# VIANNSTITUTO JUNIOR



# Algoritmos

Classes

Professor: Camillo Falcão



### Estruturas heterogêneas

 Até agora vimos as estruturas de dados homogêneas: vetores, matrizes e strings.

 Nestas estruturas todos os elementos da estrutura são de tipos de dados primitivos: inteiro, real, caractere.



# Estruturas heterogêneas

 No entanto, em muitos casos, necessitamos armazenar conjuntos de informações relacionadas, formados por diversos tipos de dados primitivos.

### Exemplos:

- Endereço;
- Fichas com dados pessoais de um cliente;
- Fichas com dados de um produto.



## Variáveis compostas

- Quando uma determinada estrutura de dados for composta por diversos tipos diferentes, primitivos ou não, temos um conjunto heterogêneo de dados.
- Essas variáveis são chamadas de variáveis compostas heterogêneas.
- Em C#, é possível utilizar estruturas (struct) ou classes (class) para a criação de tipos de dados definidos pelo usuário.



## Variáveis compostas

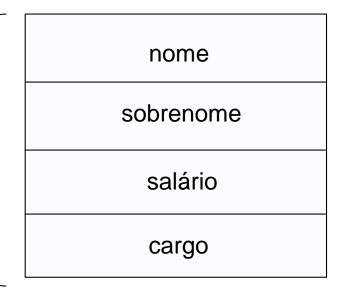
- Uma classe pode conter dados e também pode conter métodos que manipulam os dados. Nessa disciplina, vamos utilizar os métodos apenas para acessar valores de variáveis internas à classe. Outras utilizações de métodos não serão apresentadas.
- Atenção: nesta disciplina não será ensinada a Programação Orientada a Objetos. Esse assunto será abordado no próximo período em outras disciplinas.





 Uma classe pode ser definida como uma coleção de uma ou mais variáveis relacionadas (atributos) e uma coleção de métodos, sendo que cada variável pode ser de um tipo distinto.

Exemplo: empregado (4 atributos)



### Classes



- Uma classe é uma estrutura de dados que contém dados e métodos (procedimentos ou funções). Uma classe descreve comportamentos e atributos comuns.
- Um exemplo de uma classe em um sistema acadêmico seria a classe Aluno. É possível definirmos uma série de atributos comuns a todos os alunos:
  - Todo aluno possui um número de matrícula;
  - Todo aluno possui um nome;
  - Todo aluno possui um e-mail;
  - Todo aluno possui um telefone;
  - Todo aluno possui um endereço;

### VIANNA JUNIOR

### Classes

- Note que também podemos ter uma classe Endereco para representar o endereço de alunos. Alguns atributos comuns a endereços são:
  - Logradouro;
  - Número;
  - Complemento;
  - Bairro;
  - Cidade;
  - UF;
  - CEP;
- Em um sistema acadêmico, a classe aluno se refere ao conceito de um aluno no sistema (conjunto de atributos e operações que caracterizam um aluno). Já um objeto aluno se refere a um aluno específico que é uma instância da classe Aluno.



### Classes em C#

Sintaxe para definir uma classe em C#:

```
class NomeDaClasse
{
}
```

 Para adicionar uma classe, clique com o botão direito do mouse em cima do seu projeto e escolha "Add->New Item". Na tela apresentada, selecione a opção Class e, em File Name, digite o nome da sua classe.



### **Atributos**

- Até este ponto, todos os métodos e todas constantes que criamos possuem o modificador static. Esse modificador informa que o método ou constante pertence à classe e não ao objeto.
- Agora vamos criar variáveis que pertencem aos objetos e não às classes. Essas variáveis são chamadas atributos das classes.
- Nesse momento, vamos criá-las como públicas.

### **Atributos**



Sintaxe para a criação de atributos:

```
<modificador_de_acesso> <tipo> nomeAtributo;
```

Exemplo:

```
class Aluno
{
    public int matricula;
    public string nome;
}
```



### Criando objetos

 Para criar instâncias de uma classe, basta utilizar a seguinte sinaxe:

```
<Classe> nomeObjeto = new <Classe>();
```

Exemplo:

```
Aluno aluno1 = new Aluno();
```



### Manipulação de objetos

 Campos ou membros de um objeto podem ser usados da mesma forma

como as variáveis.

```
class Aluno
{
    public int matricula;
    public string nome;
}
```

 Campos são acessados usando o operador de acesso ponto (.) entre o nome do objeto e o nome do campo.



