P3

Francisco Javier Hernández Martín, Jose Luis Bueno Pachón, Carlos Marín Corbera, Carmen González Ortega, Altair Bueno Calvente

UNIVERSIDAD
DE MÁLAGA

1 jan 2022

Contents

Preámbulo

El estilo de programación utilizado se rige por los normas de programación de **Google Java Style Guide**[1].

Ejercicio 1: Expedientes médicos

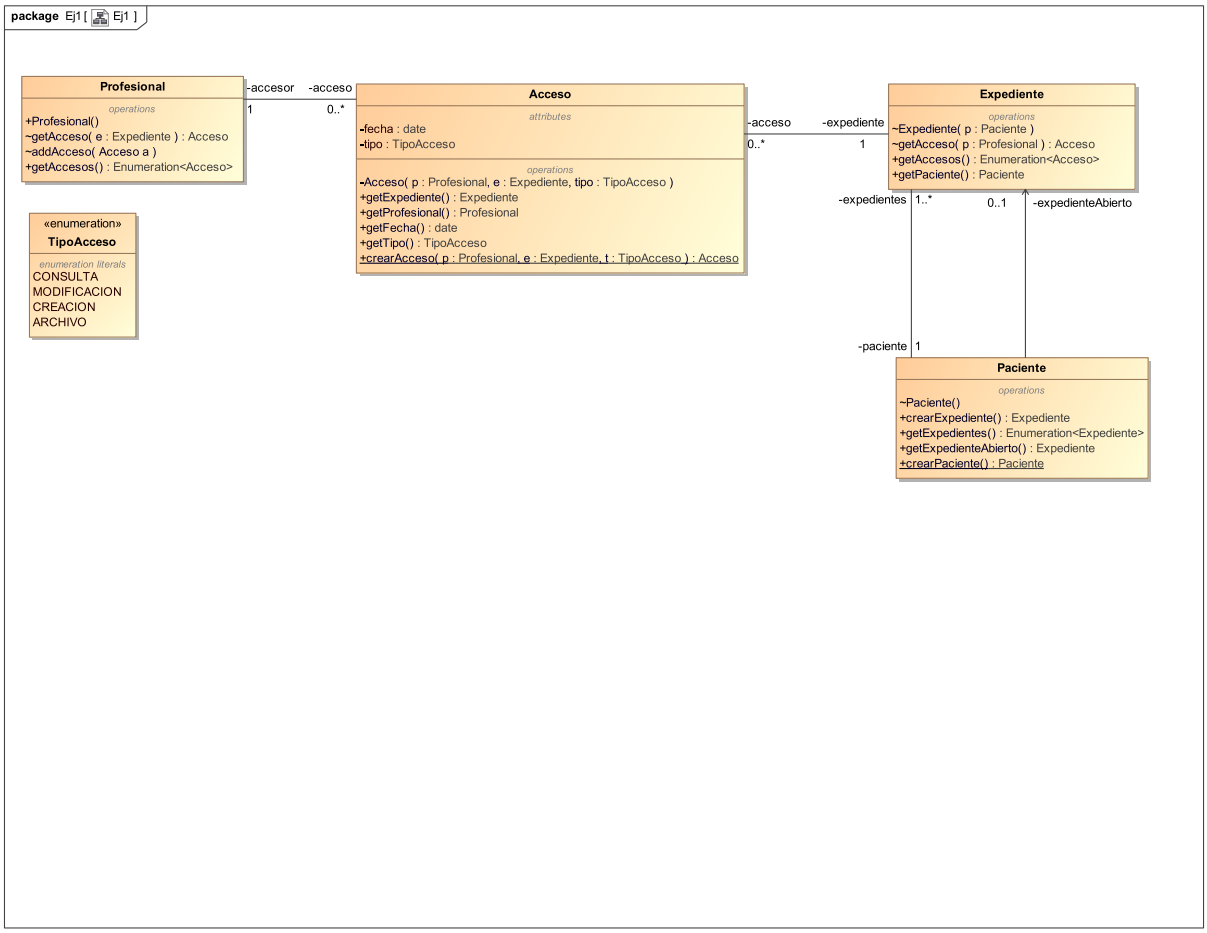

Figure 1: Modelo de diseño para Java en Magic Draw

Para la creación del modelo de diseño hemos usado la mayor parte del modelo de dominio, modificando aquellos elementos que no puedan ser directamente traducidos a Java o que pongan en peligro la consistencia del modelo.

