IPS Instituto Politécnico de Setúbal

Escola Superior de Tecnologia de Setúbal

Programação Avançada 2021-22

**[4] Introdução aos Padrões de Software & Iterator

Bruno Silva, Patrícia Macedo

Sumário 📝

- Padrões de Desenho versus Padrões de Arquitetura
- Definição de Padrão de Desenho
- O Padrão Iterator

Introdução

- Em Engenharia de Software, um padrão é uma solução geral para um problema que ocorre com frequência dentro de um determinado contexto do desenho de software.
- Existem varias classificações para os padrões, na uc de Programação Avançada vamos usar a seguinte classificação:
  - Padrões de Arquitetura (Architectural Patterns)
  - Padrões de Desenho (Design Patterns)

Introdução

Ver:

https://www.youtube.com/watch?v=I8CRI7fSw4g

Padrões de Software

- Padrões de Arquitetura (Architectural Patterns)
  - Resolvem um problema típico da arquitetura de software. Definem formas de organizar os vários componentes de um sistema de software MVC,MVP, DAO
- Padrões de Desenho (Design Patterns)
  - Resolvem problemas específicos e localizados (como criar uma variável global, como adaptar o comportamento de uma classe, como criar uma família de classes etc...)

#Vantagens da Utilização de Padrões de Software

- Aprende-se com a experiência de outros;
- Utiliza-se soluções amplamente testadas;
- Permite utilizar uma linguagem comum entre os designers e programadores. Melhora-se assim a comunicação entre a equipa e a documentação dos sistemas;
- Leva ao uso de boas práticas no desenvolvimento de software orientado a objetos, obtendo-se assim software de melhor qualidade.

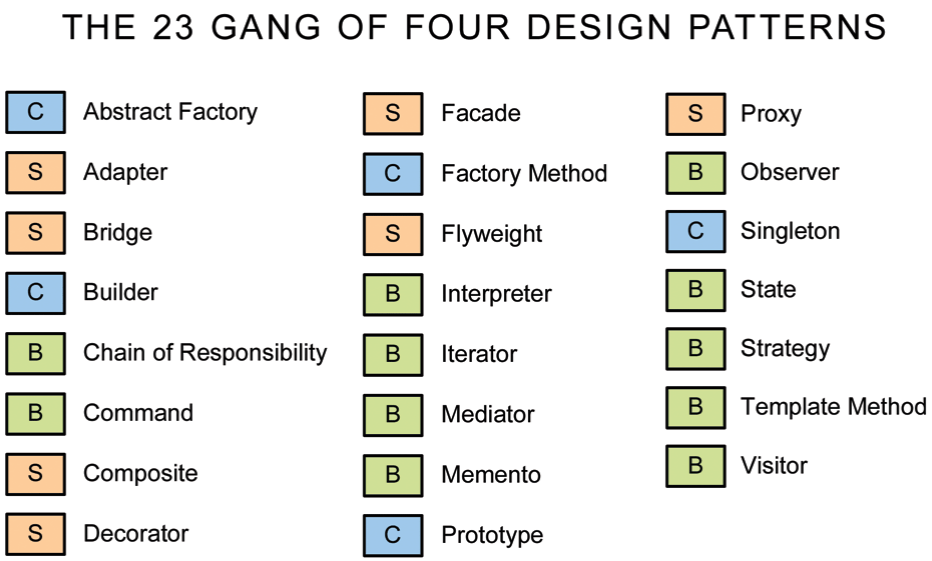
Especificação de um padrão de software

- Um padrão de desenho deverá ser especificado indicando:

```
- O nome (cria um vocabulário de padrões)
- O problema (diz quando aplicar o padrão)
- A solução (descreve os elementos do design)
- As consequências (de aplicar o padrão de desenho)
```

Padrões GoF

Design Patterns: Elements of Reusable Object-Oriented Software é um livro de engenharia de software escrito por Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides que descreve 23 padrões clássicos de desenho. Os 4 autores do livro ficaram conhecidos por **GoF** (Gang of Four).



