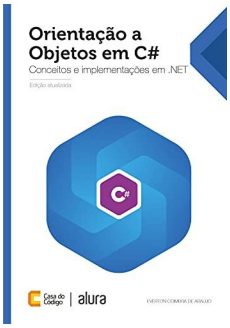

AS NOTAS DO KINDLE PARA:

Orientação a Objetos em C#: Conceitos e implementações em .NET

de Everton Coimbra de Araújo

Visualização instantânea gratuita do Kindle: https://a.co/7w7Hrb3

88 destaques

Destaque (Amarelo) | Posição 80

> Programação orientada a objetos (POO) é um paradigma que viabiliza o desenvolvimento de aplicações fazendo uso do modelo orientado a objetos.

Destaque (Amarelo) | Posição 81

> modelo orientado a objetos o conjunto de "coisas" que fazem parte

Destaque (Amarelo) | Posição 81

> do contexto atual em estudo,

Destaque (Amarelo) | Posição 82

> normalmente os objetos encontrados na análise se agrupam em classes.

Destaque (Amarelo) | Posição 82

> Estas classes, em conjunto com as associações entre elas, definem a estrutura do sistema em estudo. Uma das principais características da POO é a capacidade de reutilização, ou seja, de otimização da produtividade, aumento de qualidade, diminuição de tempo e custos de manutenção.

Destaque (Amarelo) | Posição 86

> Outro fator considerado como vantagem é a manutenibilidade,

Destaque (Amarelo) | Posição 87

> facilidade na manutenção dos projetos.

Destaque (Amarelo) | Posição 87

> Este fator depende de como o sistema foi estruturado e das técnicas de programação que foram usadas ao longo do desenvolvimento.

Destaque (Amarelo) | Posição 108

> A abstração é muito importante no processo de modelagem e implementação de soluções orientadas a objeto.

Destaque (Amarelo) | Posição 109

> Ela é considerada como a habilidade de modelar características do mundo real, de um denominado problema em questão que o programador esteja tentando resolver.

Destaque (Amarelo) | Posição 114

> Objeto é qualquer estrutura modular que faz parte de algo.

Destaque (Amarelo) | Posição 116

> Uma janela, por exemplo, é um objeto de uma casa, de um carro ou de um software com interface gráfica para o usuário.

Destaque (Amarelo) | Posição 117

> Cada objeto possui propriedades, comportamentos e métodos que o identificam e o diferenciam dentre outros objetos semelhantes. Propriedades consistem de características atribuídas aos objetos.

Destaque (Amarelo) | Posição 121

> Comportamento e eventos referem-se às ações (operações) aplicadas por um objeto ou suas reações (eventos).

Destaque (Amarelo) | Posição 124

> Os comportamentos e eventos mapeados para uma linguagem são conhecidos como a implementação dos métodos.

Destaque (Amarelo) | Posição 134

> Classe é um conjunto ou uma categoria de objetos que tem propriedades e métodos.

Destaque (Amarelo) | Posição 138

> Na OO, chamamos esta categorização de generalização e especialização respectivamente, e ambas as denominações estão relacionadas à herança.

Destaque (Amarelo) | Posição 140

herança é um termo que se refere a uma classe nova, que pode ser criada a partir de outra já existente.

Destaque (Amarelo) | Posição 141

ela pode "herdar" atributos e comportamentos da classe a ser estendida.

Destaque (Amarelo) | Posição 143

(a herança pode ser de características e de comportamentos).

Destaque (Amarelo) | Posição 145

A classe na qual é gerada a nova classe é chamada de classe

Destaque (Amarelo) | Posição 145

básica (ancestral, superclasse ou classe genérica) e a nova é chamada de classe derivada (descendente, subclasse, ou ainda, classe especializada).

Destaque (Amarelo) | Posição 147

a classe derivada herdará os atributos e comportamentos da básica.

Destaque (Amarelo) | Posição 156

polimorfismo.

Destaque (Amarelo) | Posição 160

Polimorfismo, que significa “várias formas”, é um termo designado a objetos de classes distintas, que podem reagir de uma maneira diferente ao fazerem o uso de um mesmo

---

Destaque (Amarelo) | Posição 161

método.

Destaque (Amarelo) | Posição 166

Encapsulamento é um mecanismo que trata de dar segurança aos objetos, mantendo-os controlados em relação

---

Destaque (Amarelo) | Posição 167

ao seu nível de acesso.

Destaque (Amarelo) | Posição 167

> o encapsulamento contribui fundamentalmente para diminuir os malefícios causados pela interferência externa sobre os dados, pois isola partes do código.

Destaque (Amarelo) | Posição 184

> Tudo isso são classes associadas. Interface é um tipo de classe que contém apenas assinaturas de métodos.

