**INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

Engenharia Informática e de Computadores

**Modelação e Padrões de Desenho**

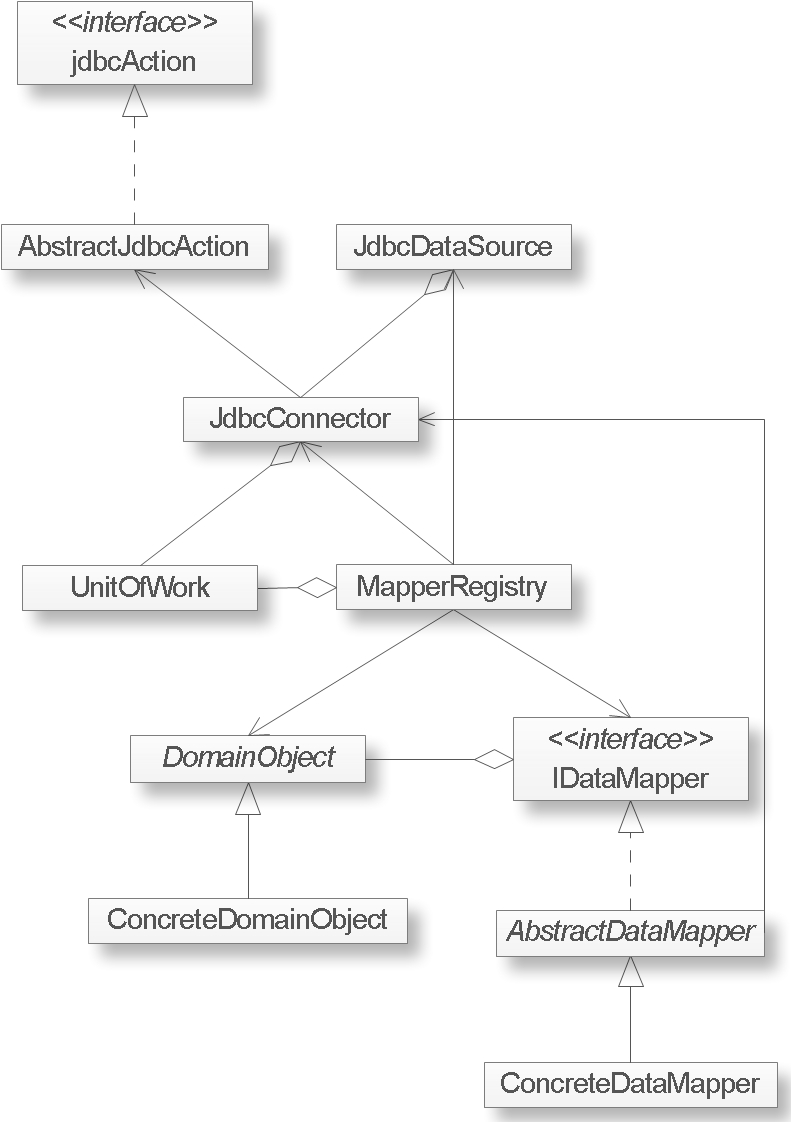
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Membros do Grupo** | | | | | |
| **31768** | **Ricardo Romão** | **31923** | **Nuno Sousa** | **26657** | **Ricardo Neto** |

**Trabalho final**

Para a realização deste trabalho foi aproveitado muito do código já realizado ao longo do semestre, quer na resolução das séries propostas, quer na abordagem ao padrão de desenho *Unit Of Work* feita nas aulas.

Desta forma, foi opção do grupo a manutenção da abordagem estudada no que ao acesso à base de dados diz respeito, nomeadamente mantendo as classes *JdcbConnector* e *AbstractJdbcAction* e a *interface JdbcAction*, bem como as classes *AbstractDataMapper* e *DomainObject*, conforme o esquema 1.

Foi também mantida a utilização quer dos *identity Map’s* quer do carregamento *lazy* dos dados.



Esquema

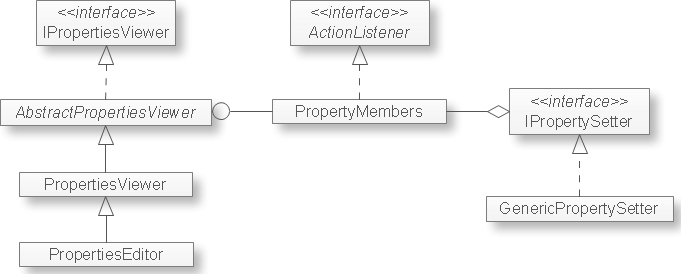
O código desenvolvido nas aulas manifestava algumas limitações ao nível das chaves, nomeadamente quanto ao seu tipo, assumia chave inteira para todas as entidades, e à sua posição no *ResultSet*, a primeira. Por outro lado também não estava contemplada a possibilidade de chaves compostas. Para resolver estes problemas foram alteradas as classes *DomainObject* e *AbstractDataMapper* por forma a suportarem chaves genéricas nas formas *DomainObject<IDType, DType extends DomainObject<IDType, DType>>* e *AbstractDataMapper<IDType, DType extends DomainObject<IDType,DType>>*.

Desta forma tornou-se possível a utilização de chaves de qualquer tipo, sendo necessário no caso de tipos *user type*, que os mesmos redefinam os métodos *equals* e *hashCode* uma vez que os mesmos serão utilizados na comparação de identidade pelos *identity map’s*.

O método *doGetId(ResultSet rs)* em *AbstractDataMapper* foi tornado abstracto delegando nos *DataMappers* concretos a responsabilidade do correcto acesso ao *ResultSet* para obtenção da chave.

Foram implementados os métodos *insert*, *update* e *delete* em *AbstractDataMapper* por forma a cumprir com os requisitos do enunciado.

Para a visualização dos objectos de domínio foi utilizada a implementação resultante da resolução das séries, nomeadamente *PropertiesEditor* genérico conforme o esquema 2.



Esquema

Ao ser criada uma instância de *PropertiesEditor*, aquando da chamada ao construtor de *PropertiesViewer*, são averiguados quais as propriedades do objecto a serem mostradas, bem como qual o seu género (*Simple*, *Complex* e *List*). É também adicionado o botão responsável pela actualização dos dados presentes no *PropertiesEditor* sendo adicionado um *listener* por cada propriedade passível de ser actualizada.

Como são reflectidas as alterações na base de dados? Há testes?