

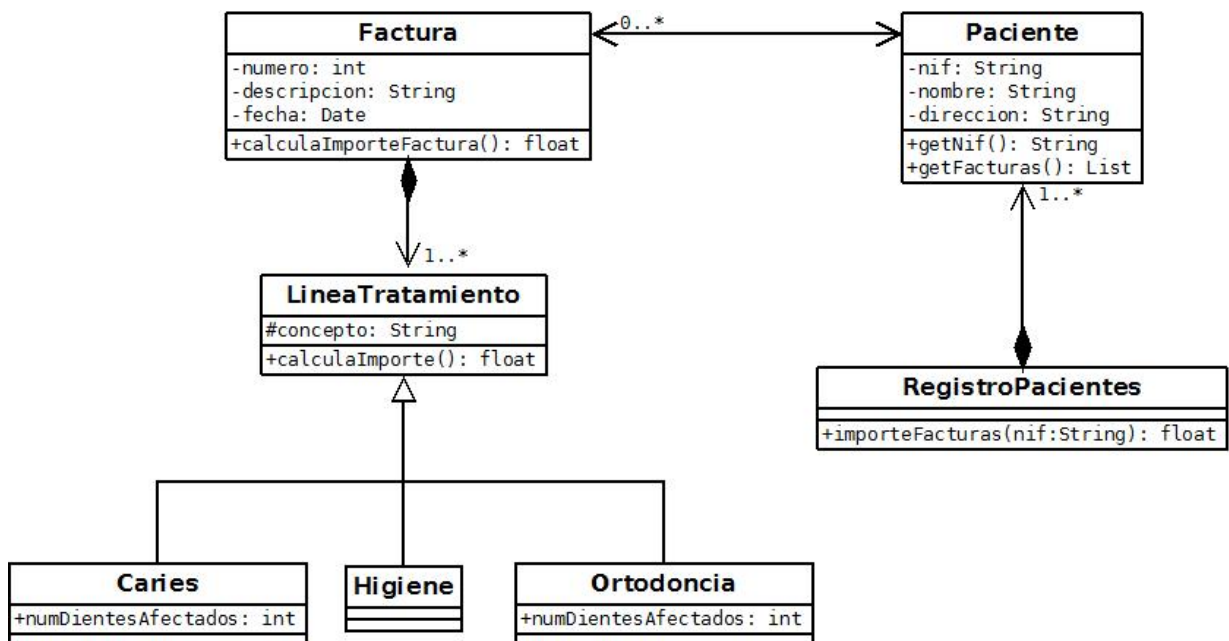
EXAMEN PARCIAL Metodología y Programación Orientada a Objetos.

Curso 2011-2012. Cuatrimestre Primavera. 25 de Abril de 2011

Estamos creando un gestor de facturas para una clínica dental que está formado por los siguientes componentes:

- Clase Paciente, que representa a los pacientes de la clínica dental, para los que se guarda su NIF, nombre y dirección.
- La Clase RegistroPacientes contiene información sobre todos los pacientes de la clínica dental.
- La Clase Factura que representa la factura que se genera para un paciente en una visita. Dicha clase contiene el número de factura, la descripción, la fecha en la que se ha generado la factura, y una línea para cada tratamiento que se ha realizado al paciente.
- Clase LineaTratamiento representa las diferentes líneas de la factura, cada una de ellas contiene el concepto, así como el importe. Las líneas de tratamiento pueden ser de los siguientes tipos: Caries, Higiene o Ortodoncia.

A continuación se muestra el diagrama de clases UML de la aplicación:



Preguntas y ejercicios:

1. (2 PUNTOS) Responde a las siguientes preguntas:

- a) La relación entre la clase Factura y LineaTratamiento es igual que la existente entre RegistroPacientes y Paciente. ¿Qué tipo de relación es? ¿Por qué crees que se ha escogido este tipo de relación?

b) Según el diagrama UML, ¿se puede saber cuántos tratamientos ha recibido un Paciente? Justifica la respuesta.

c) Si la clase LineaTratamiento tiene el siguiente constructor:

```
public LineaTratamiento (String concepto)
```

sería correcto crear un objeto de la clase Caries de la siguiente manera, si esta no tiene implementado ningún constructor? ¿Porqué?

```
Caries c = new Caries ("Empaste molar");
```

En caso que no fuera correcto, cómo lo harías?

d) Según el diagrama de clases UML, ¿Puede ser que a un paciente no se le haya generado ninguna factura? ¿Puede que una factura no contenga ninguna línea de tratamiento? Justifica las respuestas.

