YAML Editor

Paulo Aguiar[1] & Pedro Pinto[2]

[1] Universidade da Madeira, Funchal Penteada, Portugal
[2] Universidade da Madeira, Funchal Penteada, Portugal

Abstract. Projeto desenvolvido durante o primeiro semestre de dois mil e dezanove para a cadeira de Desenho e Implementação de Software. Consiste na criação de um editor de ficheiros com o formato *.yaml* (destinados ao uso na plataforma *Home Assistance*[1]), de modo a garantir uma edição fácil e correta dos campos, mesmo para utilizadores não familiarizados com esta plataforma.

Palavras-Chaves: YAML, Editor, Padrões.

1 Introdução

No âmbito da cadeira de desenho e implementação de software foi lançado como desafio a criação de uma aplicação, utilizando C#[2] e padrões de desenho[3], em que o utilizador fosse capaz de alterar ou criar um ficheiro de raiz, em formato *.yaml*. A aplicação estaria encarregue de verificar a inserção e configuração do ficheiro, ter uma funcionalidade de *auto-save* e recuperação de ficheiro (caso aconteça algo durante a execução), permitir recuar e avançar "infinitamente" nos passos anteriores/posteriores e, também, criar os componentes através de *wizards*.

2 Padrões Implementados[4]

2.1 Observer

Durante o seu desenvolvimento foi necessário criar uma forma de recuperar os ficheiros de configuração e, de *auto-save*, sendo que, implementamos uma para a outra, isto é, todas as modificações ocorridas pela aplicação (mudanças de estado) são notificadas nesse mesmo instante e são guardadas num ficheiro temporário, esse mesmo ficheiro será usado para recuperação de ficheiros (quando o programa é terminado inesperadamente na próxima execução é perguntado ao utilizador se quer recuperar as últimas mudanças). Para realizar esta operação identificamos esta situação com o padrão *observer*, sendo que, a cada mudança de estado há uma notificação ao ficheiro *recover.yaml* (ficheiro temporário e invisível).

2.2 Singleton

Para a criação do ficheiro temporário foi utilizado o padrão *singleton* para garantir que, apenas era criada uma instância do ficheiro de recuperação e, que, não utilizávamos mais recursos do que os necessários.

2.3 Command

Uma das funcionalidades que pareceu algo complexo de implementar, especialmente sobre uma *TreeView*, foi a permissão de o utilizador ser capaz de fazer *undos* e *redos* múltiplos, no entanto foi identificada uma oportunidade de simplificar a implementação desta funcionalidade utilizando o padrão de desenho *command.* Basicamente, todas as ações sobre a árvore são guardadas numa lista de comandos sendo que a mesma pode ser percorrida (fazer *redo* e *undo*), no entanto, como é obvio, caso alguma mudança seja implementada após um *undo* ou vários, deixa de ser possível fazer *redo*, relativamente às ações anteriores.

2.4 Builder

Outra das funcionalidades pretendidas para o projeto era a criação de ficheiros através de *wizards*, embora não estejam muito apropriados para utilizadores juniores existe a possibilidade de criação de um ficheiro desde a sua raiz. Para conseguir estes *wizards* foi utilizado o padrão conhecido como *Builder*, visto que este facilitava a criação de objetos diferentes utilizando o mesmo construtor.

