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 – 14 de fevereiro de 2018

1o Sem. 2017/18

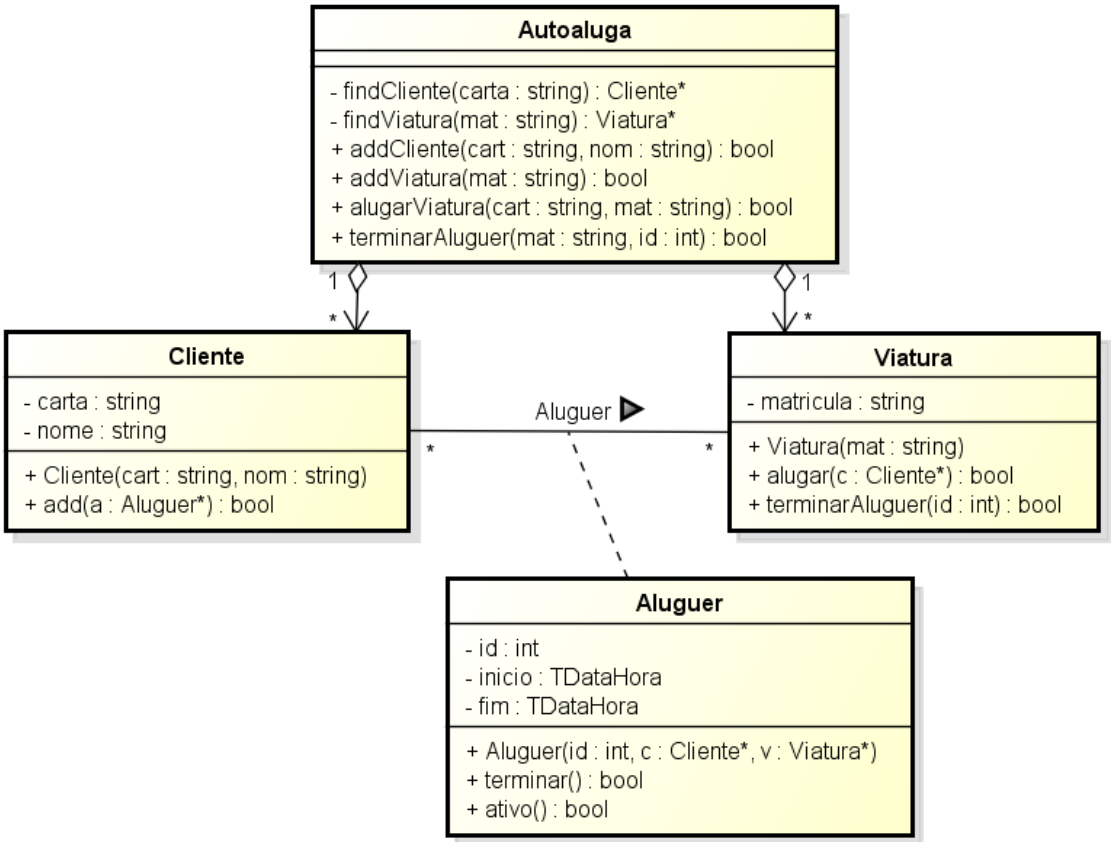
Notas importantes:

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

[14 valores]

A Autoaluga é uma empresa de aluguer de automóveis que, encontrando-se em franca expansão, necessita urgentemente de uma aplicação que lhe permita manter registado todos os alugueres das suas viaturas, atuais e passados. Segue-se o diagrama de classes UML que descreve de forma precisa a solução que se pretende para a aplicação descrita. O id dos alugueres de cada viatura deverá ser automaticamente atribuído pelo sistema, usando o código 1 para o 1º elemento, o 2 para o 2º, e assim sucessivamente.



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: [10 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;*
- *Para o tipo data/hora é usada a classe TDataHora, da qual se apresenta, no Anexo B deste enunciado, o conjunto de métodos e operadores que poderá usar na implementação das suas soluções;*

- *Admita que a aplicação está a correr em tempo real. Por isso, para as data/horas de início e fim de aluguer deve ser considerada a data/hora corrente. Usar, para o efeito, as funcionalidades da classe TDataHora;*
- *Para simplificar um pouco a solução, no aluguer de uma viatura não precisa de verificar se a viatura em causa está já alugada ou não;*
- *No momento em que se cria um novo aluguer, deve-se usar a data/hora corrente para o seu início e as 0 horas de 1 de janeiro de 1900 para o seu fim. Dessa forma, para se saber se o aluguer está ativo, bastará verificar-se se a data/hora 'fim' é anterior à data/hora 'inicio';*
- *A ação 'terminar aluguer' deve apenas consistir na atribuição da data/hora corrente ao atributo 'fim' do respetivo aluguer;*
- *As coleções deverão ser implementadas com base no template 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 dados de todos os alugueres realizados por um dado cliente. [4 val.]

Grupo II

[6 valores]

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

a) Alguma das classes é abstrata? Justifique. [0.6 val.]

b) Diga que método deveria ser definido como constante e porquê. [0.6 val.]

c) Apresente o resultado que será visualizado na saída standard com a execução do programa. [2.8 val.]

d) Apresente o resultado que seria visualizado na saída standard se o método `print()` da classe `base` não fosse virtual. Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea anterior. [0.4 val.]

e) Tendo em conta as classes definidas no problema, dê uma breve explicação do erro cometido em cada uma das instruções que se seguem: [1.6 val.]

(1) `Figura *fig = new Figura(10,20);`

(2) `Circulo circulos[10];`

(3) `Quadrado q(3); delete q;`

(4) `Circulo c1(1), c2(2); if(c1==c2) cout<< "sao iguais" << endl;`

ANEXO A

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

class Figura {
    double x, y;
public:
    Figura(double x, double y) {
        this->x = x; this->y = y;
        cout << "Criada Figura"<<endl;
    }
    virtual void print(){cout << "Com origem em " <<"("<< x <<","<< y <<")"<< endl;}
    virtual void ampliar(double fator) = 0;
    virtual ~Figura() { cout << "Xau Figura" <<endl; }
};

class Circulo : public Figura {
    double raio;
public:
    Circulo(double x, double y, double r) : Figura(x, y) {
        raio = r;
        cout << "Criado Círculo"<<endl;
    }
    void print() {
        cout << "Círculo de raio " << raio << endl;
        Figura::print();
    }
    void ampliar(double fator) { raio *= fator; }
     ~Circulo() { cout << "Xau Círculo->"; }
};

class Quadrado: public Figura {
    double lado;
public:
    Quadrado(int x, int y, int d) : Figura(x, y) {
        lado = d;
        cout << "Criado Retangulo" << endl;
    }
    void print() {
        cout << "Quadrado de lado " << lado << endl;
        Figura::print();
    }
    void ampliar(double fator) { lado *= fator; }
    ~Quadrado() { cout << "Xau Quadrado->"; }
};

void main() { Figura *f1 = new Quadrado(1, 2, 3);
              Circulo *f2 = new Circulo(4, 5, 6);
              cout << "-1-" << endl;
              Circulo c(*f2);
              cout << "-2-" << endl;
              f1->print();  f2->print();
              c.ampliar(2); c.print();
              cout << "-3-" << endl;
              delete f1;
              delete f2;
              cout << "-4-" << 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:
      // 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();
};
```