Destaque (Amarelo) | Posição 188

> Associação é o mecanismo que representa um relacionamento/ conexão entre objetos, definindo as dependências e as ligações entre eles, ou seja, o que podem utilizar de recursos de um para outro.

Destaque (Amarelo) | Posição 207

> coesão, ou seja, é importante que cada unidade, método ou classe realize apenas o que for de sua responsabilidade.

Destaque (Amarelo) | Posição 219

> Em OO, quando duas classes se associam, elas estão acopladas.

Destaque (Amarelo) | Posição 219

> discutirmos a qualidade deste acoplamento, pois quando este existe, também começa a existir uma dependência entre classes, e isso pode não ser bom.

Destaque (Amarelo) | Posição 220

> Quando existe uma alta dependência, o acoplamento também é alto, e isso é ruim.

Destaque (Amarelo) | Posição 221

> Um sintoma de alto acoplamento é a contestação de uma classe depender de várias outras.

Destaque (Amarelo) | Posição 222

> imagine a classe associada sofrendo alterações, logo, todas que dependem dela precisarão sofrer alterações também para se adequarem.

Destaque (Amarelo) | Posição 223

> O ideal é manter o foco em interfaces, desenvolver orientado a interfaces.

Destaque (Amarelo) | Posição 224

> como uma interface normalmente não sofre alterações, a associação não causa alta dependência, mantendo, assim, um baixo acoplamento.

Destaque (Amarelo) | Posição 249

> software deve se iniciar a partir de um estudo aprofundado do problema, ou seja, uma análise. Não importa a técnica ou tecnologia a ser utilizada, esta fase é extremamente importante e necessária.

Destaque (Amarelo) | Posição 271

> uma classe tem a finalidade de tipificar um objeto que possa ser usado em uma aplicação.

Destaque (Amarelo) | Posição 272

> uma classe é um Tipo Abstrato de Dados (TAD).

Destaque (Amarelo) | Posição 273

> Um TAD pode ser definido como a especificação de um conjunto de dados e das operações que podem ser executadas sobre este conjunto.

Destaque (Amarelo) | Posição 276

> Um objeto com valores atribuídos às suas características define o que é conhecido como Estado do objeto,

---

Destaque (Amarelo) | Posição 277

> É muito comum e mais correto dizermos que um objeto é uma instância de uma classe.

Destaque (Amarelo) | Posição 282

> Quando trabalhamos bem com OO, não saímos codificando logo de cara, é preciso antes um processo de análise e modelagem.

Destaque (Amarelo) | Posição 293

> A UML é uma linguagem de modelagem que oferece diversos tipos de diagramas, que podem representar todo o sistema a ser desenvolvido, como também detalhes mais específicos de requisitos levantados.

Destaque (Amarelo) | Posição 317

> Por convenção, é ideal que cada arquivo possua uma única classe e, ainda, que o nome do arquivo e da classe sejam iguais.

Destaque (Amarelo) | Posição 319

> não é uma regra.

Destaque (Amarelo) | Posição 333

> Um tipo de dado que inicia com letra maiúscula caracteriza que se refere a uma classe, e um tipo que inicia com minúscula refere-se a um tipo de dado primitivo à linguagem.

Destaque (Amarelo) | Posição 345

> Para o projeto que estará contido na solução, escolheremos o template Aplicativo de Console

---

Destaque (Amarelo) | Posição 349

> O VS permite organizar seus projetos em uma solução, que é também um tipo de projeto.

Destaque (Amarelo) | Posição 350

> é possível termos vários projetos em uma mesma solução.

Destaque (Amarelo) | Posição 351

> Nomeie corretamente sua solução e seu projeto. Uma dica seria Capitulo02 e PrimeiroProjeto.

Destaque (Amarelo) | Posição 363

> atribua o nome a essa classe, que em nosso caso é Instituicao, sem caracteres especiais

---

Destaque (Amarelo) | Posição 364

> A extensão do arquivo que conterá a classe de mesmo nome é cs.

Destaque (Amarelo) | Posição 375

> as primeiras instruções (using) referem-se a Namespaces necessários para a implementação das classes.

Destaque (Amarelo) | Posição 379

> Namespace é o nome dado para uma estrutura lógica, responsável por organizar classes, como se fossem pastas físicas, o que permite que você tenha classes com nomes iguais em locais diferentes.

Destaque (Amarelo) | Posição 380

> Sua real importância está na separação entre domínios para as classes.

Destaque (Amarelo) | Posição 381

> Poderíamos ter, por exemplo, um namespace chamado Negocio, outro chamado Controle, outro chamado Visão e, dentro deles, as classes que dizem respeito a estas áreas (camadas no exemplo).

