Relatório:

Introdução:

Para a realização da entrega final deste projecto foi-nos proposta a realização de um comando extra como por exemplo “procurar a maior entrada dentro do sistema de ficheiros” , e cuja a sua implementação fosse extensível a outros comandos semelhantes. Apos uma pequena análise de alguns dos padrões de desenhos estudados nas aulas teóricas concluímos que o “Iterator Design Pattern” era a solução mais adequada, proporcionando uma uma maneira de percorrer todos os elementos de um agregado de objectos sequencialmente sem expor as suas representações internas.

The Iterator Design Pattern:

Objectivos:

- Proporcionar uma maneira de percorrer todos os elementos de um agregado de objectos sem expor as suas representações internas.
- Um agregado de objectos é um objecto que vai agrupar/conter outros objectos. Normalmente referimo-nos a este objecto como um container ou uma collection e estão muitas vezes relacionados com outro padrão de desenho designado “Composite”.

No contexto do projecto:

No projecto esta relação de container é evidente no objecto Directory que vai conter vários objectos como SpecialEntries, Files, e outros Directories.

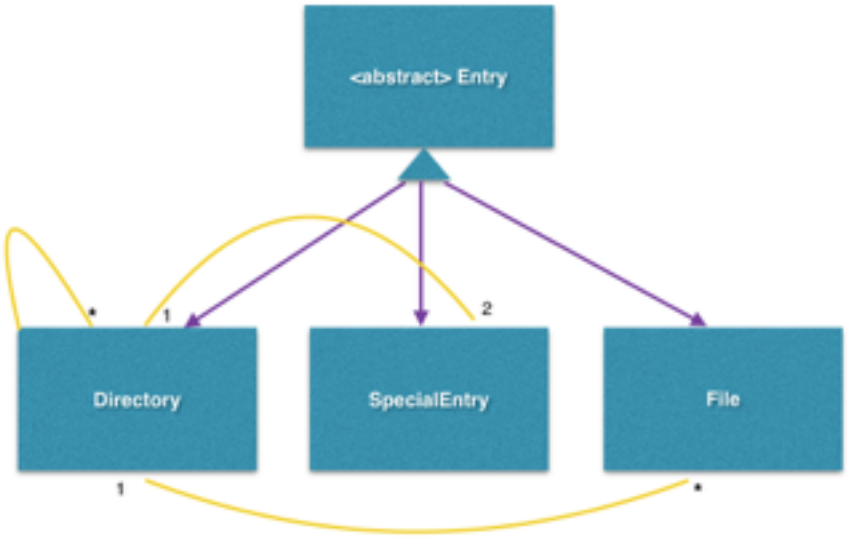

Figura 1: estrutura dos tipos Entry.

Esta implementação/estrutura vai criar uma arvore que pode ser complexa de iterar, e este padrão ira resolver esse problema sem que com isso seja necessário fazer alterações as estruturas da arvore, uma vez que elas vão tomar comportamentos diferentes (iteração polimórfica).

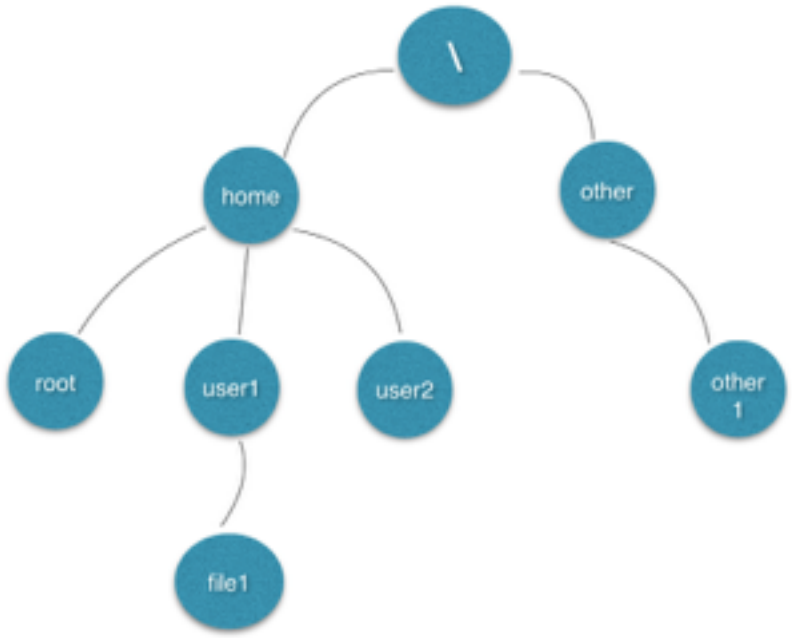

Figura 2: Arvore gerada pelo container.

