INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Engenharia Informática
Informática de Gestão

Época de Recurso – 13 de fevereiro de 2017

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 o Grupo III em folha de exame separada.*

Grupo I

(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) Diga o que entende por classe abstrata, se alguma das classes do problema é abstrata e de que forma é que se podem definir classes abstratas em C++. [1 val.]

b) Deveríamos ter definido, em ambas as classes, o método toString() como constante? Justifique. [.6 val]

c) Apresente o resultado que será visualizado na saída standard após a execução do programa. [1.8 val.]

d) Apresente o resultado que seria visualizado na saída standard se os métodos destrutor e toString(), da classe `Base`, fossem virtuais. Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea anterior. [.6 val.]

e) Se definíssemos o método construtor da classe `AErasmus` como se segue, estaríamos a introduzir, com essa alteração, dois erros. Diga quais. [1 val.]

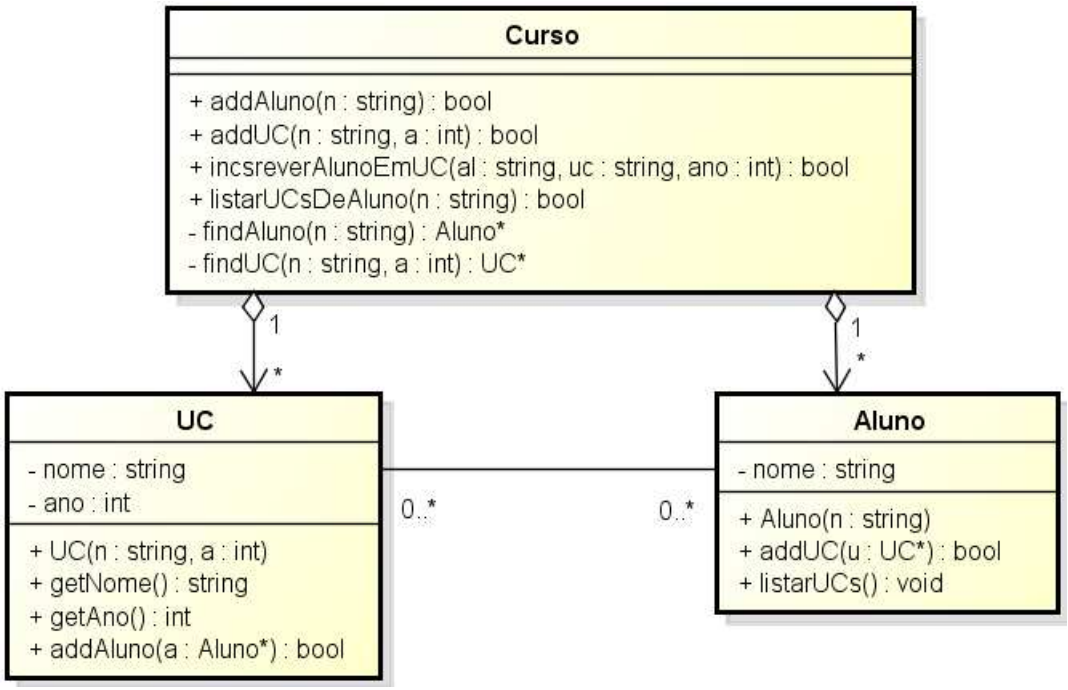
```
AErasmus(const string &nom, const string &p) {
      nome = nom;
      pais = p;
      cout << "Erasmus de " << p << endl;
}
```

Grupo II

(15 valores)

Suponha que pretende desenvolver uma pequena aplicação que lhe permita gerir as inscrições dos alunos nas diferentes unidades curriculares (UCs) do seu curso. No sentido de simplificar a solução, considere que o aluno é caracterizado unicamente pelo seu nome, e que cada UC é caraterizada quer pelo nome quer pelo ano letivo a que diz respeito. Tenha, portanto, em atenção, que no sistema vão coexistir várias UCs com o mesmo nome, mas de anos diferentes. Para além das funcionalidades básicas de gestão das entidades, a aplicação deverá permitir listar as UCs em que se encontra inscrito um dado aluno do curso.

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. [12 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.

