Patrones State y Strategy

Ejemplo 1

- Desarrollar la clase AulaVirtual que tiene un nombre, un profesor administra una lista de asistentes, mensajes de chat. La clase tiene el siguiente protocolo

>>ingresoAsistente (Asistente)

>>levantarMano(Asistente): boolean

>>enviarMensaje (Asistente, mensaje)

>>iniciarClase

>>finalizarClase

Ejemplo 1

>>ingresoAsistente: solo permite el ingreso si la clase no está empezada

>>levantarMano: solo se permite preguntar si el profesor que está hablando permitePreguntas. Antes de iniciada la clase se puede levantar sin condiciones. Retorna true si se permitió levantar la mano.

>>enviarMensaje: mientras no esté la clase empezada no se controla el tipo de mensaje. Una vez iniciada la clase, los mensajes no pueden extender la longitud de 20 letras. A los mensajes antes de empezar la clase se les antecede el apellido, una vez empezada la clase el nick.

Ejemplo 1

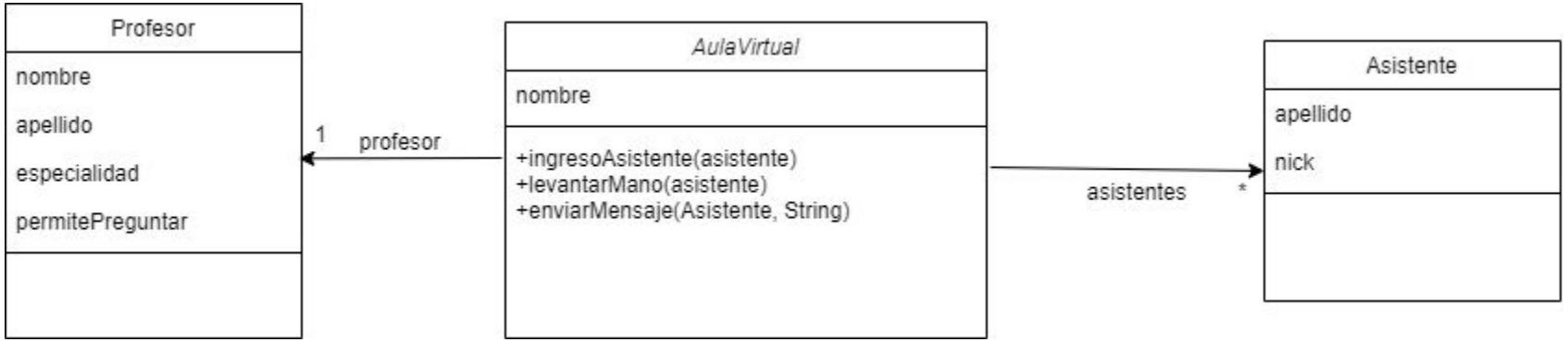
- Una vez que la clase está terminada

>>ingresoAsistente (Asistente): no se permite ingresos

>>levantarMano(Asistente): no se permite levantar la mano

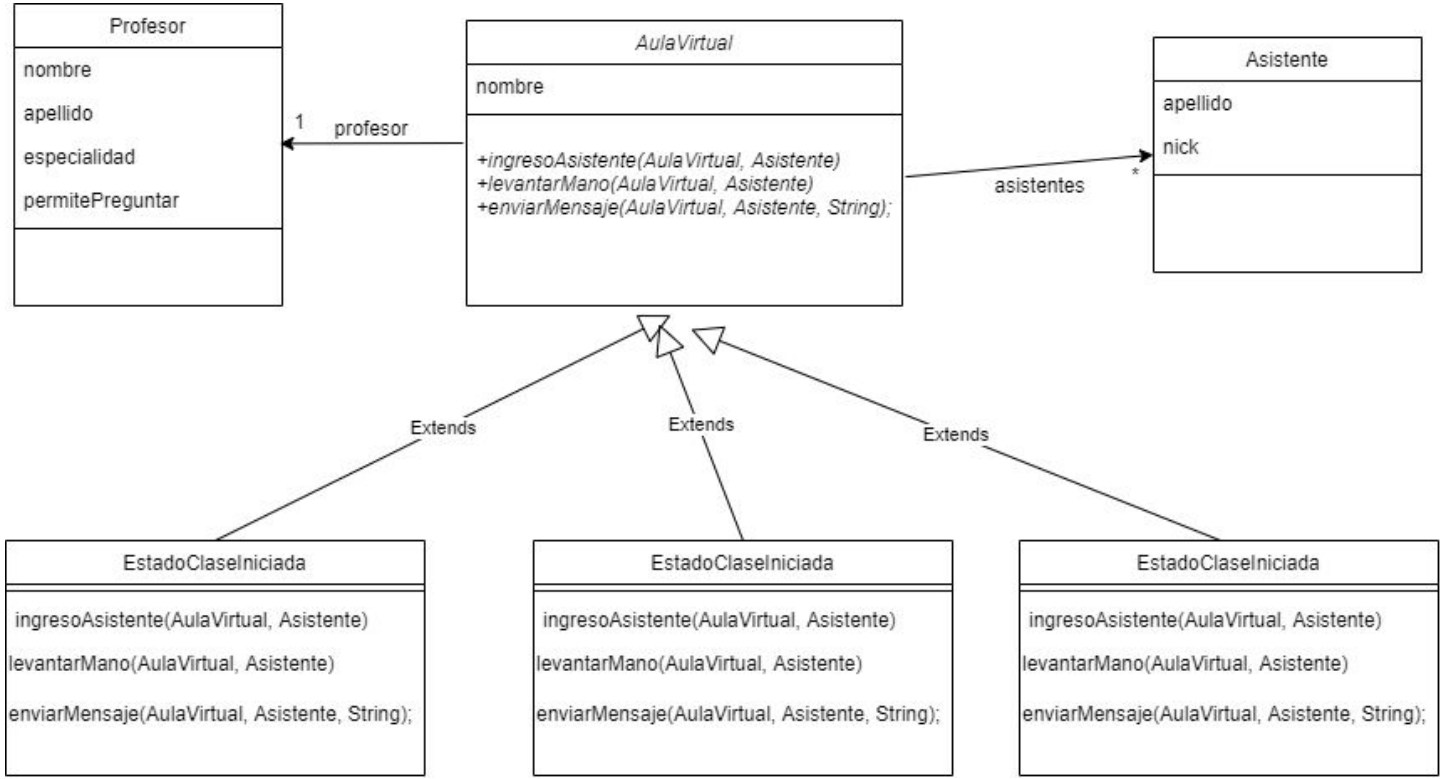
>>enviarMensaje (Asistente, mensaje): solo puede enviar mensajes aquellos asistentes que hayan enviado por lo menos 5 mensajes previamente.

