Trabajo Práctico Grupal (TPG)

Asignatura: Programación 3

Grupo Nº 4

Integrantes:

Bengoa, Sebastián
Echeverría, Noelia
Ezama, María Camila
Mateos, Juan Cruz

Introducción

Este trabajo tiene como propósito la generación de un sistema de gestión para una clínica privada. El sistema será implementado, utilizando el lenguaje de programación Java y el paradigma de programación propuesto por la cátedra es el Orientado a Objetos. Dependiendo del contexto en donde estén enmarcadas cada una de las problemática asociadas a los módulos del sistema, se utilizarán distintos patrones de diseño. El programa resultante debe poder ser extensible, dado que se espera que a futuro se agreguen nuevos requerimientos y/o funcionalidades.

Requerimientos y descripción del sistema implementado

Descripción

Se requiere el desarrollo de módulos de software para incorporar al sistema de gestión de una pequeña clínica privada. Dichos módulos son: Facturación al paciente, Reportes de actividad de los médicos (con suma de honorarios), Resolución de conflictos de sala de espera.

Información relevada

Clínica

La información de la clínica requerida es : nombre, dirección, teléfono, ciudad.

Médicos

La información del médico requerida es: dni, número de matrícula, nombre y apellido, domicilio, ciudad, teléfono, especialidad.

Existen tres tipos de especialidad: Clínico, Cirujano, Pediatra.

Existen dos tipos de contratación: Planta Permanente, Temporario (residente)

Los médicos pueden tener título de posgrado: Magister, Doctor.

Pacientes

La información del paciente requerida es: dni, número de historia clínica, nombre y apellido, rango etario, teléfono, domicilio y ciudad.

Los pacientes se clasifican según su rango etario en: Niños, Jóvenes y Mayores.

Al ingresar a la clínica se los ubica en la *Sala Vip (SV)* o en el *Patio(P)*. En la *SV* puede haber una única persona. Si *SV* está vacía, la persona ingresante se ubica en la misma; si está ocupada se elije cual paciente va *P* y cuál permanece en *SV*, de acuerdo al siguiente criterio:

- Entre un niño y un joven, el niño queda en *SV* y el joven va *P*.
- Entre un joven y un mayor, el joven queda en *SV* y el mayor va *P*.
- Entre un mayor y un niño, el mayor queda en *SV* y el niño va a *P*.

Habitaciones

La clínica cuenta con las siguientes habitaciones:

- Habitaciones compartidas
- Habitaciones privadas
- Salas de terapia intensiva

Cálculo de honorarios médicos

Los médicos tienen asignado un honorario básico, que se ve modificado según la especialidad, posgrado y tipo de contratación.

Según especialidad

- Clínica: 5% de aumento
- Cirugía: 10% de aumento
- Pediatría: 7% de aumento

Según tipo de Contratación

- Planta Permanente: 10% de aumento sobre el honorario que incluye la especialidad y el posgrado.
- Temporario: 5% de aumento sobre el honorario que incluye la especialidad y el posgrado.

Según Posgrado

Para los médicos contratados:

- Magister: 5% de aumento sobre el honorario que incluye la especialidad.
- Doctorado: 10% de aumento sobre el honorario que incluye la especialidad.

Cálculo de aranceles por internación

La clínica cobra un costo inicial por derecho a internación y luego un arancel que depende del tipo de habitación y la cantidad de días de internación.

Valores por Internación

1. Costo de asignación de habitación: <costo asignación>
2. Adicionales
    a. habitación compartida: por cada día de internación se adiciona un valor (Costo Hab. Compartida). El valor será proporcional a la cantidad de días internado.
    b. habitación privada: el costo se calcula de la siguiente forma:
        i. 1 día de internación cuesta: 1 Costo Hab Privada
        ii. 2 a 5 días de internación cuestan: cantidad de días * 1 Costo Hab Privada * 1,3
        iii. 6 o más días de internación cuestan: cantidad de días * 1 Costo Hab Privada * 2
    c. terapia intensiva: el costo tiene un crecimiento potencial y se calcula de la siguiente forma: $(1 \text{ Costo Terapia Intensiva})^{\text{cantidad de días}}$

Descripción de los Módulos, funcionamiento de la clínica

Ingreso del paciente a la clínica

El paciente llega a la ventana de atención al cliente de la clínica y se le otorga un número de orden. Luego es derivado a la Sala de Espera Privada o al Patio, según corresponda. A partir de ese momento el Paciente es atendido cuando lo llamen.