### Estruturas: Manipulação

- Para modificar ou acessar um atributo de um objeto, basta usarmos o operador (.).
- Exemplo:

```
class Aluno
{
    public int matricula;
    public string nome;
}
```

```
static void Main(string args[])
{
    Aluno aluno1 = new Aluno();
    aluno1.matricula = 123;
    aluno1.nome = "Maria da Silva";

    Console.Write("Matricula {0}", aluno1.matricula);
    // ...
}
```



### Mesma classe, diversos objetos

 Vamos ver agora um código simples que instancia dois objetos à partir da classe aluno, preenche os atributos dos objetos e imprime os atributos preenchidos.

```
class Aluno
{
    public int matricula;
    public string nome;
}
```

```
static void Main(string args[])
      Aluno aluno1 = new Aluno();
      aluno1.matricula = 123;
       aluno1.nome = "Maria da Silva";
      Aluno aluno2 = new Aluno();
      aluno2.matricula = 124;
      aluno2.nome = "José da Silva";
   Console.Write("1)\{0\} - \{1\}\n", aluno1.nome, aluno1.matricula);
   Console.Write("2){0} - {1}", aluno2.nome, aluno2.matricula);
```



### A classe como tipo de dado

Definição da classe Endereco:

```
class Endereco
{
    public string logradouro;
    public int numero;
    public string complemento;
    public string bairro;
    public string cidade;
    public string uf;
    public string cep;
}
```



### A classe como tipo de dado

Nova definição da classe Aluno:

```
class Aluno
{
    public int matricula;
    public string nome;
    public Endereco endereco;
}
```

```
class Endereco
{
    public string logradouro;
    public int numero;
    public string complemento;
    public string bairro;
    public string cidade;
    public string uf;
    public string cep;
}
```



### A classe como tipo de dado

Atribuindo e lendo o endereço de uma instância da classe Aluno:

```
static void Main(string args[])
      Aluno aluno1 = new Aluno();
      aluno1.matricula = 123;
      aluno1.nome = "Maria da Silva";
      aluno1.endereco = new Endereco();
      aluno1.endereco.logradouro = "Rua Tal";
      aluno1.endereco.numero = 1000;
   Console.Write("Nome: {0}\n", aluno1.nome);
   Console.Write ("Rua: {0}, nro{1}", aluno1.endereco.logradouro,
aluno1.endereco.numero);
```



- 1) Defina uma classe para representar as informações de um cartão de crédito. No método principal (Main) da classe Program, instancie dois objetos do tipo da classe criada, atribua valores a cada atributo dos objetos e imprima todos os atributos do segundo objeto instanciado.
- 2) Defina uma classe para representar o peso e a altura de uma pessoa. Crie um programa que pergunte ao usuário o seu peso e a sua altura, atribuindo-os aos respectivos atributos de um objeto da classe criada. Imprima os atributos da classe criada.
- 3) Considerando a classe do exercício (2) e a inicialização abaixo:

```
joao.altura \leftarrow 1.90; joao.peso \leftarrow 98; maria.altura \leftarrow 1.50; maria.peso \leftarrow 55;
```

- Escreva uma instrução que atribua 1.78 à altura de joao;
- Escreva uma instrução que atribua 75 ao peso de maria.
- Escreva um conjunto de instruções para imprimir a média das alturas e a média dos pesos de joao e maria.



- 4) Faça um programa (método principal) para leitura, via teclado, dos dados de um contato de telefone. Os dados a serem guardados na classe Contato são os seguintes: nome, endereço (logradouro, número, complemento, bairro, cidade, UF), telefone e e-mail. Ao final, imprima estas informações na tela.
- 5) Imagine que tenha sido realizada uma pesquisa com 6 pessoas a respeito de salário e idade. Crie uma classe Pessoa e faça um programa que leia os dados coletados e forneça a média salarial e a média de idades dos entrevistados.



### Classes com atributos vetor

 Em alguns casos a classe possui vetores como um dos seus campos.

```
class Aluno
{
    public int matricula;
    public string nome;
    public double[] notas = new double[3];
}
```

 O acesso a estes campos é feito da mesma maneira como acesso direto a um vetor.

```
Aluno aluno1 = new Aluno();
a.notas[0] = 100;
a.notas[1] = 90;
a.notas[2] = 95;
```



 Considere as informações de um aluno que tem NOME e 4 notas como atributos de uma classe; veja layout abaixo.

$\wedge$ 1 1		4		Ī
Cadastro	$\Delta$	notae	$\Delta$ CCC	ISTAC
Cauasiiu	uc	HULAS	しついし	ıaı cə

Nome:

Notas			
1	2	3	4

 Desenvolver um algoritmo para ler e imprimir o nome e as notas de um aluno.