Ejemplo 1

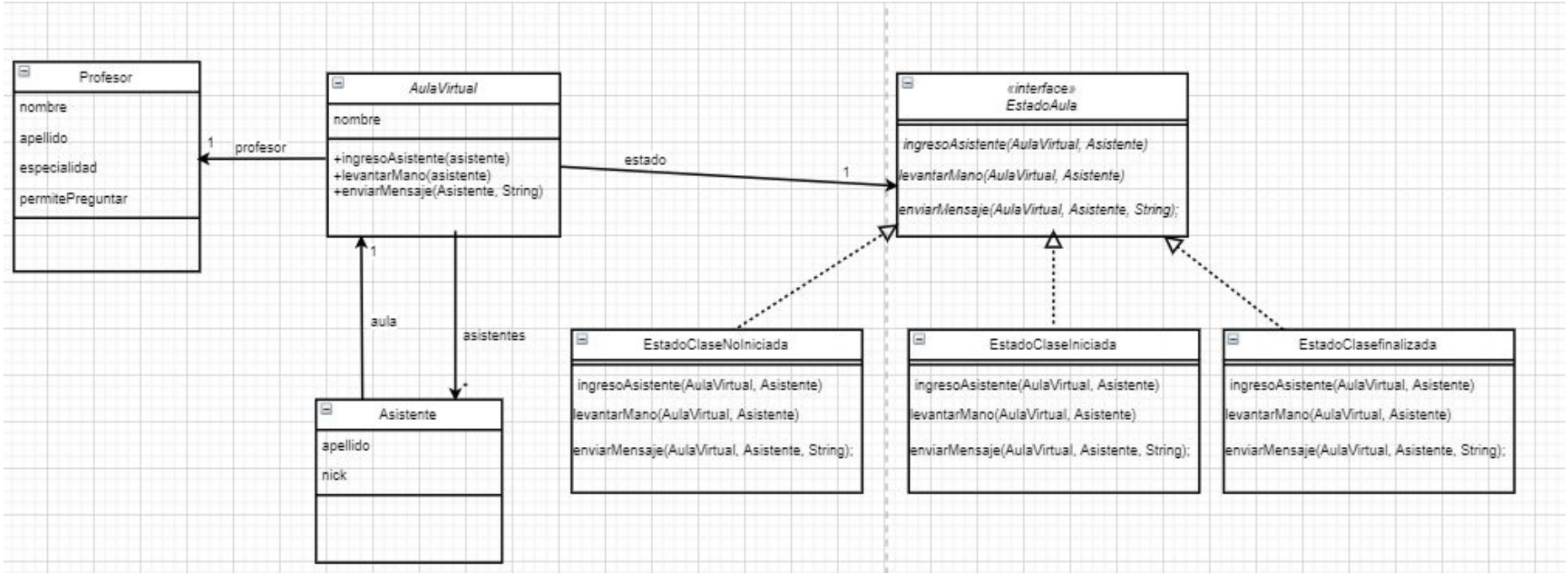


enviarMensaje: solución llena de if/switch. Poco clara y poco flexible al cambio.
¿Puedo extender el comportamiento sin tener que modificar la clase?

Ejemplo 1


Profesor
nombre
apellido
especialidad
permitePreguntar
1
profesor
AulaVirtual
nombre
+ingresoAsistente(AulaVirtual, Asistente)
+levantarMano(AulaVirtual, Asistente)
+enviarMensaje(AulaVirtual, Asistente, String);
asistentes
*
Asistente
apellido
nick
Extends
Extends
Extends
EstadoClaseIniciada
ingresoAsistente(AulaVirtual, Asistente)
levantarMano(AulaVirtual, Asistente)
enviarMensaje(AulaVirtual, Asistente, String);
EstadoClaseIniciada
ingresoAsistente(AulaVirtual, Asistente)
levantarMano(AulaVirtual, Asistente)
enviarMensaje(AulaVirtual, Asistente, String);
EstadoClaseIniciada
ingresoAsistente(AulaVirtual, Asistente)
levantarMano(AulaVirtual, Asistente)
enviarMensaje(AulaVirtual, Asistente, String);

