PROVA DE EXAME
de
Programação Orientada por Objetos

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Engenharia Informática
Informática de Gestão

Época de Recurso – 10 de fevereiro de 2020

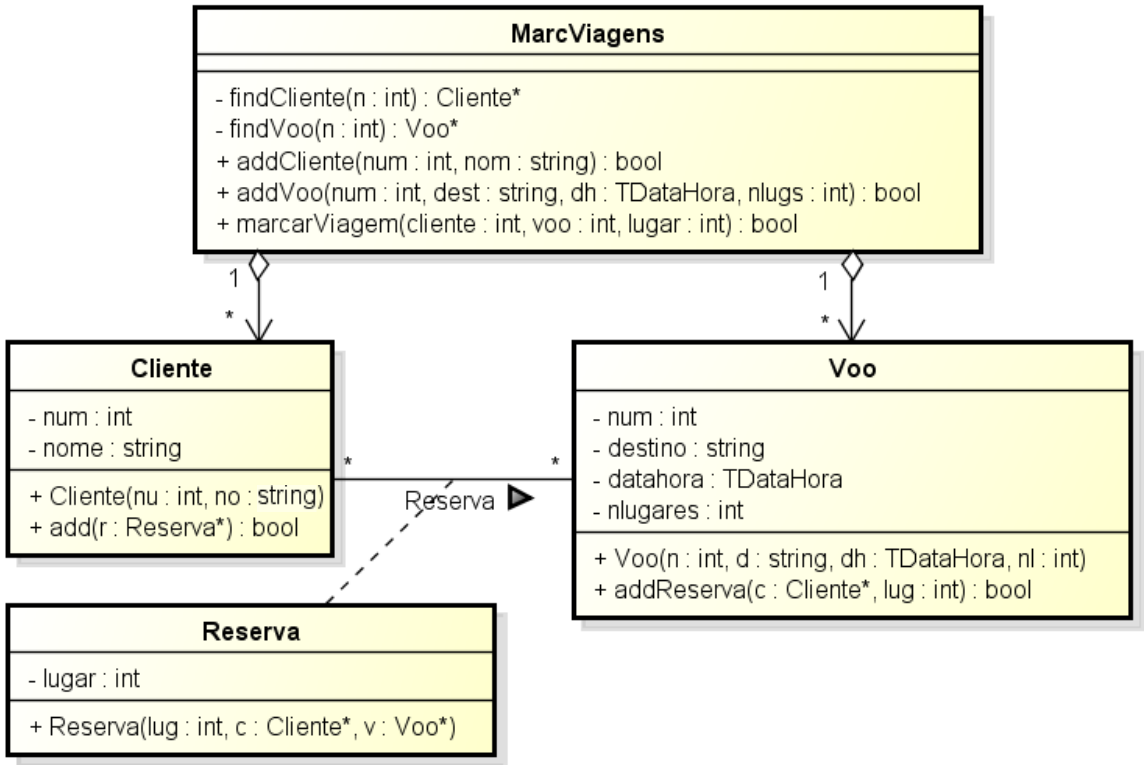
Notas importantes:

1. *A duração da prova é de 2 horas.*
2. *Comece por preencher a zona reservada à sua identificação na folha de exame.*

Grupo I

[15 valores]

Uma transportadora aérea necessita de uma pequena aplicação que permita aos seus clientes marcar a sua viagem num dos voos agendados. Nessa marcação deverá ficar reservado o lugar da aeronave que o cliente escolher. Segue-se o diagrama de classes UML que descreve de forma precisa a solução que se pretende para o problema descrito.



a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e atributos descritos no diagrama, e tendo ainda em conta todas as seguintes considerações: [9.9 val.]

- *Não defina quaisquer outros métodos ou atributos, a não ser, nas classes em que se justifique, o operador que permita que os objetos sejam colecionáveis;*
- *A marcação da viagem só deve ser permitida para um lugar da aeronave que se encontre ainda livre;*
- *Para o tipo data/hora é usada a classe TDataHora, da qual se apresenta, no Anexo B deste enunciado, o protótipo de alguns dos seus métodos e operadores mais importantes;*

- *As coleções deverão ser implementadas com base nos templates de classes Colecao ou ColecaoHibrida que utilizou nas aulas de POO. Para efeitos de consulta, disponibilizam-se os protótipos dos seus principais métodos no Anexo B deste enunciado;*
- *Defina os métodos no momento em que está a declarar as classes (não separe a declaração da implementação) e considere que todo o código reside num único ficheiro;*
- *Nesta e nas restantes alíneas, não precisa de indicar as diretivas #include.*

b) Acrescente ao problema os métodos que permitam apresentar na saída standard os voos que se encontrem lotados (mostre apenas o número de cada voo). [2.1 val.]

c) Acrescente ao problema os métodos que permitam cancelar uma reserva, sendo fornecidos os números do voo e do lugar. [3 val.]

