INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Engenharia Informática
Informática de Gestão

Época Normal – 21 de janeiro de 2013

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.*
3. *Resolva os dois grupos em folhas de exame separadas.*

Grupo I

(8 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) Propositadamente, todos os métodos da classe AveRara contêm um erro de compilação ou de execução. Identifique-os e proponha, para cada um deles, uma solução adequada (correções que deverá levar em conta na resolução das restantes alíneas). [1.6 val.]

b) Alguma das classes do problema é abstrata? Se sim, diga qual e por que motivo a considera abstrata. [0.4 val.]

c) Como sabe, para além dos métodos que são definidos de forma explícita em cada classe, existirão outros definidos automaticamente pelo C++. Diga quais são os métodos implícitos que estarão presentes em cada uma das duas classes do problema, ainda que não se vejam. [1.0 val.]

d) Apresente o resultado que será visualizado na saída standard após a execução do programa. [2.6]

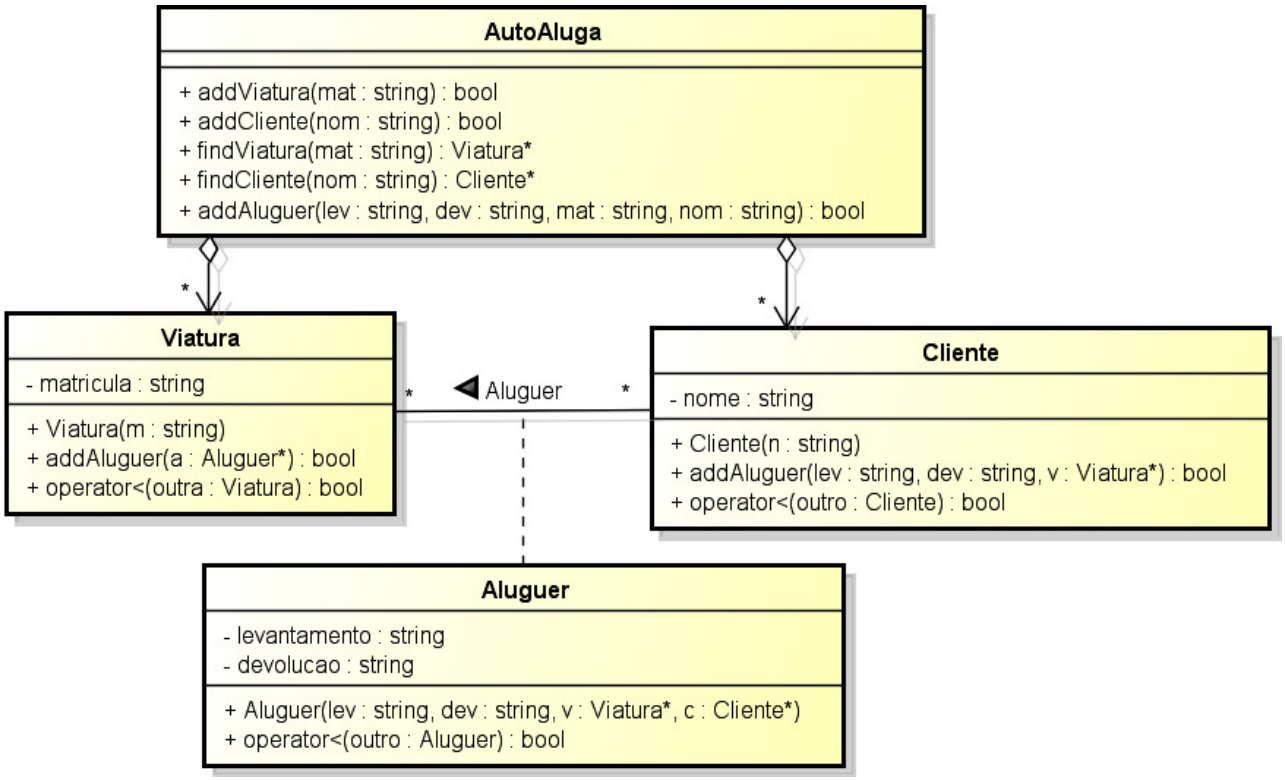
e) O que seria visualizado se o método *`mostrar`* da classe Ave não fosse virtual? Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea anterior. [0.8 val.]

f) E o que seria visualizado se fosse o método *`acrescentar`* a não ser virtual? Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea d). [0.8 val.]

g) Considerando as seguintes declarações: `AveRara a; Ave *p=&a;` diga, justificando, se a instrução `(*p)++;` é ou não válida no problema considerado. Caso não seja, introduza as alterações necessárias nas classes do problema para que a instrução passe a ser válida. [0.8 val.]

Grupo II
(12 valores)

A Auto Aluga é uma empresa de aluguer de automóveis que necessita de uma pequena aplicação que lhe permita registar todos os alugueres das suas viaturas. Dispõe de uma frota considerável de viaturas de modo a satisfazer todas as solicitações, que não param de aumentar. Relativamente a cada aluguer, pretende que sejam apenas registados, para além da viatura e do cliente envolvidos, a data-hora (apenas uma string) em que a viatura foi levantada e a data-hora em que a mesma terá sido devolvida.
Como na Auto Aluga há alguém a perceber um pouco de informática, disponibilizou já o diagrama de classes UML que se segue, para descrever de forma precisa a solução que pretende para a sua aplicação.



a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e atributos descritos no diagrama. Não implemente quaisquer outros métodos ou atributos. [8.5 val.]

NOTA 1: 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 2: 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 3: Nesta e nas restantes alíneas, não precisa de indicar as diretivas #include.

b) Acrescente ao problema os métodos necessários que permitam obter o número total de alugueres da Auto Aluga. [2.5 val.]

c) Implemente um pequeno main que faça uso de todas as funcionalidades da aplicação. [1.0 val.]

ANEXO A

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


class Ave{
	int num_penas;
public:
	Ave(int n): num_penas(n){cout<<"nova ave com "<<num_penas<<" penas\n";}
	virtual void acrescentar(int n){num_penas+=n;}
	virtual void mostrar(void) const{
		cout<<"tenho "<< num_penas << " penas\n";
	}
	virtual ~Ave(){cout<<"uma ave a menos\n";}
};


class AveRara: public Ave{
	int penas_regeitadas;
public:
	AveRara(){
		penas_regeitadas=0;
		cout<<"nova ave rara\n";
	}
	void acrescentar(int n) const{
		penas_regeitadas+=n;
		cout<<"tenho pena de nao ter penas!\n";
	}
	void mostrar(void) const{
		cout<<"regeitei "<< penas_regeitadas << " penas\n";
		Ave.mostrar();
	}
	~AveRara(){cout<<"menos uma ave rara com "<<num_penas<<" penas\n";}
};


void main(){
	AveRara a0;
	Ave &a1=a0;
	Ave a2(500);
	a0.acrescentar(3);
	cout<<"ACRESCENTAR:\n";
	a1.acrescentar(10); a2.acrescentar(20);
	cout<<"MOSTRAR:\n";
	a0.mostrar(); a1.mostrar(); a2.mostrar();
	cout<<"ABATER:\n";
}
```

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();
};
```