Instituto Superior de Engenharia de Coimbra

Engenharia Informática

Programação Orientada a Objetos
Trabalho Prático 2021/2e

Meta 2

- Filipe Alexandre Rodrigues Fernandes, 2020134826;

- Rodrigo da Silva Calçarão, 2020134575.

Índice

Introdução

O trabalho prático da disciplina de Programação Orientada a Objetos pretende que seconstrua em C++ um jogo do tipo single-player sobre construção e desenvolvimento.

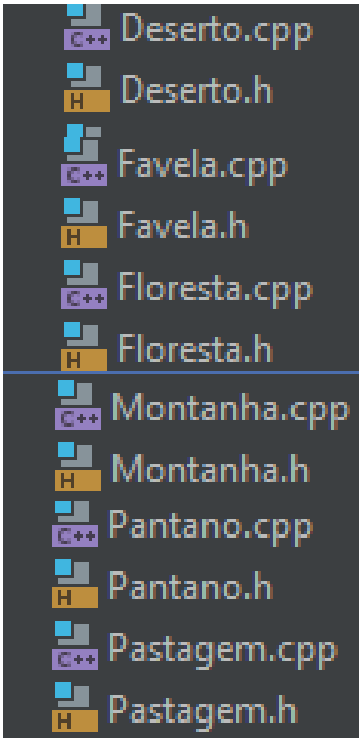
Para a realização do mesmo, foram interpretados diferentes conceitos e entidades queforam traduzidas em classes no projeto, cujo mesmos irão ser descritos na próxima secção.
Também será justificado a organização do programa e o progresso do mesmo ao longo desterelatório.

Classes – Conceitos da Versão Final

Classes da Ilha e da Zona

Nesta versão final, a ilha é constituída por vários objetos derivados (subclasses de tipo zona) da classe Zona, através de um vetor de ponteiros.

As subclasses de Zona:



Classe Zona:

```
25  class Recursos;
26  class Ilha;
27
28  class Zona {
29
30      vector<Trabalhadores* > TrabalhadoresDaZona;
31      vector<Edificio* > EdificioDaZona;
32      Recursos* recursos;
33      Ilha* ilha;
34      int N_edificios;
35      int linha;
36      int coluna;
37      const string tipo;
38      int day=1;
39
40  public:
41
42      Zona(const string t,const int linha, const int coluna)  : tipo(t),linha(linha),coluna(coluna){
43          N_edificios=0;
44      };
45
46      void pointToIlha(Ilha* abc){
47          ilha = abc;
48      }
```

Tal como acima referido, existe uma relação bidirecional entre os objetos derivados entra a classe Zona e a ilha, aqui também podemos encontrar um vetor de ponteiros do tipo Trabalhadores e edifícios, isto com intuito de armazenar a informação dos objetos de cada zona da ilha. Temos dois ponteiros a apontar para duas classes diferentes, o ponteiro “Recursos*” que aponta para uma classe móvel e o ponteiro “Ilha*” para a classe Ilha.

A Classe ilha:

```
class Ilha {

    const int linhas;
    const int colunas;
    vector<vector<Zona*>> ilhaBi;
    Recursos* recursos;
    bool find=false;
    int dia=1;


public:

    Ilha() = delete;
    Ilha(const int l, const int c);
```

A classe ilha como foi referido anteriormente contem um vetor de ponteiros de tipo Zona, um ponteiro para os recursos, e uma booleana usada com o intuito de mover os trabalhadores entre zonas.
Nesta figura podemos ver a construção de cada zona a partir do construtor da Ilha

```
Ilha::Ilha(const int l, const int c) : linhas(l), colunas(c) {
    recursos = new Recursos();
    for(int i=0; i<linhas; i++) {
        vector<Zona* > v1;
        int f = random_l_h( min: 1, max: 6);

        for(int j=0; j<colunas; j++) {

            //int f = (rand() % 6) + 1;
            f = random_l_h( min: 1, max: 6);
            switch (f) {
                case 1:
                    v1.push_back(new Deserto( linha: i, coluna: j));
                    break;

                case 2:
                    v1.push_back(new Floresta( linha: i, coluna: j));
                    break;

                case 3:
                    v1.push_back(new Pastagem( linha: i, coluna: j));
                    break;

                case 4:
                    v1.push_back(new Montanha( linha: i, coluna: j));
                    break;

                case 5:
                    v1.push_back(new Pantano( linha: i, coluna: j));
                    break;

                case 6:
                    v1.push_back(new Favela( linha: i, coluna: j));//Composição -> Zonas fazem parte da ilha
                    break;

            }
        }
```

Construtor da ilha – aqui cada zona é criada;

Respostas a questões para o relatório

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

A ilha e a zona são facilmente duas das principais classes da aplicação, sendo que a ilha é criada na main e a zona é criada na ilha através de um vetor de ponteiros do tipo zona, em relação à ilha são passados como argumentos para a criação do seu objeto o número de linhas e o número de colunas. A zona é responsável por armazenamento de edifícios e trabalhadores através de um vetor de ponteiros do tipo referente aos respetivos objetos que armazena.

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

A responsabilidade de guardar os diferentes estados de jogo está claramente atribuída aos diferentes get´s e set´s, nas diferentes classes deste projeto.

3 - De entre as classes que fez, escolha duas e justifique por que considera quesão classes com objetivo focado, coeso e sem dispersão. / Relativamente à aplicação entregue, quais as classes que considera com responsabilidades de interface com o utilizador e quais as que representam a lógica?

Podemos considerar as subclasses da classe edifício classes com um objetivo focado, coeso e sem dispersão, visto que o seu único objetivo é reagir às condições da Zona e do utilizador.
No que diz respeito ao interface com o utilizador, foram usadas a main, a classe ilha e a classe Zona e as suas subclasses para interagir com o utilizador.

4- 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 objecto da classe Ilha.

5- 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 Ilha representa a envolvente de toda a lógica. Um exemplo que pode ser dado é a execução do comando “next”, responsável pela mudança de dia.
Esta função, envolve práticamente todas as classes presentes no projeto, mas neste caso, a Classe Ilha delega uma funcionalida de na Classe Zona.

6- 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.

Na execução do comando “next”, ocorre a execução de duas funções, sendo elas a função “produzir” e a função “newDay”. A função “produzir” tem como principal objetivo, executar a função virtual presente na Classe Edificios. Uma vez que existem 6 tipos de edificos diferentes, existem 6 funções “produzir” diferentes em cada objeto de subclasses da Classe Edificio