**Abstracción**

**¿Que se solicita finalmente?**

Realizar un sistema de gestión de administración de médicos, pacientes y administradores el cual el objetivo es hacer un programa que optimice el uso y manejo de datos de estos mismos

**¿Qué información es relevante dado al problema anterior?**

Administrador Usuario <abstract> Medico

-Nombre \*nombre -nombre

-Id \*id -Id

-Correo \*correo -correo

-Telefono \*Telefono -Telefono

-Edad \*Edad -Edad

TipoSala<<Enum>> Paciente Hospital

PacientesPrioritarios -nombre -sin atributos

PacientesGeneral -id asignados

PacientesEstadoGrave -correo

-Telefono

-Edad

Sala

-NumeroSala

-Estado

-TipoSala

CitaMedica HistorialMedico

-Horario -Fecha

-Especialidad -Estado

-Sala -NumeroMedicamentos

-Medico -Enfermedad

-MotivoConsulta -Antecedentes

-Atendida -Edad

-Disponibilidad -Diagnostico

-IdPaciente -Tratamiento

HorarioConsulta <<interface>>

+establecerHorarioDisponible()

Notificacion <<interface>>

+ notificar()

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

|  |
| --- |
| **Administrador** |
| -String nombre  -String id  -String correo  -String teléfono  -int edad |

|  |
| --- |
| **Usuario**  **<<abstract>>** |
| \*String nombre  \*String id  \*String correo  \*String telefono  \*int edad |

|  |
| --- |
| **Medico** |
| -String nombre  -String id  -String correo  -String teléfono  -int edad |

|  |
| --- |
| **TipoSala**  **<<Enum>>** |
| PACIENTESPRIORITARIOS  PACIENTESGENERAL  PACIENTESESTADOGRAV |

|  |
| --- |
| **Paciente** |
| -String nombre  -String id  -String correo  -String teléfono  -int edad |

|  |
| --- |
| **Hospital** |
| Sin atributos asignados |

|  |
| --- |
| **Sala** |
| -String numeroSala  -boolean estado  -TipoSala tipoSala |

|  |
| --- |
| **CitaMedica** |
| -LocalDateTime horario  -String especialidad  -String sala  -String medico  -String motivoConsulta  -boolean atendida  -boolean disponibilidad  -String idPaciente |

|  |
| --- |
| **HistorialMedico** |
| -LocalDateTime fecha  -String estado  -int numeromedicamentos  -String enfermedad  -String antecedentes  -int edad  -String diagnostico  -String tratamiento |

|  |
| --- |
| **HorarioConsulta**  **<<interface>>** |
| +String establecerHorarioDisponible() |

|  |
| --- |
| **Notificacio**  **<<interface>>** |
| +String notificar() |

**¿Qué funcionalidades se solicitan finalmente?**

Realizar un sistema de gestión de hospital donde se maneje diferentes tareas los médicos pacientes y administradores

**Descomposición**

**¿Cómo se distribuyen las funcionalidades?**

En la carpeta model del repositorio se encuentra el diagrama de clases

**Codificación**

**¿Como escribo la solución en java?**

En la carpeta code está el código generado con todas las clases y métodos hechos

**¿Qué debo hacer para probar las funcionalidades?**

|  |  |  |
| --- | --- | --- |
| **Prueba** | **Entrada de datos** | **Salida** |
| asssertEquals(3, listaAgregada) | Debe validarse de que al momento que el admin agregue 3 listas de citas medicas tengan el mismo tamaño | Lanzar true; |
| assertnotNull(ListCitasMed  cas) | Debe validarse de que la lista de citas medicas que el admin agregue no sea vacia | Pasar la prueba y lanzar true |
| assertTrue(admin.registrar  medico)  assertEquals(1, admin.get  ListMedicos) | Debe validarse de que al momento de que el medico registrado sea true y el tamaño de esa lista sea el mismo | Lanzar true, y pasar la prueba con el equals  ya que el tamalo al agregarse la lista si  corresponde |
| assetFalse(admin.registrar  medico)  assertEquals(admin.getList  Medicos()) | Debe validarse de que  Al registrar un medico no tenga edad negativa y que no haya médicos agregados | Lanzar true, y pasara la prueba pues el tamaño de la lista no puede ser mas de 0 por la edad negativa |
| assertTrue(modificado)  assertEquals(medicoModi  ficado) | Debe validarse de que  Al modificar un medico lance true pues el método es de tipo boolean y que la lista obtenga ese medico nuevo | Lanzar true y pasar la  Prueba ya que al modificar el medico ese seria el medico que debería estar en la lista |
| assertFalse(modificado) | Debe verificarse de que al modificar un medico se necesite al menos dos, no modificar ese mismo | Lanzar true y pasar la prueba porque se verifica que al menos haya 2 medicos para modificarlos |
| assertTrue(eliminado) | Al momento de eliminar un medico se debe validar de que al obtener la lista no haya mas medicos | Lanzar true ya que no deben haber mas médicos y la lista debe tener 0 medicos al momento de eliminar |
| assertFalse(eliminado) | Al momento de eliminar un medico debe verificarse de que al menos haya sido agregado | Lanzar false y pasar la prueba ya que se espera de que haya sido agregado al menos a la lista |
| assertTrue(registrado) | Al momento de registrar un paciente debe lanzar true y verificar que el paciente cumpla con el tamaño de la lista | Lanzar true y pasar la prueba |
| assertFalse(pacienteregistr  ado) | Debe validarse de que  Un paciente no tenga  Edad 0 ya que al menos debería tener 1 año, o si son meses tocaria ver en el constructor esas condiciones | Lanzar false y pasar la prueba ya que no puede tener edad 0 |
| asserTrue(modificado) | Debe validarse de que el paciente haya sido modificado correctamente y validar esa lista correctamente el tamaño | Lanzar true y pasar la prueba |
| assertFalse(modificado) | Debe validarse de que al menos haya dos pacientes para que el admin los modifique | Debe pasar la prueba y validar que no se modifique a si mismo |
| assertFalse(pacienteElimina  do) | Debe validarse de que  No se pueda eliminarse un paciente que no ha sido agregado a la lista | Debe pasar la prueba y lanzar false ya que  Es algo que no debería pasar |
| assertTrue(String sala disponible) | Debe validarse de que  Al utilizar el método de ocupación este el mensaje de que la sala esta disponible | Debe pasar la prueba ya que la ideal es que al estar disponible, entonces pase la prueba |
| assertNotNull(resultado) | Debe validarse de que  Al momento de cambia una cita entonces que esa lista no sea nula y que la disponibilidad no puede estar ya disponible | Pasar la prueba y asegurarse de que la lista no sea nula |
| assertNull(metodoCambio  Cita) | Debe validarse de que  Al utilizar el método de cambiar cita haya al menos dos citas agregadas | Pasar la prueba y validar que sea nula ya que tiene que al menos tiene que existir dos citas medicas |

**Reconocimiento de patrones**

**¿Qué puedo reutilizar?**

No hay nada por reutilizar

**Codificación**

**¿Cómo pruebo la solución en java?**

En la carpeta code en la parte de test están las pruebas del código