

LABORATORIO Patterns en Eclipse

Leire Gesteira y Daria Paslavska

1. SIMPLE FACTORY

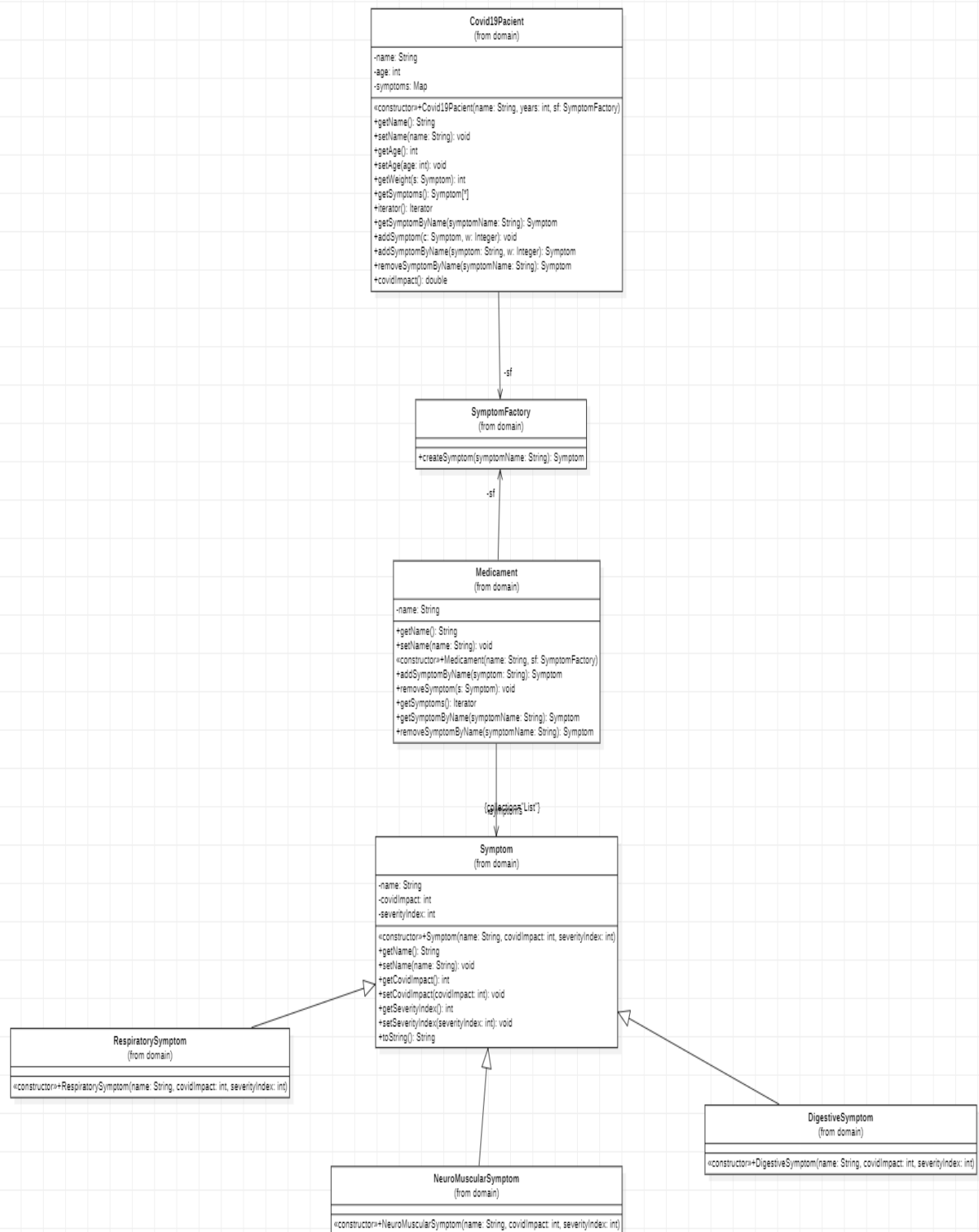
Hay muchas vulnerabilidades en esta aplicación:

- A. ¿Qué sucede si aparece un nuevo síntoma (por ejemplo, mareos)?
 - B. ¿Cómo se puede crear un nuevo síntoma sin cambiar las clases existentes (principio OCP)?
 - C. ¿Cuántas responsabilidades tienen las clases de *Covid19Pacient* y *Medicament* (**principio SRP**)?
-
- A. Habría que escribir manualmente el nuevo síntoma en el método `createSymptom` de la clase `SymptomFactory`.
 - B. Habría que cambiar la clase `Symptom` a una clase madre y que cada clase hija sea un síntoma diferente.
 - C. *Covid19Pacient* tiene 2 responsabilidades y la clase *Medicament* tiene 1 responsabilidad.

