Ivan Santiago F. Torres

3.3

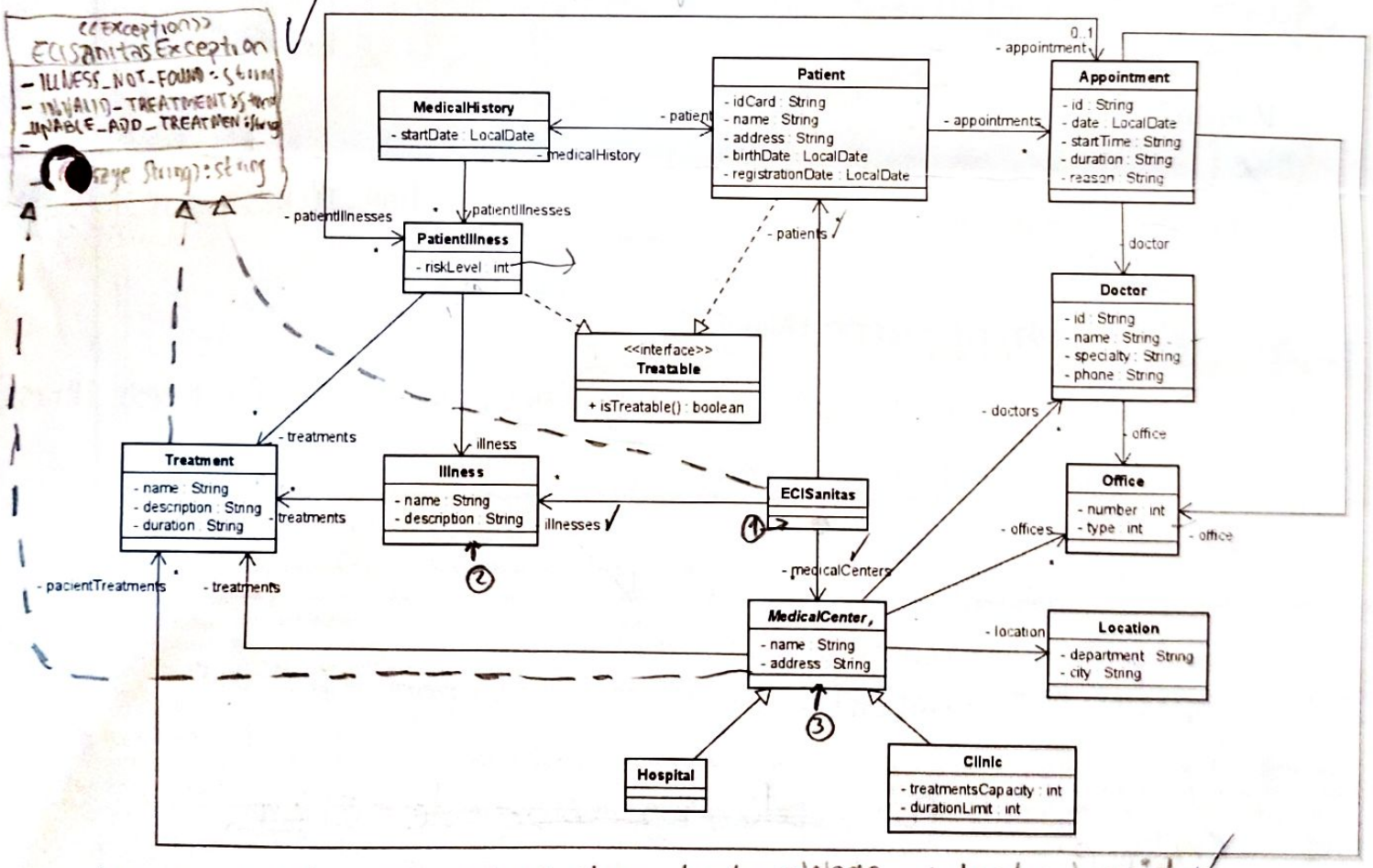
ESCUELA COLOMBIANA DE INGENIERÍA
PROGRAMACIÓN ORIENTADA A OBJETOS
Parcial segundo tercio. Nota esperada: 3.0
S11: 2024-02
G2
[redacted]SO: ECISANITAS

EciSanitas es una prestigiosa empresa de medicina prepagada enfocada en brindar servicios de salud tales como citas médicas y atención a enfermedades. Para el cumplimiento de esta misión, se ha decidido integrar mejoras que se detallan a continuación.

