INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PIAUÍ	**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ** **Curso: ADS** **Disciplina: Engenharia de Software III** **Professor: Ely**

Exercícios 02

1. Qual a relação que coesão possui com as responsabilidades de uma classe?
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;
    c. Livro Orientação a Objetos e SOLID para Ninjas Projetando classes flexíveis - Casa do Codigo. Princípio da responsabilidade única. Página 5 em diante.
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.

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/PDS-2019-1/Aula10-Complemento-AcoplamentoCoesao.pdf
6. Explique com suas palavras o que é o acoplamento entre classes.
7. Crie e exemplifique uma classse com alto acoplamento e refatore-a para ter o acoplamento reduzido.
8. Por que dizemos que o princípio ”Tell, don’t ask” mitiga problema de acoplamento. Demonstre.
9. É possivel 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 usuario que iniciou o relacionamento, o usuario 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.