Visão e análise de projetos

UC: Modelos, Métodos e Técnicas em Engenharia de Software

Prof. Eliane Faveron Maciel

UNIFACS
ecossistema ânima

2 de abril de 2024

Overview

1. Visão e análise de projetos

1.1 Coesão e Acoplamento

2. Referências

Visão e análise de projetos

Coesão

Toda classe deve implementar uma única funcionalidade ou serviço. Especificamente, todos os métodos e atributos de uma classe devem estar voltados para a implementação do mesmo serviço [Valente, 2020]

1. Facilita a implementação de uma classe, bem como o seu entendimento e manutenção.
2. Facilita a alocação de um único responsável por manter uma classe.
3. Facilita o reúso e teste de uma classe, pois é mais simples reusar e testar uma classe coesa do que uma classe com várias responsabilidades.

Exemplo de classe coesa

Exemplo 2: Suponha agora a seguinte classe:

```
class Stack<T> {
  boolean empty() { ... }
  T pop() { ... }
  push (T) { ... }
  int size() { ... }
}
```

Exemplo de classe não coesa

```
class Estacionamento {
  ...
  private String nome_gerente;
  private String fone_gerente;
  private String cpf_gerente;
  private String endereco_gerente;
  ...
}
```

Acoplamento

Acoplamento é a **força** da conexão entre duas classes. Dizemos que existem dois tipos de acoplamento entre classes: **acoplamento aceitável** e **acoplamento ruim**. [Valente, 2020]

Acoplamento Aceitável:

1. A classe **A** usa apenas métodos públicos da classe **B**.
2. A **interface** provida por **B** é estável. Os métodos públicos de **B** não mudam com frequência. São raras as mudanças em **B** que terão impacto na classe **A**.

Acoplamento ruim

1. Classe **A** realiza um acesso direto a um arquivo ou banco de dados da classe **B**.
2. Quando as classes **A e B** compartilham uma variável.
3. Quando a interface da classe **B** não é estável. Por exemplo, os métodos públicos de **B** são renomeados com frequência.

Exemplo: Acoplamento

Suponha o seguinte trecho de código, no qual existe um arquivo compartilhado por duas classes, A e B, mantidas por desenvolvedores distintos. O método B.g() grava um inteiro no arquivo, que é lido por A.f(). Essa forma de comunicação origina um acoplamento ruim entre as classes. Por exemplo, o desenvolvedor que implementa B pode não saber que o arquivo é lido por A. Assim, ele pode decidir mudar o formato do arquivo por conta própria, sem comunicar o desenvolvedor da classe A.

```
class A {
  private void f() {
    int total; ...
    File arq = File.open("arq1.db");
    total = arq.readInt();
    ...
  }
}
```

```
class B {
  private void g() {
    int total;
    // computa valor de total
    File arq = File.open("arq1.db");
    arq.writeInt(total);
    ...
    arq.close();
  }
}
```

Figura: Exemplo acoplamento [Valente, 2020].

```
class A {

  private void f(B b) {
    int total;
    total = b.getTotal();
    ...
  }
}
```

```
class B {

  int total;

  public int getTotal() {
    return total;
  }

  private void g() {
    // computa valor de total
    File arq = File.open("arq1");
    arq.writeInt(total);
    ...
  }
}
```

Objetivos

- Realizar a análise de um diagrama de classes e realizar a sua implementação em alguma linguagem de programação.
- Utilizar um projeto em um repositório de código fonte.
- Aplicar conceitos de Coesão e Acoplamento

