OBJETOS DISTRIBUÍDOS

REVISÃO

- EXERCÍCIOS AULA PASSADA.
- RETOMAR EXEMPLO DA ÚLTIMA AULA
- FALAR SOBRE COLEÇÕES

CAMADAS - DEFINIÇÃO

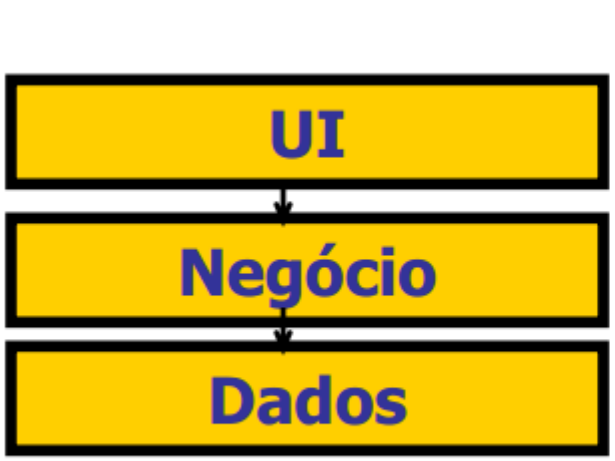
Estimula a organização da arquitetura do sistema em um conjunto de camadas coesas com fraco acoplamento entre elas.

- **Cada camada possui um propósito bem definido.**

- **A camada superior conhece apenas a camada imediatamente inferior (que fornece seus serviços através de uma interface).**

CAMADAS - DEFINIÇÃO

Cada camada é formada por um conjunto de classes com um determinado propósito



DISTRIBUIÇÃO

- **UI**: agrega as classes do sistema com as quais os usuários interagem.
- **Negócio**: mantém as classes do sistema responsáveis pelos serviços e regras do negócio.
- **Dados**: camada responsável pelo armazenamento e recuperação dos dados persistentes do sistema.
- **Comunicação**: responsável pela distribuição do sistema em várias máquinas.

VANTAGENS

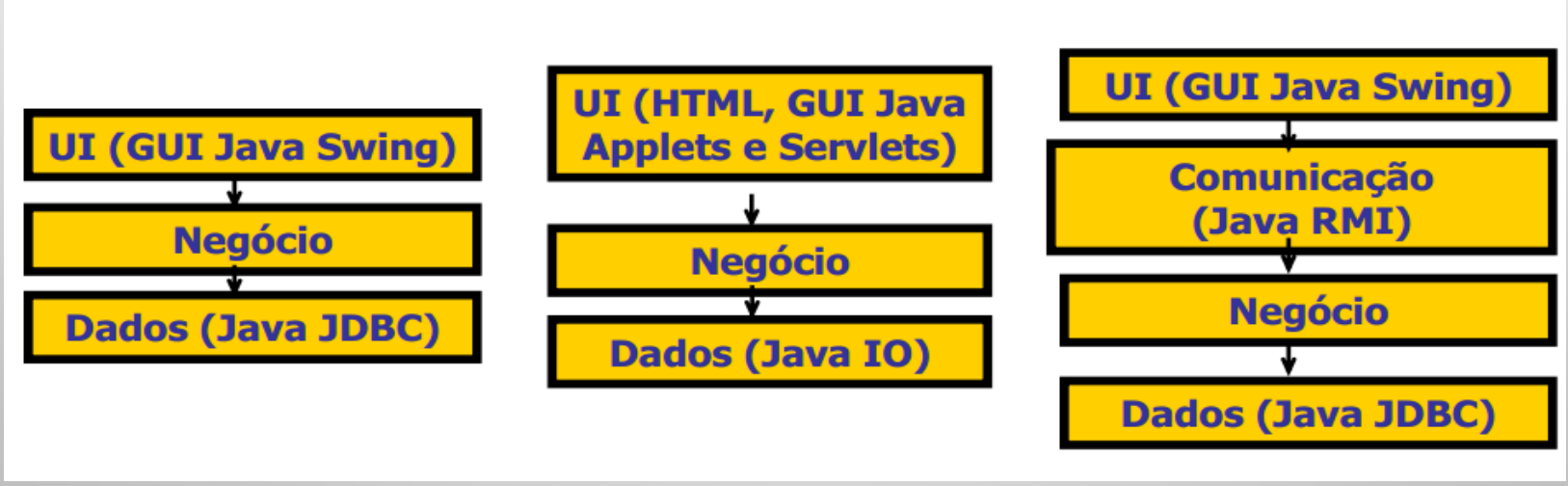
- Separação de código relativo a interface com o usuário (UI), comunicação, negócio e dados.
- Permite a mudança de implementação de uma camada sem afetar a outra, desde que a interface
entre as mesmas seja mantida.
- Possibilita que uma camada trabalhe com diferentes versões de outra camada.