Módulo de Ingreso y Módulo de Resolución de Conflictos

El módulo de ingreso dará de alta al paciente (sí es la primera vez que ingresa a la clínica) o lo ubicará de la lista de pacientes y lo pondrá en la lista de espera según su orden de llegada, también determinará sí se lo deriva a la Sala de Espera Privada o al Patio.

Atención del paciente

El paciente es atendido y se lo retira de lista de espera. También se lo retira de la Sala de Espera Privada o del Patio.

Módulo de Atención

El módulo solamente retira al paciente de la espera y lo ubica en la Lista de Pacientes en Atención.

Egreso del paciente de clínica y Facturación al Paciente

Módulo de Egreso y Módulo de Facturación

El módulo de egreso permitirá elegir un paciente de la Lista de Pacientes en Atención, y confeccionará la factura correspondiente, ingresando todas las prestaciones recibidas: Médico que lo atendió, días de internación, habitación. Luego se lo retira de la Lista

La factura de cada paciente tendrá la siguiente información: número de factura (autoincrementable), fecha, paciente, listado de prestaciones recibidas cada una con su importe (honorario médico, internación), importe total. El honorario médico que se le cobra al paciente sufrirá un 20% de incremento por sobre lo que el Médico cobra.

Cada línea de factura mostrará la siguiente información:

Prestación	Valor	Cantidad	Subtotal
Nombre médico	Valor de la consulta	Cantidad de consultas	Subtotal
Habitación	Costo	Cantidad de días de internación	Subtotal

Reporte de Actividad Médica

Se debe reportar la actividad de un médico por día, enumerando los pacientes atendidos (con la cantidad de consultas practicadas a cada paciente).

El reporte se solicita por periodos (desde - hasta) mostrando en forma cronológica las consultas realizadas por el médico, con el nombre del paciente. La fecha de la consulta es la fecha de facturación. Se debe mostrar el honorario de cada consulta y la suma total.

Desarrollo de la Aplicación

Lenguaje Java, IDEs utilizados y diagramas UML

El desarrollo de la aplicación de gestión de clínica fue programado en lenguaje Java (paradigma orientado a objetos), utilizando distintos IDEs.

La Elección de los IDEs fue personal de cada uno de los integrantes del grupo y en algunos casos determinada por la necesidad de utilizar alguna herramienta mejor desarrollada en un IDE que en otro.

Todos los diagramas UML fueron desarrollados utilizando JDeveloper. Debido a que los proyectos generados en Eclipse/IntelliJ/JDeveloper no son del todo compatibles entre sí, trabajamos transformando todos los proyectos a Maven.

Clases y relaciones de herencia y/o composición

Clase Clínica y patrón singleton

Se considera que la clínica es única y por lo tanto no debería ser instanciada más de una vez, por lo cual se decidió implementarla utilizando un patrón singleton.

Esta clase es la responsable de toda la logística relacionada a la atención de los pacientes, la facturación y los honorarios de los médicos.

Es la encargada de llevar el registro de los médicos que trabajan en ella (datos personales, tipo de contratación, especialidad y posgrado). Se decidió implementar este registro utilizando un ArrayList de médicos (ArrayList<IMedico>) siendo la relación entre la clínica y los médicos una asociación fuerte o relación de composición (la clínica tiene una lista de médicos y si la clínica desaparece, esta lista también).

La clínica además tiene a los componentes ModuloIngreso, ModuloAtencion y ModuloEgreso, que son clases en las que delega ciertas acciones relacionadas al, valga la redundancia, ingreso, atención y egreso de pacientes.

