INGENIERÍA DEL SOFTWARE II

LABORATORIO – PATRONES DE DISEÑO

2025 - 2026

AUTORES:
Ilargi Quetzalli Matilde López
Gontzal Leizabe Escartin

ÍNDICE

1. SIMPLE FACTORY

Preguntas

A. ¿Qué sucede si aparece un nuevo síntoma (por ejemplo, mareos)?
Las clases *Covid19Pacient* y *Medicament* utilizan cada una su propio método *createSymptom*. Si aparece un nuevo síntoma, hay que modificar directamente el código en ambas clases para incluirlo.
Esto viola el *principio OCP*, ya que las clases no están cerradas a la modificación.

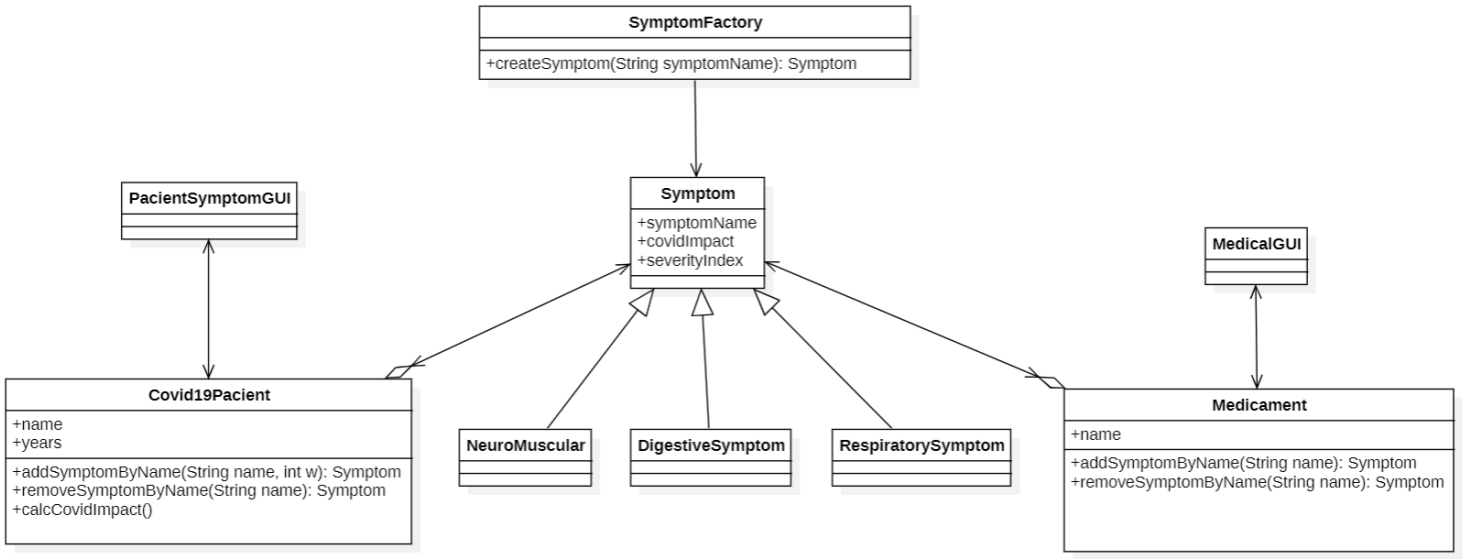
B. ¿Cómo se puede crear un nuevo síntoma sin cambiar las clases existentes (principio OCP)?
Para crear un nuevo síntoma sin cambiar las clases existentes, se puede aplicar el patrón *Simple Factory*, donde la *fábrica* será una clase *SymptomFactory* cuya única responsabilidad será crear los *síntomas*, es decir, el *producto*.
De esta forma, al añadir un nuevo síntoma solo se tendrá que modificar la clase *SymptomFactory*, manteniendo las clases cerradas a la modificación pero abiertas a la extensión, cumpliendo así con el *principio OCP*.