Destaque (Amarelo) | Posição 385

> quando o cursor estiver em uma linha de using que não é necessária para sua classe, um botão com uma lâmpada será exibido,

---

Destaque (Amarelo) | Posição 387

> o Uso das diretivas é desnecessário. Clique na setinha ao lado do ícone e clique na opção Remover usos desnecessários

---

Destaque (Amarelo) | Posição 391

> o nome qualificado da classe é PrimeiroProjeto.Instituicao

Destaque (Amarelo) | Posição 393

> Nome qualificado Toda classe possui um nome que a identifica como um tipo de dado.

Destaque (Amarelo) | Posição 394

> podemos ter classes de mesmo nome, mas com responsabilidades diferentes,

---

Destaque (Amarelo) | Posição 394

> o que leva ao uso de namespaces.

Destaque (Amarelo) | Posição 395

> nome qualificado,

Destaque (Amarelo) | Posição 395

> nome da classe precedido pelo nome do namespace em que ela está.

Destaque (Amarelo) | Posição 396

> Quando usamos métodos de outra classe (classe B) em uma classe (classe A) que não esteja no mesmo namespace, precisamos inserir uma instrução using do namespace onde a classe B esteja, ou usamos a referência à classe de maneira qualificada.

Destaque (Amarelo) | Posição 409

> Uma aplicação do tipo console, como seu próprio diz, é executada no console do sistema operacional, não gerando nenhum tipo de interface amigável com o usuário.

Destaque (Amarelo) | Posição 411

> A classe Program é definida no projeto como classe de inicialização da aplicação, e o Main() é o método estático definido como o método que será executado quando a aplicação for inicializada.

Destaque (Amarelo) | Posição 420

> um erro de compilação pudesse ser visualizado. Veja no local indicado que a palavra nome, que se refere ao campo que deveremos atribuir ou recuperar o nome de uma instituição, está com um sublinhado vermelho.

Destaque (Amarelo) | Posição 422

> Sempre que um código seu apontar este tipo de erro, coloque o cursor do mouse sobre o erro e uma mensagem orientativa será exibida.

Destaque (Amarelo) | Posição 423

> Em nosso caso, a mensagem é: 'Instituicao.nome' é inacessível devido ao seu nível de proteção

---

Destaque (Amarelo) | Posição 425

> Isso significa que o escopo definido para a propriedade nome não permite que ela seja utilizada na classe Program.

Destaque (Amarelo) | Posição 428

> Um campo de uma classe não pode ser acessado diretamente de seus objetos, pois, para isso, precisamos de métodos que ofereçam este serviço.

Destaque (Amarelo) | Posição 439

> inserção de um método chamado Nome(), que recebe uma string como argumento.

Destaque (Amarelo) | Posição 440

> O valor recebido é atribuído ao campo nome do objeto referente à chamada ao método. Isso é garantido pelo this. Para que isso funcione, precisamos alterar também o consumidor do serviço, nossa classe Program.

Destaque (Amarelo) | Posição 446

> instituicao.Nome(System.Console.ReadLine()); Note que, com a implementação anterior, mudamos a maneira de atualização do campo.

Destaque (Amarelo) | Posição 447

> Antes estava com uma simples atribuição e agora está com a chamada para um método.

Destaque (Amarelo) | Posição 448

> Precisaríamos de outro método, para recuperação do valor. Essa maneira é muito trabalhosa. Para cada campo, seriam necessários dois métodos para que um campo pudesse ter seu valor recuperado ou atribuído.

Destaque (Amarelo) | Posição 450

> Sempre que tivermos uma propriedade (campo com acesso de escrita e leitura), podemos declarar isso diretamente.

Destaque (Amarelo) | Posição 455

> A troca para a letra maiúscula não é uma regra. O nome poderia ficar com letra minúscula no início, mas é uma convenção, conhecida como Camel Case e é adotada pela comunidade desenvolvedora do C#.

Destaque (Amarelo) | Posição 467

> Caso você queira que uma propriedade seja apenas de leitura, você pode deixar de declarar a cláusula set;

Destaque (Amarelo) | Posição 503

> De maneira bem simplista, é isso que um método estático oferece; ele pode ser invocado diretamente, sem a necessidade de um objeto.

Destaque (Amarelo) | Posição 504

> Isso significa também que, em caso de termos propriedades estáticas, o valor delas é compartilhado por todos os objetos da classe.

Destaque (Amarelo) | Posição 517

> Interpolação de strings No código anterior, fiz uso do $"" para que uma string pudesse ter a inserção de uma propriedade em seu retorno.

> O processo de instanciação se dá pela cláusula new, que precede o nome da classe.