Pattern Iterator: MOTIVAÇÃO

- **Problema Genérico**
  Uma coleção é um contentor de elementos, mas muitas vezes precisamos de percorre-los sequencialmente. Como o Fazer?
- Quando estamos perante uma coleção do tipo List, normalmente percorremos os elementos em função do seu índice.

```
for (int i = 0; i < list.size(); i++)
        System.out.print(list.get(i) + " ");
```

Pattern Iterator: MOTIVAÇÃO

Mas se quisermos percorrer uma coleção do tipo Set, Map, Tree...
então normalmente usamos um ciclo foreach

```
for (Integer i: set)
            System.out.print(i + " ");
```

Os ciclos foreach só são possíveis de realizar se a coleção for **Iterable**

- Uma coleção que implementa a interface **Iterable**, é uma coleção que implementa o **padrão Iterator**

Pattern Iterator: JAVA

A interface `Iterable` e a interface `Iterator` são interfaces disponibilizadas pelo Java, pois as coleções do Java implementam esse padrão.

```
public interface Iterable<E>{
 Iterator<E> iterator();
}
```

```
public interface Iterator<E>{
   boolean hasNext(); //devolve true se existe proximo elemento
   E next();//devolve o proximo element da sequência
}
```

Pattern Iterator (EXEMPLO DE APLICAÇÃO)

Problema Concreto: Problema Concreto
Temos o **nosso** TAD Stack (que implementamos nas 1ª aulas), que é uma coleção tipo STACK e queremos percorrer uma instancia de STACK sequencialmente do topo para a base, sem destruir o stack.

```
Stack<Integer> stack = new StackArray();
for (int i = 0; i < 20; i++)
   stack.push(i);
for (Integer i: stack)
   System.out.print(i + " ");
```

Solução: Para percorrermos o stack, usando um ciclo foreach, temos que implementar o **padrão Iterator**.

Pattern Iterator (EXEMPLO DE APLICAÇÃO)

Objectivo: Tornar as classes do tipo Stack Iteráveis

- **Como?**

1. Estender a interface `Stack` da interface `Iterable`
2. Implementar o método `iterator` na classe de implementação do `Stack`
3. Criar uma **inner classe** na implementação do Stack que implemente a interface `Iterator`

Pattern Iterator (EXEMPLO DE APLICAÇÃO)

1. Tornar Stack Iterable

```
public interface Stack<E> extends Iterable<E>
```

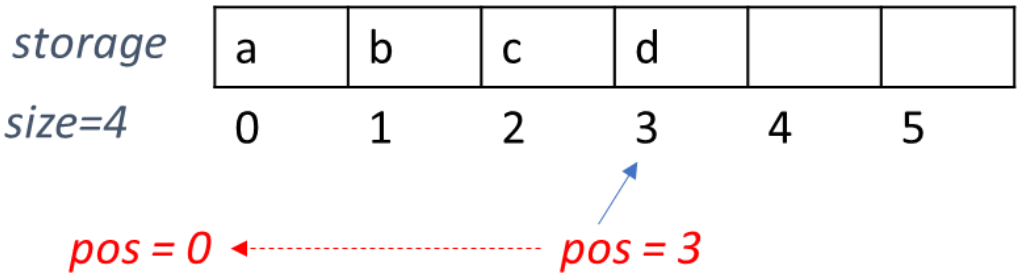
2. Implementar método iterator

```
public class StackArray<E> implements Stack<E> {

    private E[] storage;
    private int size;
    private final static int MAX = 500;
//other code from Stack Array
@Override
    public Iterator<E> iterator() {
        return new IteratorStack();
    }

  //continue
```

Pattern Iterator (EXEMPLO DE APLICAÇÃO)

3. Implementar a inner class `IteratorStack`
    - Implementar o método `hasNext()`
    - Implementar o método `next()`

Nota: A implementação depende da estrutura de dados da implementação do stack -> neste caso StackArray



Pattern Iterator (EXEMPLO DE APLICAÇÃO)

3. Implementar a inner class `IteratorStack`

```
  private class IteratorStack implements Iterator<E> {
        private int pos;

        public IteratorStack() {
            pos=size-1;
        }
        @Override
        public boolean hasNext() {
            return pos>=0;
        }
        @Override
        public E next() {
            E elem= storage[pos];
            pos--;
            return elem;
        }
    }
}
```