Se adjunta un archivo .pdf con el diagrama UML completo de todo el sistema implementado, pero en la Fig. 1 se muestra un breve extracto del mismo.

Se debe destacar que en la última versión que se generó de la clase Clinica, se utilizó un hashmap de IMedicos. Al generar el diagrama UML con esta modificación, el software Jdeveloper no genera la relación de composición entre las clases. Se adjuntan capturas de pantalla de una prueba realizada aparte que da muestra de este bug (?) del Jdeveloper.

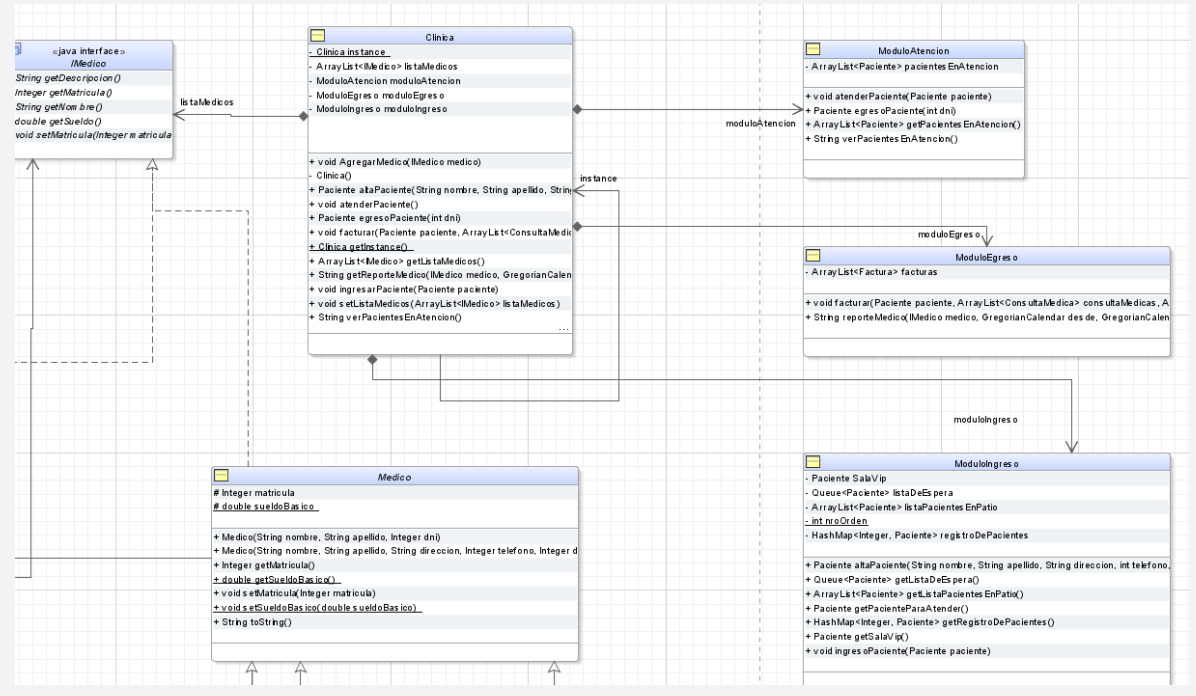

Fig. 1 - Relación de composición entre la Clínica (singleton) y distintas clases/interfaces.

Clase Persona y abstracción de características

Dado que los pacientes y los médicos comparten algunas características como tener nombre, apellido, dirección, teléfono y dni, se decidió generar una clase Persona que reúne estas características comunes. Posteriormente, se diagramaron las clases Médico y Paciente que extienden a la clase abstracta Persona. En la Fig. 2 se muestra un diagrama UML donde se puede visualizar la jerarquía de herencia entre estas clases.