Nuevos requisitos funcionales:

1. **Centros Médicos:** EciSanitas en busca de ofrecer mejores servicios ahora cuenta con dos tipos de centros médicos: **Hospitales y Clínicas**. Las clínicas ofrecen servicios médicos menos complejos y en un alcance más limitado. Las clínicas están enfocadas principalmente en la atención ambulatoria. Por otro lado, los hospitales ofrecen una gama completa de servicios médicos, incluyendo atención de emergencia, hospitalización, cirugías complejas.
2. **Agregar tratamiento:** Los centros médicos cuentan con una serie de requisitos para incluir un nuevo tratamiento dentro de sus servicios. Para agregar un tratamiento nuevo a cualquier centro médico sin importar su tipo este debe contar con al menos 10 profesional de cada especialidad (1 general, 2. odontología, 3: pediatría) además de ellos cada centro médico tiene sus propias condiciones específicas:
   - **Hospital:** Debe tener 3 consultorios por cada doctor que tengan
   - **Clínica:** Tiene un límite máximo de tratamientos que puede brindar y una duración máxima que dichos tratamientos debe tener
3. **Tratabilidad:** ECISantias desea validar la tratabilidad sobre diferentes actores:
   - **Enfermedades de Pacientes:** Una enfermedad de un paciente es tratable si en EciSantias dicha enfermedad tiene mínimo tantos tratamientos como lo indica su nivel de riesgo. La tratabilidad de la enfermedad de un paciente no puede ser evaluada si no tiene al menos una cita médica relacionada.
   - **Pacientes:** Un paciente es tratable si lleva al menos un año como paciente de EciSanitas, es decir que su historia médica tiene un año de antigüedad y todas sus enfermedades son tratables. La tratabilidad del paciente no puede ser evaluada si no tiene al menos 3 citas médicas registradas

la enfermedad tengo todos tratamientos



① + addTreatment (String: illnessName, String treatmentName, int duration): void

Contenedores de ECISanitas patients(idCard), illness(name) y medicalCenters (name) son TreeMap
Los demás contenedores son ArrayList

② + getIllness (String name): String;

③ + addTreatment (Treatment t): void;

(30%) IMPLEMENTACIÓN

MDD

1. Estudie la especificación (documentación + encabezado) del método y los diagramas de secuencia.
2. Actualice el diagrama de clase con los nuevos elementos
3. Implemente la clase responsable inicial (sólo atributos) ~~y la interfaz~~ *Summary*
4. Implemente la clase responsable de las excepciones del sistema
5. Implemente los métodos correspondientes al diseño. Incluya la documentación (si es necesario).

En EciSanitas

public Void addTreatment(String illnessName, String treatmentName, int duration)