Prática 1

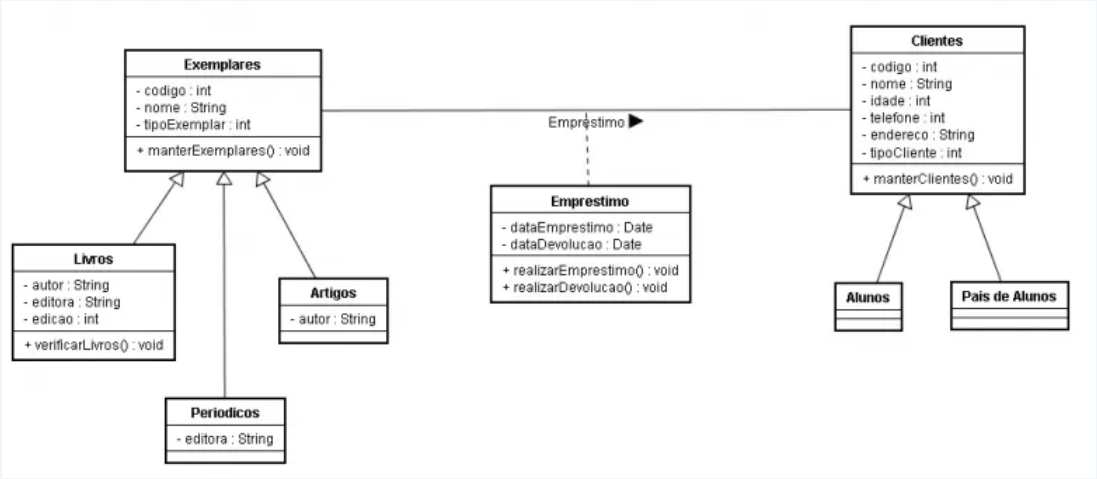
Para esta atividade crie grupo de 5 alunos. Cada aluno terá uma função específica. Realize a análise do diagrama de classes a seguir.

- Controlar o tempo
- Registrar as informação
- Apresentar os resultados
- Anotar os problemas encontrados.

Controle de Empréstimo de Livros

Você está desenvolvendo um sistema de gerenciamento de biblioteca para uma biblioteca local. O sistema permitirá que os funcionários realizem tarefas como registrar novos livros, emprestar livros para membros, renovar empréstimos, gerar relatórios de estoque, etc.

- Deve ser possível listar todos os livros emprestados a um cliente.


Exemplares
- codigo : int
- nome : String
- tipoExemplar : int
+ manterExemplares() : void
Emprestimo
Clientes
- codigo : int
- nome : String
- idade : int
- telefone : int
- endereco : String
- tipoCliente : int
+ manterClientes() : void
Emprestimo
- dataEmprestimo : Date
- dataDevolucao : Date
+ realizarEmprestimo() : void
+ realizarDevolucao() : void
Livros
- autor : String
- editora : String
- edicao : int
+ verificarLivros() : void
Artigos
- autor : String
Periodicos
- editora : String
Alunos
Pais de Alunos

Prepare o ambiente

1. Escolha a sua linguagem de programação de preferência.
2. Escolha uma IDE ou o git.dev
3. Crie um repositório no github(https://github.com/) para que todos os membros da equipe possam colaborar no desenvolvimento (adicione a professora ao repositório **elianemaciel**).

Implementação

- Implemente as classes com os atributos, métodos(vazios) e relacionamentos de acordo com o diagrama de classes propostos.
- Distribua as classes em pacotes se possível.
- Proponha melhorias para o modelo disponibilizado afim de otimizar os recursos.

Extra

- Modele as classes para o projeto A3
- Implemente as classes com os atributos, métodos(vazios) e relacionamentos de acordo com o diagrama de classes propostos.
- Distribua as classes em pacotes se possível.
- Proponha melhorias para o modelo disponibilizado afim de otimizar os recursos.

Referências

Referências


Sommerville, Ian (2011)
Engenharia de Software
Pearson Prentice Hall – 9. ed.


Pressman, Roger (2021)
Engenharia de Software: Uma abordagem Profissional
AMGH Editora Ltda – 9. ed.


Valente, Marco Tulio (2020)
Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade
Editora: Independente

Obrigada

Prof. Eliane Faveron Maciel

UNIFACS
ecossistema ânima

2 de abril de 2024