**Pensamiento Computacional**

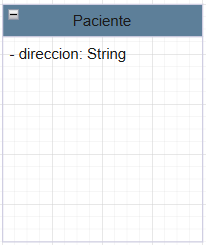
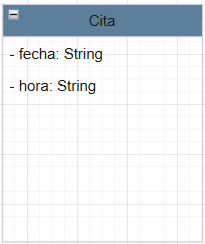
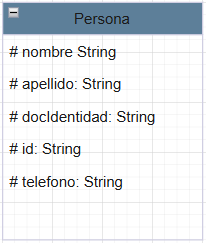
**Abstracción**

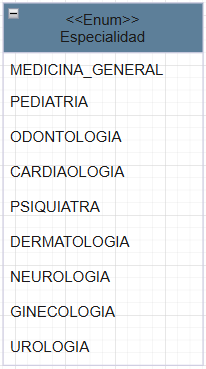
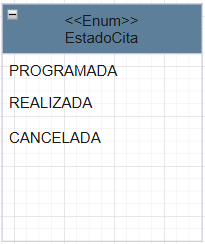
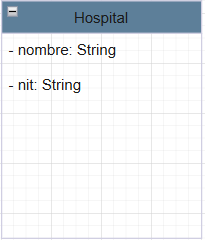
**¿Qué se solicita finalmente?**

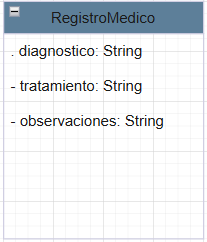
Un programa que permita gestionar y administrar un hospital.

**¿Qué información es relevante?**

Paciente, registrarPaciente,actualizarPaciente, administrarCitas,consultarHistorial, notificacionesCitasProgramadas,Medico,buscarHistorial,registroDiagnosticoYTratamiento,administracionHorarios,modificarCitas, Administradores, registrarMedicoOPaciente,modificarMedicoOPaciente,eliminarMedicoOPaciente,Cita,gestionarSalas,gestionarHorarios, asignarPacientesYMonitorearMedicos, generarReporte

**¿Cómo se agrupa la información relevante?**





**¿Qué funcionalidades se requieren?**

**C**rear paciente, crear médico, crear cita médica, crear historial médico

**R** Consultar lista de pacientes, consultar médicos disponibles, consultar citas médicas programadas, consultar historial médico de un paciente, consultar disponibilidad de salas

**U** Actualizar datos del paciente, actualizar especialidad o datos del paciente, reprogramar cita médica, modificar horarios de atención médica.

**D** Eliminar paciente, eliminar médico, cancelar cita médica, eliminar historial médico.

**Descomposición**

**¿Cómo se agrupan las funcionalidades?**

**Ver diagrama de clases adjunto: “Proyecto Final”**

**¿Cómo pruebo la solución en Java?**

| **Prueba** | **Entrada de datos** | **Salida (Resultados)** |
| --- | --- | --- |
| assertTrue(addPaciente) | boolean pruebaUno= hospital.addPaciente(paciente) | registra el paciente |
| assertTrue(BorrarPaciente) | boolean pruebaUno= hospital.eliminarPaciente(paciente) | elimina el paciente registrado con esa Id |
| assertNull(salaNoExiste) | Sala resultado= hospital.buscarSala(“S99”) | Un nulo |
| assertEquals(BuscarSalaExistente) | Sala resultado= hospital.buscarSala(“S2”) | retorna la sala encontrada en la lista |
| assertTrue(testAgregarSalaExitosamente) | boolean resultado = hospital.agregarSala(new Sala("A101", true)) | Agrega exitosamente una sala nueva |
| assertFalse(testAgregarSalaDuplicada) | hospital.agregarSala(new Sala("A101", true)) dos veces | La segunda vez no la agrega (sala duplicada). |
| assertTrue(modificarSalaExistente) | hospital.modificarSala(new Sala("S1", true)) | Modifica correctamente una sala existente. |
| assertEquals(modificarSalaExistente) | hospital.getListSalas().get(0) | Se verifica que la modificación fue guardada. |
| assertTrue(modificarSalaNoExistente) | hospital.modificarSala(new Sala("S2", false)) | No crea una nueva, sino modifica la ya existente (id igual). |
| assertEquals(modificarSalaNoExistente) | hospital.getListSalas().size() | Confirma que el tamaño de la lista no cambia. |
| assertTrue(testEliminarSalaExistente) | hospital.eliminarSala(new Sala("S1", false)) | Elimina la sala existente correctamente. |
| assertEquals(testEliminarSalaExistente) | hospital.getListSalas().size() | Se verifica que la lista tiene un tamaño menor tras eliminar. |
| assertNull(testEliminarSalaExistente) | hospital.buscarSala("S1") | Ya no encuentra la sala eliminada. |
| assertFalse(testEliminarSalaNoExistente) | hospital.eliminarSala(new Sala("S3", true)) | No elimina nada porque la sala no existe. |
| assertEquals(testEliminarSalaNoExistente) | hospital.getListSalas().size() | El tamaño de la lista no cambia. |
| assertNotNull(testBuscarSalaExistente) | Sala resultado = hospital.buscarSala("S2") | Encuentra y retorna la sala existente. |
| assertEquals(testBuscarSalaExistente) | resultado.getId() | El id de la sala encontrada es igual a "S2". |
| assertNull(testBuscarSalaNoExistente) | Sala resultado = hospital.buscarSala("S99") | No encuentra la sala, retorna null. |