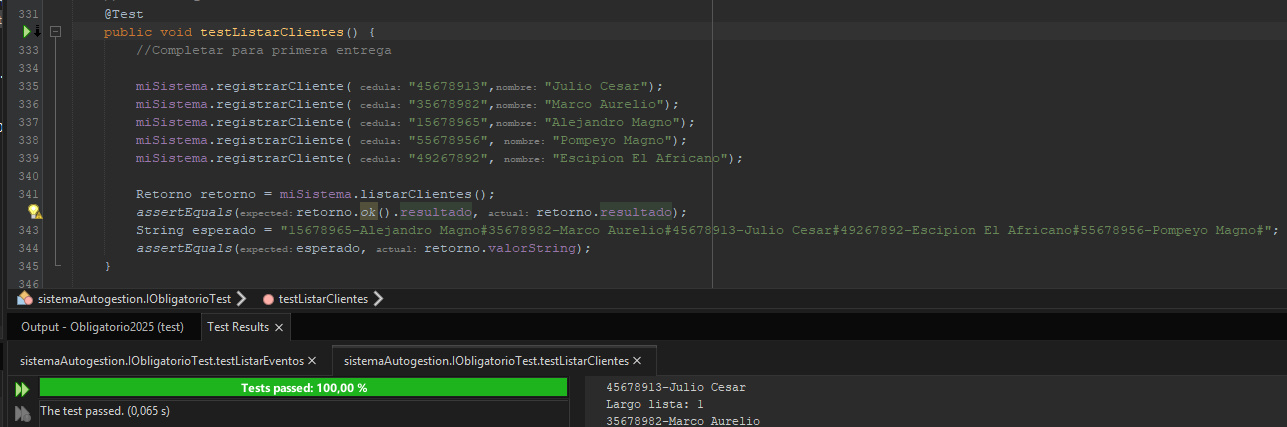
Retorno retorno = miSistema.listarClientes();

assertEquals(retorno.ok().resultado, retorno.resultado);

String esperado = "15678965-Alejandro Magno#35678982-Marco Aurelio#45678913-Julio Cesar#49267892-Escipion El Africano#55678956-Pompeyo Magno#";

assertEquals(esperado, retorno.valorString);

}



## TESTEO REQUERIMIENTO 2.4

@Override

public Retorno esSalaOptima(String[][] vistaSala) {

if (vistaSala == null || vistaSala.length == 0 || vistaSala[0].length == 0) {

Retorno ret = (Retorno.ok());

ret.valorString = "No se ingresó por parametros valores validos";

return ret;

}

int filas = vistaSala.length;

int columnas = vistaSala[0].length;

int columnasOptimas = 0;

for (int col = 0; col < columnas; col++) {

int maxOcupadosConsecutivos = 0;

int actualesOcupados = 0;

int asientosLibres = 0;

for (int fila = 0; fila < filas; fila++) {

String asiento = vistaSala[fila][col];

if (asiento.equals("O")) {

actualesOcupados++;

maxOcupadosConsecutivos = Math.max(maxOcupadosConsecutivos, actualesOcupados);

} else {

actualesOcupados = 0; // rompemos la secuencia

}

if (asiento.equals("X")) {

asientosLibres++;

}

}

if (maxOcupadosConsecutivos > asientosLibres) {

columnasOptimas++;

}

}

if (columnasOptimas >= 2) {

Retorno ret = (Retorno.ok());

ret.valorString = "Es optimo";

return ret;

} else {

Retorno ret = (Retorno.ok());

ret.valorString = "No es optimo";

return ret;

}

}

