Instituto Superior de Engenharia de Coimbra

Licenciatura em Engenharia Informática

Programação Orientada a Objetos

Leandro Adão Fidalgo | a2017017144

Pedro dos Santos Alves | a2019112789

Laboratório P6

Trabalho Prático

Coimbra, 17 de janeiro de 2021

Índice

Introdução 3

Organização do código apresentado 4

1. Quais foram as classes consideradas na primeira versão da aplicação que foi testada? 4

2. Quais os conceitos/classe que identificou ao ler o enunciado? 5

3. Relativamente a duas das principais classes da aplicação, identifique em que classes ou partes do programa são criados, armazenados e destruídos os seus objetos. 6

4. Indique um exemplo de uma responsabilidade atribuída a uma classe que esteja de acordo com a orientação dada acerca de Encapsulamento. 6

5. De entre as classes que fez, escolha duas e justifique por que considera que são classes com objetivo focado, coeso e sem dispersão. 6

6. Relativamente à aplicação entregue, quais as classes que considera com responsabilidades de interface com o utilizador e quais as que representam a lógica? 6

7. Identifique o primeiro objeto para além da camada de interação com o utilizador que recebe e coordena uma funcionalidade de natureza lógica? 6

8. A classe que representa a envolvente de toda a lógica executa em pormenor muitas funcionalidades, ou delega noutras classes? Indique um exemplo em que esta classe delega uma funcionalidade noutra classe. 6

9. Dê um exemplo de uma funcionalidade que varia conforme o tipo do objeto que a invoca. Indique em que classes e métodos está implementada esta funcionalidade. 7

10. Apresente as principais classes da aplicação através da seguinte informação: 7

Funcionalidades implementadas 9

Introdução

O presente relatório descreve o projeto desenvolvido pelos alunos: Leandro Fidalgo e Pedro Alves, no âmbito da disciplina de Programação Orientada a Objetos da Licenciatura em Engenharia Informática do Instituto Superior de Engenharia de Coimbra.

O tema do projeto é o desenvolvimento de um jogo do tipo single-player sobre conquista e expansão territorial, sendo este desenvolvido em C++.

O objetivo do presente trabalho é consolidar todos os conhecimentos adquiridos nas aulas teóricas e práticas ao longo de todo o semestre.

Organização do código apresentado

1. Quais foram as classes consideradas na primeira versão da aplicação que foi testada?

A primeira versão tinha as seguintes classes e principais funcionalidades:

- Controlador_Interface
  - Trata dos comandos provenientes do utilizador;
  - Inicia o jogo;
  - Trata do decorrer do jogo (fases);
  - Guarda os jogos gravados pelo utilizador.
- Gravacao
  - Contém uma cópia do jogo.
- Jogo
  - Indica o estado do jogo;
  - Contém o mundo;
  - Contém o próximo evento;
  - Contém a pontuação do jogador.
- Mundo
  - Guarda e trata da informação de todos os territórios;
  - Guarda a informação do império do jogador.
- Imperio_Jogador
  - Guarda os territórios conquistados;
  - Guarda as tecnologias;
  - Contém a informação do império do jogador.
- Territorio
  - Contém a informação básica do território.
- Territorio_Inicial
  - Indica o território inicial do jogador.
- Continente
  - Contém a informação básica do continente.
- Castelo
  - Deriva de Continente.
- Duna
  - Deriva de Continente.
- Fortaleza
  - Deriva de Continente.
- Mina
  - Deriva de Continente.
- Montanha
  - Deriva de Continente.
- Planicie
  - Deriva de Continente.
- Ilha
  - Contém informação básica da Ilha.
- Pescaria
  - Deriva de Ilha.

- Refugio_Piratas
    - Deriva de Ilha.
- Tecnologia
    - Contém a informação sobre as diversas tecnologias e o custo das mesmas;
    - Contém uma verificação se já foi adquirida.
- Banco_Central
    - Deriva de Tecnologia.
- Bolsa_Valores
    - Deriva de Tecnologia.
- Defesas_Territoriais
    - Deriva de Tecnologia.
- Drones_Militares
    - Deriva de Tecnologia.
- Misseis_Teleguiados
    - Deriva de Tecnologia.
- Evento
    - Contém a informação básica sobre os eventos;
- Alianca_Diplomatica
    - Deriva de Tecnologia.
- Invasao
    - Deriva de Tecnologia.
- Recurso_Abandonado
    - Deriva de Tecnologia.