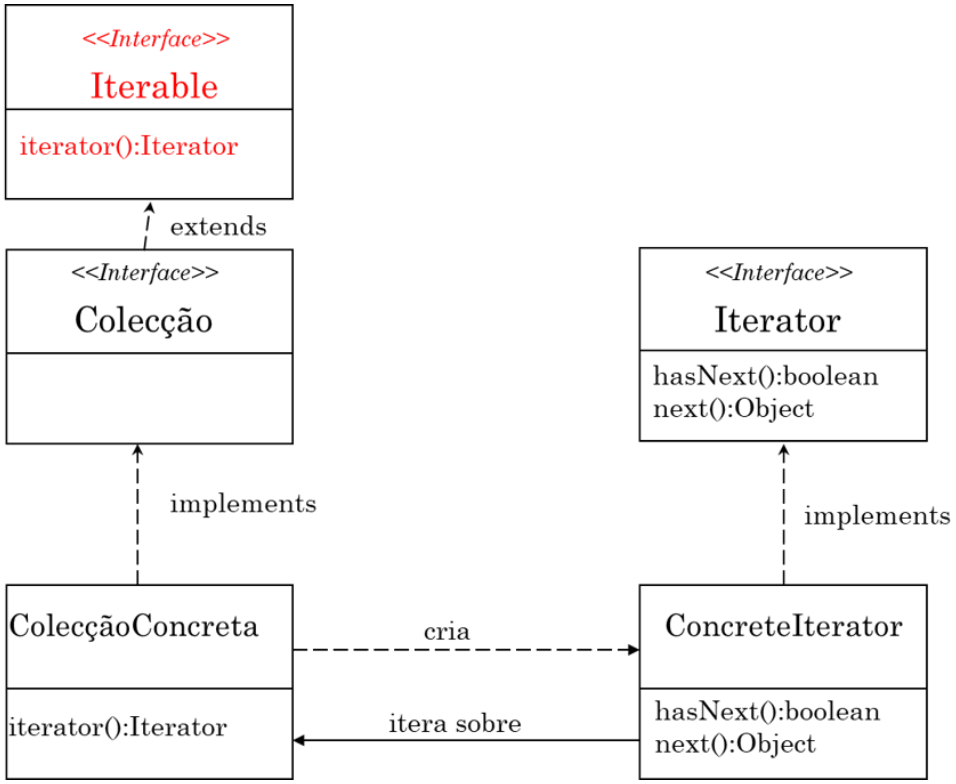
Exercício (1)

Faça Clone do projeto
https://github.com/estsetubal-pa-geral/Iterator_Template.git

- Reveja a implementação do Padrão Iterator para a classe StackArray

Padrão Iterator: Diagrama de Classes

- Um padrão é uma solução padronizada para um problema comum
- O padrão Iterator cria uma forma padronizada de iterar sequencialmente os elementos de um contentor



Padrão Iterator: Participantes

- **Iterator**
  Define uma interface para aceder e percorrer os elementos
- **ConcreteIterator**
  Implementa a interface Iterator
  Mantém a informação da posição actual de iteração
- **Coleção**
  Define uma interface para criar um objeto Iterator
- **Coleção Concreta**
  Implementa a interface que cria o Iterator para retornar o IteradorConcreto apropriado

Padrão Interator : Receita

1. Torne a interface do tipo que pretende tornar iterável a estender de Iterable.
2. Na colecção que pretenda implementar o iterador, implemente o método Iteratoriterator(), que devolve uma instancia do iterador da classe ConcreteIterator.
3. Crie uma classe interna privada ConcreteIterator que implementa a interface Iterator.
4. Implemente os métodos next() e hasNext(), na classe ConcreteIterator.

Exercício (2)

- Ainda usando o projeto do exercício 1, aplique a "Receita" acima de forma a:
  - Tornar a classe StackLinked Iteravel.
  - Tornae a classe Tree Iterável, implementando o iterador de forma a disponibilizar o elementos da árvore na sequência pre-order.
- Implemente um novo iterador para a classe StackLinked agora para iterar da base para o topo

Bibliografia e Leituras extras

Links:

https://refactoring.guru/design-patterns/iterator

https://www.tutorialspoint.com/design_pattern/iterator_pattern

Videos:
https://www.youtube.com/watch?v=wqD4fOiGep4