UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE PROCESSAMENTO DE ENERGIA ELÉTRICA

Enumerações, Listas e Associações

Curso de Engenharia de Controle e Automação

DPEE1090 - Programação orientada a objetos para automação

Prof. Renan Duarte

1º semestre de 2024

Sumário

Enumerações e Associações

- Definição e conceito de enumerações
- Vantagens do uso de enumerações
- Listas
- Definição e conceito de associação
- Tipos de associação: agregação, composição e dependência
- Exemplos

Enumerações

Definição

É um tipo especial que serve para especificar de forma literal um conjunto de constantes relacionadas

- Palavra chave em C#: `enum`
- Nota: enum é um tipo valor

Vantagem: melhor semântica, código mais legível e auxiliado pelo compilador

https://learn.microsoft.com/pt-br/dotnet/csharp/language-reference/builtin-types/enum

Enumerações

Exemplo 1 – Sem enumerações

Vamos criar uma classe com um método que retorna o dia da semana em formato de *string* a partir de um número inteiro recebido

```
public string GetDayOfWeek(int day)
{
    switch (day)
    {
        case 1:
            return "Segunda-feira";
        ...
        case 7:
            return "Domingo";
        default:
            return "Dia inválido!";
    }
}
```

Enumerações

Exemplo 1 – Sem enumerações

Problema 1 - E se agora o primeiro dia da semana deva ser o domingo?

```
public string GetDayOfWeek(int day)
{
    switch (day)
    {
        case 1:
            return "Domingo";
        ...
        case 7:
            return “Sábado";
        default:
            return "Dia inválido!";
    }
}
```

Enumerações

Exemplo 1 – Sem enumerações

Problema 2 - E se agora definirmos a variável `day` como 8?

Neste caso, nosso programa compilaria sem nenhum erro ou aviso mas a saída seria:

```
"Dia inválido!"
```

Contudo, sabemos que só existem 7 dias possíveis. Idealmente nosso programa deveria impedir que valores inválidos fossem utilizados no atributo `day`

Enumerações

Exemplo 2 – Com enumerações

- Como visto, a modificação de parâmetros é trabalhosa
- Além disso, não há nenhum mecanismo que nos impeça de utilizar o método com parâmetros inadequados

Qual a alternativa?

```
public enum DayOfWeek : int
{
    SegundaFeira = 1,
    TercaFeira,
    QuartaFeira,
    QuintaFeira,
    SextaFeira,
    Sabado,
    Domingo,
}
```

Enumeração de inteiros
Funciona como um tipo de dados

Podemos criar atributos do tipo `DayOfWeek`

O tipo da enumeração (int) e os valores de cada item são opcionais

Enumerações

tinyurl.com/y6vbfmvm

Comandos uteis

```
// Converte uma string para o valor correspondente do enum
DayOfWeek day = Enum.Parse<DayOfWeek>("QuartaFeira");

// Tenta converter uma string para o valor correspondente do enum
if (Enum.TryParse<DayOfWeek>("TercaFeira", out DayOfWeek result))
    // Sucesso: result contém o valor convertido
else
    // Falha na conversão

// Retorna um array de strings que contém os nomes dos membros do enum
string[] dayNames = Enum.GetNames(typeof(DayOfWeek));

// Retorna um array dos valores do enum
DayOfWeek[] days = (DayOfWeek[])Enum.GetValues(typeof(DayOfWeek));
```

Enumerações

https://bit.ly/4aRq3B1

Exercício 1

Você foi contratado para desenvolver um sistema de gerenciamento de pedidos para uma loja online. Sua primeira tarefa é implementar a classe `Order`, que representará os pedidos feitos pelos clientes. A classe `Order` deve possuir os seguintes atributos:

- `Id`: Representa o identificador único do pedido
- `Value`: Representa o valor do pedido
- `Status`: Representa o status atual do pedido.