2. **(2.5 PUNTOS)** Escribe el código Java para todas las clases del diagrama UML correspondiente a la definición de dichas clases, sus atributos y la implementación de las relaciones entre dichas clases. **No** es necesario que defines constructores ni métodos.

3. **(1 PUNTO)** Escribe el código Java para el método de la clase Factura cuya notación UML es:

```
+calculaImporteFactura():float
```

Dicho método calcula el importe de una factura, sumando los importes de las diferentes líneas de tratamiento presentes en dicha factura.

Nota: Suponer que siguiente el método de la clase LineaTratamiento está implementado:

```
+calculaImporte():float
```

4. **(2.5 PUNTOS)** Escribe el código Java para el método de la clase RegistroPacientes cuya notación UML es:

```
+importeFacturas(nif:String):float
```

Dicho método calcula el importe total de las facturas del usuario cuyo nif se le pasa como parámetro, y devuelve dicho importe.

Nota: Este método puedo hacer uso del método de la clase Factura que has implementado en el apartado anterior:

```
public float calculaImporteFactura();
```

5. **(1 PUNTO)** Escribid el constructor de la clase Ortodoncia.

Suponiendo que el único constructor de la clase LineaTratamiento tiene la siguiente cabecera:

```
public LineaTratamiento (String concepto);
```

6. **(1 PUNTO)** Implementad los métodos toString() de las clases Factura y Paciente, suponiendo que el método toString() de la clase LineaTratamiento ya está implementado.

Recordad que la cabecera de esos métodos es:

```
public String toString();
```

Solución

1. a) Tanto entre las clases Factura y LineaTratamiento, como entre RegistroPacientes y Paciente hay una relación de agregación fuerte o composición. Se ha escogido este tipo de relación ya que los pacientes forman parte del registro de pacientes y las líneas de tratamiento de las facturas, y debido también a que en ambos casos, si la clase que contiene a la otra se elimina (Factura y RegistroPaciente), se han de eliminar también los elementos que la componen (LineaTratamiento y Paciente, respectivamente).
- b) Sí, ya que desde la clase Paciente podemos acceder a las facturas (navegabilidad desde Paciente a Factura) y desde Factura podemos acceder las líneas de tratamiento (una factura está compuesta por diferentes líneas de tratamiento).
- c) No, ya que las subclases no heredan los constructores de su superclase. Se tendría que definir en la clase Caries el constructor *public Caries (String concepto)*.
- d) Sí, uede ser que a un paciente no se le haya generado ninguna factura (cardinalidad 0..*).
No, una factura como mínimo contendrá 1 línea de tratamiento (cardinalidad 1..*)

```
2. public class RegistroPacientes {
    private Map <String, Paciente> pacientes;
}
public class Paciente {
    private String nif;
    private String nombre;
    private String direccion;
    private List <Factura> facturas;
}
public class Factura {
    private int numero;
    private String descripcion;
    private Date fecha;
    private Paciente paciente;
    private List<LineaTratamiento> lineas;
}
public class LineaTratamiento {
    protected String concepto;
}
public class Caries extends LineaTratamiento{
    private int numDientesAfectados;
}
public class Higiene extends LineaTratamiento{...}
public class Ortodoncia extends LineaTratamiento{
    private int numDientesAfectados;
}
```

```

3. public float calculaImporteFactura(){
    float importe=0;
    Iterator<LineaTratamiento> it=lineas.iterator();
    while(it.hasNext()){
        importe+=it.next().calculaImporte();
    }
    return importe;
}

4. public float importeFacturas (String nif){
    float importe=0;
    Paciente p = pacientes.get(nif);
    if(p!=null){
        ArrayList facturas=(ArrayList)p.getFacturas();
        Iterator <Factura> it= facturas.iterator();
        while (it.hasNext())
            importe+=it.next().calculaImporteFactura();
    }
    return importe;
}

5. public Ortodoncia(String concepto, int numDientesAfectados){
    super(concepto);
    this.numDientesAfectados=numDientesAfectados;
}

6. Solución clase Factura
public String toString(){
    String res="Numero factura: "+numero+"\n";
    res += "Descripcion"+descripcion+"\n";
    res += "Fecha"+fecha.toString()+"\n";
    res += "Paciente"+paciente.toString()+"\n";
    Iterator<LineaTratamiento> it=lineas.iterator();
    while(it.hasNext()){
        res +=it.next().toString();
    }
    return res;
}

```

Solución clase Paciente:

```

public String toString(){
    String res="Nif: "+nif+"\n";
    res += "Nombre: "+nombre+"\n";
    res += "Direccion: "+direccion+"\n";
    Iterator <Factura> it = facturas.iterator();
    while(it.hasNext())
        res +=it.next().toString();
    return res;
}

```