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 – 18 de janeiro de 2016

Notas importantes:

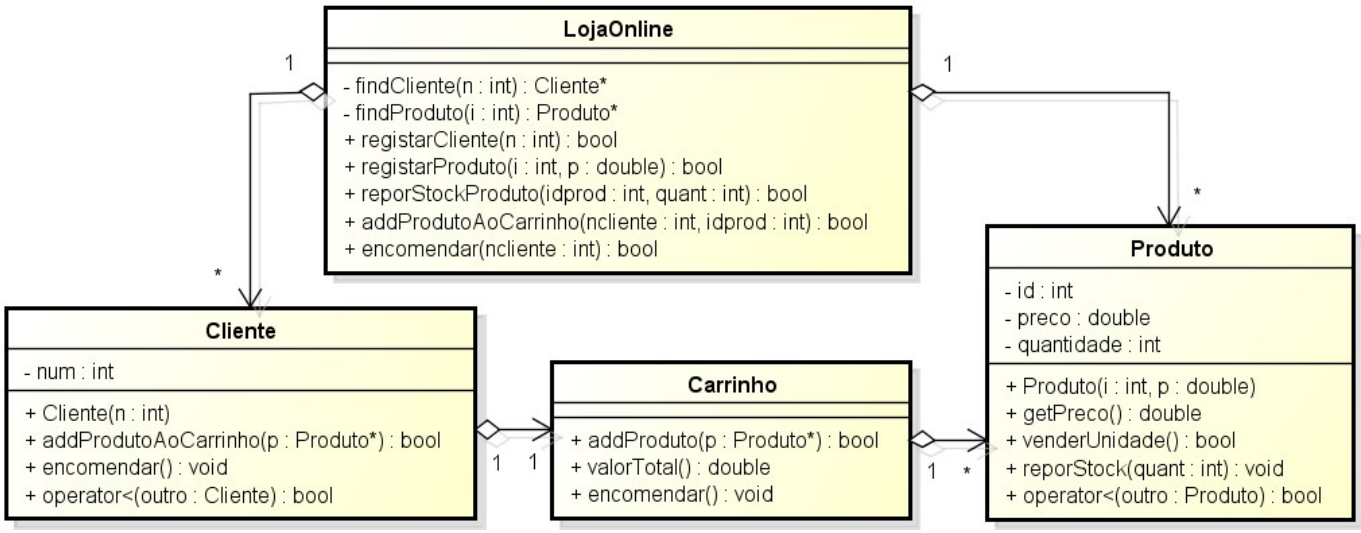
1. *A duração da prova é de 1 hora e 30 minutos.*
2. *Comece por preencher a zona reservada à sua identificação na folha de exame.*
3. *Resolva os dois grupos em folhas de exame separadas.*

Grupo I

[13.2 valores]

Uma empresa que se dedica ao comércio eletrónico necessita de uma aplicação que lhe permita implementar o típico carrinho de compras virtual, com o qual o cliente poderá constituir a sua encomenda. Relativamente a cada produto, apenas deve ser registado no sistema o código que o identifique, o preço unitário e a quantidade de unidades existente em stock, e para que um produto possa ser adicionado ao carrinho é necessário que o mesmo exista em stock, devendo este ser decrementado em uma unidade como consequência dessa operação. Depois de colocados os produtos no carrinho, efetua-se a encomenda, a qual traduzir-se-á simplesmente no esvaziamento do carrinho (repare que quando se concretiza a encomenda, passa-se para outra fase do processo de compra, não implementada na presente aplicação, ficando desde logo o carrinho disponível para se poder iniciar uma nova encomenda).

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. [9.3 val.]

> *NOTA 1: O método* venderUnidade() *da classe* Produto *deve limitar-se a decrementar, se possível, o atributo quantidade; devendo depois o seu valor de retorno dar indicação se a operação foi ou não bem sucedida (repare que pode não haver qualquer unidade para vender);*

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 recheio do carrinho de compras de um determinado cliente, com indicação final do valor total da encomenda. [2.1 val.]

c) Acrescente ao problema os métodos que permitam excluir um dos produtos do carrinho de compras de um dado cliente (de forma a cancelar a sua compra). [1.8 val.]

Grupo II
[6.8 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) Quantos métodos construtores estarão presentes em cada uma das classes do problema. [0.3 val.]

b) Alguma das classes do problema é abstrata? Diga o que entende por classe abstrata. [1.0 val.]

c) Considerando as seguintes instanciações: `A a1("um"), a2("dois");` diga, justificando, se a instrução `if(a1<a2) cout<<c1;` é ou não válida no problema considerado. [0.7 val.]

d) Apresente o resultado que será visualizado na saída standard após a execução do programa. [3.8 val.]

e) Apresente o resultado que seria visualizado na saída standard se nenhum dos métodos da classe A fosse virtual (incluindo o método destrutor). Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea anterior [1.0 val.]

ANEXO A

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


class A{
	string msg;
public:
	A(): msg("objeto da classe A") {cout << "A()" << endl;}
	A(const string &s): msg(s) {cout << "A(str)" << endl; }
	virtual string toString() { return msg; }
	virtual ~A() {cout << "~A()" << endl; }
};


class B: public A {
	string msg;
public:
	B(): msg("objeto da classe B") { cout << "B()" << endl; }
	B(const string &s1, const string &s2): A(s1), msg(s2){
		cout << "B(str,str)" << endl;
	}
	string toString() { return msg + " com sub" + A::toString(); }
	~B() { cout << "~B()" << endl; }
};


class C: public B {
	string msg;
public:
	C(): msg("objeto da classe C") { cout << "C()" << endl; }
	C(const string &s1, const string &s2, const string &s3): B(s1,s2), msg(s3) {
		cout << "C(str,str,str)" << endl;
	}
	string toString() { return msg + " com sub" + B::toString(); }
	~C() {cout << "~C()" << endl; }
};


void main() {
	cout << "-----1" << endl; //delimitador de output
	C c("solo impermeavel", "-regioes", "Area");
	cout << "-----2" << endl;
	C *pc = new C;
	cout << "-----3" << endl; //delimitador de output
	A *pa = new C;
	cout << "-----4" << endl; //delimitador de output
	cout << c.toString() << endl;
	cout << pc->toString() << endl;
	cout << pa->toString() << endl;
	cout << "-----5" << endl; //delimitador de output
	delete pa;
	cout << "-----6" << endl; //delimitador de output
	delete pc;
	cout << "-----7" << endl; //delimitador de output
}
```

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();
};
```