Módulo III
Padrões GOF: Iterator

Professores
Eduardo Bezerra – edubezerra@gmail.com
Ismael H F Santos – ismael@tecgraf.puc-rio.br

Ementa

- Padrões GOF
  - Iterator

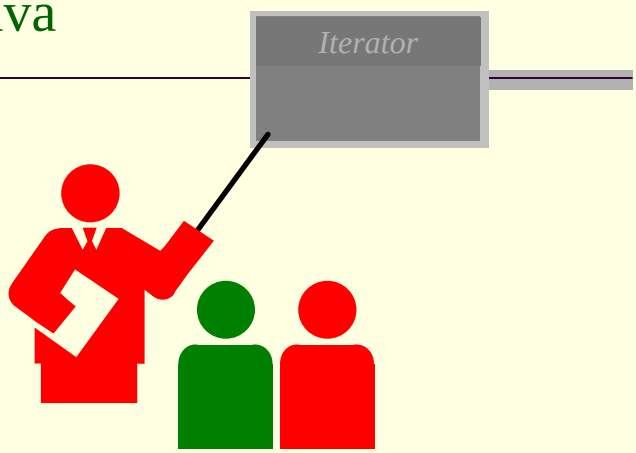
Bibliografia

- *Eric Gamma, et ali, Padrões de Projeto, Ed Bookman*
- *Martin Fowler, Analysis Patterns - Reusable Object Models, Addison-Wesley,1997*
- *Martin Fowler, Refatoração - Aperfeiçoando o projeto de código existente, Ed Bookman*

Livros

- **Core Java 2**, Cay S. Horstmann, Gary Cornell
  - Volume 1 (Fundamentos)
  - Volume 2 (Características Avançadas)
- **Java: Como Programar**, Deitel & Deitel
- **Thinking in Patterns with JAVA**, Bruce Eckel
  - **Gratuito**. http://www.mindview.net/Books/TIJ/

POO-Java



Iterator

- Toda coleção possui uma representação interna para o armazenamento e organização de seus elementos.
  - Por outro lado, essa coleção deve permitir que seus elementos sejam acessados sem que sua estrutura interna seja exposta.
- De uma maneira geral, pode-se desejar que estes elementos sejam percorridos de várias maneira, sem no entanto ter que modificar a interface da coleção em função do tipo de varredura desejado.
  - de frente para trás, vice-versa, ou mesmo em ordem aleatória.

Iterator

- O padrão **Iterator** permite descrever uma forma de percorrer os elementos de uma coleção sem violar o encapsulamento dessa coleção.
- **Intenção**: iterar sobre (percorrer seqüencialmente) uma coleção de objetos sem expor sua representação.
  - Obedecer o princípio do encapsulamento

Iterator

- **Solução:** um objeto intermediário (**iterator**) é usado entre o cliente e a coleção de objetos.
  - Este objeto conhece a estrutura interna da coleção a ser percorrida, e apresenta uma interface para percorrer tal estrutura.
  - Esta interface é independente dessa estrutura interna.
  - Os clientes que desejam percorrer a coleção utilizam a interface do objeto intermediário, em vez de se comunicarem diretamente com a coleção de objetos.

Iterator

- Requisitos de um iterador
  - Um modo de localizar um elemento específico da coleção, tal como o primeiro elemento.
  - Um modo de obter acesso ao elemento atual.
  - Um modo de obter o próximo elemento.
  - Um modo de indicar que não há mais elementos a percorrer.
- Exemplo em Java
  - As classes List, Set e Sorted são subclasses de Collection, e herdam um método iterator() que retorna um objeto iterador.
  - O objeto Iterator possui métodos hasNext() e next().

Iterator (exemplo)

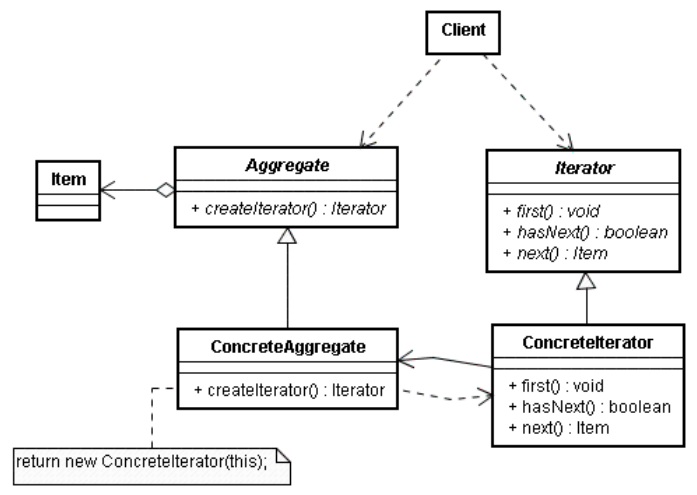
```
// ICollection.java
// interface para obtenção de Iterator para coleções

public interface ICollection
{
  // obtenção de um Iterator
  public IIterator getIterator();
  // determina existência de um elemento
  public boolean has(Object object);
  // adição de um elemento
  public boolean add(Object object);
  // remoção de um elemento
  public boolean remove(Object object);
  // remoção de todos os elementos
  public void removeAll();
}
```

Iterator (exemplo)

```
// IIterator.java
public interface IIterator {
  // verifica a existência de um próximo elemento
  public boolean hasNext();
  // retorna o próximo elemento
  public Object next();
}
```


Iterator (estrutura)



Iterator (participantes)

- *Iterator*
  - Define um interface para o acesso e varredura;
- *ConcreteIterator*
  - Implementa a interface do *Iterator*;
  - Mantém referência (cursor) ao objeto que está sendo percorrido, podendo calcular qual o elemento seguinte.
- *Aggregate*
  - Define um interface para a criação do objeto Iterator;
- *ConcreteAggregate*
  - Implementa o método da interface que retorna uma instância do ConcreteIterator.

Iterator (aplicabilidade)

- O uso do padrão Iterator se aplica quando se quer:
  - acessar o conteúdo de objeto agregados sem expor sua representação interna;
  - dar suporte a mais de uma maneira de percorrer a lista;
  - prover interface única para percorrer estruturas agregadas diferentes.

Iterator (conseqüências)

- Mantém separadas a representação interna e a responsabilidade de *navegação* pelas partes.
  - O iterador conhece a estrutura interna das partes, mas os clientes do iterador não conhecem.
- Move da coleção de objetos para o objeto iterator a responsabilidade de acesso e varredura da coleção.
- A coleção ainda é responsável por criar seus próprios iteradores e o faz através do padrão "*Factory Method*".
- Há a possibilidade de utilizar mais de um iterador simultaneamente.
  - Dá suporte a múltiplas maneiras de percorrer a coleção e, se necessário, essas varreduras podem ocorrer ao mesmo tempo.