Trabajo grupal
CLÍNICA

Facultad de Ingeneria
de la Universidad Nacional
de Mar del Plata

Profesores:

- Gellon, Ivonne
- Guccione, Leonel Domingo
- Lazzurri, Carlos Guillermo

Integrantes:

- Fernandez Ordoqui, Julian Agustin
- Stufano, Nazareno
- Ganduglia, Ciro Exequiel
- Teruel López, Leonel
- Pezzi, Esteban

1- Introducción

El presente informe documenta el desarrollo de un Sistema de Gestión para una Clínica Privada, implementado en Java. El sistema abarca tres módulos principales: Facturación de Pacientes, Reportes de Actividad Médica y Resolución de Conflictos en Sala de Espera.

2- Arquitectura del sistema

Estructura del proyecto

El proyecto sigue una arquitectura organizada por los siguientes paquetes:

- **modelo/Clinica**: cara visible del sistema. Expone los métodos que permiten el funcionamiento (registrar, ingresar, atender, internar, egresar) y coordina todo.
- **modelo/interfaces**: contratos que definen los diferentes comportamientos (IPersona, IPaciente, IMedico) y permiten el polimorfismo.
- **modelo/paciente:** diferentes pacientes por rango etario (Nino, Joven, Mayor) y la lógica de prioridad en la sala.
- **modelo/Decorator**: las “capas” que sele agregan a un IMedico para calcular honorarios según su especialidad, forma de contratación y estudios.
- **modelo/Factory**: factory para abstraer la creación de médicos (combinando los decoradores).
- **modelo/Habitacion**: tipos de internación y su forma de calcular los costos (compartida, privada, terapia intensiva).
- **modelo/Factura**: arma el detalle y el total de lo que debe abonar el paciente.

Las siguientes clases complementarias:

- **Domicilio**, **ConsultaMedica**, **Internacion**, **SalaEsperaPrivada**

Y una excepción:

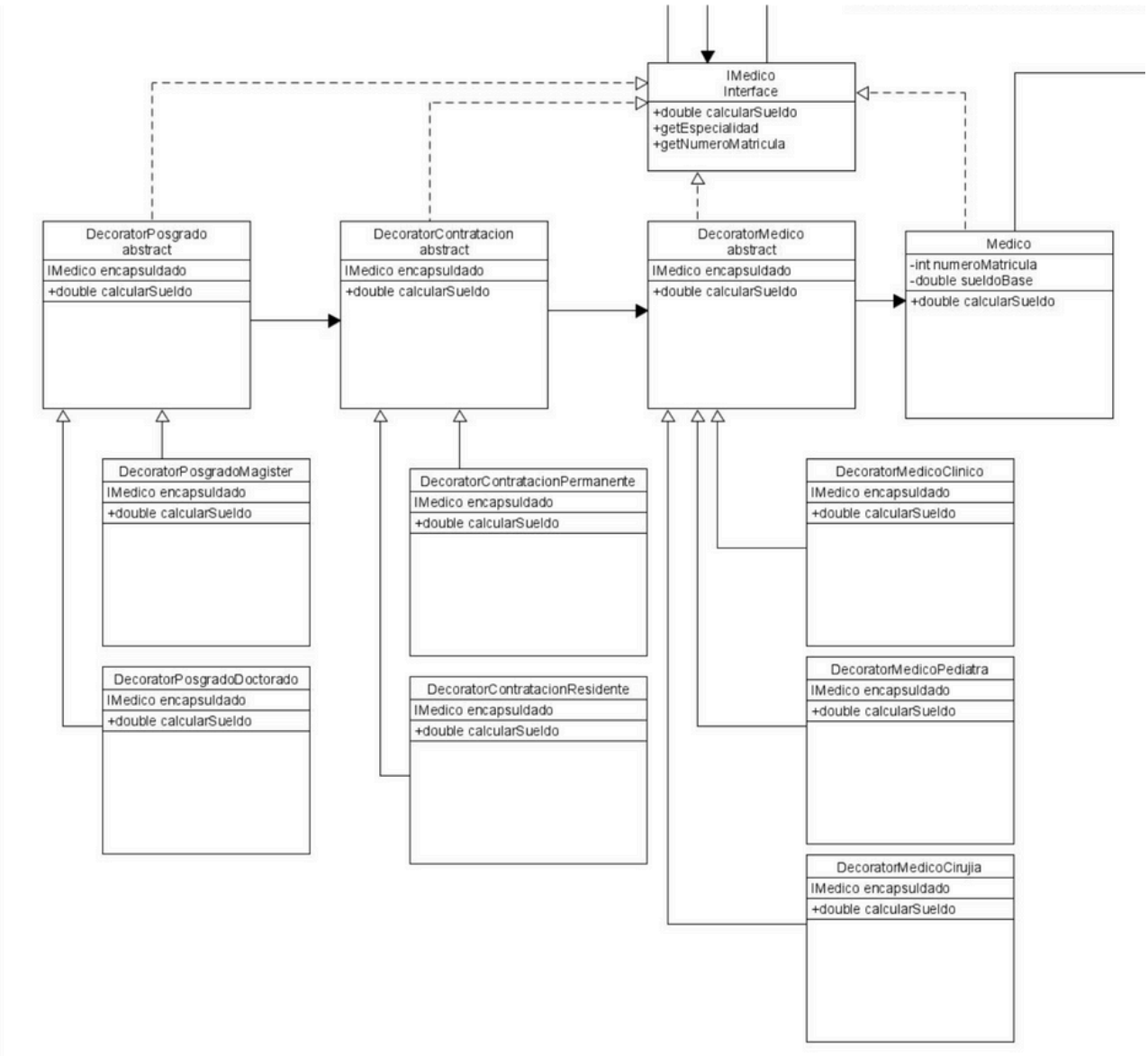
- **NoExistePacienteException**: arroja una excepción cuando se ingresa un paciente que no esta ingresado en el establecimiento.