[1] https://google.github.io/styleguide/javaguide.html

La clase asociación Acceso ha sido reificada al ser la única forma de trabajar con clases asociación en Java. Para mantener el mismo comportamiento descrito en el modelo de dominio, hemos bloqueado el acceso al constructor y ofrecer en su lugar el método público estático `crearAcceso()`. Este método se encargará de actualizar los datos en un acceso ya existente o bien de crear una nueva instancia de la clase Acceso. El resto de la interfaz publica está compuesta por *getters*.

La clase Profesional no presenta grandes cambios, tan solo hemos limitado su interfaz pública a lo esencial: Poder instanciar la clase y consultar los accesos

La clase Expediente ha sido restringida para evitar ser instanciada sin estar asociada correctamente a un paciente, siendo necesario el método `crearExpediente()` en Paciente para poder crear instancias. Además, su interfaz pública solo permite consultas sobre los accesos existentes y el paciente al que está asociado

La clase Paciente tampoco tiene su constructor visible, sino que utiliza el método estático `crearPaciente()` para generar nuevas instancias. Esto es asi para garantizar la consistencia del modelo: No podemos tener pacientes sin ningún expediente, por lo que este método se encarga de generar un primer expediente inicial y asociarlo al paciente. El resto de la interfaz pública está compuesta por *getters* para consultar el expediente abierto y todos los expedientes actuales

Valoramos la posibilidad de utilizar un patrón **factoría abstracta** con una clase Hospital, encargada de los métodos `crearPaciente()` y `crearExpediente()`. Terminamos descartando esta idea en favor del patrón **método fábrica** ya que los expedientes y los pacientes tienen alto acoplamiento.

Clase Profesional

Interfaz pública

- `Profesional()`/*{Constructor}*
- `getAccesos()`/*{Getter}*: Devuelve una enumeración[2] que itera sobre todos los accesos actuales que tiene el profesional

Interfaz visible desde paquete

- `getAcceso()`/*{Getter}*: Dado un expediente no nulo, busca cuales de todos sus accesos le relaciona con dicho expediente y lo devuelve. En caso de no existir dicho acceso

[2] `java.util.Enumeration<T>` es una colección inmutable de Java

devolverá `null`
- `addAcceso()`: Registra el acceso no nulo recibido como parámetro en su colección de accesos

Interfaz privada

- `acceso`/*{Attributes}*: Conjunto de accesos a los que el profesional tiene acceso

Clase Acceso

Interfaz pública

- `getExpediente()`/*{Getter}*: Devuelve el expediente asociado
- `getProfesional()`/*{Getter}*: Devuelve el profesional asociado
- `getFecha()`/*{Getter}*: Devuelve la fecha de registro
- `getTipo()`/*{Getter}*: Devuelve el tipo de acceso
- `crearAcceso()`/*{Static}*: Permite crear o actualizar los datos de acceso entre un profesional y un expediente. En el caso de que ya existiera un expediente entre ambas instancias, se actualiza la fecha y el tipo de acceso y se devuelve. En caso de no existir acceso previo, se crea una instancia de Acceso y se asocia al profesional y al expediente.

Interfaz privada

- `Acceso()`/*{Constructor}*
- `fecha`/*{Attributes}*: Fecha de registro
- `tipo`/*{Attributes}*: Tipo de acceso
- `expediente`/*{Attributes}*: Expediente asociado
- `profesional`/*{Attributes}*: Profesional asociado

Clase Expediente

Interfaz pública

- `getAccesos()`/*{Getter}*: Devuelve una enumeración que itera sobre todos los accesos existentes sobre el expediente
- `getPaciente()`/*{Getter}*: Devuelve el paciente al que pertenece el expediente

Interfaz visible desde paquete

- `Expediente()`/*{Constructor}*
- `getAcceso()`/*{Getter}*: Dado un profesional no nulo, busca cuales de todos sus accesos le relaciona con dicho profesional y lo devuelve. En caso de no existir dicho acceso devolverá `null`

Clase Paciente

Interfaz pública

- `crearExpediente()`: Crea un nuevo expediente y lo asocia al paciente actual. Establece dicho expediente como el expediente abierto. Devuelve la instancia de expediente creada
- `getExpedientes()`/*{Getter}*: Devuelve una enumeración que itera sobre todos los expedientes asociados al paciente
- `getExpedienteAbierto`/*{Getter}*: Devuelve el expediente abierto
- `crearPaciente()`/*{Static}*: para crear nuevos pacientes. Se encarga de instanciar un nuevo paciente, instanciar un nuevo expediente para el paciente y asignarlo como expediente abierto del paciente. Devuelve la instancia de paciente creada

