Práctica Parte 2 : If Replacement

Tomás Felipe Melli

April 12, 2025

Índice

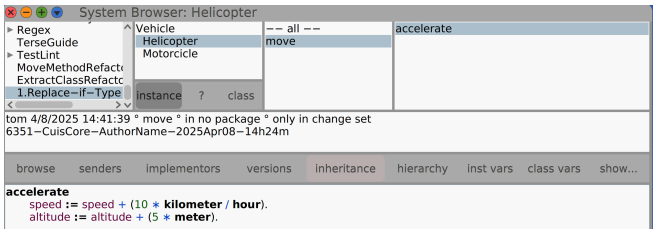
1 Introducción

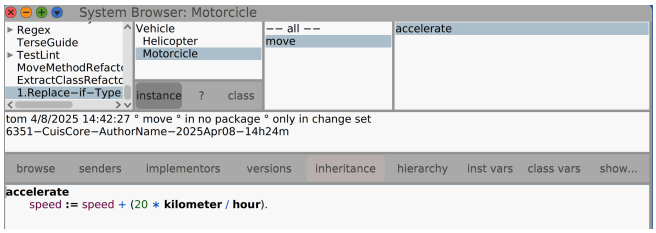
La idea de estos ejercicios es, gracias al polimorfismo (:= Se dice que los objetos del primer conjunto son polimorficos entre si respecto de los mensajes del segundo conjunto, si los objetos del primero responden los mensajes del segundo semantcamente igual) eliminar **if statements**.

2 Parte 1: Type

La idea es que inicialmente tenemos algo como, **si el type** del vehículo es : Helicopter $\implies$ hace tal cosa. **si el type** del vehículo es : Motorcycle $\implies$ hace tal cosa. Para resolver esto, segumimos los pasos : **creamos la jerarquía polimórfica** (en este caso, ya tenemos Vehicle (no hace falta crearla)) por tanto, dentro de ella creamos los dos types. Usamos el mismo **mensaje** y le movemos el cuerpo del if :

```
Vehicle subclass: #Helicopter
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
    category: '1.Replace−if−Type'
```





3 Parte 2 : State

En este ejercicio tenemos que sacar 2 Ifs. Dentro del mensaje *accelerate* del Auto. Se definen **dos** estados **On/Off**. Primer paso, creamos la jerarquía polimórfica, le podemos poner EstadoAutomobile por ejemplo. Escribimos el mensaje con el mismo nombre y ponemos el cuerpo de cada If dentro.

```
EstadoAutomobile subclass: #AutomobileOn
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
    category: '2.Replace−if−State'
```

```
EstadoAutomobile subclass: #AutomobileOff
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
    category: '2.Replace−if−State'
```

El problema es que *speed* es un colaborador interno de Automobile. Para resolver esto, la instancia de la clase EstadoAutomobile debería conocer al auto al cual representa su estado. Dicho esto, tenemos que lograr conectarlos entre sí :

```
turnOff

    powerStatus := PowerOff on: self
```

```
on: vehicle

    ^ self new initializePowerStatusOf: vehicle
```

En la inicialización del auto, que comienza apagado, le asignamos la referencia a su estado con el estado que definimos. Simultaneamente, inicializamos (como mensaje de clase) el Estado del auto (esto está en la parte de Class). A través de esta manera, el estado conocerá al auto al que referencia.
Luego, como la responsabilida de acelerar es del auto, tenemos que chequear el estado, esperar la directiva del estado que referenciamos y luego acelerar si podemos. Por ello,

```
accelerate

    vehicleToPower accelerateWhenTurnedOn
```

```
accelerate

    vehicleToPower accelerateWhenTurnedOff
```

Donde *vehicleToPower* es el colaborador interno de la clase Estado que mantiene la referencia. Esto sucede ya que el mensaje de la clase *on:* llama a initialize que se define como :

```
initializePowerStatusOf: vehicle

    vehicleToPower := vehicle
```

4 Parte 3 : Type + State

En este ejercicio tenemos en el mensaje *accelerate* de Vehicle el State y los dos types con Ifs. Tenemos unos mensajes de error si no podemos acelerar, también con ifs. La idea es análoga a lo que venimos haciendo. Jerarquía polimórfica, pasamos el cuerpo del if, y vemos cómo coordinamos todo para que funcione.