IV. (20%) Conceptos

1. ¿Cómo se puede hacer para que un método no se pueda sobreescribir? Da un ejemplo

Nosotros podemos poner final en la firma del metodo para evitar que se pueda sobreescribir mas adelante en una clase derivada, etc. Por ejemplo

public final void setUID(uid int){...}

Por ejemplo, nosotros tenemos que este metodo asigna una uid de manera general para todas las clases que se puedan derivar, ya que solo esta esa forma de asignación de un identificador y no deberia poder cambiarse.

2. ¿Cuál es la diferencia entre una excepción chequeada y una no chequeada? Da un ejemplo de cada una

Una excepción chequeada nos permite poder recuperar el código al momento que haya una excepción y se nos obliga a manejarlas y que no se acabe todo el proceso, mientras una no chequeada va a hacer que falle en tiempo de ejecución.

II. (25%) DISEÑANDO

/** Verifica si se tiene disponibilidad en un dia y horario

especificado, para asi poder saber si se puede asignar una nueva cita

- Date: Dia/Mes/Año del cual se quiere verificar la disponibilidad

- StartTime: Hora inicial para la posible cita

- EndTime: Hora Final para la posible cita

isAvailableDuring(date : LocalDate, startTime : int, endTime : int) : boolean

Y se sobre escribe en:
Para determinar si un doctor o una oficina estan disponibles se verifica

que no esten ocupados atendiendo una cita para ese dia y horario

- Appointment: Cuando se quiere crear una cita se tiene un dia (date) y horario inicial. Las

citas inicial en el horario indicado (time) y terminan 1 hora despues. No se puede crear una cita fuera de horario laboral 7 am a 7 pm

- Doctor: Los doctores adicionalmente no van a estar disponibles para una nueva cita si ya tienen asignadas 5 citas para ese mismo dia
- Office: Las oficinas necesitan un tiempo de preparacion entre cita y cita de 1 hora

III. (25%) EXTENDIENDO

3. Explique los cambios necesarios en los diseños anteriores para esta extensión.

Necesitamos un nuevo tipo de oficina y un nuevo tipo de tratamiento, por lo que podemos derivarlas de las clases ya existentes, y tambien como diagnostico tiene mas cosas que un string como un responsable por eso es una nueva clase y las relacionamos.

4. Considerando el segundo principio SOLID. ¿Se cumplen en estos diseños?

Si ya que usamos la herencia para crear estos nuevos tipos de tratamientos y oficinas sin necesidad de modificar estos.