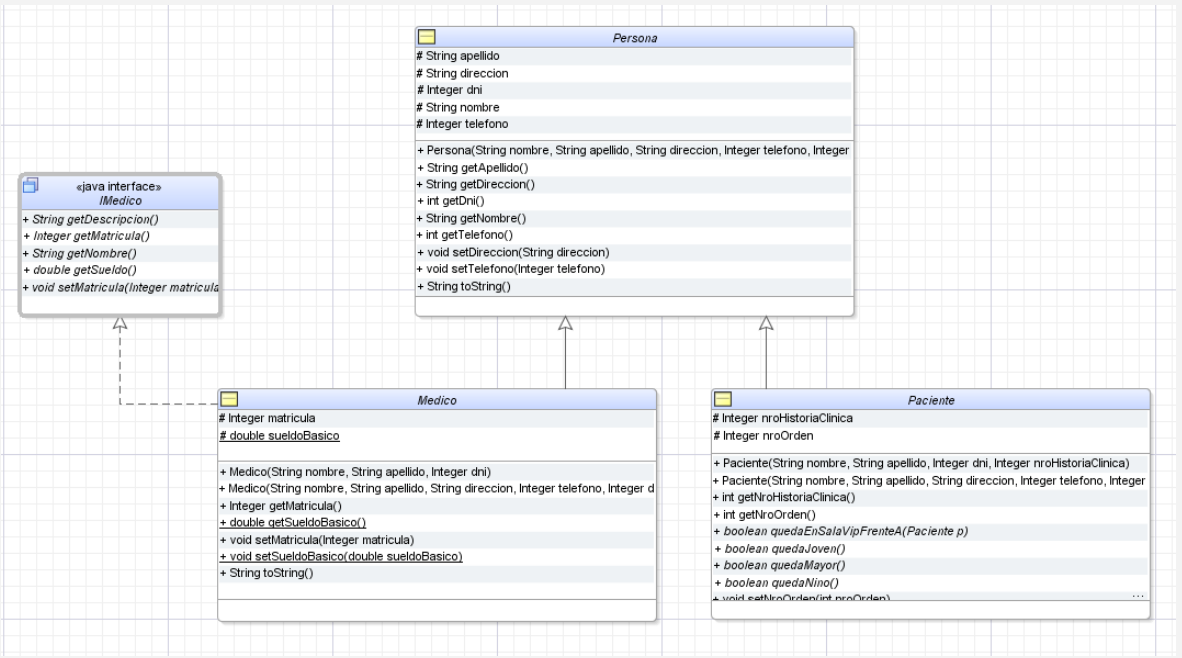

Fig. 2 - Diagrama UML de la clase abstracta Persona y sus dos clases hijas (también abstractas), Médico y Paciente

Modelado de los Médicos (interfaz y decoradores)

Para el modelado de los médicos con sus respectivas especialidades (pediatra, cirujano y clínico) se generaron 3 clases (MedicoPediatra, MedicoCirujano y MedicoClinico) las cuales heredan de una clase abstracta Medico que extiende a la clase Persona e implementa la interfaz IMedico (Fig. 3). Dentro de cada una de estas clases se modelan los comportamientos especializados, siendo el más importante para este trabajo práctico el de obtener el sueldo de cada médico.

