

EXERCÍCIO 1

OBJETIVO: Criar classes. Criar atributos, propriedades e métodos.

ENUNCIADO:

1. Abra a Solução POOAv¹.
2. Adicione à solução um novo projeto do tipo Class Library - CSAvancadoClasses;
3. Adicione ao projeto CSAvancadoClasses um ficheiro MinhasClasses.cs;
4. Adicione ao ficheiro criado a classe Empregado, com as seguintes características:
 - a. Atributos: nome, dataNascimento, departamento, salario;
 - b. Respetivas propriedades;
 - c. Métodos:
 - i. imprimeTodosDados()
imprime todos os dados do empregado;
 - ii. imprimeDados()
imprime o nome e o salário do empregado.

Atributos

private string _nome;

private DateTime _dataNascimento;

private string _departamento;

private decimal _salario;

Propriedades

public string Departamento

public string Nome

public DateTime DataNascimento

public decimal Salario

public byte Idade (só leitura)

¹ A solução encontra-se na pasta POOAv/Web/CS.

Métodos

```
public string imprimeTodosDados()
```

```
public string imprimeDados()
```

EXERCÍCIO 2

OBJETIVO: Manipular classes e instanciar objetos.

ENUNCIADO:

1. Utilize o `frmInserirEmpregados.aspx` (projeto Programação Avançada CS) para testar a classe `Empregado` criada anteriormente;
 - a. Quando se carrega no botão `Inserir Empregado`, é criado um objeto do tipo `empregado` e é colocado num `ArrayList`;
2. No formulário `frmVerEmpregados.aspx` são visualizados na lista os empregados criados anteriormente, invocando o método `ImprimeDados()` de cada empregado pertencente ao array criado;



Fig. 1

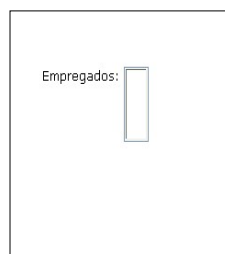


fig. 2

EXERCÍCIO 3

OBJETIVO: Manipular classes. Criar construtores. Criar overload de métodos, implementar operadores. Criar atributos e métodos shared.

ENUNCIADO:

1. Acrescente à classe empregado:
 - a. Os métodos
 - i. `public void AumentaSalario(int valor)`
 - ii. `public void AumentaSalario(decimal percentagem)`
2. Crie um atributo `_numEmp`, que deverá guardar o número de empregados criados (tem que ser comum a todas as instâncias da classe);
3. Crie o construtor default onde:
`nome= ""`; `dataNascimento=1-1-1900`; `departamento= ""`; `salario=0`
4. Crie um construtor que apenas receba o nome e data de nascimento do empregado;
5. Crie um construtor que apenas receba o nome e salário do empregado;
6. Crie um construtor que receba todos os atributos do empregado e invoque o construtor anterior;
7. Altere os construtores criados anteriormente para que o atributo `_numEmp` seja incrementado automaticamente;
8. Altere os métodos `ImprimeDados` e `ImprimeTodosDados` de modo a que o número do empregado apareça na impressão;
9. Implemente os operadores `<` e `>`. Estes operadores categorizam os empregados pelo seu salário.
10. Crie um método
`public int ComparaSalario(Empregado outroempregado)`
que deve devolver um número negativo, zero ou um número positivo, consoante a comparação do salário dos empregados. Utilize os operadores implementados na alínea anterior.

EXERCÍCIO 4

OBJETIVO: Utilizar classes com diferentes de construtores.

ENUNCIADO:

1. Utilize o `frmInserirEmpregadoEx4.aspx` para testar a classe `Empregado` criada anteriormente;
2. Quando se carrega no botão `Inserir Empregado`, é criado um objeto do tipo `empregado` de acordo com o preenchimento das características do `empregado` (preenchimento das caixas de texto) e é colocado num `ArrayList`;
3. No formulário `frmVerEmpregados.aspx` são visualizados na lista os empregados criados anteriormente, invocando o método `ImprimeDados()` de cada empregado pertencente ao array criado;

EXERCÍCIO 5

OBJETIVO: Criar classes herdadas e reescrever métodos.

