INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e de Gestão

Departamento de Informática e Comunicações

Engenharia Informática / Informática de Gestão
2º Ano
Época de Recurso

Programação Orientada por Objectos — Data: 13/2/2010
1º Semestre — Duração: 2h00

Grupo I
(10 valores)

Suponha que pretende desenvolver uma pequena aplicação que lhe permita gerir as vigilâncias que os vários docentes têm de assegurar durante uma época de exames. No sentido de simplificar a solução, considere que o docente é caracterizado apenas pelo seu nome, e cada prova de exame pelo nome da disciplina a que respeita e pela data de realização. Tenha ainda em conta que a vigilância de cada exame é, por norma, assegurada por vários docentes, e um mesmo docente tem a seu cargo um máximo de 10 vigilâncias. Considere, por fim, que podem existir vários exames para uma mesma disciplina, desde que em dias diferentes.

a) Apresente, através de um diagrama de classes em UML, a solução por si idealizada para sustentar a aplicação descrita, indicando para cada classe apenas os atributos estritamente necessários, um método construtor e outros componentes que considere essenciais ao funcionamento da solução (*getters*, *setters* e, nas classes em que se justifique, o operador que permita que os objectos sejam coleccionáveis). [2val]

b) Codifique em C++ a sua solução, tal como a descreveu no diagrama da alínea anterior. [3 val.]

NOTA: As colecções deverão ser implementadas com base no template de classes *Coleccao* que utilizou nas aulas de POO. Para efeitos de consulta, disponibilizam-se os protótipos dos seus principais métodos*:*

```
template<class K>
class Coleccao: 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();
};
```

c) Implemente o(s) método(s) necessário(s) para registar um novo exame na aplicação. [1 val.]

d) Implemente o(s) método(s) necessário(s) para mostrar todas a vigilâncias de um dado docente. [2 val.]

e) Implemente o(s) método(s) necessário(s) para remover um dado exame. [2 val.]

Grupo II

(10 valores)

Considere o seguinte programa:

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

class Cliente{
	string nome;
public:
	Cliente(const string & name): nome(name){}
	void print()const{cout<<nome<<endl;}
};

class Conta{
protected:
	int numero;
	double saldo;
	Cliente *titular1;
	static int numDeContas;
public:
	Conta(Cliente *t): titular1(t){
		numero=++numDeContas;
		saldo=0.0;
		cout<<"Criada conta n."<<numero<<endl;
	}
	virtual void depositar(double val) {if(val>0.0) saldo+=val;}
	virtual void levantar(double val) {
		if(val>0.0 && val<=saldo) saldo-=val;
	}
	virtual void print() const {
		cout<<"Conta n."<<numero<<" saldo:"<<saldo<<endl;
		cout<<"com 1o titular: ";
		titular1->print();
	}
	virtual ~Conta(){cout<<"Fechada\n";}
};
int Conta::numDeContas=0;
```

```
class ContaAPrazo: public Conta{
      Cliente *titular2;
public:
      ContaAPrazo(Cliente *t1, Cliente *t2): Conta(t1), titular2(t2){
            cout<<"<Conta a Prazo>\n";
      }
      void depositar(double val) {
            if(val>=100.0) Conta::depositar(val);
            else cout<<"Deposito invalido\n";
      }
      void print() const {
            Conta::print();
            cout<<"com 2o titular: ";
            titular2->print();
      }
      ~ContaAPrazo(){cout<<"Conta a Prazo ";}
};

void main(){
      Cliente cl1("Ana"), cl2("Rita");
      Conta *c=new Conta(&cl1);
      delete c;
      c=new ContaAPrazo(&cl1,&cl2);
      c->depositar(50);
      c->levantar(10);
      c->print();
      delete c;
}
```

a) Identifique o tipo de relação existente entre as diferentes classes definidas no programa. [1 val.]

b) Apresente o resultado que será visualizado na saída standard com a execução do programa. [4 val.]

c) Diga que alterações é que teria no resultado visualizado se na classe `Conta` [2 val.]

i) o método "`void depositar(double val)`" não fosse virtual.

ii) o método "`void levantar(double val)`" não fosse virtual.

iii) o método "`void print()`" não fosse virtual.

iv) o método "`~Conta()`" não fosse virtual.

d) Se acrescentássemos à função `main()` a linha de código que se segue, daí resultaria algum erro na nossa aplicação? Justifique. [1 val.]

```
ContaAPrazo contas[1000];
```

e) Acrescente ao programa uma classe para um novo tipo de conta, de um só titular, denominada ContaAOrdem. Esse novo tipo de conta deverá ser caracterizado por não permitir que o saldo fique, em nenhum momento, a zero e pela obrigatoriedade de se efectuar um depósito (não importa o montante) no momento da sua abertura (criação da conta). [2 val.]