Treinamento C#

Caderno de Exercícios

Considerações Gerais

Todos os artefatos que serão desenvolvidos durante o treinamento de C# (exercícios e avaliações) deverão obedecer à norma técnica de nomenclatura, a PXNX01N. Esta regra tem como objetivo padronizar nomes de artefatos de programas de acordo com sua natureza e funcionalidade.

Utilize o a sigla de sistema **PXC** para desenvolver os exercícios propostos.

Desenvolva TODOS os exercícios na pasta "c:\Desenvhome-pxc\ nomeModulo". Esta é a pasta padrão onde os programadores desenvolvem suas aplicações durante o treinamento de C#. Esta pasta é somente para o treinamento de C#, para não confundir com o repositório dos sistemas do Banrisul que é "c:\Desenvhome".

Tenha o cuidado de expor apenas os métodos necessários de cada classe. Preocupe-se em abstrair os detalhes de implementação de cada camada.

Utilize os padrões abaixo para nomear os elementos conforme a tabela 1.

- **Pascal Case:** A primeira letra de cada palavra do nome é maiúscula, as demais são minúsculas.
- **Camel Case:** Idêntico ao Pascal Case, porém a primeira letra do nome também é minúscula.
- **Upper Case:** Todas as letras são maiúsculas. As palavras do nome são separadas por underscores.

Elemento	Formato	Exemplo
Espaços de Nome	Pascal Case	namespace **NomeNamespace {}**
Classes	Pascal Case	public class **NomeClasse {}**
Estruturas	Pascal Case	public struct **NomeEstrutura{}**
Interfaces	Prefixo "I" + Pascal Case	public interface **INomeInterface {}**
Enumerações	Pascal Case	public enum **NomeEnumeracao** **{** **NomePrimeiroElemento**,
Valores de enumerações	Pascal Case	
Eventos	Pascal Case	public event System.EventHandler **NomeEvento;**
Métodos	Pascal Case	public void **NomeMetodo**() {}

Propriedades	Pascal Case	public System.Int32 **NomePropriedade**
		{
		get {}
		set {}
		}

Argumentos	Camel Case	public void Metodo(System.Int32 n**omeArgumento**) {}
Atributos	Camel Case	private System.Int32 **nomeAtributo**;
Variáveis locais	Camel Case	System.Int32 **nomeVariavel;**
Constantes	Upper Case	private const System.Int32 **NOME_CONSTANTE** = 1;

Tabela 1 – Tabela de utilização dos padrões de nomenclaturas nos elementos codificados

Para propriedades e métodos (principalmente os públicos), sempre utilize as "Seções de documentação XML". É muito importante preenchê-las de forma sucinta e completa, pois é possível extrair documentações dos componentes através destas informações.

Sempre que possível, organize o seu código com a diretiva *#region.* Como sugestão, utilize-a para separar Atributos, Propriedades, Construtores, Métodos, Operadores, Eventos (membros event e não métodos que atuam como event handlers), Tipos aninhados...

Exercício 2.1 – Utilizando variáveis, operadores e conversões

1. Inicie o Visual Studio.NET
2. Crie um novo programa do tipo Console Application
3. Faça com que o programa peça 2 valores ao usuário do seguinte modo:

```
Bem-vindo <nome do usuário> agora são <hora atual>.
Digite o 1º valor para a soma:
Digite o 2º valor para a soma:
```

4. Faça com que o programa exiba o resultado do seguinte modo:

```
<nome do usuário>, o resultado da operação é <resultado>.
```

Exercício 2.2 – Implementando Classes e utilizando construtores

1. Inicie o Visual Studio.NET
2. Crie um novo programa do tipo Console Application
3. Crie uma classe chamada Calculadora
4. Crie uma variável privada chamada valorA do tipo Double
5. Crie uma variável privada chamada valorB do tipo Double
6. Implemente um método para cada uma das 4 operações básicas (Somar, Subtrair, Multiplicar e Dividir)
7. Crie o método construtor da classe Calculadora que receba 2 valores do tipo Double e os passe para as variáveis valorA e valorB
8. Solicite ao usuário que digite 2 valores, para os campos valorA e valorB
9. Na função Main da classe Program, instancie um objeto do tipo Calculadora, passando como parâmetros para o método construtor os valores digitados pelo usuário
10. Exiba para o usuário o resultado das 4 operações básicas.

Exercício 2.3 – Criando enumeradores

1. No projeto utilizado no exercício 2.2, crie um enumerador chamado Operacao contendo os valores Somar, Subtrair, Multiplicar e Dividir
2. Altere a Main da classe program para que o usuário selecione qual operação deseja realizar
3. Faça com que o programa realize e exiba somente o resultado da operação selecionada
4. Execute o programa e teste novamente.

Exercício 2.4 – Trabalhando com Arrays, Comandos de Decisão e Repetição

1. No projeto utilizado no exercício 2.3, altere o enumerador chamado Operacao, adicionando o valor Exibir Log
2. Altere a construtora da classe para não receber parâmetros.
3. Crie as propriedades ValorA e ValorB com get e set.
4. Faça com que o programa, ao terminar a execução de cada operação, pergunte se o usuário deseja continuar na aplicação ou sair.
5. Na classe Calculadora, crie uma propriedade somente de leitura do tipo Arraylist chamada Log.
6. Faça com que, a cada operação realizada, seja adicionada na propriedade Log uma string no seguinte formato:

A <operação> de <valorA> e/por <valorB> é <resultado>.