Interfaz visible desde paquete

- `Paciente()`/*{Constructor}*

Interfaz privada

- `expedienteAbierto`/*{Attributes}*: Último expediente asociado al paciente
- `expedientes`/*{Attributes}*: Conjunto de expedientes del paciente

Enumeración TipoAcceso

Enumeración compuesta de las siguientes variantes: `Consulta`, `Modificacion`, `Creacion` y `Archivo`

TODO

ej2 Apartado a

Basically la clase `MedioPensionista` tiene 2 relaciones de generalización: Una con la clase Activo y otra con la clase Pensionista. En java no se permite la herencia multiple

Código Java

Paquete `ej1`

`Acceso.java`

```
package ej1;

import java.util.Date;

public class Acceso {
  private final Expediente expediente;
  private final Profesional accesor;

  private Date fecha;
  private TipoAcceso tipo;

  private Acceso(Profesional p, Expediente e, TipoAcceso t) {
    // Precondiciones
    assert (p != null);
    assert (e != null);
    assert (t != null);

    this.fecha = new Date(System.currentTimeMillis());
    this.accesor = p;
    this.expediente = e;
    this.tipo = t;
  }

  public static Acceso crearAcceso(Profesional p, Expediente e, TipoAcceso t) {
```

```
    // Si un profesional ya ha accedido, se modifica, si nunca ha accedido, se crea.
    if (p.getAcceso(e) == null) {
      Acceso a = new Acceso(p, e, t);
      p.addAcceso(a);
      e.addAcceso(a);
      return a;
    } else {
      // Si no, se crea y se añaden a la lista.
      Acceso a = p.getAcceso(e);
      a.fecha = new Date(System.currentTimeMillis());
      a.tipo = t;
      return a;
    }
  }

  public Expediente getExpediente() {
    return expediente;
  }

  public Profesional getProfesional() {
    return accesor;
  }

  public Date getFecha() {
    return fecha;
  }

  public TipoAcceso getTipo() {
    return tipo;
  }
}
```

Expediente.java

```
package ej1;

import java.util.Collections;
```

```
import java.util.Enumeration;
import java.util.HashSet;
import java.util.Set;

public class Expediente {
  private final Paciente paciente;
  private final Set<Acceso> acceso;

  Expediente(Paciente p) {
    assert (p != null);

    acceso = new HashSet<>();
    this.paciente = p;
  }

  Acceso getAcceso(Expediente e) {
    for (Acceso a : acceso) {
      if (a.getExpediente().equals(e)) {
        return a;
      }
    }
    return null;
  }

  void addAcceso(Acceso a) {
    acceso.add(a);
  }

  public Enumeration<Acceso> getAccesos() {
    return Collections.enumeration(acceso);
  }

  public Paciente getPaciente() {
    return paciente;
  }
}
```

Paciente.java

```
package ej1;

import java.util.Collections;
import java.util.Enumeration;
import java.util.HashSet;
import java.util.Set;

public class Paciente {
  private final Set<Expediente> expedientes;
  private Expediente expedienteAbierto;

  Paciente() {
    expedientes = new HashSet<>();
  }

  public static Paciente crearPaciente() {
    Paciente p = new Paciente();
    Expediente exp = p.crearExpediente();
    return p;
  }

  public Expediente crearExpediente() {
    Expediente exp = new Expediente(this);
    expedientes.add(exp);
    expedienteAbierto = exp;
    return exp;
  }

  public Enumeration<Expediente> getExpedientes() {
    return Collections.enumeration(expedientes);
  }

  public Expediente getExpedienteAbierto() {
    return expedienteAbierto;
  }
}
```

Profesional.java

```
package ej1;

import java.util.Collections;
import java.util.Enumeration;
import java.util.HashSet;
import java.util.Set;

public class Profesional {
  private final Set<Acceso> acceso;

  public Profesional() {
    acceso = new HashSet<>();
  }

  Acceso getAcceso(Expediente e) {
    for (Acceso a : acceso) {
      if (a.getExpediente().equals(e)) {
        return a;
      }
    }
    return null;
  }

  void addAcceso(Acceso a) {
    acceso.add(a);
  }

  public Enumeration<Acceso> getAccesos() {
    return Collections.enumeration(acceso);
  }
}
```

TipoAcceso.java

```
package ej1;
```

```
public enum TipoAcceso {
  CONSULTA,
  MODIFICACION,
  CREACION,
  ARCHIVO
}
```