

Orientación a Objetos 2 – Práctica 4

Ejercicio 1: Acceso a la base de datos

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

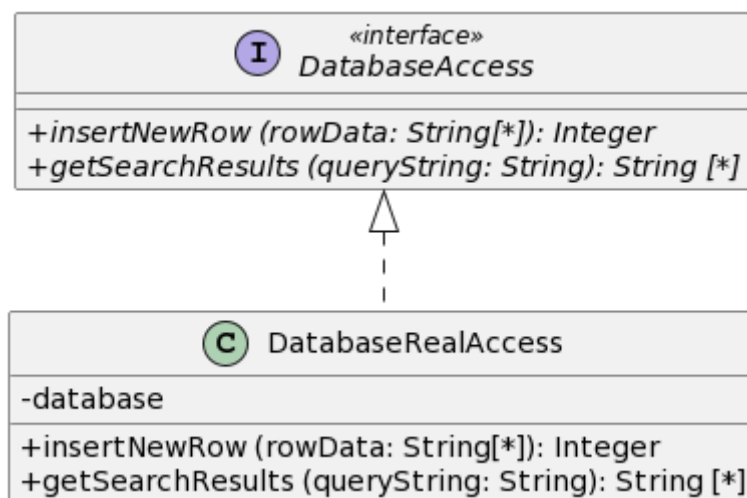


Figura 1

```

public interface DatabaseAccess {
    /**
     * Retorna una colección de acuerdo al texto que posee
     "queryString"
     *
     * @param queryString
     * @return
     */
    public Collection<String> getSearchResults(String queryString);

    /**
     * Realiza la inserción de nueva información en la base de datos y
     * retorna el id que recibe la nueva inserción
     *
     * @param rowData
     * @return
     */
    public int insertNewRow(List<String> rowData);
}
    
```



```
}
```

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

```
// Instancia una base de datos que posee dos filas
database = new DatabaseRealAccess();

// Retorna el siguiente arreglo: ['Spiderman' 'Marvel'].
database.getSearchResults("select * from comics where id=1");

// Retorna 3, que es el id que se le asigna
database.insertNewRow(Arrays.asList("Patoruzú", "La flor"));

// Retorna el siguiente arreglo: ['Patoruzú', 'La flor'], ya
que lo insertó antes
database.getSearchResults("select * from comics where id=3");
```

Tareas

En esta oportunidad, usted debe proveer un *protection proxy* para que el acceso a la base de datos lo puedan realizar solamente usuarios que se hayan autenticado previamente. Su tarea es diseñar y programar en Java lo que sea necesario para ofrecer la funcionalidad antes descrita. Se espera que entregue los siguientes productos.

- 1) Diagrama de clases UML.
- 2) Implementación en Java de la funcionalidad requerida.
- 3) Implementación de los tests (JUnit) que considere necesarios.



Ejercicio 2: File Manager

En un **File Manager** 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 File Manager 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 podría querer ver los archivos de muchas maneras. Algunas 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 podría 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 Java.
- 3) Instancie un objeto para cada uno de los ejemplos citados anteriormente y verifique escribiendo tests de unidad.