

Programación III

Informe Trabajo Práctico Grupal (TPG)

“Gestión de una Clínica”

Integrantes:

- Cazorla Martínez, Nicolás
- Dell Aguila Ureña, Franco
- Maletta, Augusto
- Rodríguez, Juan Gabriel

Universidad Nacional de
Mar del Plata
Facultad de Ingeniería.



Informe Trabajo Práctico Grupal (TPG) “Gestión de una clínica”

Flujo completo desde el ingreso hasta la salida de un paciente

A continuación se detalla brevemente cual es el flujo esperado y cuáles son los métodos que interactúan al ingresar un paciente a la clínica hasta el momento de retirarse.

Ingreso del paciente

Se ingresa un paciente a la clínica a partir del método ingresar paciente de la clínica

```
public void ingresarPaciente(String nombre, String apellido, int DNI, String rangoEtario)
```

A partir de este método, se resuelve el conflicto entre la sala de espera y el patio, para determinar donde se ubica el paciente.

También se le asigna al paciente un numero de orden (auto-incremental a medida que llegan nuevos pacientes) que será el que determine el orden de atención de los pacientes. El primer paciente en llegar será el primero en ser atendido.

Atención del paciente

En un momento determinado, se llama al siguiente paciente en espera a ser atendido, esto se resuelve a partir de un método del módulo de atención.

```
public void retiraPaciente() throws EsperaVacíaException {
```

El método quita al paciente de la sala de espera y lo pone en una lista de pacientes en atención, en este momento los médicos pueden realizar



consultas a un paciente dado te dado o internar al paciente en una o varias habitaciones hasta que se retire de la clínica. Internar al paciente en una o varias habitaciones hasta que se retire de la clínica.

Facturación y egreso del paciente

Se selecciona un paciente de la Lista de Atención tomando como clave su dni
Se le confecciona una factura al paciente con la información pertinente
Se trabaja con un hashmap para llevar el conteo de las consultas realizadas por cada médico. También se tiene en cuenta sumarle el honorario correspondiente al médico por la consulta.
Se utiliza un iterator para recorrer todas las consultas efectuadas por el paciente en la última visita, así también como para conocer los servicios utilizados en las mismas
Se muestran por pantalla los datos correspondientes a la factura en forma de tabla
Se le da el alta al paciente (se lo retira de la Lista de atención)

```
public void creaFacturapaciente(Integer dni){
```

Reporte de Actividad Médica

Se realiza un reporte de la actividad de un médico por día en un periodo de varios días presentando el id del mismo.
Para ello se recorre un hashmap el cual va recopilando las consultas de cada médico y los pacientes que fueron atendidos en dichas consultas.



Patrones aplicados

En la totalidad del trabajo se utilizaron los siguientes patrones de diseño:

- Decorator
- Double Dispatch
- Singleton
- Factory

El patrón Decorator fue utilizado en la creación de médicos y habitaciones, decorando según el tipo correspondiente

El patrón Double Dispatch fue utilizado para resolver los conflictos entre la sala de espera y el patio, resolviendo quien tenía la prioridad.

El patrón Singleton se utiliza para la clínica, que siempre tendrá una única instancia que contiene colecciones de pacientes, habitaciones, médicos, etc

El patrón Factory se utiliza para la creación de instancias de médicos, habitaciones y pacientes, ya que se debe tener en cuenta su especialidad o rango etario para el caso de los pacientes.



Javadoc

El Javadoc completo se encuentra en los archivos entregados, mediante index.html se logrará navegar por las documentaciones correspondientes de la siguiente manera:

Packages
Package
PClinica
PException
PHabitaciones
PModulos
PPersona.PMedico
PPersona.PPacientes

Package PClinica

Class Summary	
Class	Description
ClinicaSingleton	Esta clase contiene en diferentes colecciones informacion (médicos, pacientes atendidos, habitaciones) Patrón aplicado: Singleton

Package PException

Exception Summary	
Exception	Description
EsperaVacíaException	Lanzada cuando no hay pacientes para atender
FactoryHabitacionException	Lanzada cuando no se pudo crear la habitacion solicitada
FactoryMedicoException	Lanzada cuando no se pudo crear el medico solicitado
IngresoPacienteHabitacionException	Lanzada cuando no se pudo ingresar el paciente a la habitacion

Package PHabitaciones

Interface Summary	
Interface	Description
IHabitacion	Interfaz que declara los metodos a implementar por las habitaciones

