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: 11/02/2012
1º Semestre — Duração: 3h

Resolva os 3 grupos em folhas de exame separadas — 1h30 (Grupos I e II) + 1h30 (Grupo III - opcional)

Grupo I
(10 valores)

No Anexo A do presente enunciado encontra-se o código de um programa em C++ que implementa um modelo muito simples de contas bancárias, com apenas algumas funcionalidades elementares. Depois de o analisar com a devida atenção, responda às seguintes questões.

a) Alguma das classes do problema é abstracta? Se sim, diga qual e por que motivo a considera abstracta. [0.7 val.]

b) Diga quais os métodos construtores de que dispõe cada uma das classes (os explícitos e os implícitos). [1.1 val.]

c) Apresente o resultado que será visualizado na saída standard após a execução do programa. [3.2]

d) E o que seria visualizado se o método destrutor da classe Conta não fosse virtual? Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea anterior. [1 val.]

e) Como se percebe, na implementação sugerida, qualquer conta a Prazo que seja criada pode ter associada uma taxa de juro diferente das restantes. Que tipo de alteração proporia à solução apresentada, de modo a garantir que todas as contas a Prazo que viessem a ser criadas passassem a ter a mesma taxa de juro. Explique, não implemente. [1 val.]

f) Considerando as seguintes instanciações: `CPEspecial c1, c2;` diga, justificando, se cada uma das instruções `c1=c2;` e `c1++;` é ou não válida no problema considerado. [1 val.]

g) Considere agora o código substituto para a função main() que a seguir se apresenta. Identifique dois erros presentes nesse código e corrija-os efectuando apenas alterações nas classes do problema. [2 v.]

```
void main(){
      CPrazo contas[10];
      for(int i=0; i<10; i++) contas[i]+=10.0; //depósitos de 10 euros
}
```

Grupo II
(10 valores)

Um importante aeroporto necessita de uma pequena aplicação que lhe permita monitorizar, a qualquer momento, todas as aeronaves que se encontram conectadas às suas portas de embarque/desembarque (*gates*). Considere que a cada uma das *gates* apenas pode estar conectada, num dado momento, uma única aeronave (ou nenhuma), para embarque ou desembarque dos respectivos passageiros ou mercadorias, caso se trate de uma aeronave de passageiros ou de carga, respectivamente. As *gates* são identificadas por um código inteiro e, quando conectadas, deverão dar ainda a indicação se estão a operar como porta de embarque ou de desembarque para as respectivas aeronaves. Cada aeronave, sendo também identificada por um código inteiro, é ainda caracterizada pela sua lotação, no caso de se tratar de uma aeronave de passageiros, ou então pela respectiva tara, caso se trate de uma aeronave de carga.

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, nas classes em que se justifique, o operador que permita que os objectos sejam coleccionáveis. [3 val.]
Não indique quaisquer outros métodos.

> *NOTA 1: No momento em que se regista uma nova aeronave no sistema, a mesma não deve ficar logo conectada a qualquer* gate*;*
>
> *NOTA 2: No momento em que se regista uma nova* gate *no sistema, também ela não deve ficar conectada a qualquer aeronave;*

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

> *NOTA 3: As colecções deverão ser implementadas com base no template de classes Coleccao ou ColeccaoHibrida que utilizou nas aulas de POO. Para efeitos de consulta, disponibilizam-se os protótipos dos seus principais métodos no Anexo B deste enunciado;*
>
> *NOTA 4: 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 5: Nesta e nas restantes alíneas, não precisa de indicar as directivas #include.*

c) Implemente os métodos necessários para adicionar à aplicação uma nova *gate*, uma nova aeronave de passageiros e uma nova aeronave de carga. [1.5 val.]

d) Implemente o(s) método(s) necessário(s) para conectar uma aeronave a uma *gate*. [2.5val.]

Grupo III

(20 valores)

NOTA: O problema que se segue é um exercício suplementar e facultativo que se destina a substituir a classificação obtida durante o período lectivo (miniteste+trabalho). Com a sua resolução, ou tentativa de resolução, essa componente de avaliação deixa de contar para a época de recurso.

As alíneas que se seguem destinam-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.

a) Implemente o(s) método(s) necessário(s) que permita(m) verificar se uma dada *gate* se encontra conectada (ou seja, se existe alguma aeronave conectada a essa gate). [2.5 val.]

b) Implemente o(s) método(s) necessário(s) para desconectar uma dada *gate*. [2.5 val.]

c) Implemente o(s) método(s) necessário(s) para remover do sistema uma dada *gate*. [2 val.]

d) Implemente o(s) método(s) necessário(s) que permita(m) listar todas as aeronaves registadas, mostrando, para cada aeronave, o seu identificador, se se encontra conectada, e o valor da tara ou da lotação. [5 val.]

e) Implemente o(s) método(s) necessário(s) que permita(m) listar as *gates* que se encontrem conectadas, mostrando, para cada *gate*, o seu identificador, se está a ser usada para embarque ou desembarque, e os dados relativos à aeronave a que se encontra conectada. [3.5 val.]

f) Implemente o método destrutor, com o comportamento adequado, para a classe Aeroporto. [2 val.]

g) Escreva uma função main() que faça uso de todas as funcionalidades do problema. [2.5 val.]

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

class Conta{
protected:
	double saldo;
public:
	Conta(): saldo(0.0){
		cout << "Abertura de Conta ";
	}
	Conta(double s): saldo(s){
		cout << endl << "Abertura com deposito inicial " << saldo;
	}
	virtual string getTipo()const =0;
	virtual ~Conta(){cout << " fechada com saldo " << saldo <<endl;}
};

class CPrazo: public Conta{
	double txJuro;
public:
	CPrazo(double tx, double dep=0.0): Conta(dep){
		txJuro=tx;
		cout << " a Prazo ";
	}
	string getTipo()const {return " a Prazo ";};
	~CPrazo(){cout << "Conta com taxa " << txJuro;}
};

class CPEspecial: public CPrazo{
public:
	CPEspecial(): CPrazo(1.5){
		saldo+=100.0;
		cout << "Especial 100" << endl;
	}
	CPEspecial(double tx): CPrazo(tx){
		saldo+=200.0;
		cout << "Especial 200";
	}
	string getTipo()const {return "Especial ";}
	~CPEspecial(){cout << getTipo();}
};


void main(){
	CPrazo *contas[3];
	contas[0]=new CPEspecial(3.5);
	contas[1]=new CPrazo(2.5, 500.0);
	CPEspecial c;
	contas[2]=&c;
	delete contas[0];
	delete contas[1];
}
```

Template Coleccao

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

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();
};
```

Template ColeccaoHibrida

```
#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 ColeccaoHibrida: 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();
};
```