Se pide:

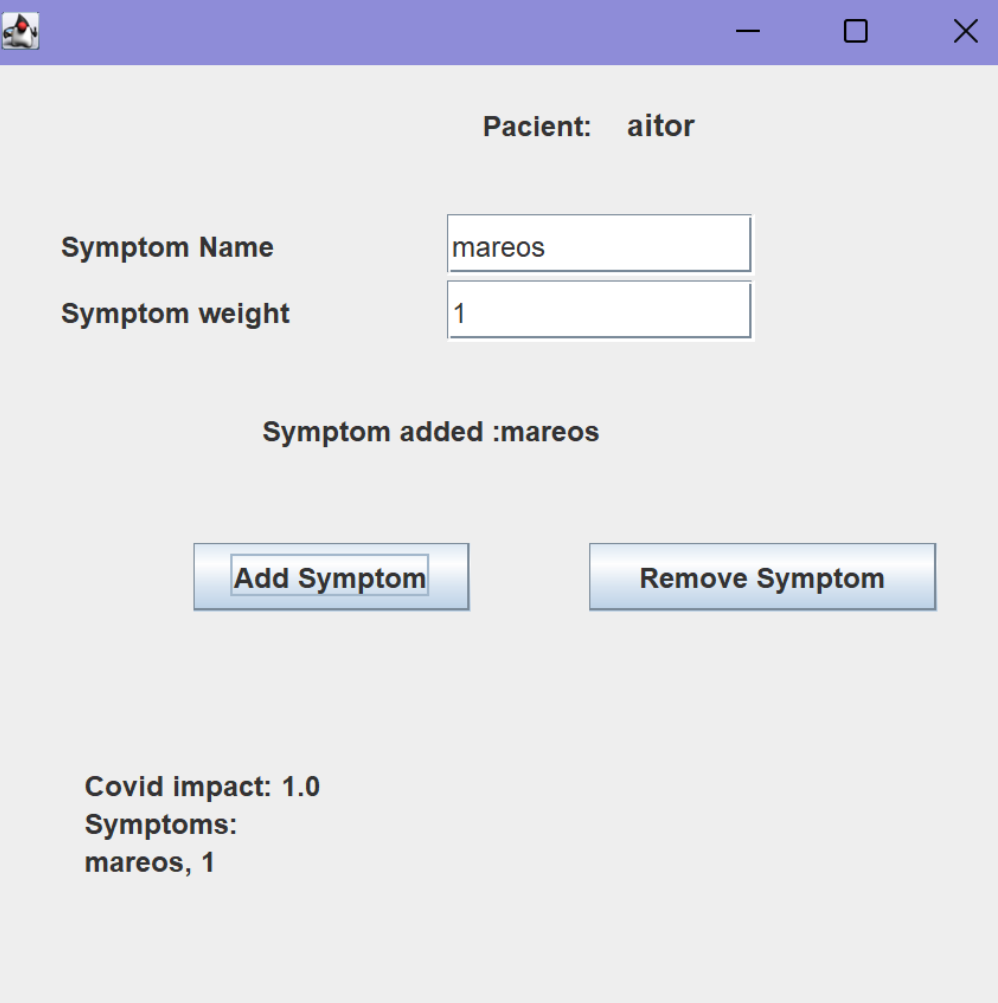
1. Realiza un nuevo diseño de la aplicación (diagrama UML) aplicando el patrón Simple Factory para eliminar vulnerabilidades anteriores y mejorar el diseño en general. Describe con claridad los cambios realizados.
2. Implementa la aplicación y agrega el nuevo síntoma "mareos" asociado a un tipo de impacto 1.
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`)

1.1



Hemos creado la clase `SymptomFactory` que implementa el método `createSymptom(String symptomName)` que añade síntomas a la lista de síntomas.