Grupo II

[5 valores]

No Anexo A do presente enunciado encontra-se o código de um pequeno programa em C++. Depois de o analisar com o devido cuidado, responda às seguintes questões.

a) Diga quais são os métodos construtores (definidos explicitamente ou não) que estão presentes em cada uma das classes do problema. [0.6 val.]

b) Implemente o método `nota()` da classe `EpNormal`, de forma a devolver o valor da nota da época normal de avaliação, calculada através da média ponderada que leve em conta 30% da nota do miniteste, 20% do trabalho prático e 50% da prova de exame, isto para o caso desta última nota (a do exame) ser maior ou igual a 7 valores. Caso seja menor, então será a nota do exame a ser usada para nota final da época normal. [0.9 val.]

c) Apresente o resultado que será visualizado na saída standard após a execução do programa. [1.5 val.]

d) Qual o resultado que seria visualizado na saída standard se o método `print()` da classe `Epoca` não fosse virtual? Apresente apenas a parte da visualização que resultaria diferente em relação ao output da alínea anterior. [0.8 val.]

e) Diga que consequências é que teria, naquilo que é visualizado, o destrutor da classe `Epoca` não ser definido como virtual. [0.4 val.]

f) Por que razão numa coleção híbrida são normalmente guardados os apontadores para os objetos que se pretende colecionar, em vez dos próprios objetos? [0.8 val.]

ANEXO A

```
#include<iostream>
using namespace std;

class Epoca {
    double exame;
public:
    Epoca(double e) {
        exame = e;
        cout << "Inicio de Epoca..." << endl;
    }
    virtual int nota() const { return (int)(exame + 0.5); }
    virtual bool aprovado() const { return nota() >= 10; }
    virtual void print() const {
        cout<<(aprovado()?"aprovado com ":"reprovado com ")<<nota()<<" valores.\n";
    }
    virtual ~Epoca() { cout << "Fim de Epoca: " << endl; }
};

class EpRecurso : public Epoca {
public:
    EpRecurso(double e): Epoca(e){cout << "Inicio da Epoca de Recurso..."  << endl;}
    void print() const { cout << "Epoca de Recurso: "; Epoca::print(); }
    ~EpRecurso() { cout << "Fim da Epoca de Recurso" << endl; }
};

class EpNormal: public Epoca{
    double miniteste, trabalho;
    static const double notaMinimaExame;
public:
    EpNormal(double mini, double trab, double exam) : Epoca(exam) {
        miniteste = mini; trabalho = trab;
        cout << "Inicio da Epoca Normal..." << endl;
    }
    int nota()const { //implementar na alinea b)
        ...
    }
    void print() const { cout << "Epoca Normal: "; Epoca::print(); }
    ~EpNormal() { cout << "Fim da Epoca Normal" << endl; }
};

const double EpNormal::notaMinimaExame = 7.0;

void main(){
    EpNormal enormal(10.0, 15.0, 14.0);
    EpRecurso erecurso(11.3);
    cout << "-1-" << endl;
    Epoca &en = enormal, &er = erecurso;
    cout << "-2-" << endl;
    en.print();
    er.print();
    cout << "Classificacao final: ";
    if (er.nota() > en.nota()) cout << er.nota() << endl;
    else cout << en.nota() << endl;
    cout << "-3-" << endl;
};
```

ANEXO B

Template Colecao

```
#include<set>
using namespace std;

template<class K>
class Colecao: public set<K>{
public:
     bool insert(const K &c);
     K *find(const K &c);
     int size() const;
     void erase(const K &);
     //void clear();
     //bool empty() const;
     //iterator begin();
     //iterator end();
};
```

Template ColecaoHibrida

```
#include<set>
using namespace std;

template<class T>
class less_pointers{
public:
     bool operator()(const T &left, const T &right) const { return (*left < *right); }
};

template<class K>
class ColecaoHibrida: public set<K, less_pointers<K>>{
public:
     bool insert(const K &c);
     K find(const K &c);
     int size() const;
     void erase(const K &);
     //void clear();
     //bool empty() const;
     //iterator begin();
     //iterator end();
};
```

Classe TDataHora

```
class TDataHora{
public: // construtor por defeito
   TDataHora();
      // cria o objeto com a data d/m/a e a hora h:min:s
   TDataHora(int d, int m, int a, int h, int min, double s);
      // cria o objeto com a data hora expressa em dt (string com o formato "x/x/xxxx h:m:s")
   TDataHora(const string & dt);
      // atribui ao objeto a data d/m/a e a hora h:min:s
   bool set(int d, int m, int a, int h, int min, double s);
      // atribui ao objeto a data hora expressa em dt (string com o formato "x/x/xxxx h:m:s")
   void set(const string &dt);
      // devolve uma string com a data hora do objeto expressa no formato "x/x/xx h:m:s"
   string toString() const;
      // verifica se a data hora do objeto é anterior a uma outra
   bool operator<(const TDataHora &outra) const;
      // verifica se a data hora do objeto é igual a uma outra
   bool operator==(const TDataHora &outra) const;
      // devolve um objeto do tipo TDataHora com a data e hora correntes retiradas do sistema
   static TDataHora hoje_agora();
};
```