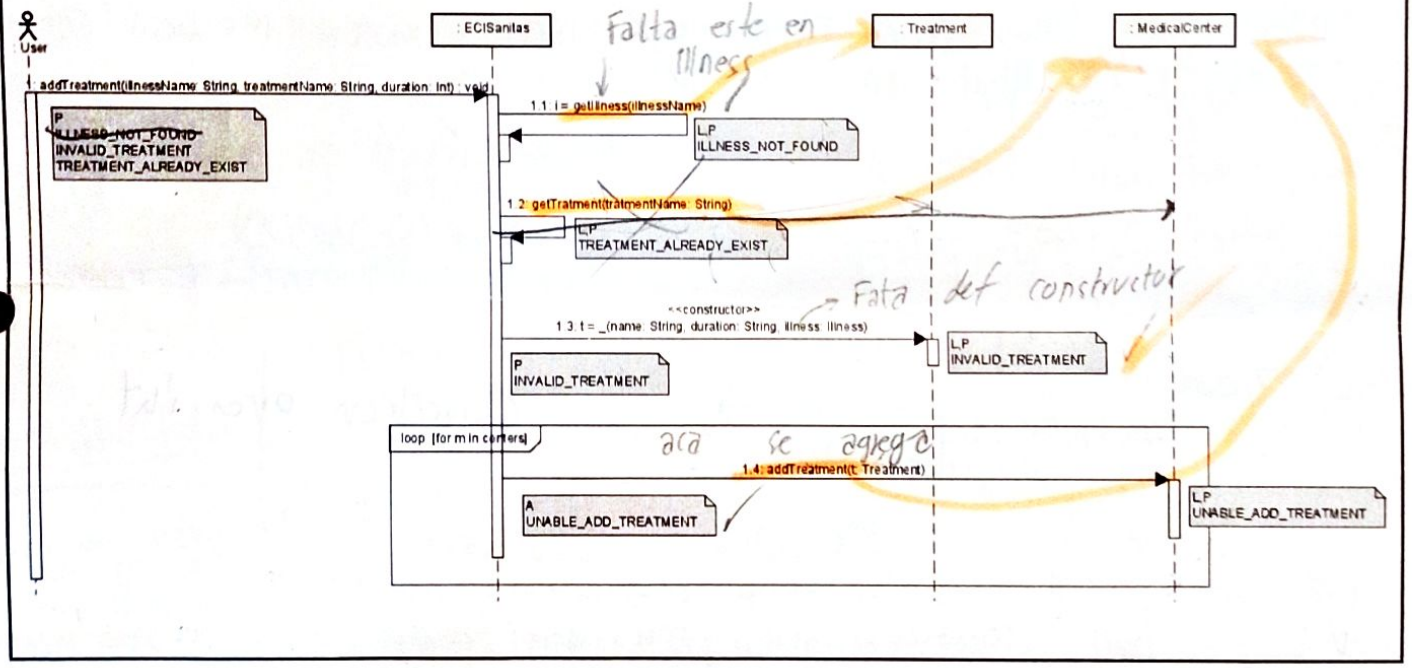
Add a treatment to all medical centers that meet the conditions. If any medical center does not meet all the conditions, it should be added to all that do meet them.

Parameters:

illnessName, Name of the illness treated with this treatment.
treatmentName, Name of the treatment .
duration, duration of the treatment.

Throws:

EciSanitasException - ILLNESS_NOT_FOUND The illness does not exist.
EciSanitasException - INVALID_TREATMENT The treatments name or duration are invalid.
EciSanitasException - UNABLE_ADD_TREATMENT The medical center does not meet the requirements to add the new treatment.
EciSanitasException - TREATMENT_ALREADY_EXIST Medical center already has that treatment.



II. (25%) DISEÑANDO

Diseñe los métodos necesarios para cumplir con los compromisos asociados a la interfaz Treatable. No olvide el manejo de excepciones.

1. Especifiqué (documentación + encabezado) el método propuesto
2. Realice los diagramas de secuencia correspondiente al diseño completo
3. Actualice el diagrama de clase con los nuevos elementos

III. (25%) EXTENDIENDO

EciSanitas desea incluir un nuevo tipo de centro médico "**Centro especializado**", el centro especializado va a estar enfocado en una única especialidad, por ello tiene un número máximo de consultorios que puede tener. Al ser un centro con recursos limitados y para tratamientos simples este es considerado una clínica. Para agregar un nuevo tratamiento a este "**Centro especializado**" todos los doctores asociados a dicho centro deben ser de la misma especialidad.

1. Realice las extensiones necesarias en el diagrama de clases.
2. Implemente dichas extensiones (encabezados y atributos de los nuevos componentes)
3. Explique los cambios necesarios en los diseños anteriores para esta extensión.
4. Considerando el segundo principio SOLID. ¿Se cumplen en estos diseños?.

IV. (20%) Conceptos

1. ¿Cuando se debe usar una **interfaz** y cuando una clase **abstracta**, Menciona 1 diferencia entre estas dos?

