Padrões de projeto na prática, com códigos Java

Kalil de Oliveira
Faculdade Senac Criciúma

Parte 1

Construção das classes do nosso exemplo

❖ Classe Livro

- Funciona como uma estrutura de dados, para armazenar nome do autor e título do livro.
- Método construtor com acesso público e parâmetros do tipo String: autor e título
- Métodos getAutor e getTitulo.

❖ Classe AcervoLivros

- Deve gerenciar os objetos do tipo Livro
- Importamos a biblioteca java.util.ArrayList
- Único atributo será um ArrayList<Livo> listaLivros;
- Construtor recebe um ArrayList<Livro> cuja ideia é que seja instanciado no método principal e passado por parâmetro;
- Possui os métodos adicionarLivro(Livro livro) e exibirAcervo() com um “for”

❖ Classe Main

- public class Main { public static void main (String[] args) {

ArrayList<Livro> livros = new ArrayList<>();

AcervoLivros acervo = new AcervoLivros(livros);

Livro livro_kalil = new Livro("Kalil", "Aula de Java");

acervo.adicionarLivro(livro_kalil);

acervo.exibirAcervo(); } }.

Parte 2

Padrão Singleton

❖ Definição

- O padrão Singleton garante que uma classe tenha apenas uma instância e fornece um ponto global para acessar essa instância.

❖ Classe AcervoLivros

- Adicionamos um novo atributo privado e estático chamado AcervoLivros instancia;
- O construtor também será privado;
- Como só é acessível dentro da classe, usamos um método público para "pegar" o atributo.

❖ Classe AcervoLivros

- public class AcervoLivros {

private static AcervoLivros instancia;

private ArrayList<Livro> listaLivros;

private AcervoLivros() {

this.listaLivros = new ArrayList<>();

}.

❖ Método público getInstancia

```
public static AcervoLivros getInstancia() {

    if (instancia == null) {

        instancia = new AcervoLivros();

    }

    return instancia;

}
```

❖ O que mudou?

- Tal qual no código antigo, a classe AcervoLivros é a classe Singleton responsável por gerenciar o acervo de livros.
  - Entretanto, criamos um método getInstancia( ) no qual garante que apenas uma instância seja criada

❖ Main

- public class Main { public static void main(String[] args) {

AcervoLivros acervo = AcervoLivros.getInstancia();

Livro livro = new Livro(“Prof Kalil", “Meu livro");

acervo.adicionarLivro(livro);

acervo.exibirAcervo(); } }

Parte 3

Padrão Adapter

❖ Definição

- O padrão Adapter é usado para permitir a interface de uma classe ser usada como interface de outra.
  - Duas interfaces diferentes trabalharão juntas (adaptação)

❖ Interface LivroInterface

- public interface LivroInterface {

  String getTitulo();

  String getAutor();

  }.

❖ Classe LivroAdapter

- class LivroAdapter implements LivroInterface {

private Livro livro;

public LivroAdapter(Livro livro) { this.livro = livro; }

@Override

public String getAutor() {

return livro.getAutor(); } }

❖ Classe AcervoLivros

- A classe AcervoLivros será praticamente a mesma.

```
public class AcervoLivros {  private ArrayList<LivroInterface>

listaLivros;

public AcervoLivros(ArrayList<LivroInterface> livros) {

this.listaLivros = livros;

}
```

❖ Método AdicionarLivro

```
public void adicionarLivro(LivroInterface livro) {

        listaLivros.add(livro);

System.out.println("Livro adicionado ao acervo: " +

        livro.getTitulo());

        }
```

❖ Método ExibirAcervo

- public void exibirAcervo() {

System.out.println("Acervo de Livros:");

for (LivroInterface livro : listaLivros) {

System.out.println("- " + livro.getTitulo());

} } }

❖ O que mudou?

- Agora temos uma interface, ou seja, uma classe que não tem métodos e precisa ser implementada. A classe Livro é adaptada pela classe LivroAdapter.
- No Main, criamos um livro e depois adaptamos. Por fim, adicionamos ao acervo e exibimos normalmente.

❖ Main

- public class Main { public static void main(String[] args) {

Livro livro = new Livro(“Prof Kalil", “Meu livro");

LivroInterface livroAdapter = new LivroAdapter(livro);

ArrayList<LivroInterface> acervo = new ArrayList<>();

acervo.adicionarLivro(livroAdapter);

acervo.exibirAcervo(); } }