ENUNCIADO:

1. Estenda a classe `Empregado`, pelas classes `Programador`, `Administrador` e `Comercial`:
 - a. Um `Programador` tem o atributo adicional `linguagem`, crie a propriedade respectiva;
 - b. O `Administrador` tem um atributo adicional `cargo`, crie a propriedade respectiva;
 - c. O `Comercial` tem um atributo adicional `vendas` e o seu salário deve incluir 1% das vendas, crie a propriedade respectiva;
 - d. Faça o overriding dos métodos `ImprimeDados()`, para ter em conta estes novos atributos;

EXERCÍCIO 6

OBJETIVO: Utilizar uma classe base e classes derivadas.

ENUNCIADO:

1. Utilize o frmInserirEmpregadoEx6 para testar a testar as classes criadas no exercício anterior.
2. Quando se carrega no botão **Inserir Empregado**, é criado um objeto do tipo empregado de acordo com o selecionado na combobox do departamento e é colocado num ArrayList;
 - a. Dependendo do departamento será visível uma caixa de texto adequada ao atributo da classe respectiva
3. No formulário frmVerEmpregados.aspx são visualizados na lista os empregados criados anteriormente, invocando o método *ImprimeDados()* de cada empregado pertencente ao array criado;
4. No formulário frmAumentaSalario.aspx são apresentados todos os empregados adicionados e os respectivos salários (apresentado ao utilizador uma listagem dos empregados criados assim como o seu salário e é também apresentado o valor total dos salários pago).

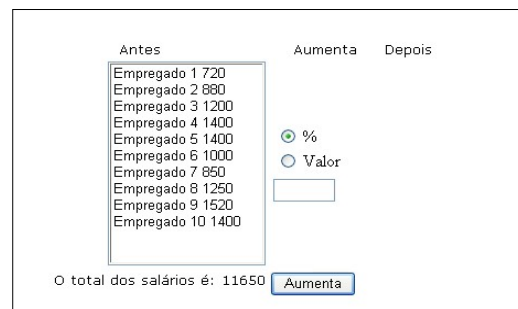
 - a. Quando se carrega no botão **Aumenta Salário** e após a escolha do tipo de aumento e indicação do valor do aumento, é apresentado ao utilizador uma listagem dos empregados assim como o seu salário atualizado e é também apresentado o valor total dos salários pago (valor atualizado).



Formulário de inserção de empregado:

- Nome:
- Data de Nascimento: ≤ Outubro de 2007 ≥
- se g e r q u a q u i s e x s á b d o m
- 24 25 26 27 28 29 30
- 1 2 3 4 5 6 7
- 8 9 10 11 12 13 14
- 15 16 17 18 19 20 21
- 22 23 24 25 26 27 28
- 29 30 31 1 2 3 4
- Departamento: Informática
- Salário:
- Linguagem:
- Inserir Empregado

Fig. 1



Formulário de aumento de salário:

- Antes Aumenta Depois
- Empregado 1 720
- Empregado 2 880
- Empregado 3 1200
- Empregado 4 1400
- Empregado 5 1400
- Empregado 6 1000
- Empregado 7 850
- Empregado 8 1250
- Empregado 9 1520
- Empregado 10 1400
- ☒ %
- ☐ Valor
-
- O total dos salários é: 11650
- Aumenta

Fig. 2

EXERCÍCIO 7

OBJETIVO: Criar classes que se relacionem entre si através relações de agregação.