Cuando clases no necesariamente comp. comunes que varias clases tengan unos mismos comportamientos

Implementación clase responsable

```
public class ECISanitas{
  private treeMap<String, Illness> illnesses = new treeMap<>();
  private treeMap<String, MedicalCenter> medicalCenters = new treeMap
  private treeMap<String, Patient> patients = new treeMap<>();
/**
 * addaTreatment to all medical centers that meet the conditions. if
 * any medical center does not meet all the conditions, it should be
 * added to all that do meet them
 * @param illnessName: Name of the illness treated with this treatment.
 * @param treatmentName: Name of the treatment.
 *
 * @param duration: duration of the treatment.
 * @Throws ...
 */

public void addTreatment (String illnessName, String treatmentName, int duratior
  throws ECISanitasException {
  Illness i = I.getIllness (illnessName);
   if (i != null) {
    return i;
  }else{
    throws new ECISanitasException (ECISanitasException.ILLNESS_NOT_FOUND);
  }

m.getTratment (string tratmentName);

  new Treatment = new Treatment (String name, string duration, Illness illness);
 for (MedicalCenter m : medicalCenters) {
    try{
      m.addTreatment (Treatment t)
    catch (ECISanitasException e){
      System.out.println (ECISanitasException.UNABLE_ADD_TREATMENT)
    }
  }
    Continue;
}
```

~~Con esto en caso de error el for va~~

era otra alternativa

// implementacion interfaz

```
public interface Treatable {
    boolean isTreatable();
}
```

// implementacion Excepciones

```
public class ECIsanitasException extends Exception {
    public static final String ILLNESS_NOT_FOUND = "The illness does not exist";
    public static final String INVALID_TREATMENT = "The treatments name or duration are invalid";
    public static final String UNABLE_ADD_TREATMENT = "The medical center does not meet the requirements to add new treatment";
    public static final String TREATMENT_AIREADY_EXIST = "medical center already has that treatment";

    public ECIsanitasException (String message) {
        super (message);
    }
}
```

// clase Treatment.

duración era int

```
public class Treatment {
    private String name, description, duration;

    public Treatment (String name, String description, String duration)
            throws ECIsanitasException {
        if ( name == null || description == null || duration == null) {
            throw new ECIsanitasException (ECIsanitasException.INVALID_TREATMENT);
        }else {
            this.name = name;
            this.description = description;
            this.duration = duration;
        }
    }
    public Illness getIllness () {
        return this.name;
    }
}
```

¿Qué es polimorfismo y como lo evidenció en el parcial?.

Rta: // el polimorfismo es la capacidad que tienen diferentes objetos de responder a un mismo mensaje. ✓

se puede evidenciar el polimorfismo de manera intrinsica en las herencias porque cada tipo de medical center puede sobreescribir metodos en medicalcenter ? en terminos generales este ejemplo es muy ambiguo

½

pregunta # 2.

