

	<p><b>INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ</b></p> <p><b>Curso: ADS</b></p> <p><b>Disciplina: Engenharia de Software III</b></p> <p><b>Professor: Ely</b></p>
--	---

### Exercícios 02

1. Qual a relação que coesão possui com as responsabilidades de uma classe?

*Uma classe com as responsabilidades bem definidas possui uma relação direta com o conceito de coesão, pois quando uma classe tem métodos e atributos que somente interagem com aquela entidade, implicitamente está usando da coesão. Em outras palavras, é aquilo que faz sentido para a entidade envolvida.*

2. Contextualize o conceito de coesão de acordo com as fontes abaixo:

a. Livro Clean Code: páginas de 140 a 151;

b. Artigo **Coesão e Acoplamento em Sistemas Orientados a Objetos**

disponível em

[https://www.researchgate.net/publication/261026207\\_Coesao\\_e\\_Acoplamento\\_em\\_Sistemas\\_OO](https://www.researchgate.net/publication/261026207_Coesao_e_Acoplamento_em_Sistemas_OO);

c. Livro Orientação a Objetos e SOLID para Ninjas Projetando classes flexíveis - Casa do Código. Princípio da responsabilidade única. Página 5 em diante.

*Com base no que foi exposto em cada uma das fontes citadas pode-se perceber que os três autores convergem para a mesma direção quanto a questão da coesão como um princípio extremamente importante a ser seguido. A coesão é retratada como uma classe ou entidade que possui apenas uma função, não se deve interpretar como uma sub-rotina de um código, mas em questão da própria existência da classe. Ou seja, aquela classe deve existir para cumprir com um único objetivo. Se ela é uma classe Pessoa, então deve apenas ter informações relacionadas à pessoa.*

*Os autores levantam as vantagens de ter classes coesas, enfatizando que classes coesas possuem menos mudanças; por conseguinte, há menos problemas no momento de gerar uma manutenção de projetos quando estes começam a adquirir grandes proporções. Além disso, os autores mostraram que certos erros são derivados de uma má compreensão do que aquela*

*classe deveria fazer. Por vezes, ela engloba atribuições de outras, gerando um código que dificilmente haveria um reuso.*

3. A classe abaixo possui problemas de coesão:

```
public class Cliente {  
    private int id;  
    private String nome;  
    private String endereco;  
    private double valorCompra;  
    private String numeroConta;  
    private double saldo;  
    public void exibirInformacoes() {  
        // exibir informações do cliente  
    }  
    public void realizarCompra() {  
        // lógica de compra  
    }  
    public void atualizarSaldo() {  
        // atualizar saldo do cliente  
    }  
}
```

Refatore o código criando classes Conta, Compra e Cliente de forma que cada uma fique coesa.

Solução da questão três se encontra na página do link:

<https://github.com/micherlane/engenhariaSoftwareIII/tree/master/Atividade-02/Codigos/src/main/java/br/com/questao03>

4. Classes utilitárias são muito comuns, pois elas agregam funcionalidades usadas por outras classes para centralizar códigos de utilidade geral. A classe abaixo entretanto possui problemas de coesão por ter diferentes aspectos de utilidades. Refatore-a criando classes necessárias de forma a deixar as classes resultantes coesas. Implemente os métodos e além disso, modifique os métodos para acessarem atributos e não usar os parâmetros passados.

```
public class Utilitarios {  
    public void ordenar(int[] array) {  
        // lógica de ordenação  
    }  
    public void embaralhar(int[] array) {  
        // lógica de ordenação  
    }  
    public String removerEspacos(String texto) {  
        // lógica de remoção de espaços  
    }  
}
```

```

}
public String[] quebrarEmPalavras(String texto) {
// lógica de remoção de espaços
}
public double calcularMedia(double[] numeros) {
// lógica de cálculo de média
}
        public double calcularDesvioPadrão(double[] numeros) {
// lógica de cálculo de média
}
}

```

5. Demonstre com Classes os principais tipos de coesão presentes no artigo

Acoplamento e Coesão, disponível em

<https://www.facom.ufu.br/~ronaldooliveira/PDS2019-1/Aula10-Complemento-AcoplamentoCoesao.pdf>

6. Explique com suas palavras o que é o acoplamento entre classes

.

*Acoplamento é uma união entre classes em que uma classe engloba a outra. Isto é a uma interdependência entre as duas classes. Quando o acoplamento entre as classes é alto, há uma extrema dependência. Consequentemente, a classe é menos imutável. Quando o acoplamento é baixo, ela pode atuar com certa independência. Ficando mais fácil o reuso e manutenção.*

7. Crie e exemplifique uma classe com alto acoplamento e refatore-a para ter o acoplamento reduzido.

8. Por que dizemos que o princípio "Tell, don't ask" mitiga o problema de acoplamento. Demonstre.

9. É possível zerar o acoplamento em um software simples ou complexo? Justifique.

10. O encapsulamento também mitiga o acoplamento alto? Discuta o exemplo da página 7 do artigo **Coesão e Acoplamento em Sistemas Orientados a Objetos**.

11. A classe abaixo no contexto de uma rede social é fortemente acoplada a Posts e Comentários.

```

public class Usuario {
private List<Post> posts;
private List<Comentario> comentarios;
public void criarPost(String conteudo) {
Post novoPost = new Post(conteudo);
posts.add(novoPost);
}
public void criarPost(String conteudo) {

```

```
Comentario novoComentario = new Comentario(conteudo);  
comentarios.add(novoComentario);  
}  
}
```

Crie uma interface interface ou classe abstrata chamada Conteudo que seja implementada tanto por Post quanto por outras classes de conteúdo, como Comentario. Dessa forma, a classe Usuario terá uma lista de Conteudo e pode criar e gerenciar qualquer tipo de conteúdo sem conhecer as implementações específicas.

12. A classe usuário abaixo está acoplada a si mesmo como uma lista de amigos. Só que nem todos são amigos, podem ser seguidores ou outro tipo de relacionamento. Proponha uma classe intermediária, como Relacionamento, que armazene o usuário que iniciou o relacionamento, o usuário que aceitou, bem como o tipo de relacionamento (enum) entre dois usuários. Refatore a classe usuário para em vez de uma lista de Usuarios, tenha uma lista de relacionamentos.

```
public class Usuario {  
    private List<Usuario> amigos;  
    public void adicionarAmigo(Usuario amigo) {  
        amigos.add(amigo);  
    }  
}
```

13. Verifique seus repositórios e repositórios públicos e identifique pelo menos 1 problemas de coesão de cada categoria acima listada na questão 5, classifique-os e proponha melhorias.

14. Verifique seus repositórios e repositórios públicos e identifique problemas de acoplamento e proponha melhorias.