1.2



The screenshot shows a web application window with a purple title bar. The main content area is light gray. At the top, it says "Pacient: aitor". Below this, there are two input fields: "Symptom Name" with the value "mareos" and "Symptom weight" with the value "1". Below the input fields, it says "Symptom added :mareos". At the bottom, there are two buttons: "Add Symptom" and "Remove Symptom". At the very bottom, it says "Covid impact: 1.0" and "Symptoms: mareos, 1".

Pacient: aitor

Symptom Name

Symptom weight

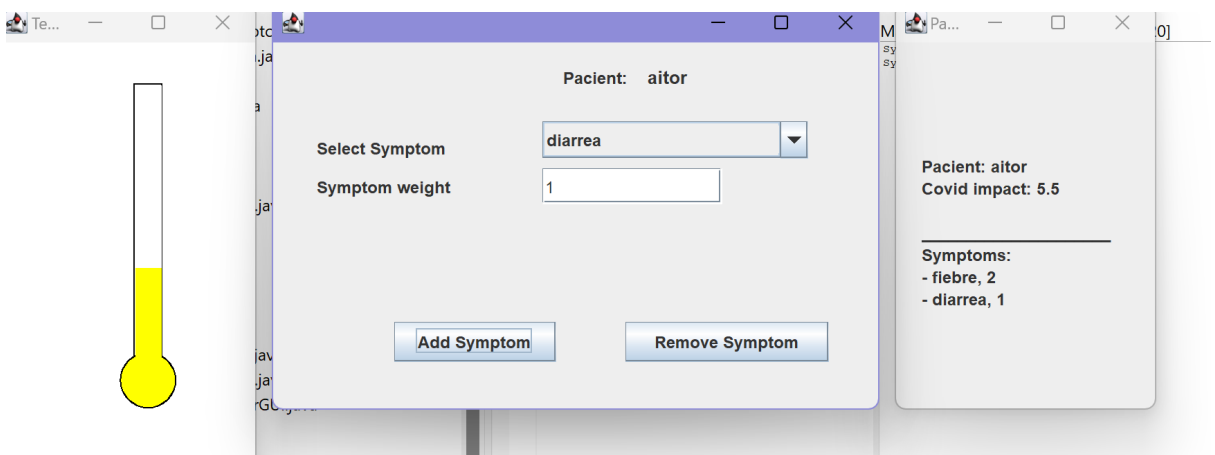
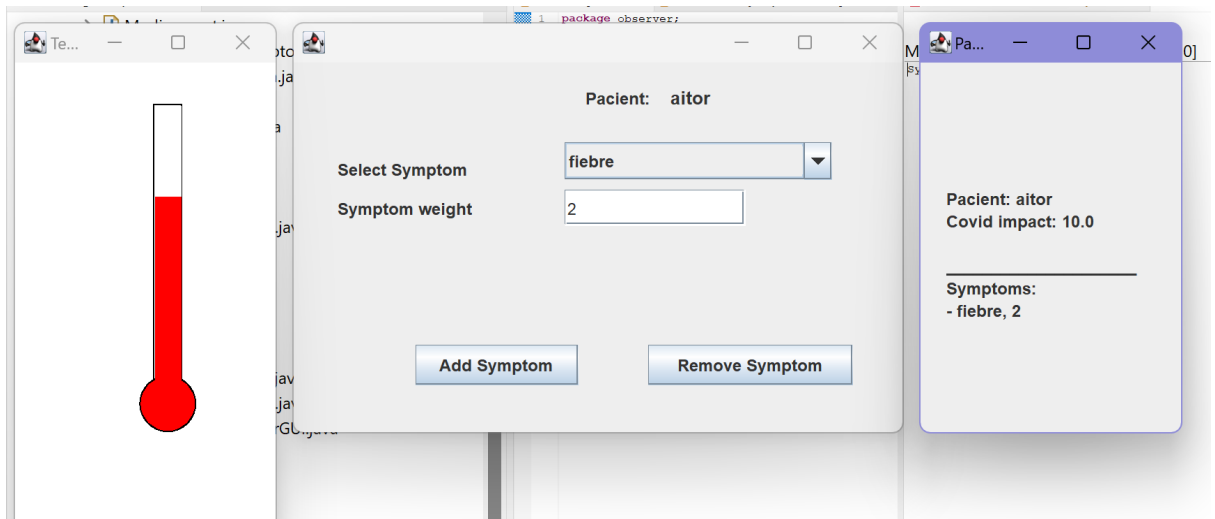
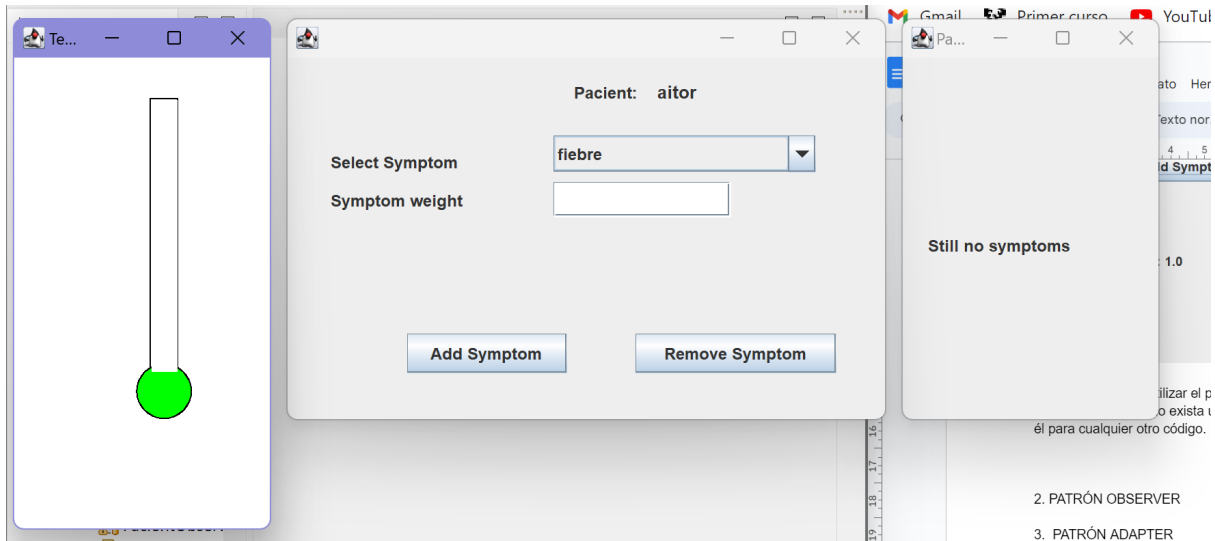
Symptom added :mareos