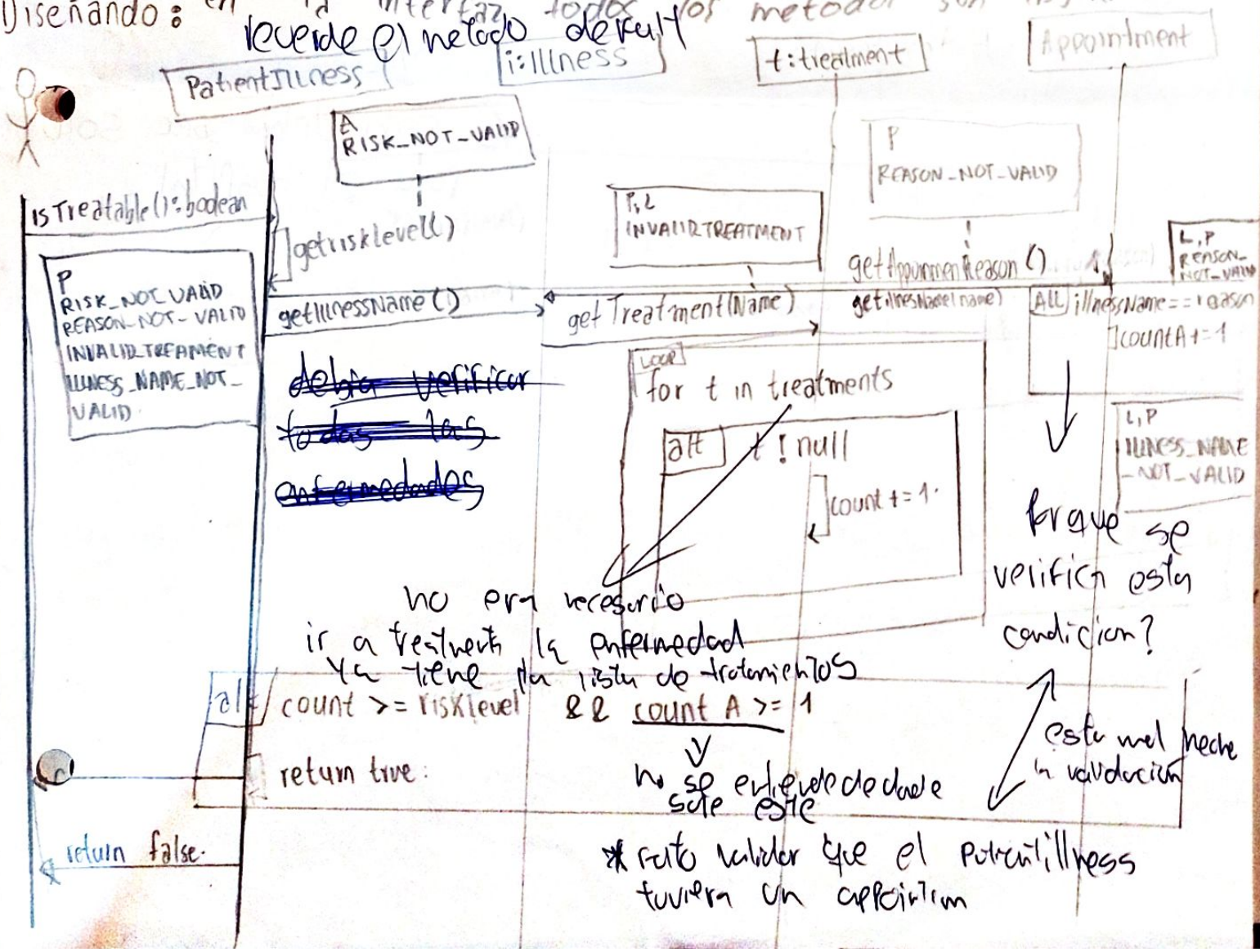
- La interfaz se utiliza cuando queremos que 2 clases no necesariamente asociadas tengan un mismo comportamiento. ✓
- La clase abstracta la utilizamos cuando queremos que otras class hereden los comportamientos.

una clase abstracta puede tener metodos implementados y no tiene comportamientos. ✓

Y si es abstracta

Diseñando: en la interfaz todos los metodos son abstractos

recuerde el metodo default



// en la clase MedicalCenter.

```
public class MedicalCenter{
  private String name, address;
  private ArrayList<Doctor> doctors; ✓
  private ArrayList<Office> offices; ✓
  private ArrayList<Treatment> treatments; ✓

  public MedicalCenter(){
  }
  public void addTreatment(Treatment t){
    for(Doctor d: doctors){
      if(countSpecialty1>10 && countSpecialty2>10 && countSpecialty3>10){
        C.verifyTreatmentCapacity();
        C.verifyDurationLimit();
        for(Office O: offices){
          countO+= O.getOffiresNumber();
        }
        if(countO == countDoctor*3){
          m.add(Treatment)
        else{
          throw new ECISanitasException(ECISanitas.UNABLE_ADD_TREATMENT
        }
      }else{
        throw new ECISanitasException(ECISanitas.UNABLE_ADD_TREATMENT);
      }
    }
```