Class Summary	
Class	Description
DHabitacionCompartida	Clase que representa una habitacion compartida por los pacientes
DHabitacionPrivada	Clase que representa una habitacion privada
DTerapiaIntensiva	Clase que representa una habitacion destinada a terapia intensiva
Habitacion	Clase que representa las habitaciones donde se alojan los pacientes
HabitacionDecorator	Clase destinada a implementar el Patron Decorator aplicado a las habitaciones Patrón aplicado: Decorator
HabitacionFactory	Esta clase tiene la responsabilidad de generar habitaciones Patrón aplicado: Factory



Package PModulos

Class Summary

Class	Description
ModuloAtencion	Modulo que se encarga de atender paciente retirandolo del lugar de espera y ubicandolo en atencion
ModuloFacturacion	Clase dedicada a tomar los datos de un paciente de la clinica para generar su factura correspondiente Retira de la lista de atencion a los pacientes atendidos
ModuloIngreso	Este modulo se encarga de ingresar al paciente, sus responsabilidades son: Asignar numero de orden al paciente Resolver conflictos en la sala de espera privada y el patio Cargar al paciente en los registros de la clinica
TEST	Clase destinada a hacer pruebas con los modulos implementados

Package PPersona.PMedico

Interface Summary

Interface	Description
IMedico	Interface que contiene los metodos a implementar por los medicos

Class Summary

Class	Description
Cirurgia	Clase que representa un cirujano
Clinico	Clase que representa a un medico clinico
Consulta	Clase que almacena la fecha y el codigo que pueden pertenecer tanto a un paciente, como a un medico
DDoctor	Clase que representa un doctor
DMagister	Clase que representa un Magister
DoctorDecorator	Clase padre destinada a aplicar el Patron Decorator a los distintos medicos
DPermanente	Clase que representa un permanente
DResidente	Clase que representa un residente
Medico	Esta es la "Master Class" que controla a todos los medicos con sus respectivos decorados.
MedicoFactory	Clase que aplica el patrón Factory para la creación de nuevas instancias de médicos Patrón aplicado: Factory
Muestra	Clase que tiene la responsabilidad de mostrar el reporte de actividad de un medico
Pediatría	Clase que representa un pediatra

Package PPersona.PPacientes

Class Summary

Class	Description
Joven	Esta clase hereda de la clase abstract Paciente, su implementacion corresponde a uno de los grupos etarios Patrón aplicado: Double Dispatch.
Mayor	Esta clase hereda de la clase abstract Paciente, su implementacion corresponde a uno de los grupos etarios Patrón aplicado: Double Dispatch.
Nino	Esta clase hereda de la clase abstract Paciente, su implementacion corresponde a uno de los grupos etarios Patrón aplicado: Double Dispatch.
Paciente	Clase que representa un paciente
PacienteFactory	Clase encargada de de generar pacientes utilizando patrón Factory



Diagrama de clases (UML)