DESVANTAGEM

- AUMENTO NO NÚMERO DE CLASSES EXISTENTES NO SISTEMA

PADRÃO

- EXEMPLOS DE DIFERENTES CONFIGURAÇÕES DO PADRÃO ARQUITETURA EM CAMADAS USANDO TECNOLOGIAS JAVA.



PADRÃO ARQUITETURA

Arquitetura em 3 camadas

- **Possui as camadas**: UI, Regras de Negócio e Acesso a Dados
- **A camada de UI**: agrega as classes de fronteira

Exemplo: GUIAluno

- **A camada de Regras de Negócio**: agrega as classes de controle e entidade

Exemplos: ControladorAluno e Aluno

- **A camada de Acesso a Dados**: agrega as classes de persistência dos dados

Exemplo: RepositorioAluno

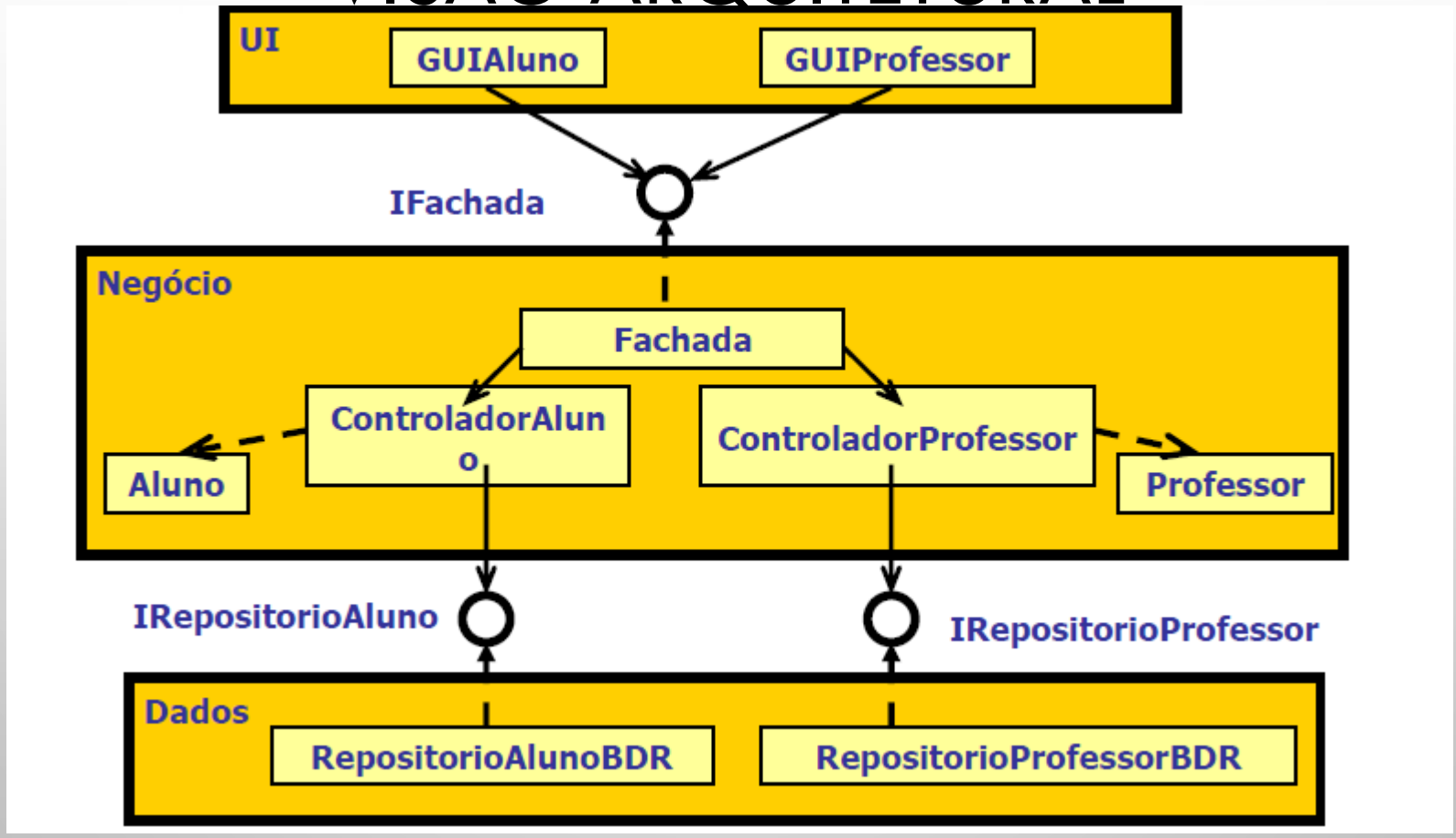
PADRÃO ARQUITETURA

- Arquitetura em 3 camadas
  - Entre as camadas UI e Negócio haverá sempre uma **interface Java** que uma classe Fachada do sistema implementará.
