Calidad de los Sistemas Informáticos, 2020-2021
Práctica 3 – Clase de mapeo (2)

Objetivo

El objetivo de esta práctica es la ampliación de la clase creada en la práctica 2. En concreto, se crearán los métodos de actualización, eliminación y búsqueda de resultados, estático este último. De igual modo, se completarán las precondiciones.

Requisitos previos

Para la realización de esta práctica se requieren los recursos y resultados de las prácticas previas.

Criterios sintácticos

- `Courier`: código fuente, nombres de archivos, paquetes, entidades, atributos, tablas y campos.
- *`Cursiva`*: términos en otro idioma.
- Negrita: contenido resaltado.
- **$**Entre dólar**$**: contenido no textual, generalmente a decidir por el desarrollador.

Instrucciones previas

- Salvo los métodos de tipo `get` y `set`, todos los métodos realizados se comentarán, incluyendo descripción y excepciones. Para ello, en Eclipse se pulsará `shift+alt+J` sobre la cabecera del método.
- En aquéllas partes del código en la que se consideren adecuadas precondiciones (generalmente, al inicio de cada método), que deberán indicarse mediante el comentario hasta el último punto de esta práctica, en el que se propone la implementación de las mismas.

```
// TODO: Preconditions
```

Pablo de la Torre Moreno
Departamento de Ingeniería Informática

Procedimiento

1. Implementación de los métodos restantes

1. Crear el método `Delete` en la clase **$**`Entidad`**$**, el cual (si la instancia no está marcada ya como eliminada [en otro caso, genera una excepción]) elimina el registro de la base de datos y marca la instancia como eliminada, en una variable `_bIsDeleted` y propiedad de lectura (y sólo lectura) correspondiente creada a tal efecto.
2. Crear el método `Update` en la clase **$**`Entidad`**$**, el cual (si el registro no está marcado como eliminado [en otro caso, genera una excepción]) envía los valores de escritura desde las variables privadas a la base de datos, en el registro correspondiente.
3. Crear el método estático `Select` en la clase **$**`Entidad`**$**, que recibe tantos parámetros de entrada como campos por los que se puede buscar[1]. Dicho método devuelve un listado de instancias de tipo **$**`Entidad`**$** (tipo `ArrayList<`**$**`Entidad`**$**`>` o similar), para lo cual, dentro de dicho método, se debe llamar a otro método privado `Where` que, dados los mismos parámetros de entrada, devuelva la cadena de consulta SQL `WHERE` *`condiciones`*. Dicho método deberá ser también implementado.

2. Pruebas

1. Implementar en la clase **$**`Entidad`**$**`Test` los métodos `testSelect`, `testUpdate` (obteniendo una instancia, modificando todos sus datos, invocando a su `Update()`, obteniéndola otra vez y comprobando que todos sus campos son los modificados) y `testDelete` (comprobando que el registro ya no está en la base de datos y que el método `getIsDeleted` de **$**`Entidad`**$** es `true`) en las condiciones y la completitud que se determinen, utilizando los métodos de `assertEquals` para comprobar el buen funcionamiento de las mismas.

3. Implementación de las precondiciones

1. Opcionalmente, implementar las precondiciones. Cada método deberá lanzar una excepción si no se cumplen.

---

[1] Todos los campos de tipo cadena se buscarán con `LIKE`, salvo que éstos incluyan caracteres comodín (`%` y/o `?`). Si un parámetro de entrada es `null` significa que no se buscará por él.