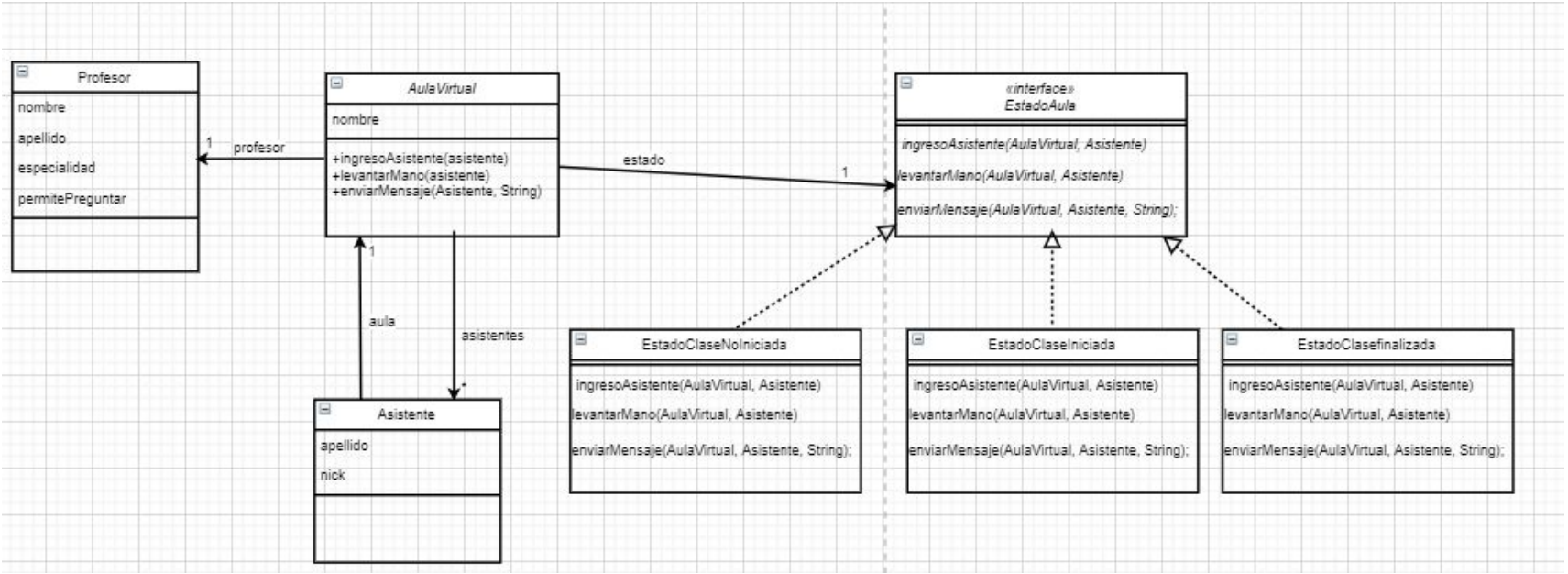
Ejemplo 1


Profesor
nombre
apellido
especialidad
permitePreguntar
1
profesor
AulaVirtual
nombre
+ingresoAsistente(asistente)
+levantarMano(asistente)
+enviarMensaje(Asistente, String)
estado
1
«interface»
EstadoAula
ingresoAsistente(AulaVirtual, Asistente)
levantarMano(AulaVirtual, Asistente)
enviarMensaje(AulaVirtual, Asistente, String);
1
aula
asistentes
*
Asistente
apellido
nick
EstadoClaseNoIniciada
ingresoAsistente(AulaVirtual, Asistente)
levantarMano(AulaVirtual, Asistente)
enviarMensaje(AulaVirtual, Asistente, String);
EstadoClaseIniciada
ingresoAsistente(AulaVirtual, Asistente)
levantarMano(AulaVirtual, Asistente)
enviarMensaje(AulaVirtual, Asistente, String);
EstadoClasefinalizada
ingresoAsistente(AulaVirtual, Asistente)
levantarMano(AulaVirtual, Asistente)
enviarMensaje(AulaVirtual, Asistente, String);

```
public void testNuevoAsistenteAntesDeEmpezar() {

public void testNuevoAsistenteClaseEmpezada() {

public void testNuevoAsistenteClaseTerminada() {

public void testLevantarManoClaseEmpezadaConProfeQueNoPermite() {

public void testLevantarManoClaseEmpezadaConProfeQuePermite() {

public void testLevantarManoAntesDeEmpezarConProfeQueNoPermite() {

public void testLevantarManoAntesDeEmpezarConProfeQuePermite() {

public void testLevantarManoClaseTerminadaConProfeQuePermite() {

public void testLevantarManoClaseTerminadaConProfeQuePermiteAntesDeEmpezar() {

public void testMensajeCortoAntesDeEmpezar() {

public void testMensajeLargoAntesDeEmpezar() {

public void testMensajeCortoClaseEmpezada() {

public void testMensajeLargoClaseEmpezada() {

public void mensajeClaseFinalizada() {

public void mensajeSinColor() {
```

Ejemplo 2

- Qué cambios tengo que hacer si quiero implementar el estado recreo?
- Puedo tener varios recreos?



