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 – 19 de janeiro de 2015

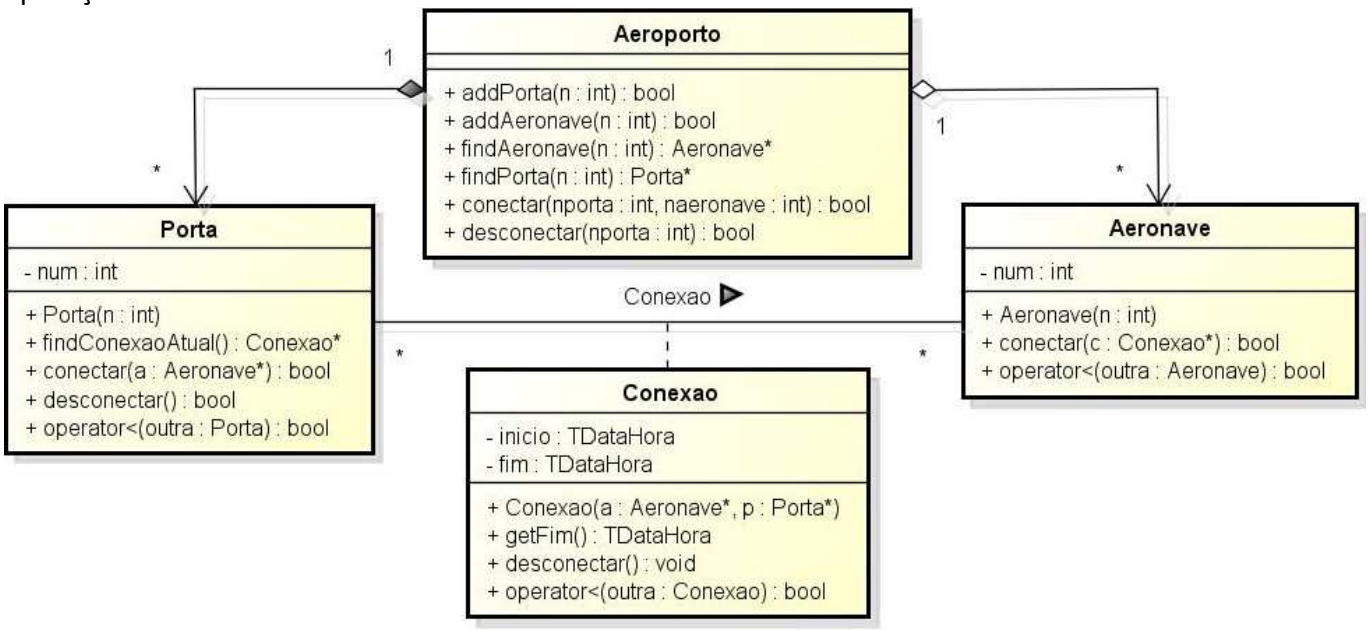
Notas importantes:

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

[12.6 valores]

Um importante aeroporto necessita de uma pequena aplicação que lhe permita manter o registo de todas as conexões de aeronaves às suas portas de embarque/desembarque (*gates*). A informação a reter relativamente a cada conexão é a data hora a que a mesma tem início e a data hora a que termina. No momento de criação de uma conexão (operação *conectar*) deve ser atribuída ao seu atributo *inicio* a data hora corrente e ao atributo *fim* a data hora `"1/1/3000 0:0:0"`. A operação *desconectar* deverá simplesmente traduzir-se na atribuição da data hora corrente ao atributo *fim*.
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 implemente o método `findConexaoAtual()` da classe `Porta`; assuma que o mesmo já se encontra implementado e que devolve NULL caso a porta não esteja conectada (i.e, todas as suas conexões já terminaram – atributo `fim` com data hora anterior à corrente). [9.0 val.]

NOTA 1: Para os tipos data hora é usada a classe TDataHora, estudada nas aulas da UC, e da qual se apresentam, no Anexo B deste enunciado, alguns métodos e operadores;

NOTA 2: Imagine que a aplicação está correr em tempo real. Por isso, para a hora e data das conexões devem ser consideradas a hora e data correntes. Use, para o efeito, as funcionalidades da classe TDataHora;

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 mostrar todas as portas que se encontrem conectadas (as que contêm uma conexão atual, i.e, que ainda não terminou), com indicação, para cada uma delas, do número da porta, do número da aeronave e da data hora a que iniciou a respetiva conexão. [2.4 val.]

c) Implemente um pequeno main que faça uso de todas as funcionalidades da aplicação. [1.2 val.]

Grupo II

[7.4 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? Se sim, diga qual e o que a torna abstrata. [0.5 val.]

b) Como pode constatar, apenas o método `print()` é um método constante. Por que razão se definiu esse método como constante? [0.5 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. [1.2 val.]

d) Apresente o resultado que será visualizado na saída standard após a execução do programa. [3.2 val.]

e) Apresente agora o resultado que seria visualizado na saída standard se nenhum dos métodos da classe `Funcionario` fosse virtual. [2.0 val.]

ANEXO A

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

class Funcionario{
protected:
	string nome;
public:
	Funcionario(const string &n) :nome(n){}
	virtual void print()const{ cout << "Funcionario " << nome <<endl;}
	virtual bool addSubordinado(Funcionario *f){
		cout << "Nao pode ter subordinados!" << endl;
		return false;
	}
	virtual ~Funcionario(){cout<<"Funcionario "<<nome<<" dispensado"<<endl;}
};

class Diretor: public Funcionario{
	static const int MaxSub = 5;
	Funcionario *subordinados[MaxSub];
	int nSub;
public:
	Diretor(const string &n): Funcionario(n){
		nSub = 0;
		cout << "Criado Diretor " << nome << endl;
	}
	void print()const{ cout << "Diretor " << nome <<endl; }
	bool addSubordinado(Funcionario *f){
		if (nSub == MaxSub) return false;
		else { subordinados[nSub++] = f; return true; }
	}
	~Diretor(){
		cout<<"Diretor dispensado com os seus "<< nSub << " subordinados"<<endl;
	}
};

void main(){
	cout << "####Criacao###" << endl;
	Diretor rui("Rui Manuel");
	Funcionario *p = &rui, f1("Pedro"), f2("Jose");
	cout << "####Adicao###" << endl;
	rui.addSubordinado(&f1);
	p->addSubordinado(&f2);
	cout << "####Print1###" << endl;
	rui.print();
	cout << "####Print2###" << endl;
	p->print();
	cout << "####Destruicao###" << 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

```
#include"TData.h"
#include"THora.h"

class TDataHora: public TData, public THora{
public:
	TDataHora();
	TDataHora(const char *dt); //dt: string no formato "x/x/xxxx h:m:s"
	bool operator<(const TDataHora &outra) const;
	...
	static TDataHora hoje_agora();//devolve data e hora corrente
};
ostream &operator<<(ostream &os, const TDataHora &dt);
```