Os status possíveis são: `PendingPayment`, `Processing`, `Shipped`, `Delivered`

A classe também deve ter um método `ToString` que imprime os dados do pedido

Listas

Coleção de dados

É uma estrutura de dados:

- Homogênea (dados do mesmo tipo)
- Ordenada (elementos acessados por meio de posições)
- Inicia vazia e seus elementos são alocados sob demanda
- Cada elemento ocupa um “nó” (ou nodo) da lista

Vantagens:

- Tamanho variável
- Facilidade para realizar inserções e deleções

Desvantagens:

- Acesso sequencial aos elementos

https://learn.microsoft.com/pt-br/dotnet/api/system.collections.generic.list-1?view=net-8.0

Listas

Significado de “acesso sequencial”

Cada elemento (ou nodo) da lista, aponta para o próximo elemento.

Assim, para acessarmos o elemento (2) do diagrama abaixo, iniciamos pelo nodo (0) e percorremos sequencialmente a lista até encontrar o elemento desejado.

Listas

Exemplo 4 – Lista de objetos

Além dos tipos básicos, podemos fazer listas com objetos:

```
// Criar uma lista de alunos
List<Aluno> alunos = new List<Aluno>();

// Adicionar alunos à lista
alunos.Add(new Aluno("João"));
alunos.Add(new Aluno("Maria"));
alunos.Add(new Aluno("Pedro"));
alunos.Add(new Aluno("Ana"));

// Exibir os nomes dos alunos na lista
Console.WriteLine("Lista de Alunos:");
foreach (Aluno aluno in alunos)
   Console.WriteLine(aluno);
```

Listas

Comandos úteis

- Inserir elemento: `Add`, `Insert`
- Tamanho da lista: `Count`
- Encontrar primeiro ou último elemento que satisfaça um predicado: `Find`, `FindLast`
- Encontrar primeira ou última posição de elemento que satisfaça um predicado: `FindIndex`, `FindLastIndex`
- Filtrar a lista com base em um predicado: `FindAll`
- Remover elementos da lista: `Remove`, `RemoveAll`, `RemoveAt`, `RemoveRange`

Associação

Definição

Em programação orientada a objetos, a associação é um dos conceitos fundamentais que descreve a relação entre classes e objetos.

A associação possibilita um relacionamento entre classes (ou objetos) no qual estes podem utilizar os serviços ou métodos de outra classe (ou objeto) para representar de forma completa o conceito no qual se destinam.

Neste tipo de relacionamento, as classes e os objetos interagem entre si para atingir seus objetivos