2. Quais os conceitos/classe que identificou ao ler o enunciado?

Os conceitos/classe identificados foram:

- Imperio do jogador
- Mundo
- Território
    - Território inicial
    - Continente
        - Planície
        - Montanha
        - Fortaleza
        - Mina
        - Duna
        - Castelo
    - Ilha
        - Refúgio dos Piratas
        - Pescaria
- Tecnologia
    - Banco Central
    - Bolsa Valores
    - Defesas Territoriais
    - Drones Militares

    - Misseis Teleguiados
- Jogo
- Evento
    - Aliança Diplomática
    - Invasão
    - Recurso Abandonado

3. Relativamente a duas das principais classes da aplicação, identifique em que classes ou partes do programa são criados, armazenados e destruídos os seus objetos.

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

Tecnologia: os objetos desta classe são criados, armazenados e destruídos na classe Imperio_Jogador.

4. Indique um exemplo de uma responsabilidade atribuída a uma classe que esteja de acordo com a orientação dada acerca de Encapsulamento.

A responsabilidade “adicionar territórios” está atribuída à classe Mundo porque tem a coleção de Territórios.

5. De entre as classes que fez, escolha duas e justifique por que considera que são classes com objetivo focado, coeso e sem dispersão.

Classe Mundo: tem dados e responsabilidades relativos apenas ao Império e Territórios, como a adição de territórios, conquista e a sua listagem.

Classe Territorio: tem dados base de cada território, tal como a verificação se este pode ser conquistado e a sua cópia, sendo estas funções virtual e que irão ser redefinidas nas classes que a descendem.

6. Relativamente à aplicação entregue, quais as classes que considera com responsabilidades de interface com o utilizador e quais as que representam a lógica?

Responsabilidade de interface: Controlador_Interface

Responsabilidades da lógica da aplicação: Restantes classes já descritas acima.

7. Identifique o primeiro objeto para além da camada de interação com o utilizador que recebe e coordena uma funcionalidade de natureza lógica?

As ordens vindas da camada de interação com o utilizador são recebidas e processadas por um objeto da classe Jogo.

8. A classe que representa a envolvente de toda a lógica executa em pormenor muitas funcionalidades, ou delega noutras classes? Indique um exemplo em que esta classe delega uma funcionalidade noutra classe.

A classe Mundo representa a envolvente de toda a lógica. Para adicionar um território ainda não conquistado ao Império, delega a adição desse território ao Império.

9. Dê um exemplo de uma funcionalidade que varia conforme o tipo do objeto que a invoca. Indique em que classes e métodos está implementada esta funcionalidade.

Classe Evento

- O método efeito (define o efeito de cada evento) e o método novo (serve para duplicar o próprio objeto). Exemplo: Quando o jogo está na fase de eventos e invoca o método efeito, o método varia de acordo com o evento que irá decorrer.

Classe Tecnologia

- O método comprar (verifica se é possível adquirir a tecnologia nas diversas tecnologias) e o novo (serve para duplicar o próprio objeto).

Classe Territorio

- Os métodos set_criacao_produtos (alterar a criação de produtos dependendo do território), set_criacao_ouro(alterar a criação de ouro dependendo do território), ser_conquistado (verifica se o território pode ser conquistado) e novo (serve para duplicar o próprio objeto).

10. Apresente as principais classes da aplicação através da seguinte informação:

Classe: Controlador_Interface

Responsabilidades:

- Ler comandos do utilizador;
- Inicia o jogo;
- Guarda os jogos gravados pelo utilizador;
- Trata do decorrer do jogo (fases).

Colaborações: Jogo, Gravacao

Classe: Gravacao

Responsabilidades:

- A gravação contém o nome da gravação e uma cópia do jogo;
- Trata das gravações do jogo que o utilizador faz.

Colaborações: Jogo

Classe: Jogo

Responsabilidades:

- Inicia o Mundo;
- Inicia os Eventos;
- Estado do jogo (turno, fator sorte, indicação se o jogo está a correr, pontuacao).

