



Recuperacion Entorno

Alejandro Ramon Rolon
Vazquez



1. Diseño de Casos de Prueba

Se han diseñado pruebas unitarias con JUnit 5 para todos los métodos clave de la clase GestorTurnos. Las pruebas cubren:

Casos normales:

- Añadir pacientes.
- Atender pacientes según prioridad.
- Cancelar turno correctamente.

Casos límite:

Atender cuando no hay pacientes.

Cancelar un turno inexistente.

Casos excepcionales:

Verificar comportamiento si se añaden pacientes con la misma prioridad.

Se usa @BeforeEach para inicializar el gestor en cada test, y se cubren todos los flujos de ejecución con assertEquals, assertTrue, assertNull, etc.

2. Implementación con JUnit 5

Las pruebas están automatizadas y bien estructuradas en el archivo GestorTurnosTest.java, con los siguientes métodos:

```
testAñadirPaciente()
testAtenderPacientePorPrioridad()
testCancelarTurnoPorNombre()
testContarPacientesPorPrioridad()
```

Todas las pruebas pasan correctamente (resultado verde al ejecutarlas desde Eclipse). La cobertura es total respecto a los requisitos funcionales.

Plan de Pruebas e Incidencias



Planificación de pruebas:

Método probado	Tipo de prueba	Resultado esperado	Estado
añadirPaciente()	Normal	Se añade correctamente	✓
atenderSiguientePaciente()	Límite	Atiende según prioridad	✓
cancelarTurnoPorNombre()	Excepcional	Devuelve false si no existe	✓
verSiguientePaciente()	Límite	Devuelve null si está vacío	✓

Incidencias encontradas:

Inicialmente no se podía usar JUnit por no estar en el build path. Se resolvió añadiendo las librerías manualmente desde Eclipse.

Refactorización y Depuración

Se refactorizó el código original para mejorar:

Nombres de métodos (añadirPaciente, verSiguientePaciente, etc.).

Claridad de variables y funciones.

Se extrajo la lógica de prioridad en un método auxiliar privado obtenerSiguientePaciente() para mejorar la legibilidad y reducir duplicación.

El código está ordenado, modularizado y documentado brevemente donde es necesario.

Uso de GitHub y Documentación Técnica

- El proyecto está subido en un repositorio público en GitHub.
- Se han realizado commits frecuentes y descriptivos durante el desarrollo.
- Este archivo README.md incluye:
 - Descripción general del proyecto.
 - Plan de pruebas.



Detalles sobre refactorización.
Instrucciones para ejecutar el proyecto.

Crear la clase Paciente

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure: 'Recuperacionentorno' -> 'GestorTurnosConsulta' -> 'src' -> 'modelo'. The 'Paciente.java' file is selected. The main editor shows the following code:

```
1 package modelo;
2
3 public class Paciente {
4     String n;
5     String m;
6     int p;
7
8     public Paciente(String n, String m, int p) {
9         this.n = n;
10        this.m = m;
11        this.p = p;
12    }
13 }
14
```

Crear la clase GestorTurnos

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure: 'Recuperacionentorno' -> 'GestorTurnosConsulta' -> 'src' -> 'modelo'. The 'GestorTurnos.java' file is selected. The main editor shows the following code:

```
1 package modelo;
2
3 import java.util.*;
4
5 public class GestorTurnos {
6     List<Paciente> lista = new ArrayList<>();
7
8     public void a(String n, String m, int p) {
9         lista.add(new Paciente(n, m, p));
10    }
11
12    public Paciente b() {
13        for (int prio = 1; prio <= 3; prio++) {
14            for (Paciente x : lista) {
15                if (x.p == prio) {
16                    lista.remove(x);
17                    return x;
18                }
19            }
20        }
21        return null;
22    }
23
24    public int c(int prio) {
25        int cont = 0;
26        for (Paciente x : lista) {
27            if (x.p == prio) {
28                cont++;
29            }
30        }
31        return cont;
32    }
33
34    public String d() {
35        for (int prio = 1; prio <= 3; prio++) {
36            for (Paciente x : lista) {
37                if (x.p == prio) {
```



```
        if (x.p == prio) {  
            return x.n;  
        }  
    }  
    }  
    return null;  
}  
  
public boolean e(String n) {  
    for (Paciente x : lista) {  
        if (x.n.equals(n)) {  
            lista.remove(x);  
            return true;  
        }  
    }  
    return false;  
}  
}
```

Crear clase Main

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure: 'GestorTurnosConsulta' containing 'JRE System Library [JavaSE-22]', 'src' (with 'GestorTurnos.java', 'Main.java', 'Paciente.java', and 'module-info.java'), and 'modelo'. The main editor shows the code for 'Main.java'.

```
1 package modelo;  
2  
3 public class Main {  
4     public static void main(String[] args) {  
5         GestorTurnos g = new GestorTurnos();  
6  
7         g.a("Ana", "Dolor de cabeza", 2);  
8         g.a("Luis", "Urgencia", 1);  
9         g.a("Pedro", "Consulta general", 3);  
10  
11         System.out.println("Siguiente en ser atendido: " + g.d());  
12         System.out.println("Paciente atendido: " + g.b().n);  
13         System.out.println("Pacientes con prioridad 1: " + g.c(1));  
14         System.out.println("Cancelar turno Ana: " + g.e("Ana"));  
15     }  
16 }  
17
```



Verificacion

The screenshot shows the Eclipse IDE interface. The main editor displays the `GestorTurnosTest.java` file, which contains the following code:

```
1 package modelo;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 public class GestorTurnosTest {
6
7     private GestorTurnos gestor;
8
9     @BeforeEach
10    public void setUp() {
11        gestor = new GestorTurnos();
12    }
13
14    @Test
15    public void testAñadirPaciente() {
16        gestor.agregarPaciente("Mario", "Dolor de muelas", 2);
17        assertEquals("Mario", gestor.verNombreSiguientePaciente());
18    }
19
20    @Test
21    public void testAtenderPacientePorPrioridad() {
22        gestor.agregarPaciente("Pedro", "Consulta", 3);
23        gestor.agregarPaciente("Luis", "Urgencia", 1);
24        gestor.agregarPaciente("Ana", "Dolor cabeza", 2);
25
26        Paciente atendido = gestor.atenderSiguientePaciente();
27        assertNotNull(atendido);
28        assertEquals("Luis", atendido.getNombre());
29    }
30
31    @Test
32    public void testCancelarTurno() {
33        gestor.agregarPaciente("Ana", "Dolor cabeza", 2);
34        assertTrue(gestor.cancelarTurnoPorNombre("Ana"));
35        assertFalse(gestor.cancelarTurnoPorNombre("Ana")); // ya cancelado
36    }
37 }
```

The left sidebar shows the Package Explorer with the `GestorTurnosTest` class selected. Below it, the Failure Trace is empty. The bottom status bar indicates 0 errors, 1 warning, and 0 others.

A la hora de verificarlo no se ha producido ningún error .