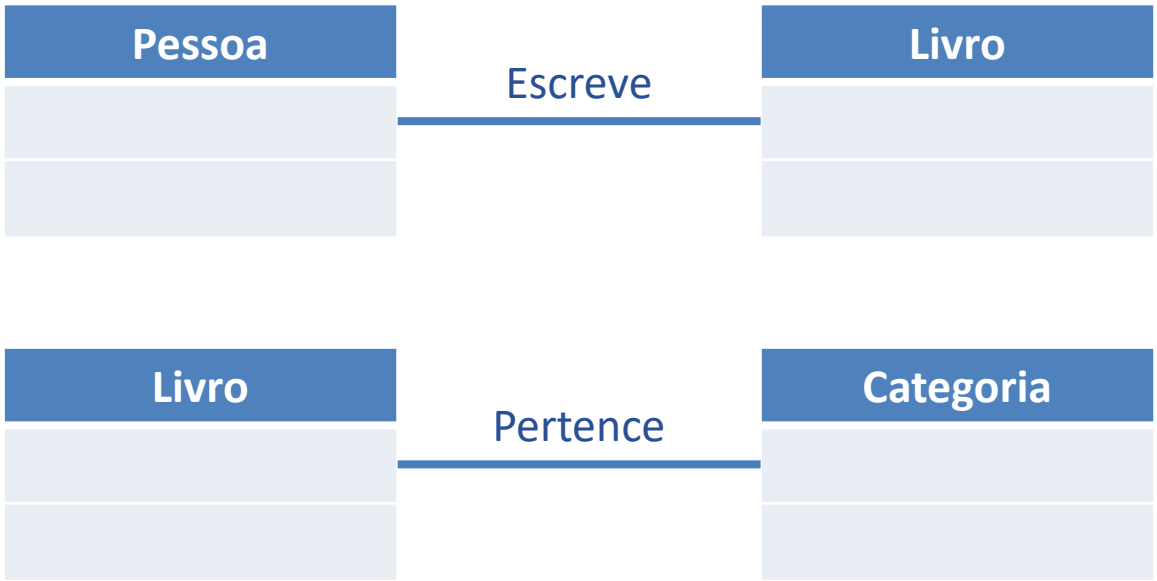
Associação

Definição

Atributos de um objeto que se referem a outros objetos

Associação

Relacionamentos em UML



Relacionamento direcionado

Associação

Cardinalidade no relacionamento

Restrições que limitam a possibilidade de combinações de entidades em relacionamentos → Mínima e máxima

Razão de cardinalidade

É a razão (ou proporção) de participação em um relacionamento

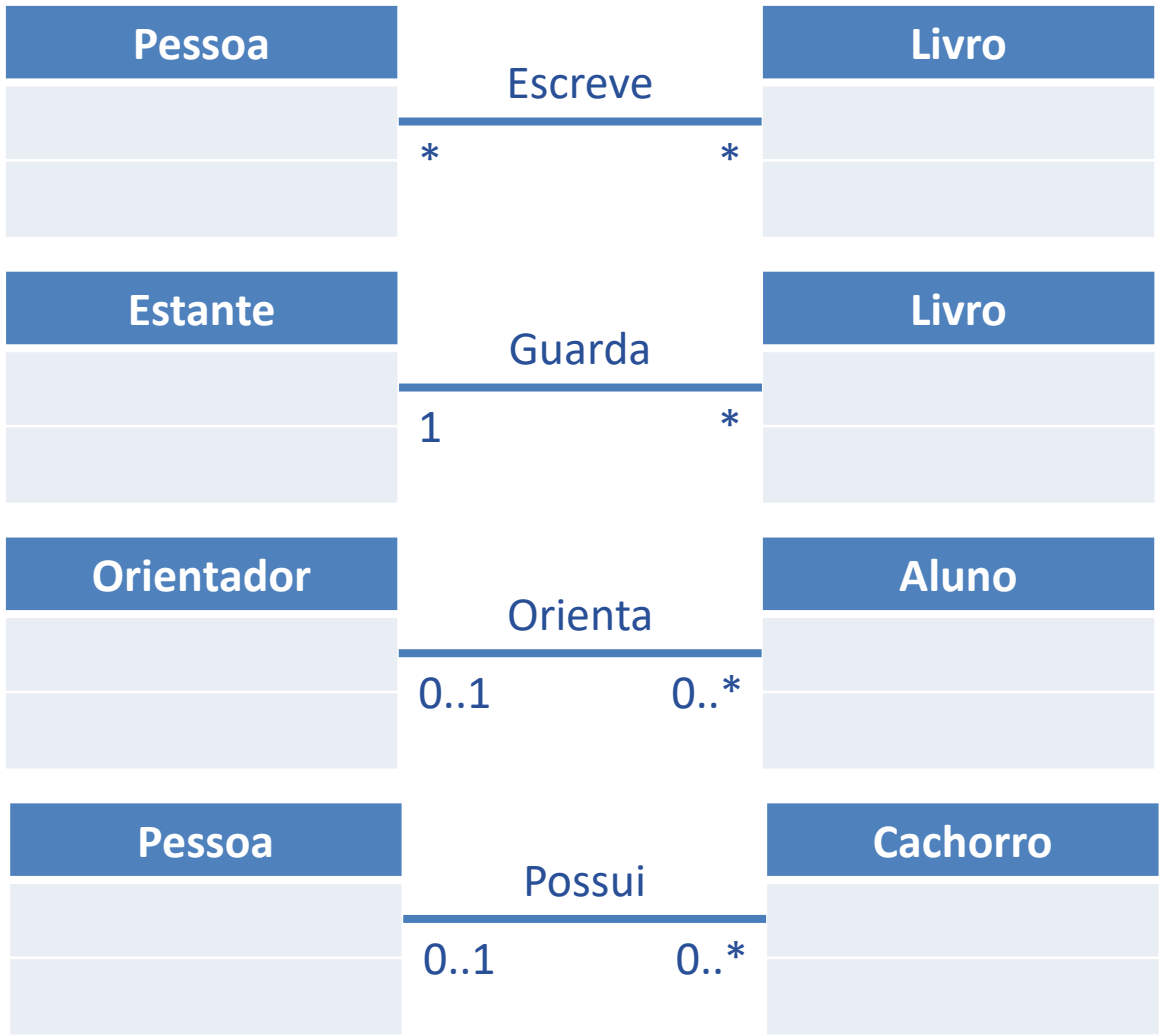
Possibilidades: **1:1, 1:N, N:1, N:N** (Símbolo * indica N)