  - A classe **Fachada** é utilizada para oferecer um caminho único para acesso aos serviços da camada de regras de negócio.
  - **As classes da UI, portanto, comunicam-se apenas com a classe Fachada**, que por sua vez colabora com as outras classes internas da camada de regras de negócio para oferecer os serviços.

PADRÃO ARQUITETURA

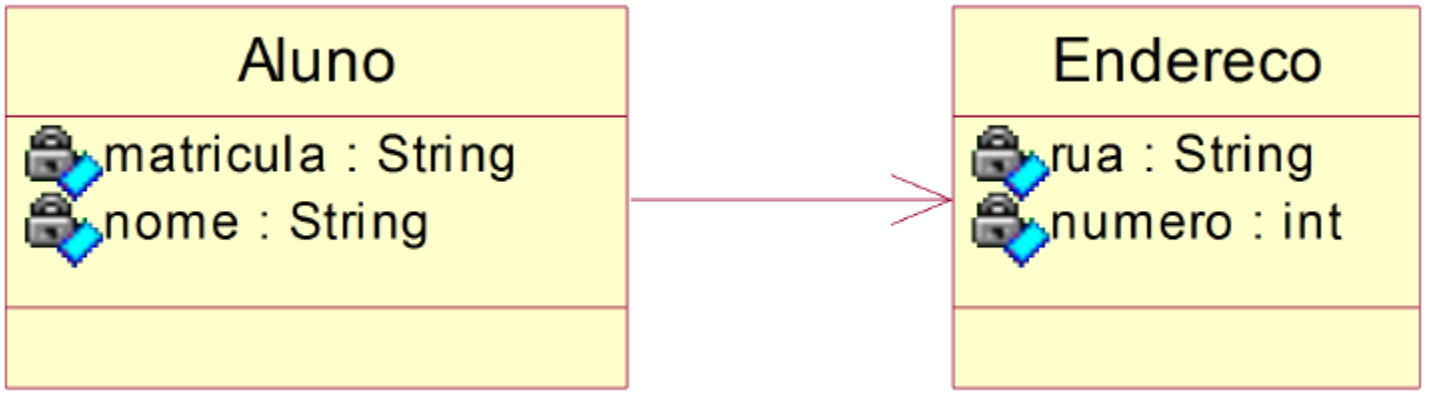
- Arquitetura em 3 camadas
  - O **Controlador** pode conter regras de controle do sistema e delega ações da fachada para a camada de acesso a dados.
  - Entre as camadas Negócio e Dados haverá sempre uma **interface Java** que uma classe Repositório implementará.
  - O **Repositório** armazena os objetos persistentes do sistema em algum meio de armazenamento físico (banco de dados, arquivo, etc.).

