Orientación a Objetos 2 – Práctica 6

Ejercicio 1: Acceso a la base de datos

Queremos acceder a una base de datos que contiene información sobre comics. Este acceso está dado por el comportamiento de la clase DatabaseRealAccess con el siguiente comportamiento y modelado como muestra la Figura 1.

```
>>getSearchResults: queryString
El cual retorna un arreglo de acuerdo al texto que posee
queryString.

>>insertNewRow: rowData
Que realiza la inserción de nueva información en la base de datos
y retorna el id que recibe la nueva inserción.
```

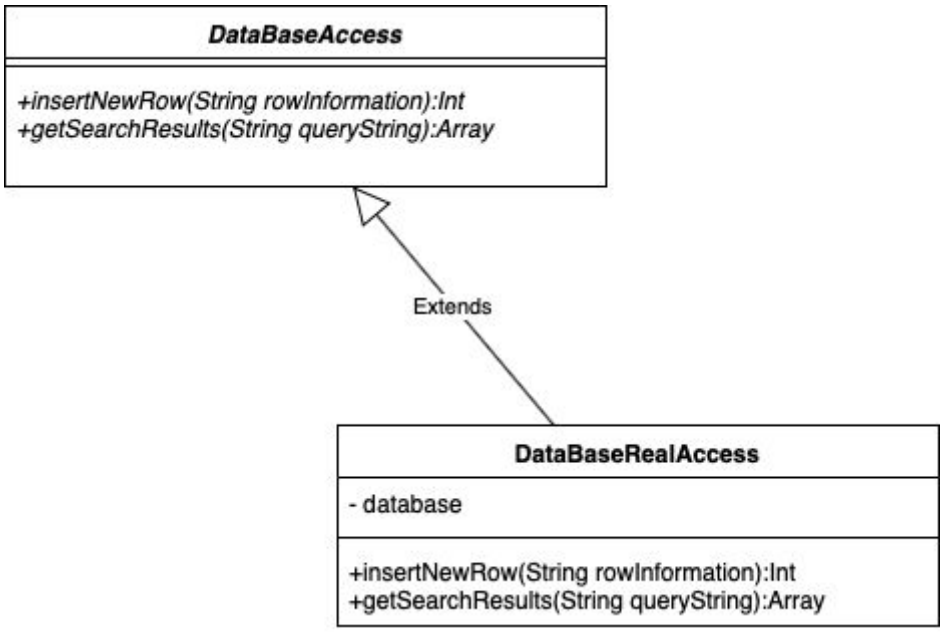

Figura 1

En este caso, ustedes recibirán una implementación prototipica de la clase DatabaseRealAccess (ver material extra) que simula el uso de una base datos de la siguiente forma (mire el código y los test para entender cómo está implementada).

```
database:= DatabaseRealAccess new. "Instancia una base de datos
que posee dos filas"

database getSearchResults: 'select * from comics where id=1'.
"retorna el siguiente arreglo #('Spiderman' 'Marvel')."
```

```
rowData:= #('Paturuzu' 'La flor').
database insertNewRow: rowData.
"retorna 3, que es el id que se le asigna"

database getSearchResults: 'select * from comics where id=3'.
"Retorna un arreglo igual a rowData ya que lo inserto antes"
```

Desafío

En esta oportunidad, usted debe implementar un *protection proxy* para que puedan realizar el acceso a la base de datos, solamente usuarios que se hayan autenticado antes en forma correcta.

1) Realice el diseño de su solución.
2) Implemente completamente en Pharo.

Ejercicio 2: FileManager

En un **FileManager** se muestran los archivos. De los archivos se conoce:

- Nombre
- Extensión
- Tamaño
- Fecha de creación
- Fecha de modificación
- Permisos

Implemente la clase **FileOO2**, con las correspondientes variables de instancia y accessors.

En el FileManager el usuario debe poder elegir cómo se muestra un archivo (instancia de la clase FileOO2), es decir, cuáles de los aspectos mencionados anteriormente se muestran, y en qué orden. Esto quiere decir que un usuario puede querer ver los archivos de *6 factorial* maneras, alguna de ellas son:

- `nombre - extensión`
- `nombre - extensión - fecha de creación`
- `permisos - nombre - extensión - tamaño`

Para esto, el objeto o los objetos que representen a los archivos en el FileManager debe(n) entender el mensaje `#prettyPrint`.

Es decir, un objeto cliente (digamos el FileManager) le enviará al objeto que Ud. determine el mensaje `#prettyPrint`. **De acuerdo a cómo el usuario lo haya configurado se deberá retornar un String con los aspectos seleccionados por el usuario en el orden especificado por éste**. Considere que un mismo archivo podría verse de formas diferentes desde distintos

puntos del sistema, y que el usuario puede cambiar la configuración del sistema (qué y en qué orden quiere ver) en runtime.

Tareas
1) Discuta los requerimientos y diseñe una solución. Si aplica un patrón de diseño, indique cuál es y justifique su aplicabilidad.
2) Implemente en Pharo.
3) Cree en un Playground un objeto para cada uno de los ejemplos citados anteriormente.