**Guia de Implementação**

SUMÁRIO

[1. Introdução 2](#_Toc485801465)

[1.1. Referências 2](#_Toc485801466)

[2. Organização e Estilo do Código 2](#_Toc485801467)

[3. Comentários 2](#_Toc485801468)

[4. Nomeação 2](#_Toc485801469)

[5. Declaração 3](#_Toc485801470)

[6. Reutilização 3](#_Toc485801471)

[7. Tratamento de Erros e de Exceções 3](#_Toc485801472)

[8. teste unitário 3](#_Toc485801473)

[9. Diretrizes Gerais 3](#_Toc485801474)

# Introdução

Este documento fornece um padrão de codificação para o projeto DDPJ (Documentação e Design Patterns para Jogos) e como este padrão será implementado no jogo exemplo GladArena.

O jogo GladArena é um jogo de batalha de gladiadores no qual a jogabilidade depende da criação da estratégia no início da partida através de fluxograma; após isto, o jogador assiste seu gladiador obedecendo suas estratégias.

## Referências

* Guia – C# coding convetions;

# Organização e Estilo do Código

* Tamanho de classe: não mais que 1000 linhas
* Tamanho demétodo: não mais que 40 linhas

# Comentários

Comentário no início das classes [exceto em classes de Tranfer Object]

//

// <Descrição da funcionalidade da Classe>

// @author: <Fulano da Silva>

//

Comentários dos métodos [exceto em classes de Transfer Object]

//

// <Descrição completa das funcionalidades do método>

// @return <tipo do retorno do método: String, int, etc>

// @param <parâmetro1> <Descrição do parâmetro>

// @param <parâmetro2> <Descrição do parâmetro>

// @exception <exceção retornada>

//

# Nomeação

Classe de Transfer Object

* NomeClasse.cs

Demais classes

* PrefixoCasoDeUsoSufixoNomePacote.cs (Ex.: ManterEstrategiaDao).

Arquivo PNG

* NomeSprite.png

Arquivo Animação

* NomeAnimacao.anim

Scripts SQL

* NOMESCRIPT.sql

# Declaração

1. Padrão de indentação de declaração de métodos:

someMethod(int anArg, Object anotherArg, String yetAnotherArg,

Object andStillAnother) {

...

}

# Reutilização

* Criar interfaces entre as camadas arquiteturais: Apresentação, Controle e Dados;
* Aplicar os padrões de atribuição de responsabilidade nas classes (ver padrões GRASP).

# Tratamento de Erros e de Exceções

* Criar uma casse de tratamento de exceções na camada de apresentação.
* Garantir que as exceções subam para a camada de apresentação, usando a clausula Throw nos métodos envolvidos.

# teste unitário

* Criar um pacote que exiba a quantidade de frames for segundo em partida.

# Diretrizes Gerais

As classes em Unity devem estar vinculadas em algum objeto de jogo. Se estas não estiverem, devem ser implementadas algumas estratégias para que se possa fazer as relações entre classes.

* Referências de classes de objetos de jogo que estão no mesmo pacote e na mesma cena, serão feitas desta forma:

NomeClasse nomeObjeto = FindObjectOfType (typeof(NomeClasse)) as NomeClasse;

* Demais referências devem ser desta forma:

NomeClasse nomeObjeto = gameObject.AddComponent <NomeClasse> ();