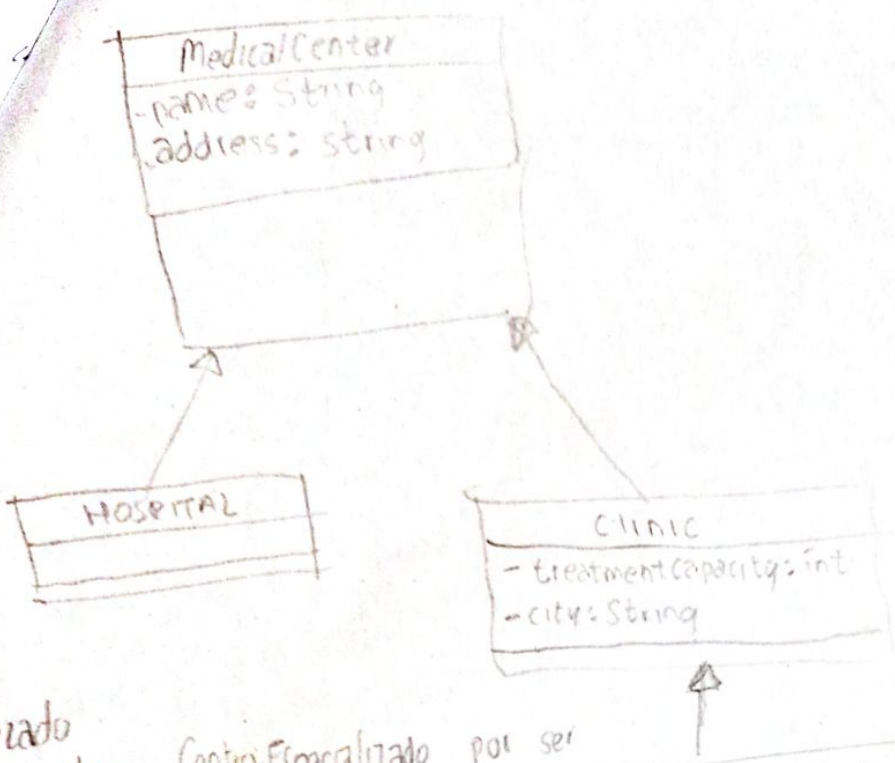
Se aplico el polimorfismo

falto declarar los contadores

¿que es C? → Syntaxis

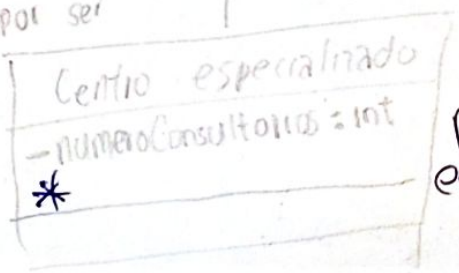
debia ser solo para las clinicas

esto debia ser solo para el hospital



- encabezado

```
/** La clase CentroEspecializado por ser
 * de recursos limitados cuenta como
 * clinica, tiene numero maximo
 * de consultorios.
 */
```



falto definir la especialidad del centro

```
public class CentroEspecializado extends Clinic {
    private int numeroConsultorios;
    public CentroEspecializado() {
    }
}
```

→ el constructor no debería ser vacío

- Los cambios necesarios para esta nueva clase es realizar la extensión de clinica, ademas de eso tener en cuenta el numero de consultorios maximos; puesto que la clase padre tiene una clase padre que se llama MedicalCenter y este a su vez tiene relación con las clases Doctor, office no necesitamos mas relación.
- Considerando el segundo principio Solid SI se cumplen estos diseños

por cuestion de tiempo:
Actualizacion clases en el extendiendo

en PatientIllness agregamos el metodo
+getRiskLevel(): int

en Illness
+getIllnessName(): string

en treatment
+getTreatmentName(): string

en Appointment
+getArgumentReason(): string

en las excepciones
agregamos RISK_NOT_VALID
REASON_NOT_VALID
INVALID_TREATMENT
ILLNESS_NAME_NOT_VALID