3- Patrones de diseño

3.1 Patrón decorator

Objetivo: Calcular honorarios médicos aplicando incrementos dinámicos según especialidad, tipo de contratación y estudios.

Implementación:



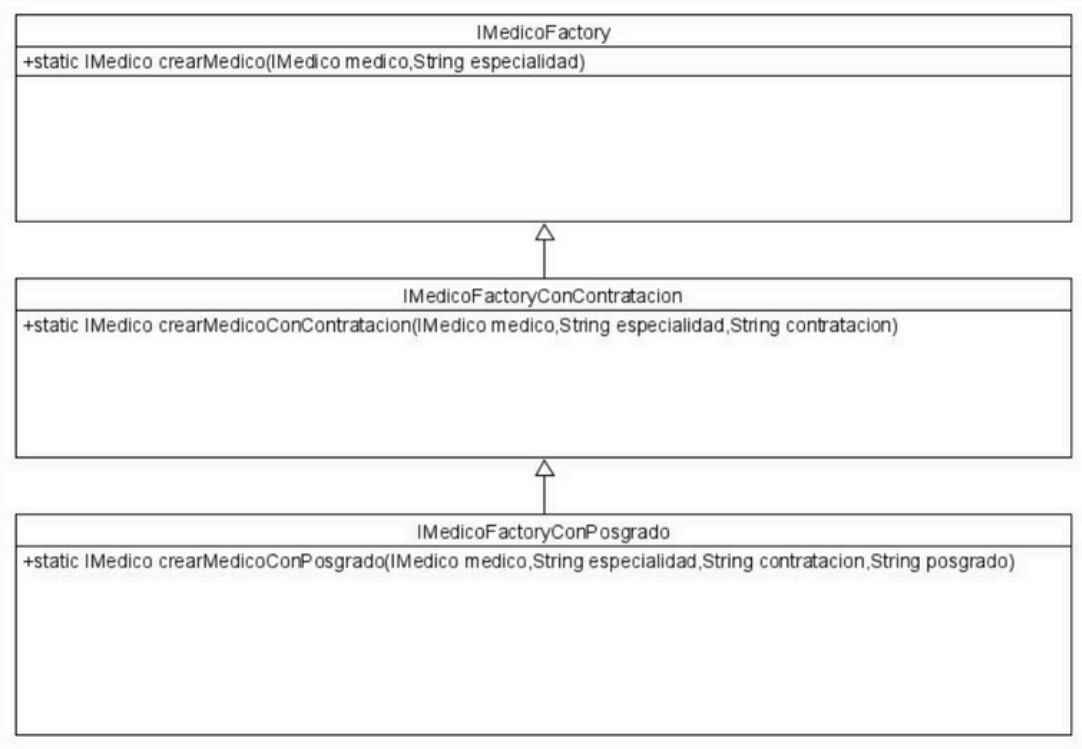
Ventajas:

- Permite combinar muchas características sin implementar una cantidad enorme de subclases.
- Fácil extensión para nuevas especialidades o tipos de contratación.
- Cálculo de honorarios variable según especialidad y contratación, con una base en la clase concreta.

3.2 Patrón Factory

Objetivo: Simplificar la creación de médicos con múltiples decoradores aplicados.

Implementación:



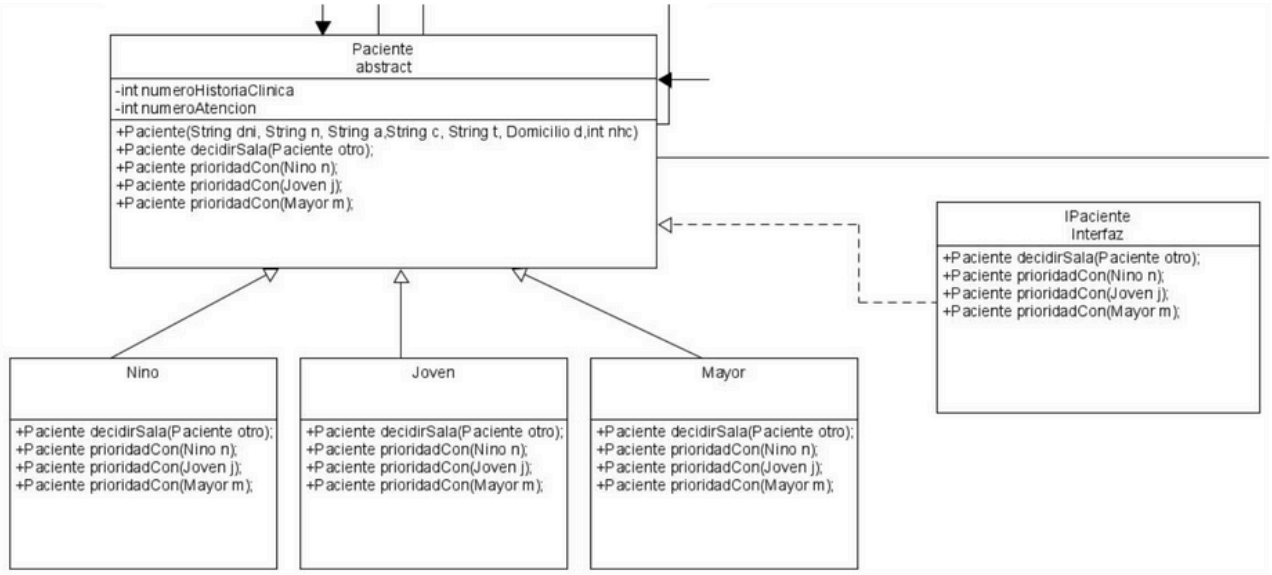
Ventajas:

- Permite agregar nuevas variantes sin modificar el código.
- Funciona como una sola entrada para la creación y asegura invariantes al crear objetos.

3.3 Patrón Double Dispatch

Objetivo: Resolver prioridades en la sala de espera privada según el rango etario de los pacientes

Implementación:



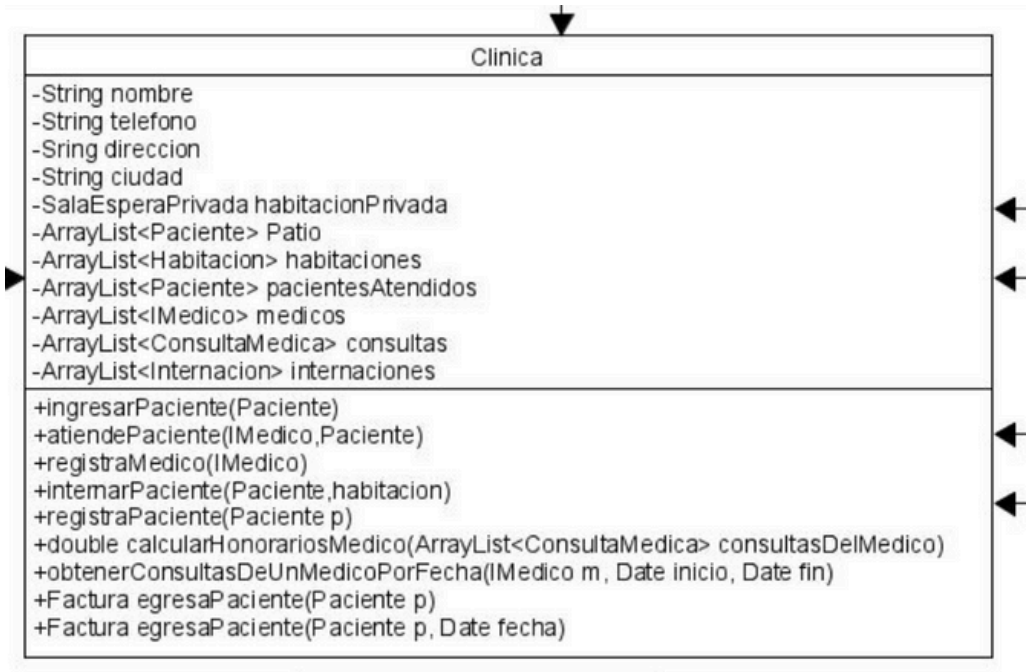
Ventajas:

- Evitamos la utilización de condicionales anidados.
- Facilita agregar nuevos rangos etarios.
- Código mantenible y fácil de leer.

3.4 Patrón Facade

Objetivo: Simplifica la interacción con el sistema mediante una interfaz unificada.

Implementación:



4. Modelo de negocio

4.1 Jerarquía de clases principales

Personas:

IPersona (interface)
↑
Persona (abstract)
↑
├── Medico → IMedico
└── Paciente (abstract) → IPaciente
├── Nino
├── Joven
└── Mayor

Persona: clase base con datos comunes.
Medico: médico base sin especialización.
Paciente: numero de historia clínica y orden de llegada.

Habitaciones:

Habitacion (abstract)
↑
├── HabitacionCompartida
├── HabitacionPrivada
└── SalaTerapiaIntensiva

HabitacionCompartida: días x precioBase
HabitacionPrivada:

- 1 día: precioBase.
- 2-5 días: dias * precioBase * 1,3.
- >= 6 días: dias * precioBase * 2.

4.2 Clases de soporte

Domicilio: Encapsula calle y número.
ConsultaMedica: Registra la atención médica con fecha.
Internacion: Gestiona pacientes internados.
Factura: Genera factuas con número autoincremental.
SalaEsperaPrivada: Maneja la sala VIP (1 solo paciente).

5. Funcionalidades principales

5.1 Gestión de Pacientes

Flujo de atención:

1. **Registro: registrarPaciente()** - Alta en el sistema
2. **Ingreso: ingresarPaciente()** - Asignación de número de orden y ubicación (Sala privada o Patio)
3. **Atención: atiendePaciente()** - Retiro de espera y registro de consulta médica
4. **Internación (opcional) : internarPaciente()** - Asignación de habitación
5. **Egreso: egresaPaciente()** - Generación de factura

5.2 Sistema de facturación

La clase **Factura** genera documentos con:

- Número de factura autoincremental.
- Datos del paciente.
- Fechas de ingreso y egreso.
- Detalle de consultas médicas.
- Costos de internación (si aplica).
- Total general

Formato de salida

N° Factura: 3
Nombre Paciente: Julian Ganduglia
Fecha Ingreso: 2025/10/10
Fecha Egreso: 2025/11/14
Cantidad de días: 35
Habitación tipo: Compartida Costo: $ 350000.0

Consultas Médicas:
Médico: Juan Alvarez
Especialidad: Clinico
Subtotal: $ 138,92

Total: 350138,92

5.3 Reportes de actividad médica

El método **calcularHonorariosMedico()** en **Clinica** permite:

