isec Engenharia

Programação Orientada a Objetos

Ano Letivo 2020/2021

Empire

Relatório do Trabalho Prático

Ana Rita Santos Videira – 2015012218
a21250074@isec.pt

ÍNDICE

INTRODUÇÃO
ORGANIZAÇÃO E PRINCIPAIS CLASSES E CONCEITOS IDENTIFICADOS

Na sequência da unidade curricular Programação Orientada a Objetos foi proposto a elaboração de um pequeno jogo de forma a desenvolver e melhorar competências em linguagem C++.

De acordo com o enunciado proposto, o jogo a implementar consiste num mundo com vários elementos: territórios a conquistar e seus produtos, armas e tecnologias. Este jogo possui várias fases e tem como objetivo a criação de um império, conquistando o maior número de territórios e de pontuação possível. Para facilitar a navegação no jogo encontram-se ao dispor do jogador uma série de comandos que permitem a tomada de decisões tanto na configuração como na duração do próprio jogo.

O código de todo o programa foi estruturando recorrendo a conceitos e estratégias expostas durante as aulas da unidade curricular: conceito de classe, atributo, relação de composição e agregação, encapsulamento, entre outros.

Consideram-se as classes de maior importância a classe Jogo e a classe Mundo pois é a partir destas que se desenvolvem a maioria das classes e funcionalidades disponíveis. Assim, estas classes e as suas relações com outras irão ser apresentadas com maior pormenor seguidamente.

RESUMO DE CLASSES IDENTIFICADAS

Jogo

- → **Imperio**
  - ↘ **Tecnologias**
    - ➢ DronesMilitares, MisseisTeleguiados, DefesasTerritoriais, BolsaValores, Banco Central

- → **Mundo**
  - ↘ **Territorio**
    - ➢ Inicial
    - ➢ Ilha
      - RefugioPiratas, Pescaria
    - ➢ Continente
      - Planicie, Montanha, Fortaleza, Mina, Duna,Castelo

  - ↘ **Evento**
    - ➢ Abandonado, Invasao, Alianca

ALGUNS CONCEITOS IDENTIFICADOS

Mundo - turnos, impérios , mundo
Imperio – nome, pontuação, armazém, cofre,
Tecnologias – drones, defesas…
Territorios – inicial, planície, montanha …
Territorio Continente
Territorio Ilha
Conquista (de Territorios)
Recolha e Compra (atualização de atributos de acordo com o indicado)
Evento

CLASSES

CLASSE JOGO

A classe Jogo é a classe mais importante deste programa por ser a responsável por toda a organização do jogo e pela interação jogo / jogador. É através desta que é executado o menu inicial do programa e todos os menus correspondentes às diversas fases do jogo. É a responsável também pelo pedido/receção e interpretação de comandos inseridos pelo jogador, necessários ao acesso às diversas funcionalidades disponíveis.

```
Imperio *imp;
Mundo *m;
int fase;
static int lastLuck;
vector < Snapshot * > vsnap;
```

FIGURA 1 - MEMBRO PRIVADO DA CLASSE JOGO EM JOGO.H

Nesta classe foi criada uma função para cada fase do jogo de forma a obter a melhor organização possível: Na fase zero faz-se a configuração de um objeto da classe Mundo e avança-se para a fase um. A fase um corresponde à conquista de territórios e a dois à recolha de produtos e ouro. Na fase três é possível a compra de força militar e tecnologias. Os eventos ocorrem na fase quatro e finalmente, na fase cinco é calculada e apresentada a pontuação final. O jogo termina após 12 turnos (correspondente a 2 anos) ou pela ação de derrota no jogo.

Um aspeto também relevante desta classe é que é a responsável, na fase zero, pela construção de também um objeto da classe Imperio. Os objetos das classes Mundo e Imperio criados nesta fase zero serão os utilização durante o decorrer de todo o jogo.

