Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Ambientes Virtuais de Execução – Semestre de Verão – 2017/2018
Terceiro Trabalho

Objectivos: Prática com Delegates e Genéricos
Data limite de entrega: **10 de Junho**

Este trabalho dá continuidade ao projecto `AutoSql` mantendo-se disponíveis todas as funcionalidades implementadas nos trabalhos anteriores. As novas funcionalidades devem ser construídas sobre o `ReflectDataMapper` mantendo-se as classes **`EmitDataMapper` e `DynamicDataMapper` inalteradas.**

Adicione todos os testes unitários necessários para validar o correcto funcionamento dos requisitos do enunciado.

1ª Parte

De modo a suportar genéricos e iteradores *lazy*, torne `ReflectDataMapper` compatível com a nova interface `IDataMapper<K, V>`:

```
public interface IDataMapper<K, V> : IDataMapper
{
        V GetById(K id);
        new IEnumerable<V> GetAll();
        K Insert(V target);
        void Update(V target);
        void Delete(V target);
}
```

Os parâmetros-tipo `K` e `V` presentes na especificação de `IDataMapper<K,V>` designam, respectivamente, o tipo da propriedade usada como chave, e o tipo do objecto de domínio. Para tal crie uma nova classe `AbstractDataMapper<K,V>` que implementa a interface `IDataMapper<K,V>` e faça `ReflectDataMapper` estender esta nova classe abstracta, passando a ser `ReflectDataMapper<K,V>`.

DICA: nos casos de dependências entre *data mappers* (exemplo de `Product ---> Supplier`) terá que instanciar o *data mapper* referido concretizando os seus parâmetros genéricos (bibliografia *Constructing an Instance of a Generic Type*).

Além do suporte para genéricos a nova classe `AbstractDataMapper<K,V>` deve usar iteradores *lazy*. Ou seja, baseando-se na implementação **não genérica** de `AbstractDataMapper` remova todas as ocorrências de `List` ou `IList`, substituindo pela utilização de `IEnumerable<V>`. **Não** poderá usar nenhuma estrutura de dados em memória.

2ª Parte

Implemente uma forma de registar *listeners* (também designados de *observers* ou *handlers*) dos acessos a dados feitos pelos *data mappers*. O objectivo é poder adicionar uma função a um *data mapper* que é chamada sempre que este faça um acesso a dados (base de dados ou *cache*).

Usando esta funcionalidade elabore um teste unitário onde é verificada a existência de um acesso a dados *eager* ou *lazy* consoante se trate de um *data mapper* que estenda de `AbstractDataMapper` ou `AbstractDataMapper<K,V>`.

3ª Parte

Uma relação de *Foreign Key* pode ser implementada em OO (*Object Oriented*) como uma associação de 1:1 ou 1:N. No primeiro trabalho foi dado suporte apenas a associações 1:1. Ou seja, se na tabela `Products` a coluna `SuppierId` é uma *Foreign Key* para `Suppliers`, então no modelo OO a entidade `Product` tem uma propriedade do tipo `Supplier`. Contudo, esta mesma relação pode ser implementada em sentido contrário com uma associação de 1:N, em que o tipo `Supplier` tem uma propriedade do tipo `IEnumerable<Product>`.

Em suma, sendo `A` e `B` entidades de domínio correspondentes às tabelas `A'` e `B'`; se `A'` tem uma FK para `B'`, então no modelo OO:

- `A` pode ter uma propriedade do tipo `B` – associação de 1:1
- `B` pode ter uma propriedade do tipo `IEnumerable<A>` – associação de 1:N

Note que, por questões de simplicidade, não é necessário a biblioteca `autosql` suportar os dois tipos de associação em simultâneo entre as mesmas entidades.

Adicione suporte para associações de 1:N com carregamento *lazy*. Por exemplo, no modelo relacional `Orders` tem uma FK para `Employees`. Assim, no modelo OO, uma instância de `Employee` pode estar associada a várias ordens, tendo por isso uma propriedade `IEnumerable<Order>`. Neste caso o tipo `Order` não terá nenhuma propriedade do tipo `Employee`.

Altere a sua implementação de `ReflectDataMapper<K,V>` para que passe a suportar propriedades deste tipo. De notar que neste exemplo uma instância de `ReflectDataMapper<int, Employee>` irá depender de uma instância de `ReflectDataMapper<int, Order>` através da qual obterá a sequência de ordens que estão associadas a um determinado empregado.

Para poder executar um comando SQL com cláusula WHERE pode tornar público o método `Get(string sql)` de `AbstractDataMapper`.

Implemente testes unitários que verifiquem o comportamento *lazy* da associação de 1:N entre `Employee` e `Order`.