- Ideia básica do algoritmo :
  - Definir a classe
  - Declarar variáveis
  - Ler os dados de um aluno
  - 4. Imprimir os dados do aluno



### Definir a classe

```
class Aluno
{
    public string nome;
    public double[] notas = new double[4];
}
```



- Definir a classe
- 2. Declarar variáveis

```
Aluno a = new Aluno();
int i;
```

```
class Aluno
{
    public string nome;
    public double[] notas = new double[4];
}
```



- 1. Definir a classe
- 2. Declarar variáveis
- 3. Ler dados de um aluno

```
class Aluno
{
    public string nome;
    public double[] notas = new double[4];
}
```

```
a.nome = Console.ReadLine();
for (i = 0; i < 4; i++)
{
   a.nota[i] = Convert.ToDouble(Console.ReadLine());
}</pre>
```



- 1. Definir a classe
- 2. Declarar variáveis
- 3. Ler dados de um aluno
- 4. Imprimir os dados

```
Console.WriteLine("{0}", a.nome);
for (i = 0; i < 4; i++)
{
    Console.WriteLine(" {0}", a.nota[i]);
}</pre>
```



### Classes e sub-rotinas

 Um objeto pode ser passado por parâmetro para uma sub-rotina. Um objeto é sempre passado por referência.



- Pode-se criar vetores de objetos como se criam vetores de tipos primitivos.
- Os programas apresentados até o momento só fizeram menção a uma única instância da classe.
- É necessário possuir uma definição da classe antes de declarar um vetor da mesma.



- Suponha que deseja-se manter um registro de informações relativas a passagens rodoviárias de todos lugares (poltronas) de um ônibus.
- Pode-se utilizar uma classe referente a cada poltrona (Passagem) e para agrupar todas elas utiliza-se um vetor de objetos.



Um ônibus possui 44 lugares numerados de 0 a 43:

0	Nome: Número:		
1	De: Para:	-	
	Data:// Horário:	-	
43	Poltrona: Distância:	<u></u>	



Definição da estrutura:

```
class Passagem
{
   string nome;
   int numero;
   string origem;
   string destino;
   string data;
   string horario;
   int poltrona;
   double distancia;
}
```

Declaração do vetor de objetos:

```
Passagem[] vet_passagens = new Passagem[44];
```



### Acessos:

```
vet_passagem[3].numero

vet_passagem[34].distancia

vet_passagem[2].origem
```



### Vetores de objetos - Exemplo

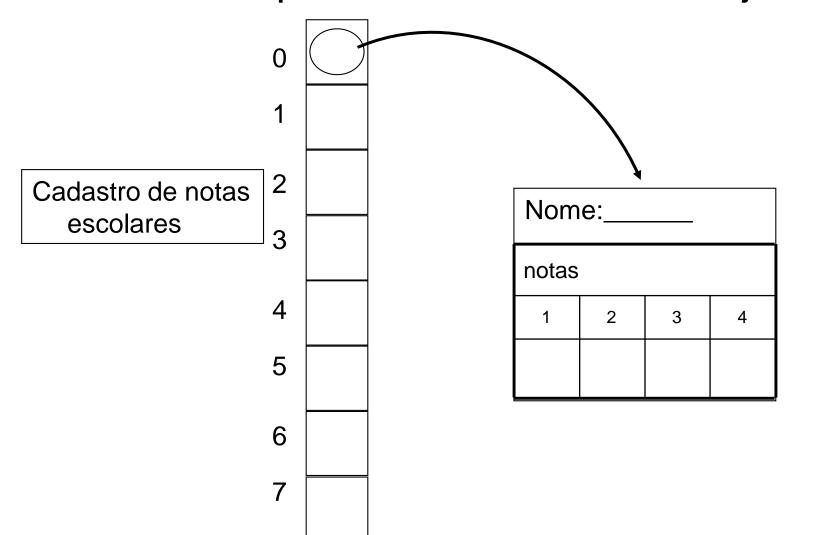
 Considere que você está fazendo um programa que leia o nome e as 4 notas escolares de 8 alunos.



Esquematicamente: Nome:\_ Nome: Cadastro de notas escolares Nome:\_ 4 Nome: Nome: 6 4 Nome: 5 Nome: Indices do vetor Nome: notas 2 3 4 36



Visão esquemática do vetor de objetos:





 Ao utilizarmos o modificador public em nossos atributos da classe Aluno, esses mesmos atributos podem ser setados fora da classe Aluno. Um programador pode esquecer-se de fazer uma validação do dado que está sendo inserido e isso pode deixar o sistema em um estado inconsistente.



 Por exemplo: o usuário poderia informar como número de matrícula um número negativo, o que é impossível. Então o programador poderia criar um método para checar se o número é válido ou não, mas qual o melhor lugar para se colocar esse método? Um segundo programador se lembraria de utilizá**l**0?



 Na orientação a objetos existe o conceito de encapsulamento. Logo, os atributos de suas classes não devem ser públicos.