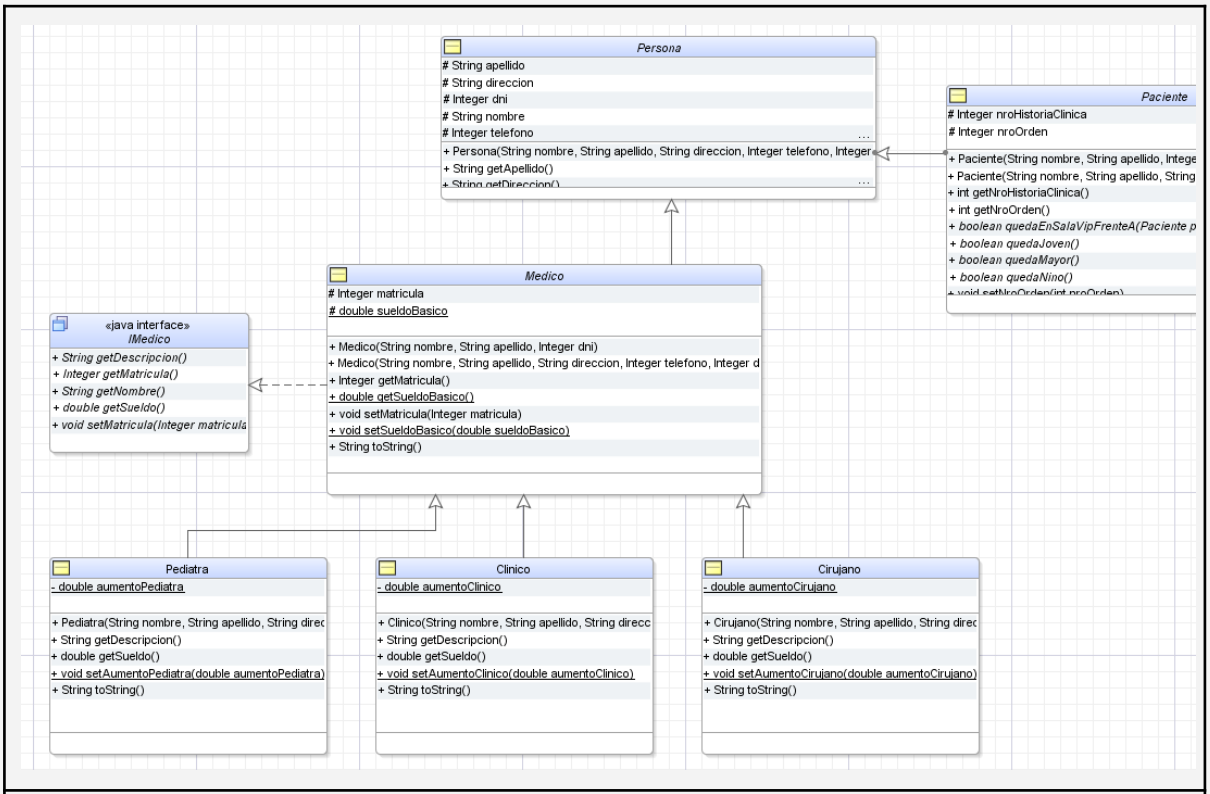

Fig. 3 - Clase abstracta médico y 3 clases concretas que heredan de ella.

Debido a que el sueldo de cada médico se ve afectado no solo por la especialidad, sino también por el título de posgrado (Doctor o Master) y el tipo de contratación (permanente o temporario), teniendo además en cuenta todas las posibles combinaciones (por ejemplo, Médico Cirujano con Doctorado y contratación permanente o Médico Pediatra con Master y contratación temporal) y con el fin de evitar una explosión de clases, se decidió utilizar un patrón decorator para modelar el comportamiento referente al sueldo (Fig. 4).