Exemplos:

- Uma pessoa pode escrever N livros e um livro pode ser escrito por N pessoas
- Uma estante pode conter N livros, mas um livro só pode estar em uma estante

Associação

Cardinalidade no relacionamento



Tipos de associação

Agregação

Relação do tipo parte-todo

Uma classe agrega outra (não existencial)

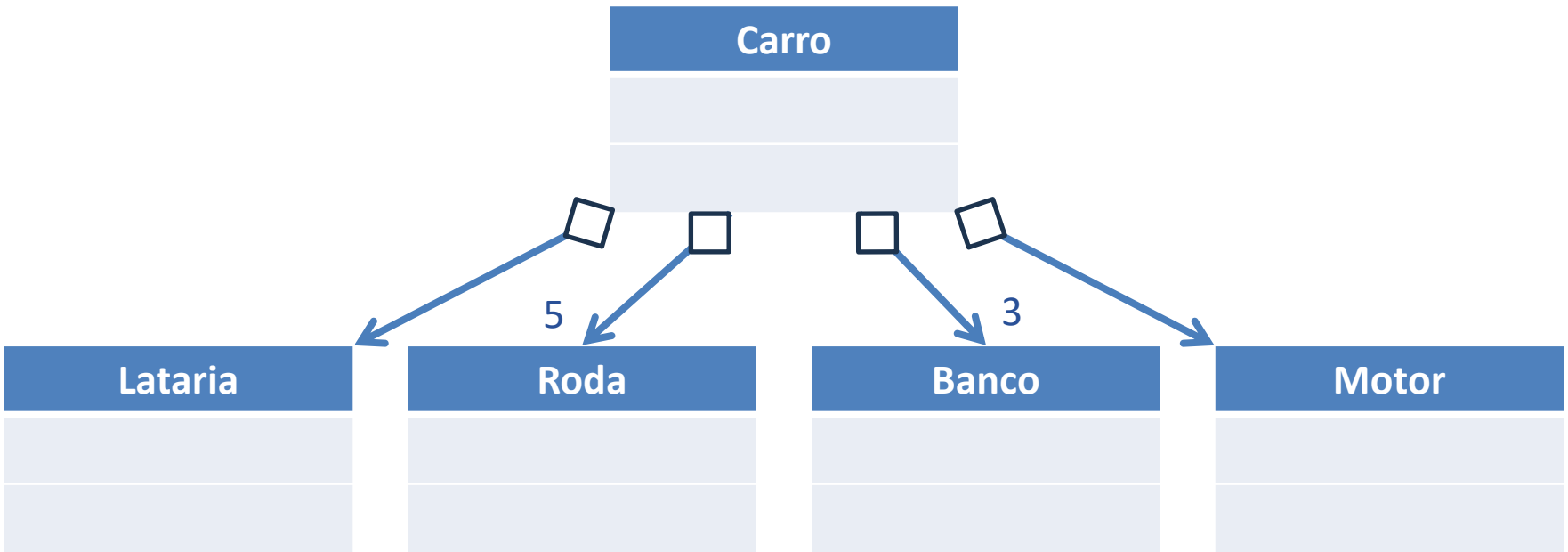
- Objetos "parte" podem existir mesmo sem o relacionamento (não são criados exclusivamente para esse fim)
- Objetos "parte" podem ser compartilhados (um mesmo autor pode ter vários livros ou um mesmo livro pode ter vários autores)