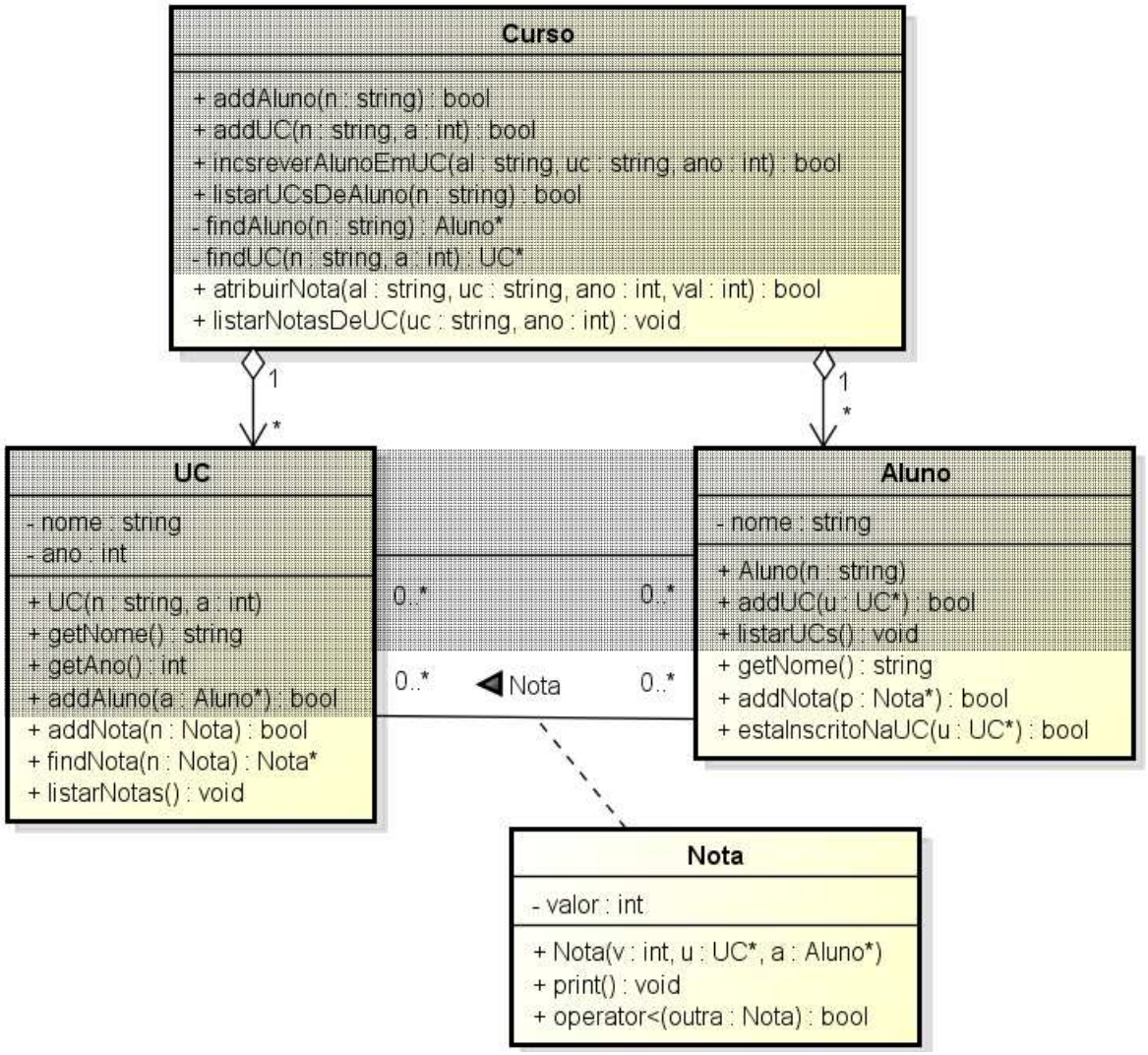
b) Acrescente ao problema os métodos que permitam remover do sistema uma dada unidade curricular, garantido a eliminação prévia de qualquer associação que possa existir dessa UC a outras entidades dos sistema. [3 val.]

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 passará a ser determinada pela seguinte fórmula: (GI+GII+GIII)*2/3.*

No seguimento do exercício do grupo II, pretende-se agora acrescentar à aplicação a capacidade de registar as notas dos alunos nas diferentes UCs. 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++ a nova classe Nota e acrescente às restantes os métodos e atributos que assegurem as novas funcionalidades do problema (métodos não sombreados do diagrama). [10 val]

NOTA: A adição de uma nova nota a uma UC apenas deverá ser permitida se o aluno em causa estiver já inscrito nessa UC.

.

ANEXO A

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

class Aluno {
	string nome;
public:
	Aluno(const string &n) {
		cout << "Novo aluno ";
		nome = n;
	}
	string toString() { return "Nome: " + nome; }
	~Aluno() { cout << nome << " cancelou a sua matricula" << endl; }
};

class AErasmus : public Aluno {
	string pais;
public:
	AErasmus(const string &nom, const string &p) :Aluno(nom) {
		pais = p;
		cout << "Erasmus de " << p << endl;
	}
	string toString() { return Aluno::toString() + " Aluno de " + pais; }
	~AErasmus() { cout << "O aluno de " << pais << " "; }
};

void main() {
	Aluno *v[3];
	cout << "OUTPUT 0:" << endl;
	v[0] = new AErasmus("Pirlo", "Italia");
	cout << "OUTPUT 1:" << endl;
	AErasmus ae("Paco", "Espanha");
	v[1] = &ae;
	cout << "OUTPUT 2:" << endl;
	Aluno a("Ana");
	v[2] = &a;
	cout << endl << "OUTPUT 3:" << endl;
	int i = 0;
	cout << "0: " << v[0]->toString() << endl;
	cout << "1: " << v[1]->toString() << endl;
	cout << "2: " << v[2]->toString() << endl;
	cout << "OUTPUT 4:" << endl;
	delete v[0];
	cout << "OUTPUT 5:" << endl;
}
```

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();
};
```

Paulo Gouveia