3 Modelo Implementado

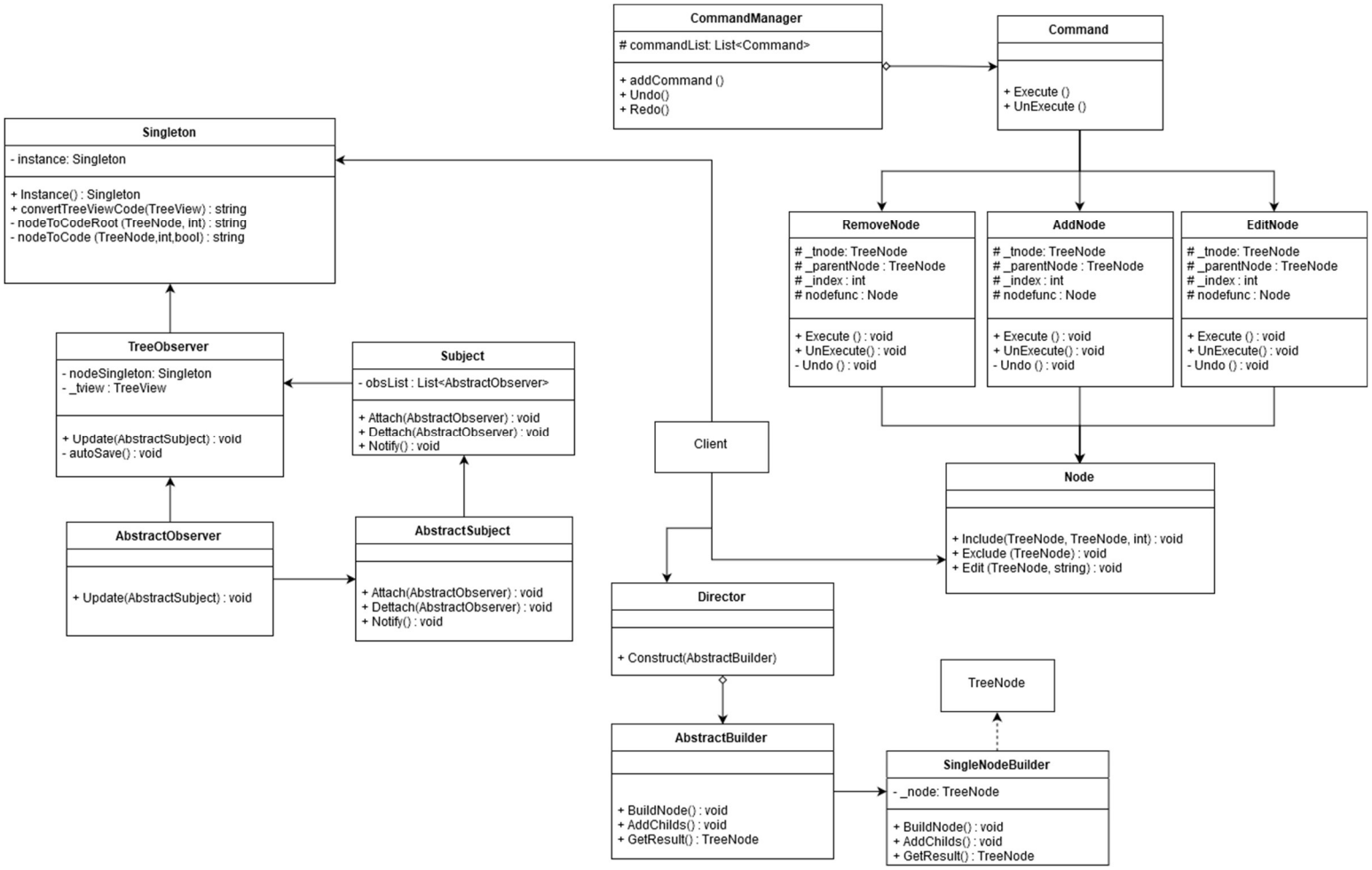

Figura 1 - UML

4 Conclusão

Com a evolução deste projeto, as nossas capacidades de desenvolvimento em C# e utilização de padrões de programação evoluíram, sendo que podemos afirmar que os padrões para além de terem facilitado algumas tarefas complicadas (como por exemplo os *redos* e *undos*) tornaram o código menos extenso. Após um uso e estudo dos padrões neste projeto apercebemo-nos que, por vezes, as melhores soluções para algo complexo são algo simples e, que, podem ser facilmente implementadas com um padrão, padrões estes que também conseguimos identificar mais facilmente após o trabalho desenvolvido.

Em suma, apesar do projeto efetuado ser o mínimo apresentado, foi feito na totalidade e com sucesso.

5 Bibliografia

1. Home Assistance - YAML Example, https://www.home-assistant.io/cookbook/, última vez acedido 2019/01/29.
2. C# Guide Microsoft, https://docs.microsoft.com/en-us/dotnet/csharp/index, última vez acedido 2019/01/29.
3. .NET Design Patterns, https://www.dofactory.com/net/design-patterns, última vez acedido 2019/01/30.
4. Design-patterns-for-humans, https://github.com/kamranahmedse/design-patterns-for-humans, última vez acedido a 2019/01/30.