ENUNCIADO:

1. Crie uma classe Empregados de forma a que tenha:
 - a. Atributos:
 - i. `private ArrayList _empregados;`
 - ii. `private Empregado _empMaisCaro;`
 - b. Construtores:
 - i. `public Empregados ()`
 - ii. `public Empregados(Empregado novoempregado)`
 - c. Propriedades:
 - i. `public Empregado this[int i]`
para indexar o ArrayList
 - ii. `public Empregado EmpMaisCaro`
só leitura devolve o empregado mais bem pago;
 - iii. `public decimal MediaRemun`
só leitura devolve a média de remunerações pagas;
 - iv. `public int NrEmpregados`
só leitura devolve o número de empregados;
 - d. Métodos:
 - i. `public void AdicionaEmpregado(Empregado novoempregado)`
que permite a inserção de novos empregados
 - ii. `public static Empregado ComparaEmpregados`
`(Empregado empregado1, Empregado empregado2)`
que dados dois empregados devolve o empregado mais bem pago
 - e. Evento:
 - i. `public delegate void delCincoEmpregados();`
`public event delCincoEmpregados eCincoEmpegados;`
que é despoletado assim que a classe tem cinco empregados

EXERCÍCIO 8

OBJETIVO: Criar objetos.

ENUNCIADO:

1. Utilize o `frmInserirEmpregadoEx8` para testar a classe criada no exercício anterior.
 - a. Quando de carga no botão Inserir Empregado é adicionado um empregado a um objeto do tipo `Empregados`, quando o número de elementos criados for 5 é apresentado ao utilizador uma label avisando que já existem 5 empregados criados (evoque o evento `eCincoEmpegados ()`)
2. No formulário `frmEstatisticas.aspx` é apresentado ao Utilizador:
 - a. O total de empregados
 - b. O nome do empregado mais bem pago
 - c. A média de remunerações pagas
 - d. Quando se pressiona o botão `Compara` e após se terem escolhido os empregados (nas combobox respetivas) é apresentado ao utilizador o empregado com o salário mais alto, dos 2 empregados escolhidos.

EXERCÍCIO 9

OBJETIVO: Criar e implementar interfaces.

ENUNCIADO:

1. Implemente a interface `Comparable` na classe `Empregado`, e codifique o método `CompareTo` de modo a que a comparação seja feita pelo salário do empregado.
2. Na classe `Empregados` crie um novo método `Ordena` que permite ordenar o array dos empregados.
3. Crie uma interface - `iEmpregados` - que contenha as seguintes assinaturas de métodos:
 - a. `void RemoveEmpregado(string nomeempregado);`
 - b. `int ExisteEmpregado(string nomeempregado);`
4. Implemente a interface `iEmpregados` na classe `Empregados`
 - a. `RemoveEmpregado`
É removido o empregado que contenha o nome passado no argumento.
 - b. `ExisteEmpregado`
É devolvido o índice do arraylist onde se encontra o empregado cujo nome é passado no argumento, caso não existe o empregado é devolvido -1.

EXERCÍCIO 10

OBJETIVO: Criar objetos.

ENUNCIADO:

1. No formulário frmRemoverEmpregados.aspx são apresentados todos os empregados adicionados
 - a. Quando o botão Remover é pressionado são removidos os empregados selecionados e o utilizador é redirecionado para a Homepage.
 - b. Quando se carrega no botão Ordenar Empregados os empregados inseridos serão apresentados ao utilizador de uma forma ordenada, na caixa de listagem.
2. No formulário frmPesquisarEmpregados.aspx é possível ao utilizador pesquisar um empregado.
 - a. Quando o botão Pesquisar é pressionado é pesquisado o empregado, pelo nome que é colocado na caixa de texto e caso o empregado existir deverá aparecer uma label com o departamento ao qual o mesmo pertence.

EXERCÍCIO 11

OBJETIVO: Criar classes abstratas.

ENUNCIADO:

1. Crie a classe Empresa como uma classe não implementável diretamente.
 - a. Atributos:
 - i. `string _nomeEmpresa;`
 - ii. `byte _NIF;`
garanta que este atributo tem apenas 9 caracteres numéricos
 - iii. `Empregados _empregados;`
 - b. Propriedades:
 - i. `public string NomeEmpresa`
 - ii. `public byte NIF`
 - iii. `public Empregados Empregados`
1. Crie a classe NovaEmpresa que herda a classe Empresa.

