APLICAÇÃO DOS PILARES DA PROGRAMAÇÃO ORIENTADA A OBJETOS EM UM SISTEMA DE GERENCIAMENTO DE ZOOLÓGICO

RESUMO

Este artigo apresenta a aplicação prática dos pilares da Programação Orientada a Objetos (POO) por meio do desenvolvimento de um sistema simples de gerenciamento de zoológico. O sistema contempla o cadastro de animais (mamíferos, aves e répteis) e funcionários (veterinários e zeladores), além da interação entre eles. São abordados os principais conceitos da POO: encapsulamento, herança, polimorfismo e abstração. Também são utilizados recursos como construtores, sobrecarga, propriedades, métodos, interfaces, classes e métodos abstratos.

---

1. INTRODUÇÃO

A Programação Orientada a Objetos (POO) é um paradigma essencial para o desenvolvimento de sistemas modernos. Ela permite a modelagem de sistemas complexos com base em objetos do mundo real. Este artigo explora os pilares da POO implementados em um sistema que simula o gerenciamento de um zoológico.

---

2. DESENVOLVIMENTO

2.1 Encapsulamento

O encapsulamento é aplicado por meio do uso de propriedades com modificadores de acesso. Exemplo:

```
public string Nome { get; set; }
public int Idade { get; set; }
```

Essas propriedades estão presentes nas classes `Animal` e `Funcionario`, permitindo controlar o acesso aos dados internos das classes.

2.2 Herança

A herança é utilizada para generalizar comportamentos comuns e permitir especialização. A classe `Animal` é uma classe base abstrata que é herdada por `Mamifero`, `Ave` e `Reptil`.

```
public abstract class Animal
{
   // propriedades e métodos comuns
}

public class Mamifero : Animal
{
   // implementação específica
}
```

O mesmo ocorre com a classe `Funcionario`, herdada por `Veterinario` e `Zelador`.

2.3 Polimorfismo

O polimorfismo permite que métodos sejam sobrescritos em classes derivadas, adaptando o comportamento conforme o tipo do objeto.

```
public abstract void EmitirSom();
public override void EmitirSom()
{
   Console.WriteLine($"{Nome} está emitindo sons(Piu).");
}
```

Métodos como `EmitirSom` e `Movimentar` têm diferentes implementações em `Mamifero`, `Ave` e `Reptil`.

2.4 Abstração

A abstração foi utilizada com classes e métodos abstratos. A classe `Animal` define métodos abstratos que devem ser implementados por todas as subclasses:

```
public abstract class Animal
{
   public abstract void EmitirSom();
   public abstract void Movimentar();
}
```

Isso força cada tipo de animal a definir seu próprio comportamento.

---

3. OUTROS CONCEITOS

3.1 Construtores

Todas as classes utilizam construtores para garantir a inicialização correta de seus objetos:

```
public Mamifero(string nome, int idade, double peso, string especie)
    : base(nome, idade, peso, especie) { }
```

3.2 Sobreposição de Métodos

As classes derivadas implementam seus próprios comportamentos com `override`, como nos métodos `EmitirSom()` e `Movimentar()`.

3.3 Interfaces

As interfaces `ICuidador` e `ITratamentoAnimal` são utilizadas para definir comportamentos específicos de certos funcionários:

```
public interface ICuidador
{
    void CuidarHabitat(Animal animal);
}
```

A classe `Zelador` implementa `ICuidador`, enquanto `Veterinario` implementa `ITratamentoAnimal`.

---

4. FUNCIONALIDADES DO SISTEMA

- **Cadastro de Animais**: usuário pode cadastrar mamíferos, aves ou répteis.
- **Cadastro de Funcionários**: cadastro de veterinários ou zeladores.
- **Interações**: veterinários podem consultar e tratar animais; zeladores podem alimentar e cuidar do habitat.

Exemplo de saída do sistema:

```
Veterinário João consultou o animal Simba com sucesso.
Zelador Marcos alimentou o animal Simba com sucesso.
```

5. CONCLUSÃO

O sistema de gerenciamento de zoológico desenvolvido demonstra de forma prática a aplicação dos pilares da Programação Orientada a Objetos, promovendo uma estrutura de código clara, reutilizável e extensível. Através do uso de herança, encapsulamento, polimorfismo, abstração e outros recursos, foi possível construir um sistema funcional e didático.

REFERÊNCIAS

- MICROSOFT. *Programação orientada a objetos (OOP) em C#*. Microsoft Learn. Disponível em: https://learn.microsoft.com/pt-br/dotnet/csharp/fundamentals/tutorials/oop.
- MICROSOFT. *Guia de programação C#: classes e objetos*. Microsoft Learn. Disponível em: https://learn.microsoft.com/pt-br/dotnet/csharp/programming-guide/classes-and-structs/.
- MICROSOFT. *Fundamentos da linguagem C#*. Microsoft Learn. Disponível em: https://learn.microsoft.com/pt-br/dotnet/csharp/fundamentals/.