7. Faça com que, quando o usuário selecionar a opção Exibir Log, sejam exibidas todas as entradas do ArrayList Log ou a mensagem “Nenhuma operação realizada.”.

Exercício 2.5 – Utilizando funções de String e a classe StringBuilder

1. No projeto utilizado no exercício 2.4, altere os métodos de operação para que ao realizar uma operação, troque o ArrayList para StringBuilder no Log
2. Utilize a função String.Format() ao adicionar no StringBuilder do Log
3. Execute o programa e teste novamente.

Exercício 3.1 – Entendendo a estrutura de classes do C#

1. Inicie o Visual Studio.NET
2. Crie um novo programa do tipo Console Application
3. Crie uma classe abstrata chamada FiguraGeometrica, com uma variável protegida do tipo double chamada “área” e um método abstrato chamado CalcArea do tipo void. Crie uma propriedade somente leitura “Area” para essa variável.
4. Crie uma classe chamada Retangulo que herde de FiguraGeometrica, com duas variáveis protegidas chamadas largura e altura do tipo Double.
5. Faça com que o construtor da classe receba os valores para os campos largura e altura.
6. Sobrescreva o método CalcArea, que implemente o cálculo da área do retângulo (área = largura * altura) e ponha o resultado na variável área.
7. Crie uma classe chamada Quadrado que herde de Retangulo.
8. Faça com que o construtor da classe receba apenas 1 valor para o campo lado do tipo Double.
9. Crie uma classe chamada Elipse que herde de FiguraGeometrica, com duas variáveis protegidas chamadas raio1 e raio2 do tipo Double.
10. Faça com que o construtor da classe receba os valores para os campos raio1 e raio2.
11. Sobrescreva o método CalcArea, que implemente o cálculo da área da elipse (área = "PI" * ((r1 + r2) / 2) ^ 2 e ponha o resultado na variável área.
12. Crie uma classe chamada Circulo que herde de Elipse.
13. Faça com que o construtor da classe receba apenas 1 valor para o campo raio do tipo Double.
14. Faça com que o programa apresente a opção de cálculo de área para cada um dos tipos acima.

Exercício 3.2 – Entendendo a estrutura de classes do C#

1. Modifique o exercício 3.1 incluindo um ArrayList figuras. Toda vez que for instanciado um objeto da hierarquia de FiguraGeometrica, adicione o mesmo no ArrayList.
2. Modifique a classe Retangulo incluindo o método Perimetro público do tipo Double. (2 * lado1 + 2 * lado2)
3. Na saída do programa, percorra a lista exibindo todos os cálculos de área realizados e indique o tipo do objeto (Quadrado, Retângulo, Elipse, Círculo). Caso seja um objeto da hierarquia de Retangulo, então execute o método Perimetro mostrando na tela o perímetro do objeto.

Exercício 4.1 – Utilizando Dispose e a interface IDisposable

1. No projeto utilizado no exercício 2.5, faça com que a classe Calculadora implemente a interface IDisposable
2. Crie o método Dispose(), e faça com que nele seja atribuído 0 para os campos valorA e valorB e o ArrayList Log seja limpo
3. Execute o programa e teste novamente

Exercício 5.1 – Tratamento estruturado de exceções

1. No projeto utilizado no exercício 4.1, crie um tratamento de exceções na classe Program, exibindo a cada erro a mensagem trazida pela exceção na tela.
2. Faça com que a aplicação trate alguma exceção que poderá ser lançada ao tentar pegar algum elemento que não exista no ArrayList Log
3. Faça com que a aplicação trate alguma exceção que poderá ser lançada ao tentar converter tipos de dados incompatíveis
4. Altere o programa, utilizando o bloco *finally* para exibir a mensagem perguntando se o usuário deseja continuar ou sair da aplicação.
5. Execute o programa e teste novamente.

Exercício 6.1 – Delegates e Eventos

1. No projeto utilizado no exercício 5.1, crie um delegate chamado DelegateOperacao, do tipo Double que não receba parâmetros
2. Faça com que, ao usuário selecionar a operação a ser realizada, seja direcionado ao DelegateOperacao o método correspondente da classe Calculadora
3. Execute o delegate para realizar a operação
4. Após, execute o programa e teste novamente.

Exercício 7.1 – Entendendo Generics

1. Crie uma propriedade abstrata chamada Lados dentro da classe FiguraGeometrica.
2. Implemente a propriedade nas classes que herdam e que retorne a quantidade de lados da respectiva figura.
3. Altere a classe Main do exercício 3.1 para que cada vez que for instanciado uma figura geométrica, a mesma seja armazenada em um List<FiguraGeometrica>.
4. Crie uma opção no menu para mostrar a quantidade de lado das figuras adicionadas.

Exercício 8.1 – .Net Assembly

1. Modifique o exercício 7.1 criando um novo projeto Class Library para as classes de FiguraGeometrica.
2. Referencie no projeto da Console Application, a nova DLL criada e faça funcionar.

Exercício 10.1 – Explorando menu e leitura de dados com a Pxcoiexn.dll

1. Modifique o exercício anterior e faça o menu utilizando o método ControlaMenu e as classe Tela e Menu.
2. Modifique a leitura das informações utilizando o método Ler da classe Tela.
3. Troque a solicitação de confirmação de continuação ao final do programa pelo método Confirma da classe Tela.
4. Crie um novo item de menu e utilize o método ImprimeLista para imprimir a lista de figuras geométricas para listar o tipo da figura, a quantidade de lados, a área e o perímetro no caso de retângulo.