- Calcular honorarios totales del período

El método obtenerConsultasDelMedicoPorFecha() se complementa con el anterior y:

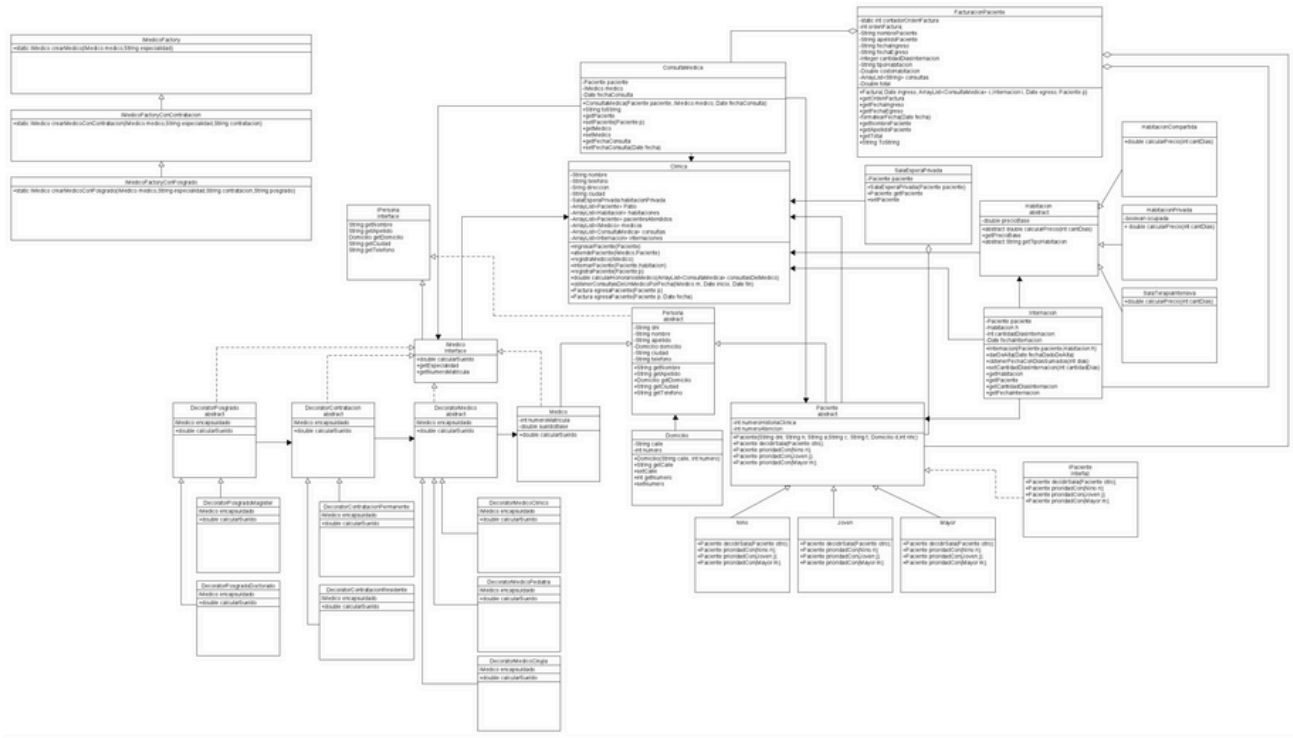
- Permite filtrar consultas por médico y rango de fechas

6. Manejo de Excepciones

Se implementó **NoExistePacienteException** para:

- Validar que el paciente esté registrado antes de ingresar.
- Verificar la existencia del paciente al momento de atender
- Proporcionar mensajes de error descriptivos

7. Diagrama de clases UML



Puede verlo también en este link

8. Conclusiones

El sistema implementa los tres modulos requeridos:

- Facturación: Generación automática con cálculos complejos.
- Reportes médicos: Filtrado y resumen de honorarios.
- Gestión de espera: Resolución de conflictos mediante double dispatch.

Los patrones de diseño aplicados proporcionan:

- Facilidad para agregar nuevas especialidades y tipos de habitación.
- Código organizado, fácil de mantener
- Escalable, preparado para futuras extensiones.