INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Engenharia Informática
Informática de Gestão

Época de Recurso – 7 de fevereiro de 2015

Notas importantes:

1. *O Grupo III é de resolução facultativa e destina-se a substituir as notas do trabalho prático e miniteste*
2. *Duração da prova: 1h30 (Grupos I e II) + 1h (Grupo III)*
3. *Comece por preencher a zona reservada à sua identificação na folha de exame;*
4. *Resolva os três grupos em folhas de exame separadas.*

Grupo I

(7.2 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) Identifique o tipo de relação existente entre as classes *Desenho* e *Figura*. [0.2 val.]

b) Alguma das classes é abstrata? Se sim, diga qual e o que a torna abstrata. [0.7 val.]

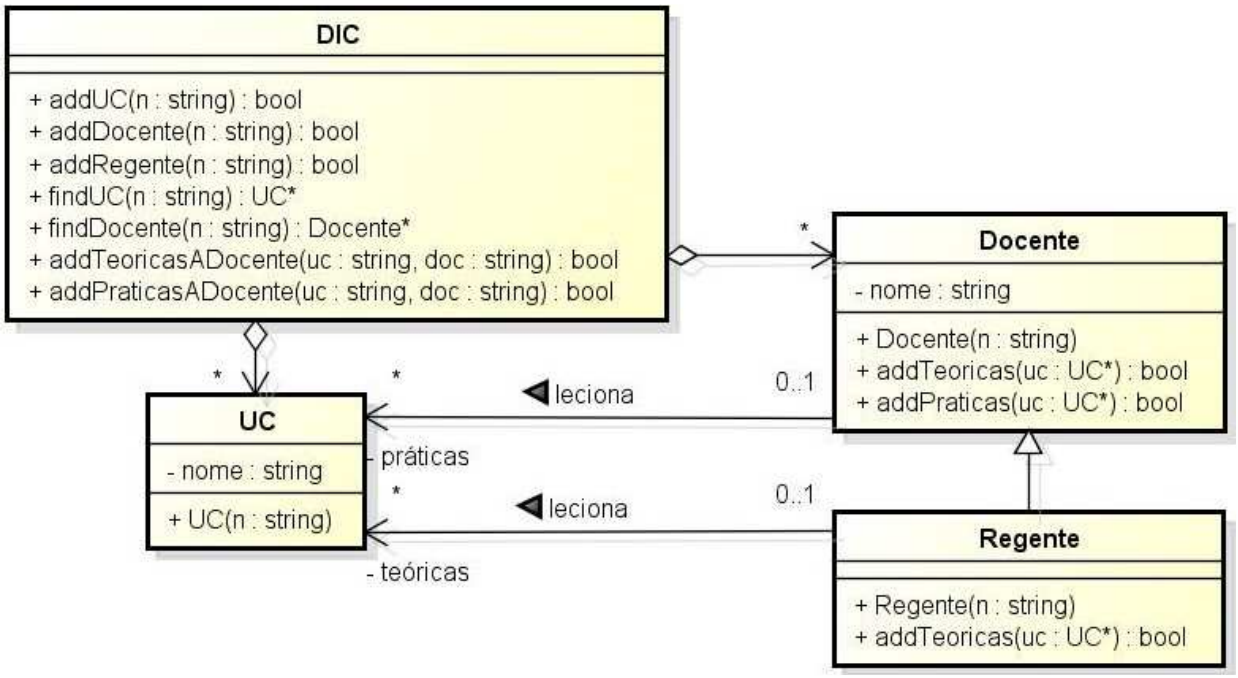
c) Diga quais os métodos construtores que estarão presentes em cada uma das classes do problema. [1.6 val.]

d) Apresente o resultado que será visualizado na saída standard com a execução do programa. [4.0 val.]

e) O que seria visualizado se o método *print* da classe *Figura* passasse a virtual? Refira-se apenas à parte da visualização que achar que se altera em relação ao output da alínea anterior. [0.7 val.]

