Padrão de projeto de software

Paulo Venancio Lopes e Daniel Sguillaro

Nome: Roupa suja lava-se em casa.

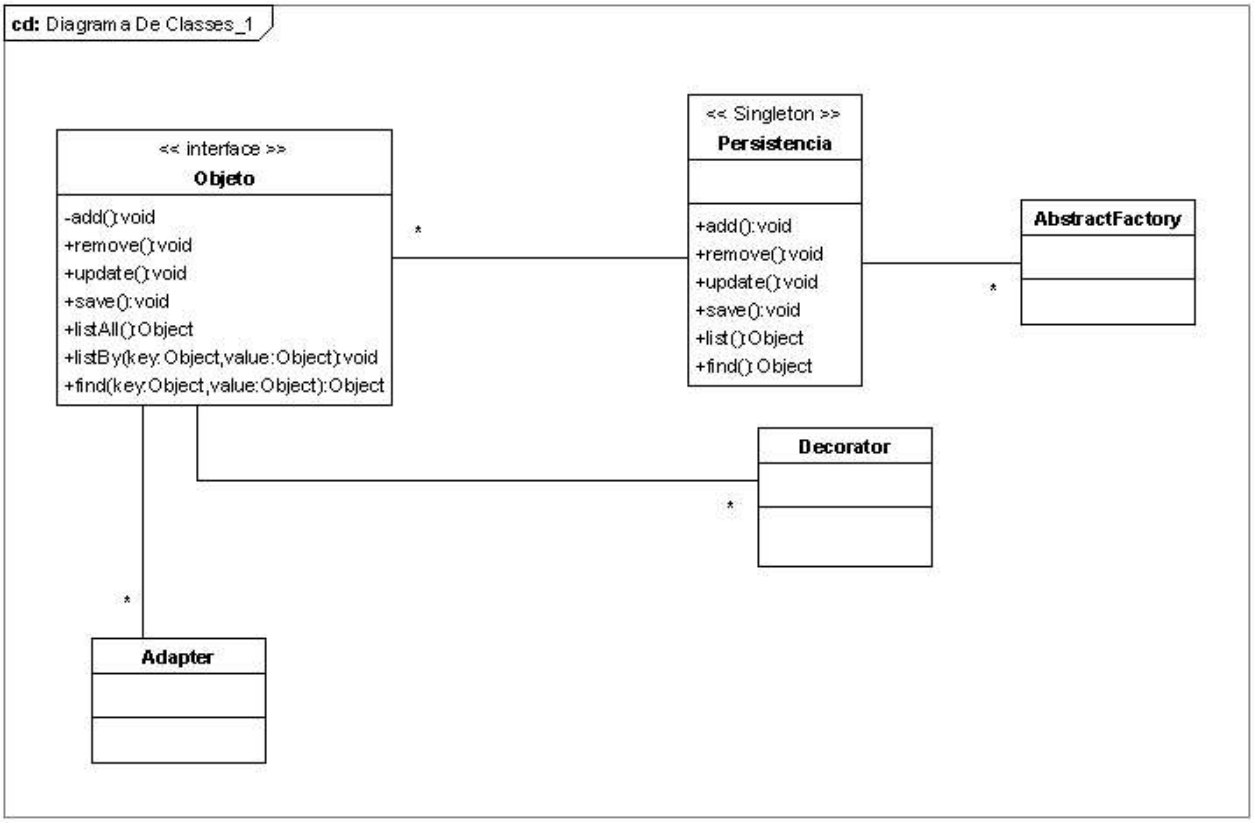
Problema:
No modelo MVC de projeto de software a camada de modelo tem como responsabilidades fazer a conexão com o banco de dados e persistir o objeto, recuperar, listar, apagar e outros. Mas por que essa camada tem que fazer este trabalho, já que ela deve apenas se preocupar com as regras de negócio? Se o próprio objeto se responsabilizasse toda essa parafernália a camada de modelo ficaria apenas preocupado com a lógica de negócios.

Solução:
Para solucionar este problema implementa-se o padrão Objeto Pesistente, que direciona o desenvolvedor a deixar tudo que se referir ao controle do objeto ser feito por ele mesmo, desta forma o objeto é que deve saber se os seus dados serão persistidos e se for ele definirá qual será sua política. O objeto deve saber salvar-se, recuperar-se, apagar-se, listar seus semelhantes e ainda implementar soluções que ajudem a camada de modelo a resolver suas lógicas de negócio.

Objetivo:
Prover um modelo de interfaces para criação de objetos relacionados que darão
suporte a uma entidade fazer todo seu controle e ainda disponibilizar
funcionalidades auxiliares para quem utilizá-la.

Motivação:
Considere uma aplicação que queira salvar seus objetos em um banco de dados,
bem como manipulá-los (apagar, trocar, listar, fazer contas, etc...). Numa aplicação normal teremos sempre o problema de estar fazendo a conexão com o banco de dados e não importa sua forma (JDBC, Hibernate, etc...), sempre que precisarmos manipular este objeto persistido. Desta forma teremos sempre o trabalho extra na camada de modelo, o que é desperdício de tempo, deixa sujo o código e tira a real função de lógica de negócios da camada de modelo. Pense que podemos criar uma interface que faz as manipulações básicas (salvar, apagar, trocar, listar) e criar interfaces auxiliares para manipulações especificas.

Estrutura:



Participantes:

- Abstract Factory.
- Singleton.
- Descorator.
- Adapter.

Colaborções:
O Singleton fará com só tenhamos uma única instancia de persistência, enquanto que um Abstract Factory proverá instâncias para os possíveis objetos de controle de persistência, e um Decorator adicionará novas responsabilidades ao objeto criado. Um Adapter proverá adaptação da interface do objeto para que seja ele visto por clientes diferentes.

Consequências:
O objeto criado saberá se deve ser persistido e qual meio será usada para este processo, se for persistido terá um mínimo de métodos para a sua manutenção no banco de dados. O objeto poderá ter funcionalidades extras para garantir eficiência para a camada de modelo e ainda poderá ser visto por outros clientes.

Implementação:
A Abstract Factory será implementada como um Singleton para gerar apenas uma instância de cada tipo de conexão para persistência. O Singleton será usado também para gerar apenas uma instância do objeto que faz a persistência. O
Decorator adicionará funcionalidades especiais para o objeto e o Adapter proverá mudança na interface do objeto de tal forma que possa ser vista por diferentes clientes.