 Para que alguém consiga utilizá-los, você deve criar métodos públicos. As validações dos dados podem ser feitas nesses métodos.



```
class Aluno
   private int matricula;
    public int getMatricula()
        return matricula;
    public void setMatricula(int novaMat)
        if (novaMat >= 0)
            matricula = novaMat;
   private string nome;
    public string getNome()
        return nome;
    public void setNome(string novoNome)
        nome = novoNome;
```



# Propriedades em C#

```
class Aluno
   private int matricula;
   public int Matricula
        get { return matricula; }
        set
            if (value >= 0)
                matricula = value;
   private string nome;
   public string Nome
        get { return nome; }
        set { nome = value; }
```



- 6) Crie uma classe chamada ponto contendo apenas as coordenadas x e y (inteiros) do ponto. Declare 2 pontos, leia as coordenadas x e y de cada um e calcule a distância entre eles. Apresente no final a distância entre os dois pontos.
- 7) Utilizando a classe ponto definida no exercício anterior, faça um programa que leia 4 pontos. Em seguida imprima qual o ponto mais próximo do primeiro ponto lido.
- 8) Faça um programa que receba três nomes de no máximo 100 caracteres cada e as idades das respectivas pessoas em um vetor de objetos. Após o recebimento, listar os três nomes armazenados neste vetor por ordem crescente de idades.
- 9) Faça um programa que armazene informações de restaurantes. Cada restaurante é identificado pelo nome, endereço, o tipo de comida (brasileira, chinesa, francesa, italiana, japonesa, etc.) e uma nota para a cozinha (entre 0 e 5). O programa deverá ler todas as informações e imprimir ao final a lista de todos os restaurantes, e o endereço do restaurante com maior nota para a cozinha.



- 10) Faça um programa que leia os dados de 10 ingressos. As informações que deverão ser lidas de cada ingresso são: preço, local e atração. Ao final, informe os eventos de ingresso mais barato e mais caros.
- 11) Utilizando a classe definida no exercício anterior. Faça um procedimento que recebe um parâmetro do tipo ingresso e outro contendo um novo preço que atualiza o preço do ingresso para o novo valor. E um procedimento que recebe um parâmetro do tipo ingresso e mostra na tela os valores de seus campos.
- 12) Faça um programa que avalie o preço de um eletrodoméstico em diferentes lojas. Faça a leitura das informações das lojas, que contenham os seguintes campos: nome da loja, telefone e preço do eletrodoméstico. Ao final, informe os dados de todas as lojas. Informe também a média dos preços cadastrados e uma relação contendo o nome e o telefone das lojas cujo preço estava abaixo da média. Utilize funções para realizar operações de leitura e escrita e faça um menu que possibilite ler todas as informações das lojas, ler uma loja com um endereço específico, caso ela esteja cadastrada, e imprimir as informações citadas anteriormente.



13) Faça um programa que permita o cadastro de veículos. A classe veículo deverá conter os campos placa, marca, modelo e ano. Faça um menu com as seguintes opções:

### Menu:

- 1 Ler as informações de um veículo
- 2 Verificar se uma placa está no formato correto (AAADDDD)
- 3 Imprimir por ano
- 4 Pesquisar veículo por placa
- 5 Imprimir todos os veículos cadastrados

### O programa deverá ter as seguintes características:

- No primeiro item, peça inicialmente o índice do vetor que deseja alterar.
- No terceiro item, peça o ano mínimo e máximo e imprima os veículos que estão nesse intervalo.
- Faça funções para realizar as operações de cada um dos itens do menu. 45



- 14) Elabore um programa que auxilie no controle de uma fazenda de gado que possua um total de 100 cabeças de gado. O programa deverá conter uma classe que comporte:
  - código: código da cabeça de gado;
  - leite: número de litros de leite produzido por semana;
  - alimento: quantidade de alimento ingerida por semana (kg);
  - mês de nascimento;
  - ano de nascimento;
  - abate: 'N"(não) ou 'S' (sim).



### 14) (continuação...)

- O seu programa deverá **conter um menu** com as seguintes funcionalidades:
- (a) Ler a base de dados (código, leite, alimento, nascimento) informados pelo usuário e armazenar em um vetor de estruturas.
- (b) Preencher o campo *abate*, considerando que a cabeça de gado irá para o abate caso:
  - tenha mais de 5 anos, ou;
  - produza menos de 40 litros de leite por semana, ou;
  - produza entre 50 e 70 litros de leite por semana e ingira mais de 50 quilos de alimento por semana.
- (c) Imprimir a quantidade total de leite produzida por semana na fazenda.
- (d) Imprimir a quantidade total de alimento consumido por semana na fazenda.
- (e) Imprimir a quantidade total de leite que vai ser produzido por semana na fazenda, após o abate
- (f) Imprimir a quantidade total de alimento que vai ser consumido por semana na fazenda, após o abate
- (g) Imprimir número de cabeças de gado que irão para o abate.

(h) Inclua uma opção para sair do menu.