VISÃO ARQUITETURAL



IMPLEMENTAÇÃO ARQUITETURA 3

- Exemplo: Implementação de um Sistema para cadastro de alunos
  - Classes Básicas de Negócio (Entidades): Aluno, Endereco



IMPLEMENTAÇÃO ARQUITETURA 3 CAMADAS

- Divisão em Pacotes
  - Um **pacote** é um conjunto de classes.
  - Agrupar em um pacote classes fortemente relacionadas.
  - O pacote deve ser visto como um elemento altamente coeso:
    - Classes de um mesmo pacote devem ter acoplamento e colaboração relativamente altos.
    - Por outro lado, o acoplamento e colaboração entre classes agrupadas em diferentes pacotes devem ser relativamente baixos.

IMPLEMENTAÇÃO ARQUITETURA 3 CAMADAS

- Interface da Fachada
  - Irá oferecer todos os serviços do sistema.
    - Exemplo: Serviços básicos de manutenção de um cadastro tais como inserir, alterar, remover e buscar.

IMPLEMENTAÇÃO ARQUITETURA 3 CAMADAS

```
public interface IFachada {

public void inserirAluno(Aluno aluno) throws
                    ExcecaoElementoJaExistente, ExcecaoRepositorio;

public void alterarAluno(Aluno aluno) throws
                    ExcecaoElementoInexistente, ExcecaoRepositorio;

public void removerAluno(String mat) throws
                    ExcecaoElementoInexistente, ExcecaoRepositorio;

public Aluno buscarAluno(String mat) throws
                    ExcecaoElementoInexistente, ExcecaoRepositorio;

}
```

IMPLEMENTAÇÃO ARQUITETURA 3 CAMADAS

- Fachada
  - Obrigatoriamente, deve implementar todos os métodos da interface da fachada.
  - Tem um ou mais atributos do tipo **Controlador**.
  - Delega para os controladores as chamadas de métodos.

IMPLEMENTAÇÃO ARQUITETURA 3

- Fachada

```
public class Fachada implements IFachada {
  private ControladorAluno controladorAluno;
  public Fachada( ) {
    this.controladorAluno = new ControladorAluno( );
  }
  public void inserirAluno(Aluno aluno) throws
                ExcecaoElementoJaExistente, ExcecaoRepositorio {
     this.controladorAluno.inserirAluno(aluno);
  }
  /* Continua aqui a implementação dos demais métodos! */
}
```

IMPLEMENTAÇÃO ARQUITETURA 3 CAMADAS

- Controlador
  - Um controlador para cada entidade persistente.
  - Contém os métodos que serão chamados pela fachada.
    - Exemplo: Serviços básicos de manutenção de um cadastro tais como inserir, alterar, remover e buscar.
  - Tem um atributo do tipo **IRepositorio** para acessar os métodos da camada de acesso a dados.

IMPLEMENTAÇÃO ARQUITETURA 3

- Controlador

```
public class ControladorAluno {
  private IRepositorioAluno repAlunos;
  public ControladorAluno( ) {
      this.repAlunos = new RepositorioAlunos( );
  }
  public void inserirAluno(Aluno aluno) throws
                ExcecaoElementoJaExistente, ExcecaoRepositorio {
      this.repAlunos.inserirAluno(aluno);
  }
  /* Continua aqui a implementação dos demais métodos! */
}
```

IMPLEMENTAÇÃO ARQUITETURA 3 CAMADAS

- Interface do Repositório
  - Irá oferecer os serviços de persistência de dados
  - Uma interface do repositório para cada entidade persistente.
    - Exemplo: Serviços básicos de persistência tais como inserir, alterar, remover, buscar e verificar se um determinado objeto existe.

IMPLEMENTAÇÃO ARQUITETURA 3

- Interface do Repositório

```
public interface IRepositorioAluno {

public void inserirAluno(Aluno aluno) throws
                    ExcecaoElementoJaExistente, ExcecaoRepositorio;

public void alterarAluno(Aluno aluno) throws
                    ExcecaoElementoInexistente, ExcecaoRepositorio;

public void removerAluno(String mat) throws
                    ExcecaoElementoInexistente, ExcecaoRepositorio;

public Aluno buscarAluno(String mat) throws
                    ExcecaoElementoInexistente, ExcecaoRepositorio;

public boolean verificarExistenciaAluno(String matricula);

}
```

IMPLEMENTAÇÃO ARQUITETURA 3 CAMADAS

- Repositório
  - Implementa a persistência dos dados.
  - Obrigatoriamente, deve implementar todos os métodos da interface do repositório.
  - Contém os métodos que serão chamados pelo controlador.
    - Exemplo: Serviços básicos de persistência tais como inserir, alterar, remover, buscar e verificar se um determinado objeto existe.