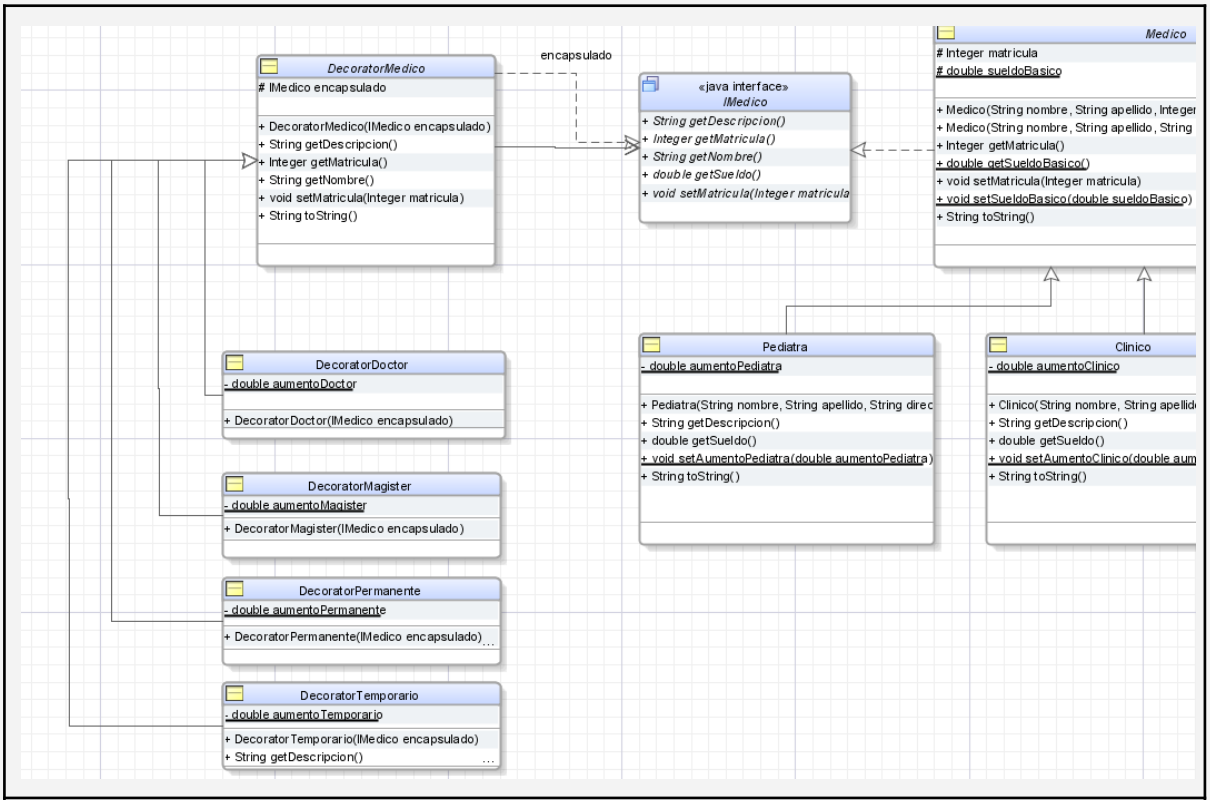

Fig. 4 - Decorator abstracto (DecoratorMedico) y sus 4 clases hijas (Decorators concretos)

Instanciación de nuevos médicos y patrón factory

El agregado de nuevos médicos, con su especialidad, tipo de contratación y título de posgrado (decoradores) se realizó utilizando un patrón factory, es decir se delegó en una clase especializada la generación de nuevas instancias de médicos.

Se decidió que dentro de la factoría de médicos, se puedan generar excepciones relacionadas con ingresar erróneamente el tipo de posgrado, el tipo de contratación o bien campos null o cadenas vacías. Posiblemente cuando se genere la interfaz gráfica, las excepciones relacionadas al posgrado y a la contratación no sean necesarias, pero con la información disponible al momento de generar este informe trabajar con excepciones parecía adecuado. Otra opción (que no se implementó) podría haber sido no lanzar excepciones y agregar pre-requisitos y validaciones en un nivel superior.

Modelado de Pacientes y double dispatch

Para el modelado de los pacientes, en primer lugar se generó una clase abstracta Paciente que extiende de Persona y posteriormente se generaron 3 clases hijas concretas diferenciadas por el rango etario (PacienteNino, PacienteJoven, PacienteMayor).

La decisión de generar estas 3 clases se fundamentó en la posibilidad de arbitrar la resolución de los conflictos relacionados a la sala Vip y al patio utilizando el mecanismo de double dispatch. Cuando dos pacientes coinciden en la sala Vip, la

elección de quien permanece y quien va al patio, depende de la interacción entre los dos objetos (instancias de Paciente).

Módulo de Ingreso

El módulo de ingreso tiene varias tareas, todas relacionadas al ingreso del paciente a la institución.

En primer lugar, tiene la responsabilidad a través del método altaPaciente, de buscar un paciente en los registros históricos de clínica y en caso de encontrarlo, devolver una referencia al objeto Paciente creado con anterioridad. En caso de que el paciente no se encuentre registrado, crea un nuevo objeto, guarda su referencia en el registro histórico y devuelve una referencia a este nuevo objeto creado.