Tipos de associação

https://bit.ly/4ccKdqv

Exemplo 5 – Agregação

Agregação de diferentes objetos para modelar um carro



Tipos de associação

Composição

Relação do tipo parte-todo

Uma classe é parte de outra (relacionamento “mais forte”)

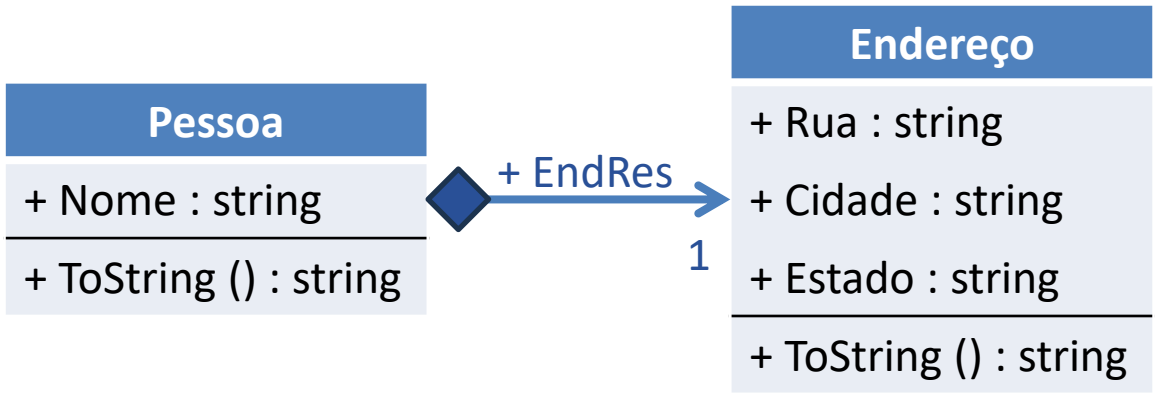
- Se objeto “todo” for destruído, objeto(s) “parte” também deixam de existir
- Objeto “todo” é responsável pela criação e destruição das partes

Tipos de associação

https://bit.ly/3XqXmI0

Exemplo 6 – Composição

Cadastro de pessoas e seus respectivos endereços



Embora seja possível criar objetos "Endereço" de forma independente, quando criamos uma "Pessoa", o objeto "Endereço" correspondente não tem sentido fora do contexto do objeto "Pessoa"

Revisão

https://bit.ly/3KREfPR

Exercício 2 – Enumerações, listas, agregação e composição

Ler os dados de um pedido com N itens (N fornecido pelo usuário). Depois, mostrar um resumo do pedido conforme exemplo (próximos slides).

