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 – 30 de janeiro de 2019

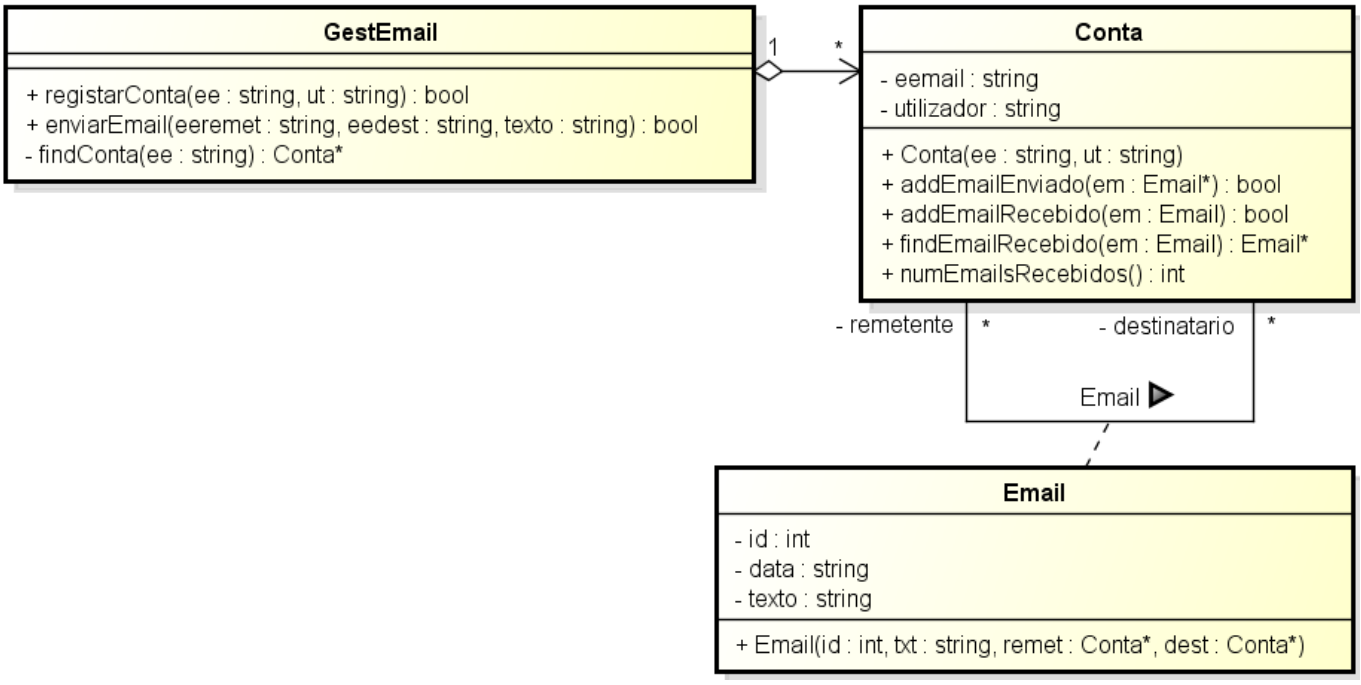
Notas importantes:

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

[14 valores]

Com o diagrama UML que se segue pretende-se modelar um sistema simplificado de troca de correio eletrónico (email) dentro de uma mesma organização (*intranet*). O sistema de gestão de emails em causa é responsável por gerir um conjunto de contas de correio e as mensagens (de texto) que entre elas são trocadas. O envio de correio eletrónico será assim a principal operação assegurada pelo sistema, devendo consistir na composição do email, seguindo-se a salvaguarda do mesmo quer na conta do destinatário, quer na conta do remetente. (Para um melhor entendimento do diagrama, leia-se sempre a dupla vogal 'ee' com o significado de 'endereço eletrónico'.)



a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e atributos descritos no diagrama, e tendo ainda em conta todas as seguintes considerações: [11.4 val.]

- *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;*
- *O id do email deverá ser automaticamente atribuído pelo sistema e ser único dentro de cada inbox (emails recebidos por uma conta) usando o código 1 para o 1º elemento, o 2 para o 2º, e assim sucessivamente;*

- *Admita que a aplicação está a correr em tempo real. Por isso, para a data do email deverá ser usada a função* `agora()`*, a qual se assume devolver uma string com a data corrente, retirada do sistema operativo;*
- *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;*
- *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;*
- *Nesta e nas restantes alíneas, não precisa de indicar as diretivas #include.*

b) Acrescente ao problema os métodos que permitam apresentar na saída standard todos os *emails* recebidos numa dada conta, com indicação do endereço de *email* do remetente, da data e do respetivo conteúdo. [1.8 val.]

c) Implemente um pequeno *main* que faça uso de todas as funcionalidades do sistema de gestão de *emails* que acabou de implementar. [0.8 val.]

Grupo II

[6 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) Quais das classes dispõem do construtor por defeito? E quais delas dispõem do construtor de cópia? [0.8 val.]

b) Apresente o resultado que será visualizado na saída standard com a execução do programa. [2.7 val.]

c) Qual o resultado que seria visualizado na saída standard se o método `print()` da classe `Animal` não fosse virtual? Apresente apenas a parte da visualização que resultaria diferente em relação ao output da alínea anterior. [0.8 val.]

d) Tendo em conta as classes definidas no problema, dê uma breve explicação do erro cometido em cada uma das instruções que se seguem: [0.9 val.]

```
(1) Animal *a = new Animal("Farrusco");
(2) Animal *a = new Domestico();  Domestico *d; d = a;
(3) Domestico d("Farrusco"); DeEstimacao &de = d;
```

e) Faça as alterações que achar necessárias nas classes do problema para que a linha de código que se segue possa surgir na função *main*. [0.8 val.]

```
DeEstimacao de;
```

ANEXO A

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

class Animal {
public:
	Animal() {cout << "Novo Animal ";}
	virtual void print() const { cout << "->Indiferenciado" << endl; }
	virtual ~Animal() { cout << "->Menos um animal" << endl; };
};


class Domestico : public Animal {
	string nome;
public:
	Domestico(const string &n) : nome(n) {
		cout << "de nome " << nome << endl;
	}
	void print() const {
		cout << "->"<<nome<<endl;
	}
	~Domestico() { cout << "->Xau " << nome; }
};


class DeEstimacao : public Domestico {
public:
	DeEstimacao(const string &n) : Domestico(n) {
		cout << "Novo Domestico " << endl;
	}
	void print() const {
		cout << "->Bicho mimado";
		Domestico::print();
	}
	~DeEstimacao() { cout << "->Kaput"; };
};


void main() {
	Animal *a1 = new DeEstimacao("Boby");
	cout << "-1-" << endl;
	Domestico *a2 = new Domestico("Lacy");
	cout << "-2-" << endl;
	DeEstimacao a3("Kitty");
	cout << "-3-" << endl;
	a1->print();
	cout << "-4-" << endl;
	a2->print();
	cout << "-5-" << endl;
	a3.print();
	cout << "-6-" << endl;
	delete a1;
	cout << "-7-" << endl;
	delete a2;
	cout << "-8-" << 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();
};
```