C. ¿Cuántas responsabilidades tienen las clases de Covid19Pacient y Medicament (principio SRP)?
Antes, tanto la clase *Covid19Pacient* como la clase *Medicament* tenían 2 responsabilidades: una era gestionar la información de sus *síntomas*, y la otra era crear los *síntomas*. Esto violaba el *principio SRP* ya que cada clase debería de tener una única responsabilidad.
Tras aplicar el patrón *Simple Factory*, las clases *Covid19Pacient* y *Medicament* solo se encargan de gestionar la información de los síntomas, mientras que la nueva clase *SymptomFactory* se encarga de la creación de nuevos síntomas, cumpliendo así en *principio SRP.*

Tareas

1. Realiza un nuevo diseño de la aplicación (diagrama UML) aplicando el patrón Simple Factory para eliminar vulnerabilidades anteriores y mejorar el diseño en general. Describe con claridad los cambios realizados.

Diagrama UML



Se ha creado la clase *SymptomFactory*, cuya única responsabilidad es la creación de objetos de tipo *Symptom*.
En esta clase se encuentra el método *createSymptom*, que ha sido abstraído de las clases *Covid19Pacient* y *Medicament*, evitando así la duplicidad del código y cumpliendo con el *principio SRP*.
Este método devuelve un objeto de tipo *Symptom* o de sus posibles subclases, como *NeuroMuscular*, *DigestiveSymptom* o *RespiratoySymptom*, según corresponda.
De esta forma si en el futuro se quisiera añadir un nuevo síntoma solo sería necesario modificar la clase *SymptomFactory*, manteniendo el resto de las clases cerradas a la modificación, cumpliendo así con el *principio OCP*.

2. Implementa la aplicación y agrega el nuevo síntoma "mareos" asociado a un tipo de impacto 1.
Se ha eliminado el método *createSymptom* de las clases *Covid19Pacient* y *Medicament*, y se ha creado la clase *SymptomFactory* y añadido el método *createSymptom* en ella.
Además se ha agregado el nuevo síntoma "*mareos*" asociado a un tipo de impacto 1.

```
package factory;

import java.util.Arrays;

import java.util.List;

import domain.DigestiveSymptom;

import domain.NeuroMuscularSymptom;

import domain.RespiratorySymptom;

import domain.Symptom;

public class SymptomFactory {
```

```
      public static Symptom createSymptom(String symptomName) {

       List<String> impact5 = Arrays.asList("fiebre", "tos seca", "astenia","expectoracion");

       List<Double> index5 = Arrays.asList(87.9, 67.7, 38.1, 33.4);

       List<String> impact3 = Arrays.asList("disnea", "dolor de garganta",
"cefalea","mialgia","escalofrios");

       List<Double> index3 = Arrays.asList(18.6, 13.9, 13.6, 14.8, 11.4);

       List<String> impact1 = Arrays.asList("nauseas", "vómitos", "congestión
nasal","diarrea","hemoptisis","congestion conjuntival", "mareos");

       List<Double> index1 = Arrays.asList(5.0, 4.8, 3.7, 0.9, 0.8, 4.9);



       List<String> digestiveSymptom=Arrays.asList("nauseas", "vómitos","diarrea");

       List<String> neuroMuscularSymptom=Arrays.asList("fiebre", "astenia", "cefalea",
"mialgia","escalofrios");

       List<String> respiratorySymptom=Arrays.asList("tos seca","expectoracion","disnea","dolor
de garganta", "congestión nasal","hemoptisis","congestion conjuntival");

       int impact=0;

       double index=0;

       if (impact5.contains(symptomName)) {impact=5; index=
index5.get(impact5.indexOf(symptomName));}

       else if (impact3.contains(symptomName)) {impact=3;index=
index3.get(impact3.indexOf(symptomName));}

       else if (impact1.contains(symptomName)) {impact=1; index=
index1.get(impact1.indexOf(symptomName));}



       if (impact!=0) {

            if (digestiveSymptom.contains(symptomName)) return new
DigestiveSymptom(symptomName,(int)index, impact);

            if (neuroMuscularSymptom.contains(symptomName)) return new
NeuroMuscularSymptom(symptomName,(int)index, impact);

            if (respiratorySymptom.contains(symptomName)) return new
RespiratorySymptom(symptomName,(int)index, impact);

       }

       return null;



      }

}
```