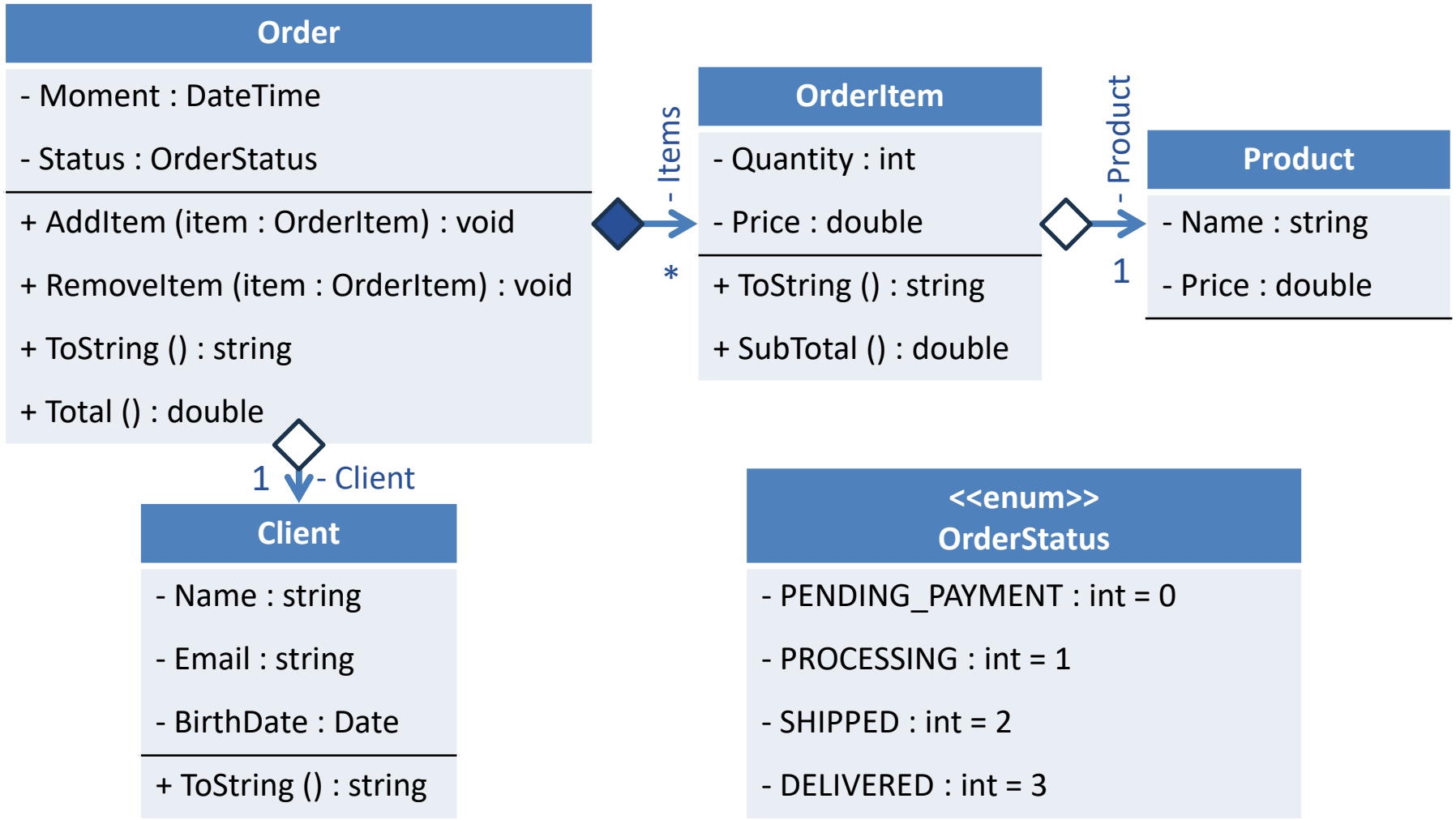
Utilize o diagrama UML do próximo slide como base para seu desenvolvimento.

Dica: O instante do pedido por ser capturado automaticamente com uma variável do tipo DateTime:

```
DateTime Moment = DateTime.Now;
```

Revisão

Exercício 2 – Continuação



Revisão

Exercício 2 – Continuação

```
Enter cliente data:
Name: João da Silva
Email: joao@gmail.com
Birth date (DD/MM/YYYY): 10/10/2000

Enter order data:
Status: Processing
How many items to this order? 2

Enter #1 item data:
Product name: TV
Product price: 1000.00
Quantity: 1
Enter #2 item data:
Product name: Mouse
Product price: 40.00
Quantity: 2

ORDER SUMMARY:
Order moment: 19/06/2024 11:25:09
Order status: Processing
Client: João da Silva (10/10/2000) - joao@gmail.com
Order items:
TV, $1000.00, Quantity: 1, Subtotal: $1000.00
Mouse, $40.00, Quantity: 2, Subtotal: $80.00
Total price: $1080.00
```

Revisão

Próxima aula

Métodos e Classes Abstratas

- Introdução ao conceito de métodos e classes abstratas
- Implementação de métodos e classes abstratas
- Diferenças entre classes abstratas e interfaces

Dúvidas?

renan.duarte@gedre.ufsm.br

GEDRE – Prédio 10 – CTLAB