Covid impact: 1.0
Symptoms:
mareos, 1

1.3

Una solución sería utilizar el patrón Singleton que es un patrón de diseño creacional que garantiza que tan solo exista un objeto de su tipo y proporciona un único punto de acceso a él para cualquier otro código.

2. PATRÓN OBSERVER



3. PATRÓN ADAPTER

The screenshot shows two Java Swing windows titled "Covid Symptoms aitor" and "Covid Symptoms maria". Both windows display a table with three columns: "Name", "Symptom", and "Weight".

| Name | Symptom | Weight |
|-------|---------|--------|
| aitor | fiebre | 1 |
| aitor | disnea | 2 |
| aitor | nauseas | 3 |

| Name | Symptom | Weight |
|-------|---------|--------|
| maria | mialgia | 3 |
| maria | diarrea | 2 |
| maria | fiebre | 1 |

Below the tables, there is a code editor showing the implementation of the adapter pattern. The code includes a `ShowPacientTableGUI` class with a `gui2=new ShowPacientTableGUI(paciente2);` line and an `@Override` method.

4. PATRÓN ITERATOR Y ADAPTER

```
<terminated> Main (6) [Java Application] C:\Users\Владелец\.p2\pool\plugins\org.eclipse.justj.openj
Síntoma: s1, Índice de gravedad: 8
Síntoma: s2, Índice de gravedad: 10
Síntoma: s3, Índice de gravedad: 5
Síntoma: s4, Índice de gravedad: 10
Síntoma: s5, Índice de gravedad: 10

Síntoma: s3, Índice de gravedad: 5
Síntoma: s1, Índice de gravedad: 8
Síntoma: s5, Índice de gravedad: 10
Síntoma: s4, Índice de gravedad: 10
Síntoma: s2, Índice de gravedad: 10
```