Grupo II
(12.8 valores)

Suponha que o incumbiram de desenvolver uma pequena aplicação que permita gerir a distribuição de serviço docente do depart. de informática e comunicações (DIC). Pretende-se com essa aplicação que cada docente saiba quais as unidades curriculares (UCs) em que leciona as aulas práticas e quais as UCs em que leciona as aulas teóricas. Se as aulas práticas podem ser asseguradas por um qualquer docente, as teóricas só poderão ser asseguradas por regentes. Com o objetivo de simplificar o problema, assume-se que a UC desconhece os docentes que a lecionam (associação unidirecional) e que cada uma das suas componentes, teórica ou prática, é lecionada por inteiro por um único docente. 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 as associações, os métodos e os atributos descritos no diagrama. Não implemente quaisquer outros métodos ou atributos; apenas acrescente, nas classes em que se justifique, o operador que permita que os objetos sejam colecionáveis. [9.0 val]

   NOTA 1: Quando associa a componente teórica ou prática duma UC a um docente/regente (addTeoricas() e addPraticas()) não precisa de verificar se essas componentes já estão a ser asseguradas por outros docentes;

   NOTA 2: 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 3: 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 4: Nesta e nas restantes alíneas, não precisa de indicar as diretivas #include.

b) Acrescente ao problema os métodos que permitam mostrar o nome de todas as UCs lecionadas por um dado docente (que pode ser regente). [2.4 val.]

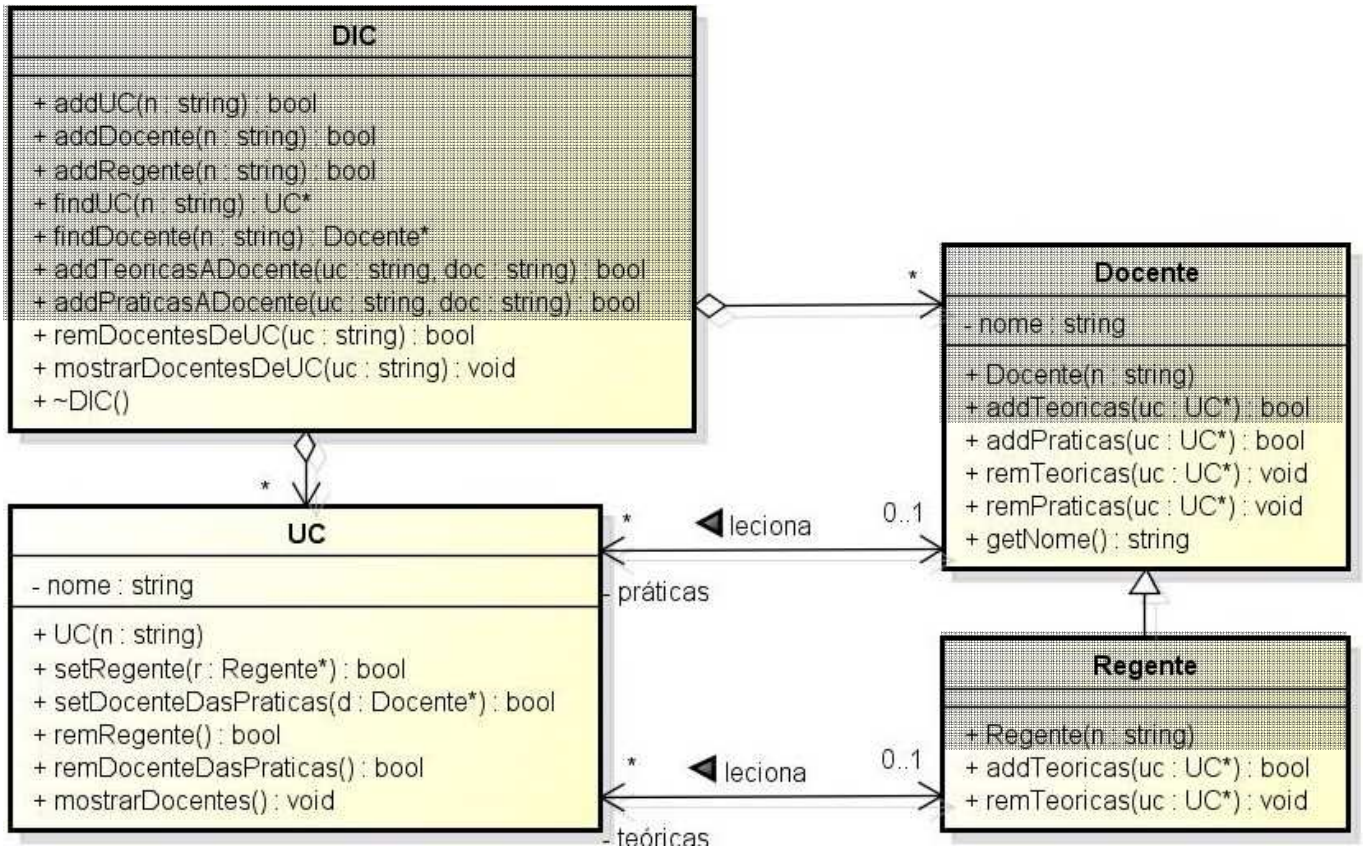
c) Escreva um pequeno *main* que faça uso de todas as funcionalidades da aplicação. [1.4 val.]