CLASSE SNAPSHOT

Sendo Jogo uma classe que interliga as diferentes classes existentes no programa faz também parte dela o vetor de ponteiros Snapshot. A classe Snapshot é uma classe que é responsável apenas pelo armazenamento de informações sobre um determinado estado do jogo. É também na classe Jogo que estão implementados os comandos grava, ativa e apaga responsável pela ação sobre o então vetor **vsnap** .

CLASSE MUNDO

```
protected:
    vector < Territorio *> vt;
    vector < Evento *> ve;

private:
    int turno;
    static Evento * event;
```

FIGURA 2 – MEMBROS PRIVADOS E PROTEGIDOS DA CLASSE MUNDO EM MUNDO.H

A classe mundo é a responsável pela configuração e gestão dos territórios e dos eventos presentes durante o programa.

Para fazer gestão dos territórios foi utilizado um vetor de ponteiros Territorio. Este vetor é preenchido durante a fase um do programa sendo que pode incluir informação escrita pelo jogador ou informação proveniente de um ficheiro texto com a devida formatação para o efeito.

Possui um vetor de ponteiros Evento que , sendo carregado automaticamente na fase um do jogo detém as informações sobre os eventos a aplicar no aleatoriamente no mundo durante o jogo.

Existe também nesta classe um elemento **turno** que é a variável que simula a passagem de tempo durante o jogo e **static Evento *evento** detém o evento que será utilizado futuramente na fase 4.

Sendo esta a classe que gere e configura territórios e eventos neste programa é natural que se encontrem então funções como **addTerritorio();** e **addEvento();** que permitem o preenchimento dos respetivos vetores.

Outra função de grande importância pode-se considerar a **UpdateMundo();** . É utilizada na fase 4 depois da ação do evento. É a responsável pela atualização do turno, simulando a passagem de tempo, e chama todas as funções virtuais **update();** dos territórios responsáveis pela atualização de campos como a produção de produtos e ouro, que variam ao longo do jogo.

CLASSE IMPERIO

```
private:
    int pontuacao;
    int armazem;
    int maxArmazem;
    int cofre;
    int maxCofre;
    int fMilitar;
    int maxFmilitar;
    vector < Tecnologia *> vtec;
```

FIGURA 3 – MEMBRO PRIVADO DA CLASSE IMPERIO EM IMPERIO.H

A classe Imperio detém todas as informações sobre o império a ser construído pelo utilizador durante o jogo.

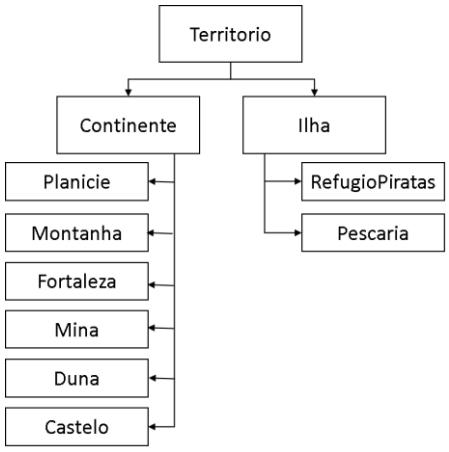
Detém a função **void war(Territorio * t, Mundo * m, int luck);** que é a responsável pela conquista ou não de territórios para o império. A classe Territorio detém uma variável **bool conquista** que é mudada de acordo com a fórmula presente na função.

*A ideia definida na organização deste jogo foi que os **Territórios** vão sempre pertencer ao **Mundo** mesmo que temporariamente possam ser associados ao **Imperio .***

Assim devido á grande interligação das duas classes , a classe Imperio é também uma class friend da classe Mundo.

CLASSE TERRITORIO

Esta classe possuiu uma relação de composição com a classe Mundo. Os objetos da classe Territorio são criados explicitamente para o preenchimento de vetores incluídos na classe Mundo. Falando também de uma relação de hereditariedade, esta é também a classe mãe de outras duas classes, Ilha e Continente. Por sua vez, estas duas classes são as classes-mãe de outras classes, tal como o exemplificado no esquema seguinte :



CLASSE EVENTO

Classe-mãe das classes Abandonado, Alianca e Invasao . Estas classes são as principais responsáveis pela ação na fase 4. Cada uma n a sua função **virtual bool eventAction(Imperio * imp, Mundo * m);** que é ajustada á medida da ação de cada classe filha sobre o Imperio e o Mundo presentes no jogo.

