Instituto Superior de Engenharia de Coimbra
Engenharia Informática
Programação Orientada a Objetos

Relatório do Trabalho Prático

Programação Orientada a Objetos

2020/2021

Meta 1

Trabalho realizado por:

Hugo Gomes - 2019136085 - P4

Fábio Santos - 2018019155 - P8

Índice

Fábio Moreira Santos – a2018019155
Hugo Gomes - a2019136085

Introdução

Neste ano letivo de 2020/2021, na cadeira de Programação Orientada a Objetos, foi-nos proposto criar um jogo do tipo *single-player* sobre a conquista e expansão territorial. Na meta 1, o nosso objetivo seria criar a base do projeto (criação do mundo, tanto por teclado, como por ficheiro; comando "conquista"; comando "lista") de maneira a que ficássemos a entender melhor como iria ser o projeto.

Fábio Moreira Santos – a2018019155
Hugo Gomes - a2019136085

Classes Consideradas

A primeira versão tinha as seguintes classes e principais funcionalidades:

- Mundo
  - Contém um vetor com todos os territórios existentes
  - Tem a possibilidade de criar territórios novos
  - Comando "lista()" que permite visualizar todos os territórios
  - Comando "lista(nomeTerritorio)" para obter os pormenores do território
- Território
  - Nome do território
  - A sua resistência a invasões
  - A quantidade de criação de Produtos
  - A quantidade de criação de Ouro
  - Os pontos de Vitória
  - Um indicativo se está conquistado ou não
- Império
  - Armazém de produtos
  - Armazém de Ouro
  - A força militar que o império tem, força esta que permite a conquista de novos territórios
  - Vetor com todos os territórios pertencentes ao império
- Interface
  - Esta é a classe que cria e armazena o Mundo
  - Configura o Mundo tanto através de comandos lidos do teclado como de um ficheiro
  - Controla as fases de turnos
  - Processa os comandos recebidos do utilizador por uma *stream* de *input*
  - Mostra informação vinda do Mundo por uma *stream* de *output*

Criação e destruição de classes

Interface: os objetos desta classe são criados, armazenados e destruídos na função main.

Mundo: os objetos desta classe são criados, armazenados e destruídos na classe Interface.

Império: os objetos desta classe são criados, armazenados e destruídos na classe Mundo.

Território: os objetos desta classe são criados e destruídos na classe Mundo, e referenciados em vetores de ponteiros para **Território** nas classes Mundo e Império.

Exemplo de Encapsulamento

- A responsabilidade “conquistar território” conquistar um território está na classe Império, porque esta é que armazena os territórios conquistados pelo império.

```
bool Imperio::conquistar(Territorio * territorio) {
    int fatorSorte = randomNumEntre(MAX,MIN);
    int soma = fatorSorte + forcaMilitar;

    if (soma >= territorio->getResistencia()) {
        this->addTerritorio(territorio);
        territorio->setIsConquistado(true);
        return true;
    }
    else {
        territorio->setIsConquistado(true);
        if(forcaMilitar != 0)
            forcaMilitar--;
        return false;
    }
}
```

- Ao executar o comando “lista”, podem ser listados vários tipos de informação, como os dados do império, dados de um território específico, lista de territórios no mundo, e lista de territórios conquistados pelo império.
- A responsabilidade “listar informação do império” está atribuída ao método “listaInfo()” na classe Império, dado que é o mesmo que guarda os dados que se pretendem.

```
string Imperio::listaInfo() const {
    ostringstream os;
    os << "Armazem Produtos: " << armazemProdutos << "/" << maxUnidades << endl
        << "Armazem Ouro: " << armazemOuro << "/" << maxUnidades << endl
        << "Forca Militar: " << forcaMilitar << "/" << maxMilitar << endl
        << "Territorios conquistados: " << reinado.size() << endl;

    return os.str();
}
```

- A responsabilidade “listar informação de um território específico” está atribuída ao método “listaInfo()” na classe Território, pois a mesma é que guarda os dados que pretendem ser apresentados ao utilizador.

```
string Territorio::listaInfo() const {
    ostringstream os;

    os << "Resistencia: " << resistencia << endl
        << "Criacao Produtos: " << criacaoProdutos << endl
        << "Criacao Ouro: " << criacaoOuro << endl
        << "Pontos Vitoria: " << pVitoria << endl
        << "Status: " << getStatusConquitado();

    return os.str();
}
```

Exemplos de Classes com objetivo focado

Classe Interface: Esta classe tem o objetivo de fazer a ligação entre o utilizador e o mundo, controlando as fases do jogo, input do utilizador e output de informação para a consola.

Classe Império: Esta classe tem como objetivo gerir o império do utilizador.

Classes de interface com o utilizador e de lógica

Responsabilidade da **interface de interação com o utilizador**: Classe Interface

Responsabilidade da **lógica da aplicação**: Classes Mundo, Império e Território.

Primeiro objeto para além da camada de interação com o utilizador

As instruções recebidas e processadas pela camada de interação, são enviadas para um objeto da classe Mundo de modo a que estas sejam executadas (comandos cria, conquista e lista).

Funcionalidades implementadas

Componente do trabalho	Realizado	Realizado parcialmente	Não realizado
Comando cria (leitura por teclado)	x		
Comando carrega (leitura por ficheiro)	x		
Comando conquista	x		
Comando lista	x		

Conclusão

Nesta primeira meta, começámos a ter uma ideia de como iniciar o desenvolvimento do jogo, através da criação das classes básicas (Interface, Mundo, Império, Território) e da implementação de alguns dos comandos (cria, carrega, conquista e lista), tendo sempre em consideração os conceitos de Orientação a Objetos e da linguagem C++ lecionados nas aulas, conseguindo assim realizar todas as tarefas propostas para esta meta.