Grupo III

(10 valores)

*NOTA: O problema que se segue é um exercício suplementar e facultativo que se destina a substituir a classificação obtida durante o período letivo (miniteste+trabalho). Com a sua resolução, ou tentativa de resolução, essa componente de avaliação deixará de contar para a época de recurso, e a sua nota final será: (GI+GII+GIII)*2/3.*

A ideia agora é permitir que cada UC reconheça quem é o regente que está a lecionar as suas aulas teóricas e quem é o docente que está a lecionar as suas aulas práticas (repare que as associações entre essas entidades são agora bidirecionais). Adicionalmente, pretende-se acrescentar à aplicação a capacidade de dissociar os docentes/regentes das UCs, de mostrar os docentes de uma determinada UC e de destruição adequada da entidade centralizadora (DIC). Segue-se o diagrama de classes UML que descreve de forma precisa a nova solução que agora se pretende.



Este exercício destina-se a acrescentar novas funcionalidades ao problema que começou a implementar no grupo de questões anterior. Por isso, nas suas repostas deve ter sempre em conta o código que já implementou anteriormente para o problema.

Defina então em C++ novamente toda a classe UC e acrescente às restantes os métodos que asseguram as novas funcionalidade do problema (métodos não sombreados do diagrama). [10 val]

NOTA: Agora a associação de uma componente (teórica ou prática) da UC a um docente/regente (addTeoricas() e addPraticas()) não deve ser permitida se essa componente já estiver a ser assegurada por outro docente;

ANEXO A

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

class Figura{
	int x, y;
public:
	Figura(){ x = y = 0;
		  cout << "Criada Figura na origem" << endl;
	}
	Figura(int x, int y){
		this->x = x;
		this->y = y;
		cout << "Criada ";
		print();
	}
	void print(){ cout << "Figura na posicao (" << x << ',' << y << ')' << endl; }
	~Figura(){ cout << "Destruida ";
		   print();
	}
};

class Desenho{
	Figura *figuras;
	int n, max;
public:
	Desenho(){
		n = max = 0;
		cout << "Criado Desenho que nao vai poder incluir figuras" << endl;
	}
	Desenho(int m){
		max = m; n = 0;
		figuras = new Figura[max];
		cout << "Criado Desenho com capacidade para incluir " << max
		     << " figuras" << endl;
	}
	void addFigura(const Figura &f){ if (n<max) figuras[n++] = f; }
	void print(){
		for (int i = 0; i<n; i++)    figuras[i].print();
	}
	~Desenho(){
		if (figuras!=NULL) delete[] figuras;
		cout << "Desenho com " << n << " figuras destruido" << endl;
	}
};

void main(){
	cout << "####Criacao 1###" << endl;
	Figura f0, f1(2,3);
	cout << "####Criacao 2###" << endl;
	Desenho d(3);
	cout << "####Adicao###" << endl;
	d.addFigura(f0);
	d.addFigura(f1);
	cout << "####Print###" << endl;
	d.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();
};
```