En las clases Cocid19Pacient y Medicament se ha modificado el método addSymptomByName de la siguiente manera:
Antes:

```
public Symptom addSymptomByName(String symptom, Integer w){
         Symptom s=null;
         s=createSymptom(symptom);
         if (s!=null)
             symptoms.put(s,w);
         return s;
        }
```

Ahora:

```
public Symptom addSymptomByName(String symptom, Integer w){

        Symptom s=null;

        s=SymptomFactory.createSymptom(symptom);

        if (s!=null)

            symptoms.put(s,w);

        return s;

    }
```

3. Cómo se puede adaptar la clase Factory, para que los objetos Symptom que utilicen las clases Covid19Pacient y Medicament sean únicos. Es decir, para cada síntoma sólo exista un objeto. (Si hay x síntomas en el sistema, haya únicamente x objetos Symptom)

Para que de cada síntoma exista solo un objeto, se puede implementar un *HashMap* en la clase *SympomFactory*.
Se crearía una variable estática donde la clave sería el nombre del síntoma, de tipo *String*, y el valor sería el objeto de tipo *Symptom* correspondiente.
Además habría que modificar el método *createSymptom* de manera que, antes de crear un nuevo síntoma, compruebe primero si existe ya en el *HashMap*.
En caso de que el objeto esté registrado, devolvería el mismo objeto, y en caso contrario, se crearía y se guardaría en el *HashMap* y finalmente se devolvería.

2. PATRÓN OBSERVER

Tarea

Cambia el programa principal para crear 2 pacientes Covid19Pacient con sus interfaces PacientSymptomGUI y PacientObserverGUI.

```
public class Main {
	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		Observable  pacient=new Covid19Pacient("aitor", 35);
		Observable pacient2=new Covid19Pacient("Jaime",48);
		new   PacientObserverGUI      (pacient);
		new   PacientSymptomGUI((Covid19Pacient) pacient);
		new   PacientObserverGUI      (pacient2);
		new   PacientSymptomGUI((Covid19Pacient) pacient2);
	}
}
```

3. PATRÓN ADAPTER

Tareas

Añade el código necesario en la clase Covid19PacientTableModelAdapter y ejecuta la aplicación para comprobar que funciona correctamente.

```
public class Covid19PacientTableModelAdapter extends AbstractTableModel {
	protected Covid19Pacient pacient;
	protected String[] columnNames =
	new String[] {"Symptom", "Weight" };
	public Covid19PacientTableModelAdapter(Covid19Pacient p) {
	this.pacient=p;
	}
	public int getColumnCount() {
	// Challenge!
		return 2;
	}
	public String getColumnName(int i) {
	// Challenge!
		return columnNames[i];
	}
	public int getRowCount() {
	// Challenge!
		return pacient.getSymptoms().size();
	}
	public Object getValueAt(int row, int col) {
		Set<Symptom> losSintomas = pacient.getSymptoms();
		Object[] symptoms =losSintomas.toArray();
		Symptom s= (Symptom) symptoms[row];
		if(col==0) return s.getName();
		if(col ==1) return pacient.getWeight(s);
		return null;
```

```
        }
    }
```

Añade otro paciente con otro síntomas, y ejecuta la aplicación para que aparezcan los 2 pacientes son sus síntomas.

```
public static void main(String[] args) {
            Covid19Pacient pacient=new Covid19Pacient("aitor", 35);

            pacient.addSymptomByName("disnea", 2);
            pacient.addSymptomByName("cefalea", 1);
            pacient.addSymptomByName("astenia", 3);

            ShowPacientTableGUI gui=new ShowPacientTableGUI(pacient);
            gui.setPreferredSize(
                    new java.awt.Dimension(300, 200));
            gui.setVisible(true);
    }
```