Colaborações: Mundo, Evento, Controlador_Interface

Classe: Evento

Responsabilidades:

- Contém a informação básica sobre os Eventos (nome, descricao);
- Trata do efeito dos eventos e ainda da sua cópia.

Colaborações: Jogo

Classe: Mundo

Responsabilidades:

- Inicia e contém o império do jogador;
- Guardar os territórios;
- Trata de tudo o que envolve os territórios.

Colaborações: Imperio_Jogador, Território, Jogo

Classe: Imperio_Jogador

Responsabilidades:

- Contém a informação do Império (tecnologias, armazém, max_armazem, cofre, max_cofre, força_militar, max_forca_militar, fator_sorte);
- Trata dos territórios conquistados.

Colaborações: Mundo, Território, Tecnologia

Classe: Territorio

Responsabilidades:

- Contém a informação base dos territórios (nome, pontos_vitória, resistencia, criacao_produtos, criacao_ouro).

Colaborações: Mundo, Imperio_Jogador

Classe: Tecnologia

Responsabilidades:

- Contém a informação base sobre as tecnologias (nome, objetivo, custo e se as tecnologias foram adquiridas ou não);
- Trata da compra das tecnologias;
- Trata do tomar e ainda da sua cópia.

Colaborações: Imperio_Jogador

Funcionalidades implementadas

Componente do trabalho	Realizado	Realizado parcialmente	Não realizado
Configuração do mundo através de comandos (cria)	**x**		
Configuração do mundo através de comandos (carrega)	**x**		
Conquista de territórios (comando conquista)	**x**		
Visualização dos dados do jogo (comando lista)	**x**		
Neste turno não se pretende conquistar nenhum território (comando passa)	**x**		
obtém mais 1 de ouro, perdendo 2 de produtos; se não tiver 2 de produtos esta troca não é possível (comando maisouro)	**x**		
obtém mais 1 de produtos, perdendo 2 de ouro; se não tiver 2 de ouro esta troca não é possível (comando maisprod)	**x**		
compra uma unidade militar: reduz uma unidade de produtos e uma de ouro para obter uma unidade de força militar adicional, desde que a força militar não ultrapasse o valor máximo possível (maismilitar)	**x**		
Dá a ordem ao império para adquirir uma determinada tecnologia. O parâmetro tipo indica qual a tecnologia a comprar (drone, missil, etc.). Todas as regras anteriormente descritas são aqui seguidas: se é possível adquirir tal objeto o seu custo é descontado dos recursos existentes do império. Esta nova aquisição será efetivamente aplicada ao império na fase correspondente do turno (comando adquire)	**x**		
Termina a fase de recolha de comandos e desencadeia as ações necessárias a cada fase (comando avanca)	**x**		
Grava o estado do jogo em memória, associando-lhe um	**x**		

nome. Esta ação consiste em fazer uma espécie de savegame para memória, possibilitando ao jogador manter em memória diversos snapshots do jogo, correspondentes a diversos momentos, permitindo-lhe a qualquer momento recuperar um desses momentos. O jogo continua ativo, mas a cópia feita para memória já não será afetada pelos comandos entretanto escritos a partir deste momento (comando grava)			
Recupera um dado estado do jogo em memória, identificado ao nome indicado, e carrega-o. O jogo recuperado passa a ser o que está em efeito: os comandos passam a agir sobre este (comando ativa)	**X**		
Apaga da memória um dado estado do jogo previamente gravado e associado ao nome especificado (comando ativa)	**x**		
Toma de assalto um determinado território ou tecnologia desde que esteja disponível. O parâmetro qual pode ser terr – território ou tec - tecnologia. O parâmetro nome indica o nome do território ou da tecnologia. Este comando serve apenas para DEBUG, não sendo seguida qualquer regra para a sua aquisição ou mesmo aplicado o custo associado. Este comando tem efeito imediato, não sendo necessário esperar pela fase correspondente (comando toma)	**x**		

Modifica os dados do império: quantidade de ouro no cofre ou quantidade de unidades de produtos a ter nos armazéns. Este comando também é essencialmente para DEBUG (para efeitos de testes e demonstração) (comando modifica)	**x**		
força a ocorrência de um evento indicado pelo seu nome, tal como descrito na lista indicada atrás. Este comando serve para DEBUG e teste/avaliação (comando fevento)	**x**		