IMPLEMENTAÇÃO ARQUITETURA 3 CAMADAS

- Repositório (versão em array)

```
public class RepositorioAlunoArray implements IRepositorioAlunos {
  private Aluno[ ] alunos;
  private int quantAlunos;
  public RepositorioAlunoArray( ) {
      this.alunos = new Aluno[100];
      this.quantAlunos = 0;
  }
  public void inserirAluno(Aluno aluno) throws ExcecaoElementoJaExistente,
                                                          ExcecaoRepositorio {
      if (this.verificarExistenciaAluno(aluno.getMatricula( )) == false) {
        this.alunos[quantAlunos++] = aluno;
      } else {
        throw new ExcecaoElementoJaExistente("Aluno Já Cadastrado!");
      }
  }
  /* Continua aqui a implementação dos demais métodos! */
}
```

IMPLEMENTAÇÃO ARQUITETURA 3

- Aplicação (Classe de interação com o usuário representando a camada de UI da arquitetura)

```
public class Aplicacao {
  private static IFachada fachada = new Fachada( );
  public static void main(String[ ] args) {
    try {
      Endereco end = new Endereco(“Masc. Moraes", 111);
      Aluno al = new Aluno("123", “João", end);
      fachada.inserirAluno(al);
    } catch (ExcecaoDadoInvalido e) {
      System.out.println(e.getMessage( ));
    } catch (ExcecaoElementoJaExistente e) {
      System.out.println(e.getMessage( ));
    } catch (ExcecaoRepositorio e) {
      System.out.println(e.getMessage( ));
    }
  }
}
```

