

*ISEC - I See, Eu Conquisto!*

*Programação Orientada a Objetos – Ano Letivo 2020/2021*

Por João Oliveira e José Marques.

# Índice:

[Índice:](#_heading=h.gjdgxs) **2**

[Organização do Código](#_heading=h.30j0zll) **3**

[Introdução](#_heading=h.1fob9te) 3

[Classes implementadas](#_heading=h.3znysh7) 4

[Classes/conceitos encontradas no enunciado](#_heading=h.j1yyv9nf0vno) 4

[Classes onde são criados/destruídos objetos](#_heading=h.px94voz97ykf) 5

[Classes com responsabilidades e encapsulamentos](#_heading=h.vnvf8kjy33od) 5

[Classes com objetivos focados](#_heading=h.qvk6merhw22j) 5

[Classes com responsabilidades de interface com o utilizador com e sem lógica](#_heading=h.2671fgtrvsoy) 5

[Primeiro objecto para além da camada de interacção com o utilizador](#_heading=h.mdpougxyzd47) 5

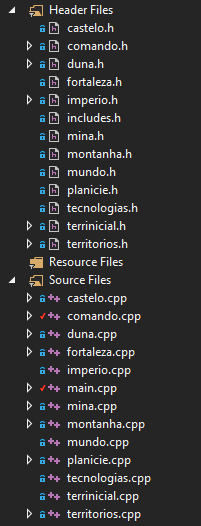
[A classe que representa a envolvente de toda a lógica, executa em pormenor muitas funcionalidades, e delega noutras classes](#_heading=h.m6b6nnn8jqe4) 5

[Uma funcionalidade que varia conforme o tipo do objecto que a invoca. Indique em que classes e métodos está implementada esta funcionalidade. (Não é necessário responder a esta pergunta na meta 1).](#_heading=h.s3bw5nm65v82) 6

[Apresente as principais classes da aplicação através da seguinte informação: 2 a. Classe: nome da classe b. Responsabilidades: i. o que permitem consultar ii. o que permitem fazer (Neste ponto pede-se uma descrição resumida e objectiva das responsabilidades das classes). c. Colaborações: as classes com que colaboram (por exemplo, utilizando objectos, tendo ponteiros, invocando funções)](#_heading=h.3yja5lyqadvc) 6

# Organização do Código

## Introdução



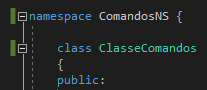
Apesar de ser apenas a primeira meta, decidimos adiantar já pelo menos o processo da criação tanto dos *headers* necessários como dos *source* *files*. Há uma enorme probabilidade de haver mudanças ainda, mas até agora, são esses os ficheiros criados.

Quero dizer já que o programa foi feito no standard C++14 para não haver conflito com os conteúdos lecionados nas aulas.

Vou entrar mais em detalhe nas classes criadas até agora.

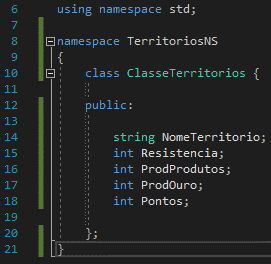
## Classes implementadas

A primeira classe de que vou falar é a Classe *Comandos:*



A classe “*comandos*” é a mais importante do programa até agora, pois é ela segura as funções principais do programa, é a responsável por chamar a criação de terrenos, manipular os ficheiros – tanto para gravar como para abrir e retornar o progresso do jogador, e de listar o progresso do jogador.

Outra classe que temos de igual importância é a classe *Territórios:*



Onde todos os territórios que criamos usam as variáveis que os definem. Esta classe é crucial na criação de tipos de territórios, sendo a origem para classes derivadas para uma próxima fase do projeto.

## Classes/conceitos encontradas no enunciado

Os conceitos que identificámos foram:

* *Comandos*
* *Territórios*
* *Território Inicial*
* *Castelo*
* *Duna*
* *Fortaleza*
* *Império*
* *Mina*
* *Montanha*
* *Planície*
* *Tecnologias*

## Classes onde são criados/destruídos objetos

**Territórios**: os objetos desta classe são criados e destruídos na classe Comandos.

**Comandos**: os objetos desta classe são criados e destruídos na classe Comandos.

## Classes com responsabilidades e encapsulamentos

A responsabilidade “criar territórios” está atribuída à classe **ClasseComandos** porque esta tem a coleção de comandos.

## Classes com objetivos focados

**Classe Territórios**: Tem responsabilidade relativa à criação dos territórios (*construtor*) e dado que pertencem a cada território.

## Classes com responsabilidades de interface com o utilizador com e sem lógica

Responsabilidade de interface: main.cpp

Responsabilidade de lógica da aplicação: Comandos, Territórios

## Primeiro objecto para além da camada de interacção com o utilizador

As ordens vindas da camada de interação com o utilizador são recebidas no *main* e processadas por um objeto da **classe Comandos**.

## A classe que representa a envolvente de toda a lógica, executa em pormenor muitas funcionalidades, e delega noutras classes

A classe **Comandos** representa a envolvente de toda a lógica. Para criar todos os territórios, delega à classe **Territórios** para que esta os crie.

## Uma funcionalidade que varia conforme o tipo do objecto que a invoca. Indique em que classes e métodos está implementada esta funcionalidade. (Não é necessário responder a esta pergunta na meta 1).

## Apresente as principais classes da aplicação através da seguinte informação: 2 a. Classe: nome da classe b. Responsabilidades: i. o que permitem consultar ii. o que permitem fazer (Neste ponto pede-se uma descrição resumida e objectiva das responsabilidades das classes). c. Colaborações: as classes com que colaboram (por exemplo, utilizando objectos, tendo ponteiros, invocando funções)

**Classe: Territórios**

Responsabilidades:

*Criação de atributos para os territórios*

*Criação de objetos Território*

*Colaboração:*

*Comando*

**Classe: Comando**

Responsabilidades:

*Criação e conquista de territórios*

*Gravar e carregar ficheiros*

*Listar progresso*