Motivação:

Assim sendo como motivação a utilização deste design pattern temos:

- A existência de uma lista em que vamos querer aceder aos seus elementos sem expor a sua estrutura interna.
- Necessidade de percorrer a lista de diferentes maneiras dependendo da tarefa que queremos realizar.
- Retirar a responsabilidade de acesso transversal da class container e coloca-la num objecto independente, o iterador.

Estrutura:

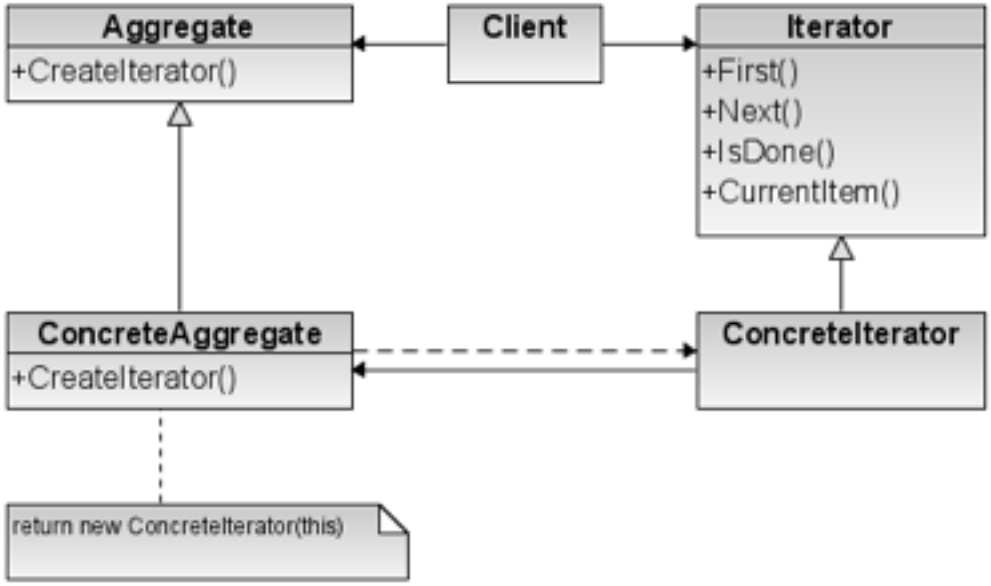

Figura 3: UML do padrão Iterator.

O Iterator:

Define a interface que nos vai permitir percorrer e aceder aos elementos da lista. No projecto foi usada a interface disponibilizada pela biblioteca do java (java.util.*) para tipos genéricos.

Concrete Iterator:

Implementa a interface Iterator e mantém o controlo sobre a posição atual na travessia do agregado. A class CompositeIterator será o nosso concrete iterator.

Aggregate:

Irá ser a class que vai conter a interface para a criação de um iterador. Esta interface será implementada pela class Entry que irá ter um método iterator() que vai ser herdado por todas as subclasses.

Concrete Aggregate:

Class que vai reescrever o método responsável pela criação de um iterador devolvendo um iterador do tipo concreto, ou seja no contexto do projecto, do tipo CompositeIterator. Esta class será a class Directory, uma vez que todas as outras subclasses de Entry herdam o métodos iterator() (sem o reescrever) que devolve um NullIterator.

Nota: A utilização da class NullIterator é apenas auxiliar. Na class abstracta entry poderíamos fazer com que o metodo iterator retorna-se null de forma a que todas as subclasses sobre o qual não nos interessa continuar a iterar não devolvessem nenhum iterador, mas isso implica que quando se esta a realizar a iteração seja necessário verificar se o retorno do método iterator para uma dada entrada é null ou não. Um NullIterator será um iterador que retorna sempre False quando se realiza o método hasNext(), eliminando necessidades de verificação.

Conclusão:

Através deste padrão de desenho foi possível definir o comando que nos foi proposto de forma simples e sobretudo tornar a solução extensível a outros comandos semelhantes, sem que para isso seja necessário fazer alterações nas estruturas internas da classes do core.

Permitiu também um primeiro contacto com os Design Patterns em situação de projecto, mostrando-nos que devem ser usados sempre que possível uma vez que na maioria dos casos se revelam a solução mais simples.

Bibliografia:

sites:

http://www.slideshare.net/ppd1961/design-patterns-3159605
http://www.tutorialspoint.com/design_pattern/iterator_pattern.htm

livros:

Head First Design Patterns , Eric Freeman, Elisabeth Freeman, Kathy Sierra, Bert Bates, 2004 (Outubro), O'Reilly. ISBN 0596007124