El diagrama de clases puede apreciarse en su totalidad en el trabajo

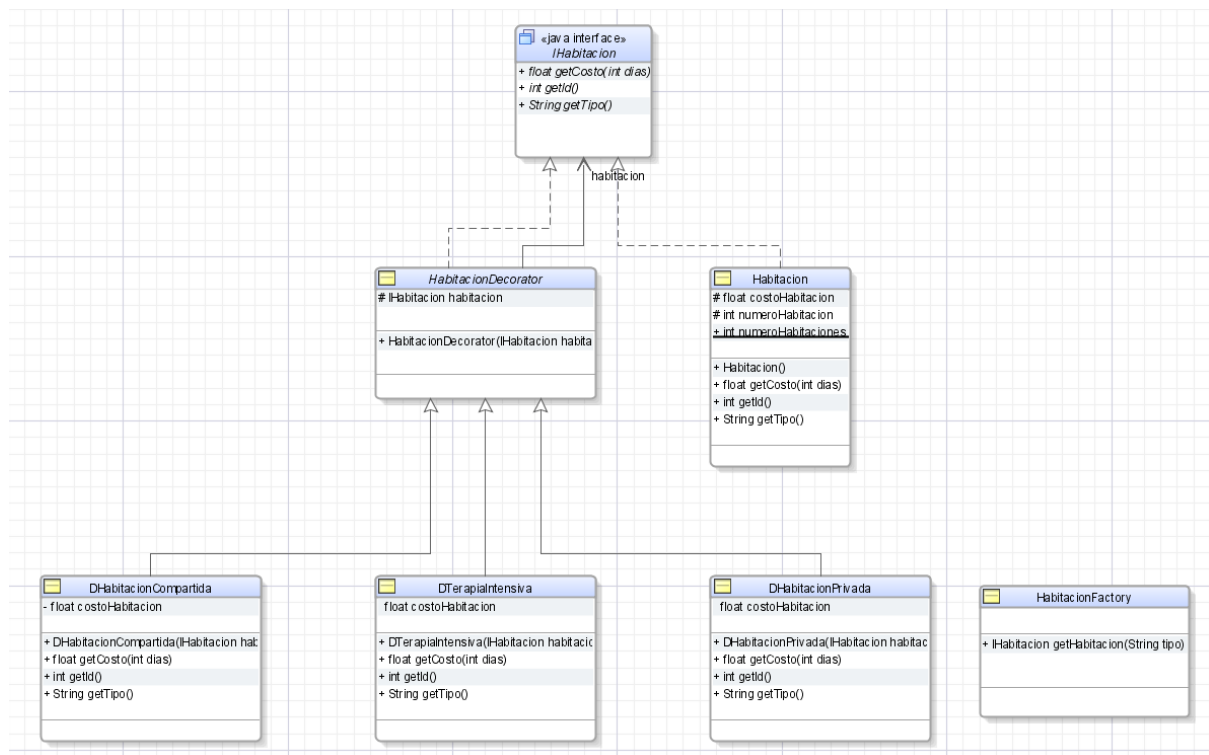
entregado, dentro del paquete PUML



Java Class Diagram.java_diagram

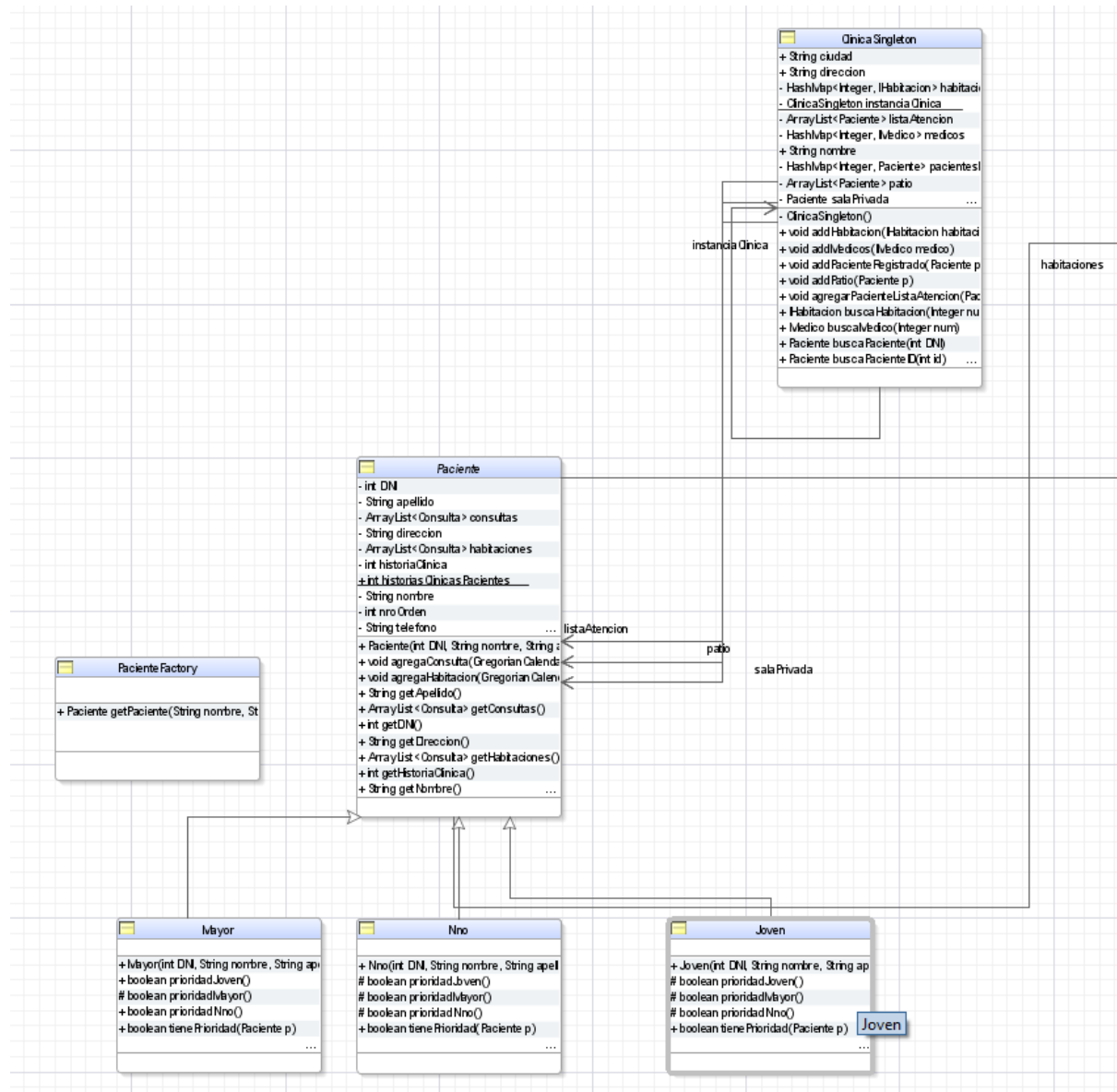
Sin embargo, a fines de complementar el informe, se presentan a continuación los diagramas de habitacion, paciente y medico respectivamente:

Habitaciones:

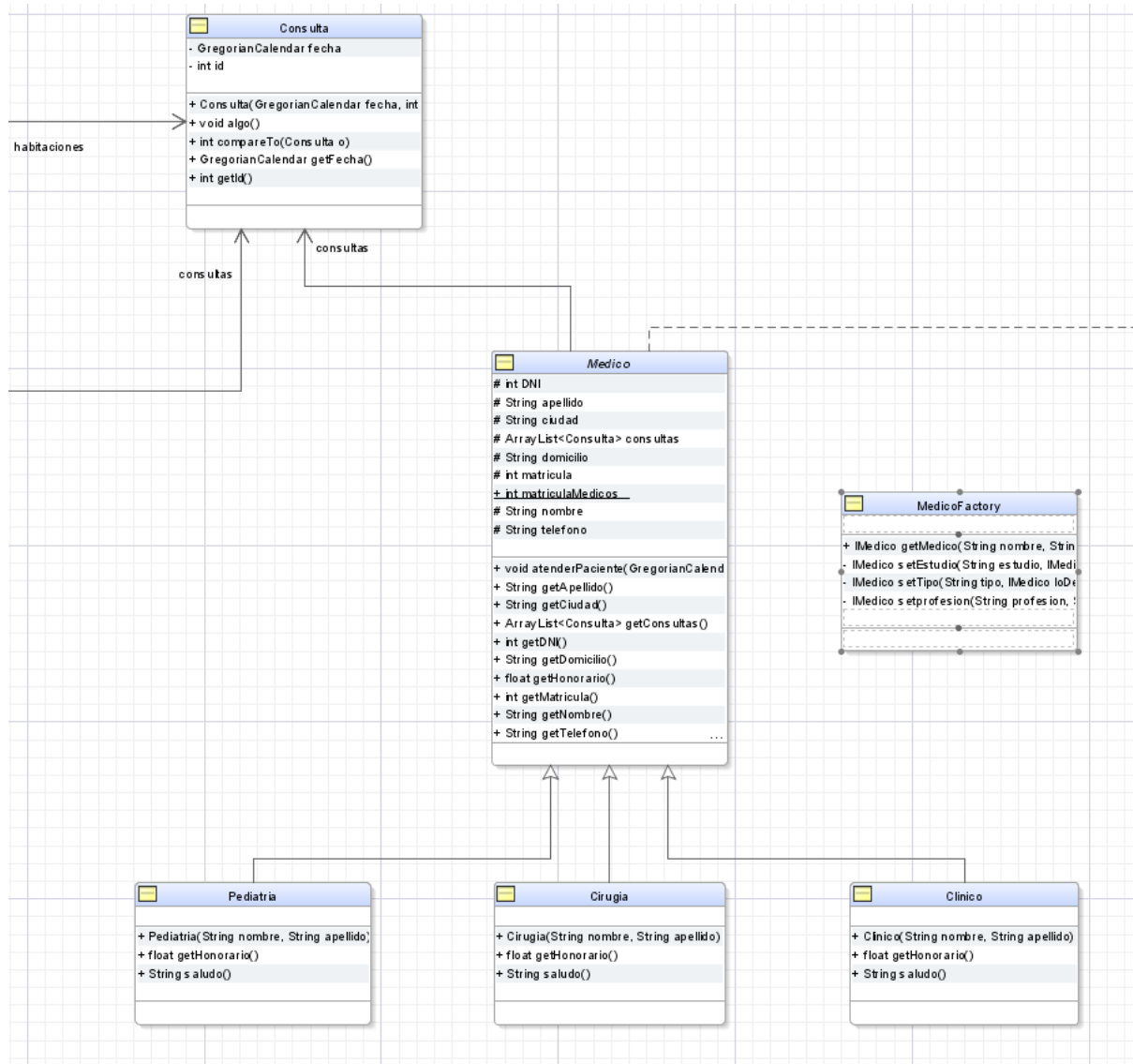




Pacientes:

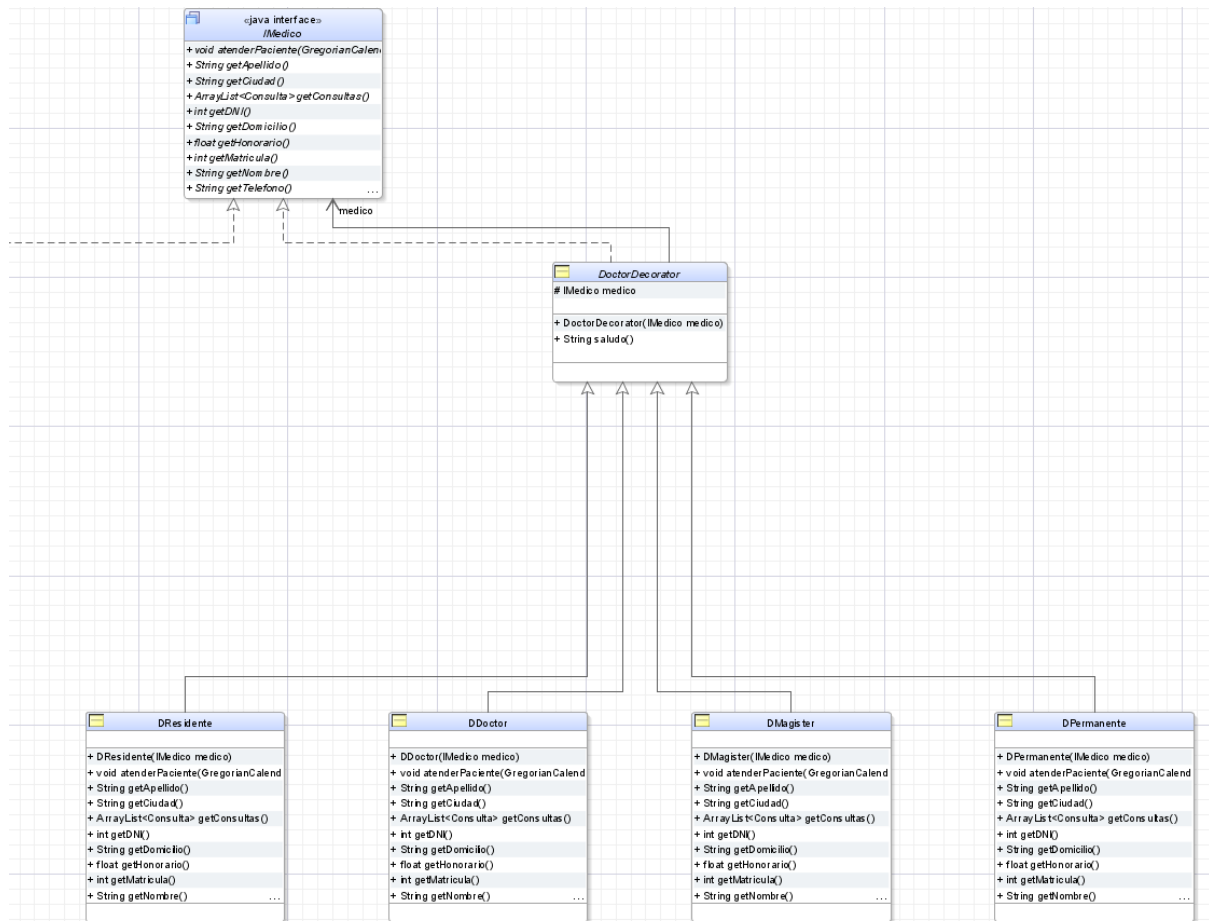


Médicos: (1ra parte)





Medicos (2da parte):





Observaciones

- El plantel médico, los pacientes ingresados y las habitaciones se encuentra en colecciones de la clase Clínica, esto facilita el ingreso a los datos ya que la instancia de la clínica es única y se pueden solicitar todos los datos a misma.
- Se consideran las siguientes excepciones:
 - **EsperaVacíaException:** El módulo de atención llama al siguiente paciente a ser atendido, pero no hay ninguno.
 - **FactoryHabitacionException:** El rango etario del paciente no es válido, por lo tanto no se pudo crear un paciente nuevo.
 - **FactoryMedicoException:** Igual que la excepción anterior, pero en este caso la especialidad del médico no es válida.
 - **IngresoPacienteHabitacionException:** Se lanza cuando no es posible ingresar a un paciente a una habitacion dada.
 - **NoHayPacienteException:** Lanzada cuando no se encontro paciente atendido por un medico en un tiempo determinado