Curso de Orientação a Objetos em C#	Aula03
Professora: Luana Fernandes	Turma: Quinta – 19:30h

Conteúdo desta aula:

- Modificadores de acesso
- Encapsulamento

MODIFICADORES DE ACESSO

Os modificadores de acesso são padrões de visibilidade de acessos às classes, atributos e métodos. Esses modificadores são palavras-chaves reservadas pela linguagem, ou seja, não podem ser usadas como nome de métodos, classes ou atributos.

Public → Uma declaração com o modificador public pode ser acessada de qualquer lugar e por qualquer entidade que possa visualizar a classe a que ela pertence.

Private → Os membros da classe definidos como private não podem ser acessados ou usados por nenhuma outra classe, nem pelas classes herdadas. Esse modificador não se aplica às classes, somente para seus métodos e atributos.

Protected → O modificador protected torna o membro acessível às classes do mesmo pacote ou através de herança, seus membros herdados não são acessíveis a outras classes fora do pacote em que foram declarados.

Internal (padrão) → A classe e/ou seus membros são acessíveis somente por classes do mesmo pacote, na sua declaração não é definido nenhum tipo de modificador, sendo este identificado pelo compilador.

Os modificadores de acesso ficam alocados no início da declaração dos elementos. Veja:

```
private int idade; → Atributo privado
public void MostraIdade( ) { → Método público
    Console.WriteLine("Idade: " + idade);
}
```

ENCAPSULAMENTO

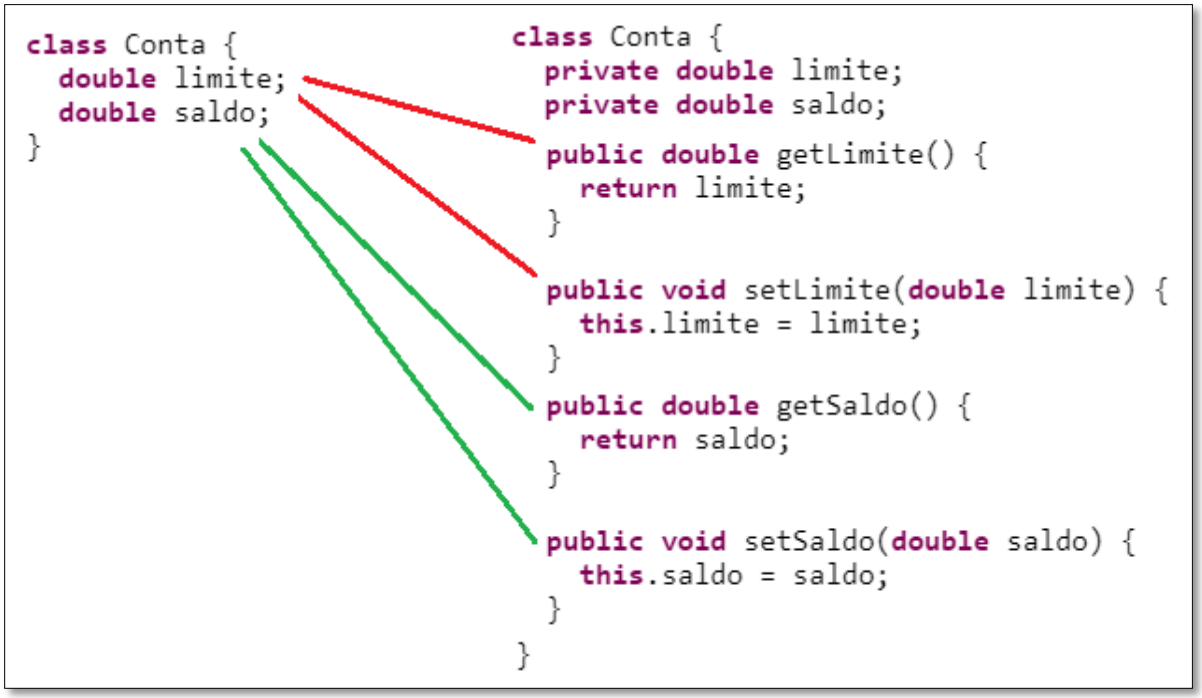
Encapsulamento é a técnica que faz com que detalhes internos do funcionamento dos métodos de uma classe permaneçam ocultos para os objetos. Com isso, o conhecimento a respeito da implementação interna da classe é desnecessário do ponto de vista do objeto, uma vez que isso passa a ser responsabilidade dos métodos internos da classe.

Tendo em mente que os métodos e os atributos de uma classe podem ser definidos como públicos ou privados, temos a seguinte situação:

- Tudo o que o usuário externo precisa conhecer a respeito de uma classe encontra-se em métodos declarados como públicos (public).
- Somente os métodos da classe são capazes de acessar seus atributos privados. Isso garante que não ocorrerão ações inadequadas, mas exige que a interface pública seja bem planejada para que o funcionamento interno da classe não seja muito exposto.

Getters e Settters - Métodos Acessores

Devemos sempre fornecer campos privados e métodos de acesso (getters) públicos, e caso a classe seja mutável devemos também fornecer os métodos modificadores (setters). Veja o exemplo da classe Conta não encapsulada, em seguida usando Encapsulamento, ou seja, tornando seus atributos privados e criando os setters e getters de acesso.

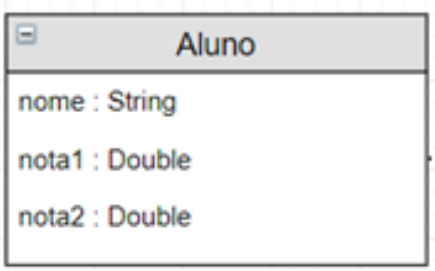


Desse modo, passaremos a usar os getters e setters dos atributos e não teremos mais acesso direto aos atributos, pois mudamos os modificadores para privados. Sendo assim, qualquer regra de negócio que venha a surgir no projeto pode ser implementada nos métodos. Por exemplo, dentro do setSaldo(double saldo) podemos fazer uma validação para verificar se o valor de saldo é maior que zero.

EXERCÍCIOS

Questão 1

Após a explicação apresenta, crie um projeto chamado **escola** com a classe **Aluno** Siga o diagrama a seguir para incluir os atributos e encapsule-os. Não se esqueça de alterar a visibilidade dos atributos para **private**.



OBS: Algumas bibliográficas criticam o uso genérico de getters e setters que nunca serão usados. Não adotaremos essa prática aqui, mas sugiro que leia sobre o assunto.

Questão 2

Execute as seguintes validações dentro dos métodos criados:

a) Dentro de **SetNota1** e **SetNota2**, valide se o parâmetro recebido está entre 0 e 10.

b) Dentro de **SetNome**, verifique se o nome contém até 30 caracteres. Se houver mais, mostre na tela a mensagem: "**O nome deve conter até 30 caracteres**."

OBS: Pesquise pelo comando Length do C#.

Questão 3

a) Crie uma classe chamada **ControleAluno** e inclua o método Main().

b) Use os getteres e setters criados para preencher os dados de um Aluno.

Veja um exemplo:

```
Aluno aluno1 = new Aluno();
aluno1.setNome("Maria");
aluno1.setNota1(8.3);
aluno1.setNota2(7.2);
```

c) Teste se as validações dos métodos estão funcionando passando como parâmetro os seguintes exemplos de valores:

- Um nome com mais de 30 caracteres;
- Uma nota1 maior que 10;
- Uma nota2 com valor negativo.