En segundo lugar, se da ingreso al paciente a través del método ingresoPaciente, asignándole un número de orden para poder ser atendido cuando corresponda y poniéndolo en cola para ser atendido. Posteriormente, se pasa al paciente a la sala VIP o al Patio, delegando la resolución de conflictos si los hubiera en los pacientes mismos como se mencionó previamente utilizando el mecanismo de double dispatch.

Por último, utilizando el método getPacienteParaAtender, se obtiene la referencia al próximo paciente a ser atendido. Esto se puede hacer llevando un registro del próximo número de orden a ser atendido o utilizando una cola (Queue), siendo este último el método elegido para llevar a cabo esta tarea.

Se debe mencionar que si bien es el módulo de ingreso el responsable de realizar estas tareas, es a través de la clase Clínica que se gestionan las mismas (la clínica delega esta responsabilidad, pero es la encargada de hacer los llamados a los métodos correspondientes).

Módulo de Atención

Las responsabilidades del modulo de atencion son únicamente dos: ingresar un paciente para que sea atendido y, una vez finalizada su atención, removerlo para del módulo a fin de que pase al sector de egreso. Este comportamiento es modelado mediante dos métodos. El método atenderPaciente(Paciente) recibe un paciente (que proviene del módulo de ingreso; acción gestionada por la clínica) y lo agrega a la lista de pacientes en atención. Finalmente, cuando termina el proceso de atención de un paciente, este puede ser removido de la lista de atención mediante el método egresoPaciente(int) que, dado un dni, busca al paciente cuyo dni es el indicado y lo retorna removiéndolo de la lista de pacientes en atención. Este comportamiento de remover pacientes de la lista de atención buscando por dni será reemplazado en la segunda parte del trabajo que, al implementar una interfaz de usuario, permitirá escoger visualmente al paciente de una lista haciendo click sobre él.

Módulo de Egreso

Entre las responsabilidades del módulo de egreso se encuentran la facturación a los pacientes, la búsqueda de una determinada factura a fin de ver su detalle y la generación de los reportes médicos ordenados cronológicamente. Este módulo, además, es el encargado de mantener el registro de todas las facturas y, dado que al momento de generar el reporte médico es de gran utilidad que dicho registro se encuentre ordenado cronológicamente, se especializó la clase ArrayList en un nueva clase SortedArrayList, sobreescribiendo los métodos add() e iterator(), a fin de que las inserciones en la estructura se mantuvieran ordenadas.

En el caso de la facturación, para generar la factura un paciente basta con indicar el paciente, la fecha de facturación y los listados de consultas médicas e internaciones. Por otro lado, el detalle de una determinada factura se obtiene suministrando el número de factura; si no se encuentra ninguna factura registrada con ese número la salida es indefinida. Por último, el reporte médico se obtiene indicando un médico y dos fechas (desde - hasta) que serán los márgenes dentro de los cuales se buscará las actividades del médico indicado.

Conclusión

Durante el desarrollo de este trabajo y debido a su complejidad hemos podido explorar distintos patrones de diseño, tales como el Singletón, el Decorator, Factory entre otros, los cuales nos han ayudado a desarrollar las tareas asignadas de una forma más estructurada.

Hemos aprendido la importancia de generar código con posibilidad de ser extendido, sin la necesidad de hacer un refactor completo del mismo. Esto último ha sido de vital importancia principalmente a la hora de dividir el trabajo para acelerar los procesos y posteriormente poder unir sus partes.

Ha sido de mucha utilidad ir documentando cada una de las clases y métodos, como así también las justificaciones en la elección de los patrones de diseño utilizados, permitiendo que el trabajo en equipo sea más ameno y ágil.

Por último, ha sido desafiante y a la vez gratificante aprender a distribuir las tareas, siendo de vital importancia la puesta en común de criterios a la hora de generar el código.