PROVA DE EXAME
de
Programação Orientada por Objetos

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Engenharia Informática
Informática de Gestão

Época Normal – 2 de fevereiro de 2018

1º Sem 2017/18

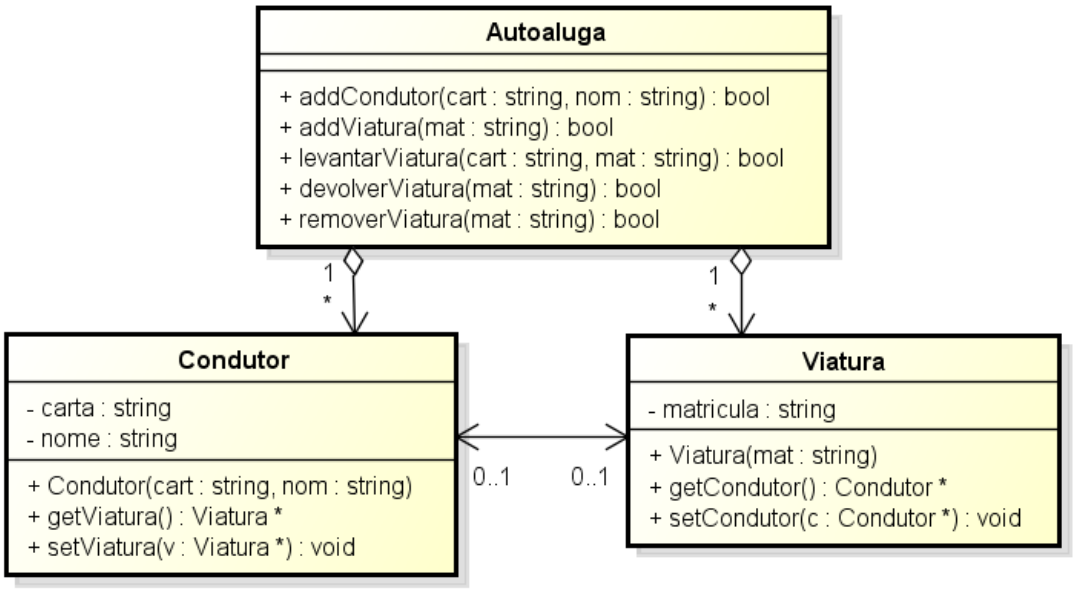
Notas importantes:

1. *A duração da prova é de 1 hora e 30 minutos.*
2. *Comece por preencher a zona reservada à sua identificação na folha de exame.*

Grupo I

[15 valores]

A Autoaluga é uma empresa de aluguer de automóveis que, encontrando-se em franca expansão, necessita urgentemente de uma aplicação que lhe permita, a todo o momento, monitorizar as viaturas que se encontrem alugadas. Segue-se o diagrama de classes UML que descreve de forma precisa a solução que se pretende para a aplicação descrita.



a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e atributos descritos no diagrama. 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 e um ou outro método auxiliar de que precise, definindo-os, nesse caso, como privados. [11 val.]

NOTA 1: Um condutor só poderá levantar uma viatura se já estiver registado na aplicação, não estiver já na posse de outra viatura e se aquela que pretenda estiver disponível;

NOTA 2: Uma viatura só poderá ser removida do sistema se a mesma não estiver associada a nenhum condutor;

NOTA 3: 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;

NOTA 4: 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;

NOTA 5: Nesta e nas restantes alíneas, não precisa de indicar as diretivas #include.

b) Acrescente ao problema os métodos que permitam apresentar na saída standard as viaturas que estejam a ser utilizadas (alugadas), mostrando para cada uma delas a respetiva matrícula, e o nome e carta de condução de quem a está a utilizar. [4 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 a devida atenção, responda às seguintes questões.

a) Desenhe o diagrama de classes UML do problema, indicando somente as classes, relações e respetivos atributos. Não inclua, portanto, qualquer método. [1.0 val.]

b) Indique três métodos (construtores ou não) que, embora não definidos explicitamente, vão estar presentes em ambas as classes do problema. [0.9 val.]

c) Apresente o resultado que será visualizado na saída standard após a execução do programa. [2.4 val.]

d) Apresente o resultado que seria visualizado na saída standard se nenhum dos métodos da classe `base` fosse virtual. [0.7 val.]

ANEXO A

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

class Pessoa{
	string nome;
public:
	Pessoa(const string &n):nome(n) {cout << "Criado "<< nome << endl; }
	string getNome() { return nome; }
	virtual void print() { cout << "Solteiro"; }
};

class PCasada: public Pessoa {
	Pessoa *conjuge;
	PCasada(const string &n, PCasada *c):Pessoa(n) {
		conjuge = c;
		cout << "Conjuge 1" << endl;
	}
public:
	PCasada(const string &n, const string &nc):Pessoa(n){
		conjuge = new PCasada(nc,this);
		cout << "Conjuge 2" << endl;
	}
	void operator+=(PCasada &outra) { conjuge = &outra; }
	void print() { cout << getNome() <<" + " << conjuge->getNome() << endl; }
};

void main() {
	cout << "-0-" << endl;
	PCasada *p1=new PCasada("Manuel","Maria");
	cout << "-1-" << endl;
	p1->print();
	cout << "-2-" << endl;
	PCasada p2("Ana", "Luis");
	cout << "-3-" << endl;
	p2.print();
	*p1 += p2;
	p2 += *p1;
	cout << "-4-" << endl;
	p1->print();
	cout << "-5-" << endl;
	p2.print();
}
```

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();
};
```