CLASSE TECNOLOGIA

Classe-mãe das classes BancoCental , BolsaValores, DefesasTerritoriais, DronesMilitares e MisseisTeleguiados. Também possuem funções virtuais adequadas á sua ação.

CARACTERÍSTICAS DAS CLASSES IMPLEMENTADAS

ENCAPSULAMENTO

No que diz respeito ao conceito de encapsulamento todas a classes existentes podem ser consideradas visto que cada uma possui dados e funcionalidades autónomos.

CLASSE DE OBJETIVO FOCADO , COESO E SEM DISPERSÃO

As classes Mundo e Imperio são as que melhor apresentam as características de objetivo focado, coeso e sem dispersão.

A classe Mundo foca-se apenas nas questões relacionadas com os territórios e eventos que o compõem. A classe Imperio segue os mesmos moldes contendo maioritariamente funções que alteram e atualizam os seus atributos.

O objetivo é sempre deixar todas as classes o mais possível com as características indicadas: com objetivo focado, coeso e sem dispersão. Neste trabalho prático a única classe que se afasta destas características é a classe Jogo que por ser o “motor do jogo“ faz a ligação entre elementos de diversas classes.

CLASSES COM RESPONSABILIDADES DE INTERFACE VS CLASSES DE NATUREZA LÓGICA

A classe Jogo tem responsabilidades de interface, de interação com o utilizador. Esta organiza o jogo, lança os menus e recebe os principais comandos de acesso às funcionalidades do jogo.

Todas as restantes classes representam a logica do programa, as suas ações são ocultas do utilizador.

FUNCIONALIDADE VARIÁVEL CONFIRME O OBJETO QUE A INVOCA

string show(Mundo * mun) const;
string show() const;

A função **show();** apresenta numa string única a informação sobre o objeto que a invoca.

virtual void update(int turno) = 0; em Mundo.h
virtual bool eventAction(Imperio * imp, Mundo * m); = 0 em Evento

A utilização de funções virtuais é também um exemplo da ação variável de uma função sobre o objeto em questão.

CONCLUSÃO:
FUNCIONALIDADES IMPLEMENTADAS E PROBLEMAS ENCONTRADOS

Todo o programa funciona partindo da classe Jogo. Esta lança os vários menus disponíveis para aceder às funcionalidades existentes. As funcionalidades foram implementadas por fases sendo que cada fase tem o seu objetivo e comandos próprios.

Ao longo a realização do trabalhos prático existiram algumas dificuldades na realização dos construtores e destrutores de classes como a classe Mundo devido à sua complexidade.
Ainda assim, considera-se que as funcionalidades descritas no enunciado deste trabalho prático foram implementadas com um alto grau de sucesso.

Tabela Resumo das Funcionalidade Implementadas:

Funcionalidade	Comandos	Realizado	Realizado Parcialmente	Não Realizado
Fase 0: Configuração do mundo	carrega<ficheiro> cria<território><número>	X		
Fase 1: Conquista de Territórios	conquista<território> passa	X		
Fase 2 :Recolha de Produtos e Ouro	maisouro maisprod	X		
Fase 3 : Compra de unidades militares e de tecnologia	maismilitar adquire<tecnologia>	X		
Fase 4 : Ocorrência de Eventos	-	X		
Comandos de visualização de resultados	lista lista mundo lista imperio lista tecnologias lista eventos	X		
Snapshots de Jogo	grava<nome> ativa<nome> apaga<nome>	X		
Restantes comandos de Debug	toma <qual><nome> modifica<ouro/prod> <n> fevento<nomeEvento>	X		