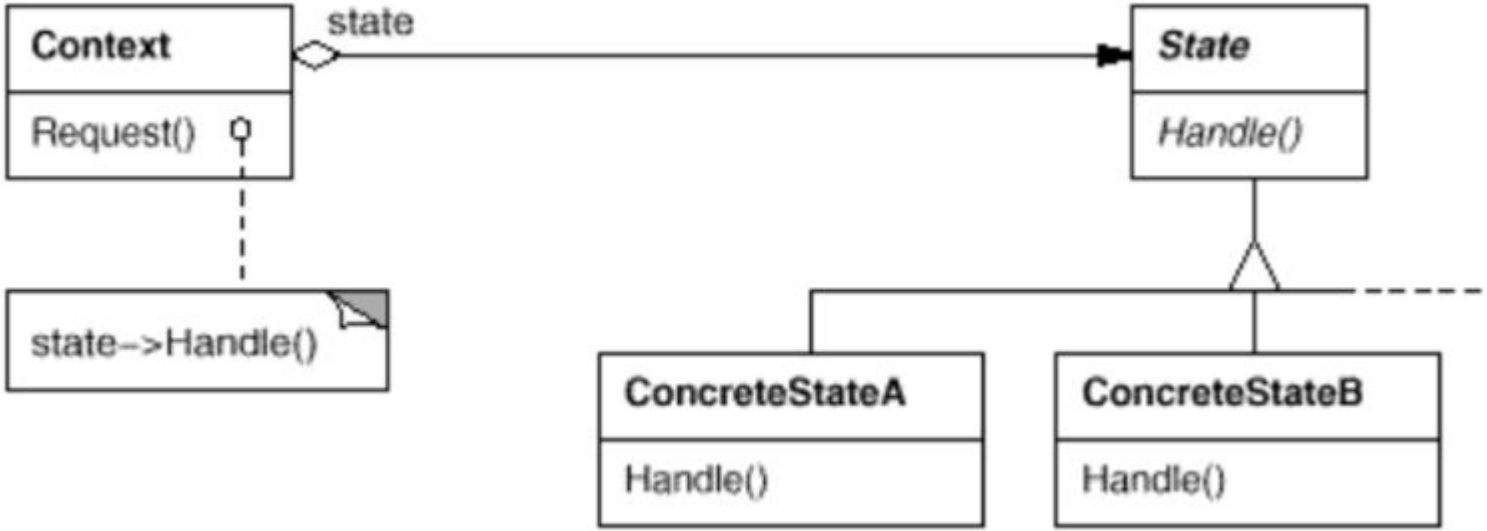
Ejemplo 2

- Desarrollar un sistema que permita a un maratonista profesional:
  - entrenar
  - dieta
  - proximaCompetencia
- Cualquiera de estas actividades será diferente según si el maratonista está en forma o lesionado.
  - El entrenamiento si está lesionado será solo de caminata. Si está en forma, será correr a máxima velocidad 3 veces al día.
  - La dieta en caso de que esté lesionado será manejada por un equipo especializado en lesiones. De lo contrario seguirá la dieta recomendada por su entrenador.
  - La próxima competencia será según el cronograma de maratones en caso que este en forma. Si está lesionado debe indicar que no es conocida.

Patrón State

- Aplica cuando el comportamiento de un objeto (AulaVirtual) depende de su estado.
- El estado puede cambiar en runtime
- Alternativa?
  - 1-Si son dos estado: boolean e if/switch por todos lados. No!!
    - Qué pasa si agrego un estado más?
  - 2-Represento con Strings los distintos estados para soportar más de dos estados. No!!
    - Cada método que dependa del estado será un embroolo de if/switch. Poco claro y flexible al cambio.
  - Herencia? Que pasa cuando cambia de estado?
  - State
    - Represento cada estado con una clase y defino el comportamiento propio en cada uno. El objeto delega en el estado todo comportamiento que dependa del mismo.

State


Context
state
Request()
state->Handle()
State
Handle()
ConcreteStateA
Handle()
ConcreteStateB
Handle()

Patrón State

- Algunos puntos para discutir
  - ¿Dónde se inicializa el estado inicial?
  - ¿Qué ocurre si los estados necesitan del objeto contexto (AulaVirtual) para operar?
  - ¿Quién hace el cambio de estado?
  - Instancio cada vez que cambio de estado?
  - Está bien que el estado delegue en el contexto (AulaVirtual)?