Orientación a Objetos 2 – Práctica 2

Rev 2018: Diego Torres - Leandro Antonelli
Rev 2016: Julián Grigera - Emiliano Perez - Mariela Zurbano

Material adicional de este TP: https://goo.gl/LVDcMV

Para esta práctica lea el capítulo 4 del libro Design Patterns de Gamma et al. y responda a las siguientes preguntas:

1. ¿Qué es un patrón estructural?
2. El capítulo menciona dos formas de implementar el patrón Adapter: como patrón estructural de clase y como patrón estructural de objeto. ¿Cuáles son esas dos formas? ¿Cuál de ellas no es implementable en Smalltalk? ¿Por qué?

Ejercicio 1: Publicaciones en redes sociales

Ud. ha implementado un exitoso cliente desktop que permite publicar en Facebook su estado de ánimo.

En su sistema la clase `Facebook` implementa el mensaje `#post:`, donde recibe un string (sin límite de longitud) para ser publicado.

Dada la popularidad de Twitter, ahora ud. desea que su cliente también permita publicar en dicha red social. Lamentablemente se encuentra con 2 problemas:

- Los tweets son strings de a lo sumo 140 caracteres. Si el mensaje supera este tamaño, solo publica los primeros 140 caracteres.
- La clase Twitter no entiende el mensaje `#post:` sino `#publish:`. También recibe como parámetro un String.

Además, como su software funciona correctamente, ud. desea introducir el menor número posible de modificaciones. A la vez, la clase Twitter pertenece a una librería de terceros y no debe modificarla.

Tareas:

1. Diseñe la aplicación original (considerando Facebook pero no Twitter).
2. Discuta con el ayudante la inclusión del soporte a Twitter.
3. Explique cómo el patrón resuelve los problemas planteados.
4. Implemente en Smalltalk.

Ejercicio 2: Friday the 13th en Smalltalk

La clase Biblioteca implementa la funcionalidad de exportar el listado de sus socios en un formato JSON. Para ello define el método #exportarSocios de la siguiente forma:

```
Biblioteca>>exportarSocios
"Retorna la representación JSON de la colección de socios."
      ^ self exporter export: (self socios).
```

La Biblioteca delega la responsabilidad de exportar en una instancia de la clase VoorheesExporter que dada una colección de socios retorna un texto con la representación de la misma en formato JSON, esto lo hace mediante el mensaje de instancia **`#export:`**.

De un socio es posible conocer el nombre, el email y el número de legajo. Por ejemplo, para una biblioteca que posee una colección con los siguientes socios:

● Nombre: Arya Stark ● e-mail:needle@stark.com ● legajo: 5234/5	● Nombre: Tyron Lannister ● e-mail:tyron@thelannisters.com ● legajo: 2345/2

Haciendo “Print it” de las siguientes expresiones en un Playground:

```
| miBiblioteca arya tyron |
miBiblioteca:= Biblioteca new: VoorheesExporter new.
arya:= Socio nombre:'Arya Stark' email:'needle@stark.com' legajo: '5234/5'.
tyron:= Socio nombre:'Tyron Lannister' email:'tyron@thelannisters.com' legajo:'2345/2'.
miBiblioteca agregarSocio: arya.
miBiblioteca agregarSocio: tyron.
miBiblioteca exportarSocios.
```

El JSON que se generará es :

```
[
      {
            "nombre" : "Arya Stark",
            "email" : "needle@stark.com",
            "legajo" : "5234/5"
      },
      {
            "nombre" : "Tyron Lannister",
            "email" : "tyron@thelannisters.com",
            "legajo" : "2345/2"
      }
]
```

Note los corchetes de apertura y cierre de la colección, las llaves de apertura y cierre para cada socio y la coma separando a los socios.

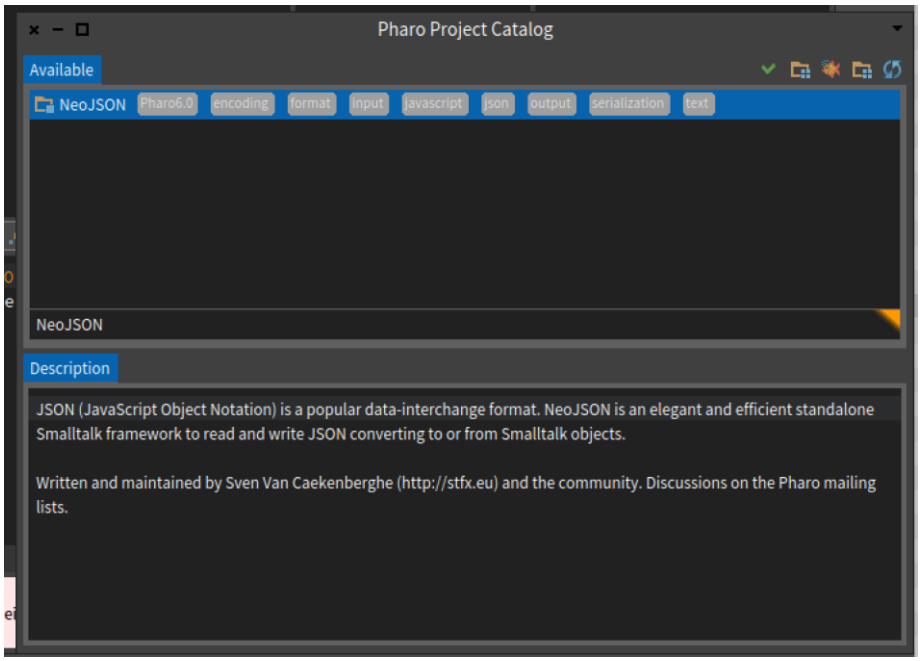
Usando el paquete NeoJSON

El paquete NeoJSON para Pharo incluye a la clase NeoJSONWriter la cual permite generar la representación JSON de diferentes objetos. Mediante el mensaje de clase #toStringPretty: es posible enviarle un diccionario o una colección con diccionarios para obtener su representación en formato JSON.

Su nuevo desafío consiste en utilizar la clase NeoJSONWriter para imprimir en formato JSON a los socios de la Biblioteca en lugar de utilizar la clase VoorheesExporter. Pero con la siguiente condición: **nada de esto debe generar un cambio en el código de la clase Biblioteca.**

Tareas

1. Analice la implementación de la clase Biblioteca y VoorheesExporter que se provee con el material adicional de esta práctica (Archivo Biblioteca.st).
2. Instale en el ambiente Pharo el paquete NeoJSON. Para ello debe dirigirse a:
    a. World / Tools / Catalog Browser.
    b. En la ventana escribir NeoJSON, seleccionar el paquete con ese nombre y presionar en el botón Install Stable Version, como muestra la siguiente imagen:



3. Inspeccione las siguientes expresiones en un playground y analice el resultado.
    a. col:= OrderedCollection with: (Dictionary new at: #x put: 1; at: #y put: 2; yourself) with: (Dictionary new at: #x put: 3; at: #y put: 4; yourself).
    b. NeoJSONWriter toStringPretty: col.
4. Ahora debe utilizar la clase NeoJSONWriter para imprimir en formato JSON a los socios de la Biblioteca en lugar de la clase VoorheesExporter sin que esto genere un cambio en el código de la clase Biblioteca.
    a. Modele una solución a esta alternativa utilizando un diagrama de clases UML. Si utiliza patrones de diseño indique los roles en las clases utilizando estereotipos.
    b. Implemente en Smalltalk la totalidad de la solución.