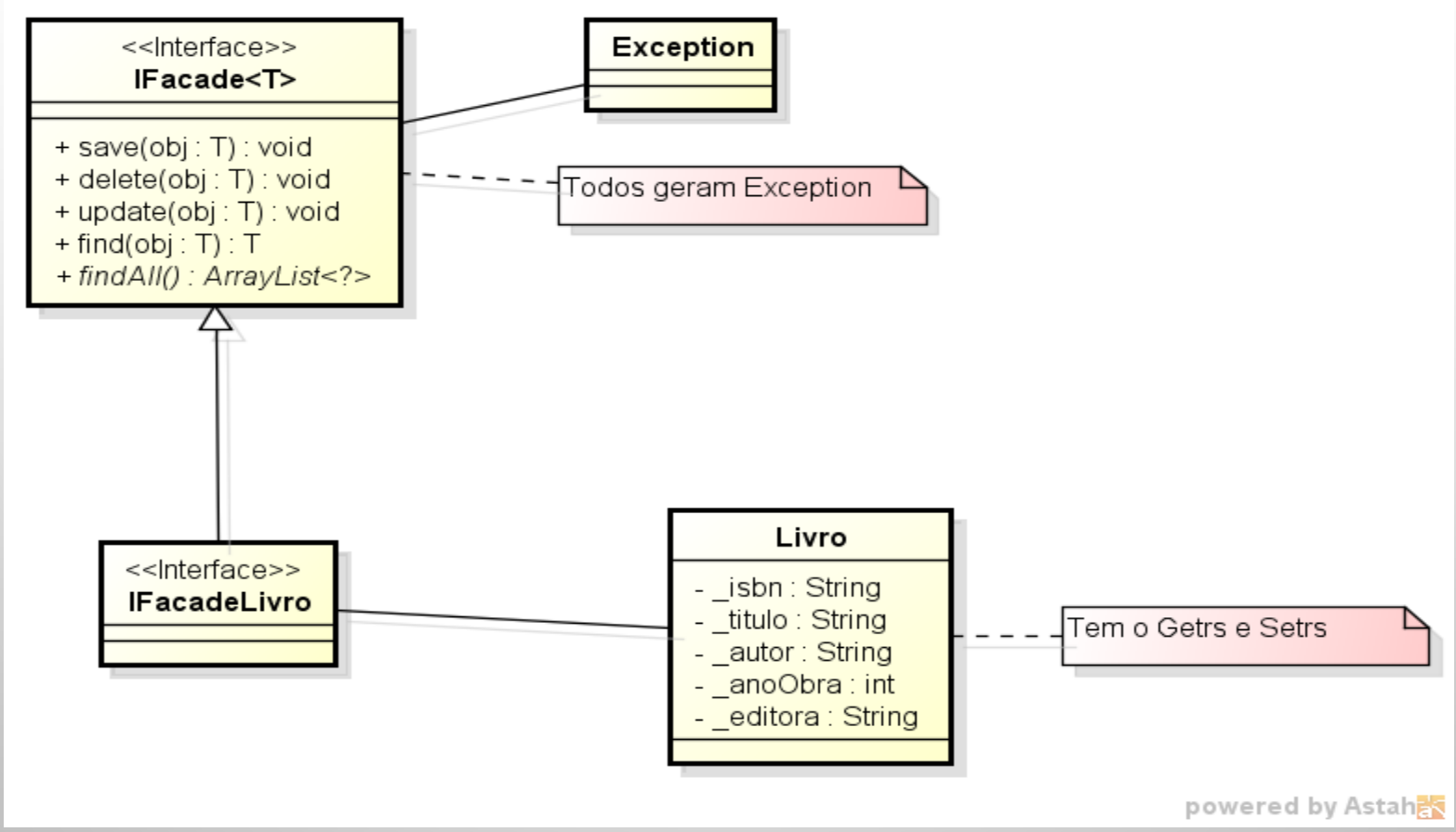
TRABALHO1

- IMPLEMENTAR A ARQUITETURA EM 3 CAMADAS NO CADASTRO DE LIVROS (SAVE, FIND, DELETE, UPDATE, FINDALL E IMPRIMIR RELATÓRIO COM TODOS OS LIVROS CADASTRADAS E DISPONÍVEIS).

IMPORTANTE: SEGUIR EXEMPLO(MODELO) A PARTIR DO SLIDE 11

- OBS 1: UTILIZE ARRAYLIST NO REPOSITÓRIO.
- OBS 2: IMPRIMIR DADOS NA TELA É FUNÇÃO DA CAMADA DE INTERFACE DE USUÁRIO.
- OBS: BOLAR UMA FORMA DE APRESENTAR SOMENTE OS LIVROS DISPONÍVEIS

TRABALHO1


<<Interface>>
IFacade<T>
+ save(obj : T) : void
+ delete(obj : T) : void
+ update(obj : T) : void
+ find(obj : T) : T
+ findAll() : ArrayList<?>
Exception
Todos geram Exception
<<Interface>>
IFacadeLivro
Livro
- _isbn : String
- _titulo : String
- _autor : String
- _anoObra : int
- _editora : String
Tem o Getrs e Setrs
powered by Astah

BIBLIOGRAFIA

- MATERIAL: PADRÃO ARQUITETURA EM CAMADAS. AUTORES: PROF. MÁRCIO BUENO, PROF. KARINA OLIVEIRA

  (HTTP://MARCIOBUENO.COM/ARQUIVOS/ENSINO/POO/POO_15_ARQCAMADAS.PDF)

  ARTIGO: INTEGRANDO JAVA COM BANCOS DE DADOS RELACIONAIS. AUTORES: EURICÉLIA VIANA E PAULO BORBA (HTTP://WWW.CIN.UFPE.BR/~PHMB/PUBLICATIONS.HTM)

- ARTIGO: PDC: THE PERSISTENT DATA COLLECTIONS PATTERN. AUTORES: TIAGO MASSONI, VANDER ALVES, SÉRGIO SOARES E PAULO BORBA (HTTP://WWW.CIN.UFPE.BR/~PHMB/PUBLICATIONS.HTM)

- LIVRO: DESIGN PATTERNS: ELEMENTS OF REUSABLE OBJECT-ORIENTED SOFTWARE. AUTORES: ERICH GAMMA ET AL. EDITORA ADDISON-WESLEY