EXERCÍCIO 12

OBJETIVO: Utilização de classes.

ENUNCIADO:

1. Utilize o frmEmpresa.aspx para testar a classe criada no exercício anterior.
2. Crie um objeto do tipo NovaEmpresa para teste das funcionalidades da classe NovaEmpresa.
3. Cada um dos botões do formulário permite visualizar a respetiva zona de trabalho e suas funcionalidades.

The image shows a web form with the following elements:

- Two text input fields: "Nome:" and "NIF:", each followed by a text box.
- A button labeled "Inserir" positioned below the "NIF:" field.
- A row of three buttons at the bottom: "Criar Empresa", "Inserir Empregado", and "Ver Empregado".
- A second row of three buttons below the first row: "Remover Empregado", "Pesquisar Empregado", and "Estatística".

EXERCÍCIO 13

OBJETIVO: Criar e utilizar User Controls.

ENUNCIADO:

1. Crie uma nova pasta - UserControl;
2. Crie um novo user control com o nome DataEscolhida.ascx - fig. 1- e adicione-o à pasta criada. O controlo deve ter as seguintes características :
 - a. Atributos:
 - i. `private DateTime _dataInicial`
 - b. Propriedades:
 - i. `public DateTime DataInicial`
 - c. Quando se pressiona o botão Ver, aparecerá por extenso, na lblData, a data colocada em txtData. - fig. 2;
 - d. Se o programador não atribuir um valor à propriedade DataInicial, esta deverá ter o valor da data corrente;
 - e. Se o utilizador não colocar uma data válida, deverá ser apresentado a data corrente.
3. Crie um novo web form com o nome frmTesteUserControl.aspx e adicione-o à pasta criada, atualize o ficheiro web.sitemap.
 - a. Adicione e teste o user control criado anteriormente.

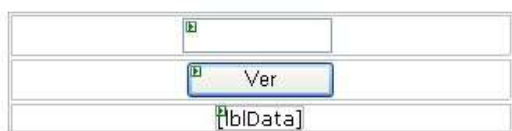


Fig. 1

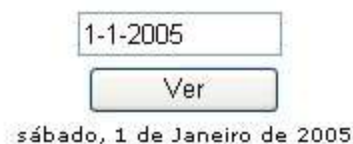


Fig. 2

EXERCÍCIO 14

OBJETIVO: Criar e utilizar User Controls.

ENUNCIADO:



1. Crie um novo user control com o nome `MinhaCaixaTexto.ascx` - fig. 1- e adicione-o à pasta criada. O controlo deve ter as seguintes características:
 - a. Atributos:
 - i. `private int _valorMinimo = 0;`
 - ii. `private int _valorMaximo = 1000;`
 - iii. `private int _valorActual = 0;`
 - b. Propriedades:
 - i. `public int ValorMinimo`
 - ii. `public int ValorMaximo`
 - iii. `public int ValorActual`
 - c. Quando se pressiona o botão  é diminuída uma unidade ao valor apresentado na caixa de texto; se o valor já é o mínimo é apresentado o valor máximo - Figs 2a e 2b.
 - d. Quando se pressiona o botão  é adicionada uma unidade ao valor apresentado na caixa de texto; se o valor já é o máximo é apresentado o valor mínimo - Figs 3a e 3b.
2. Adicione o user control criado anteriormente ao web form `frmTesteUserControl.aspx` e teste-o.



Fig. 1



Figs 2a , 2b



Figs 3a e 3b

EXERCÍCIO 15

OBJETIVO: Utilizar LINQ.

ENUNCIADO:

1. Reescreva os métodos de Pesquisa dos exercícios 9 e 10, utilizando LINQ.