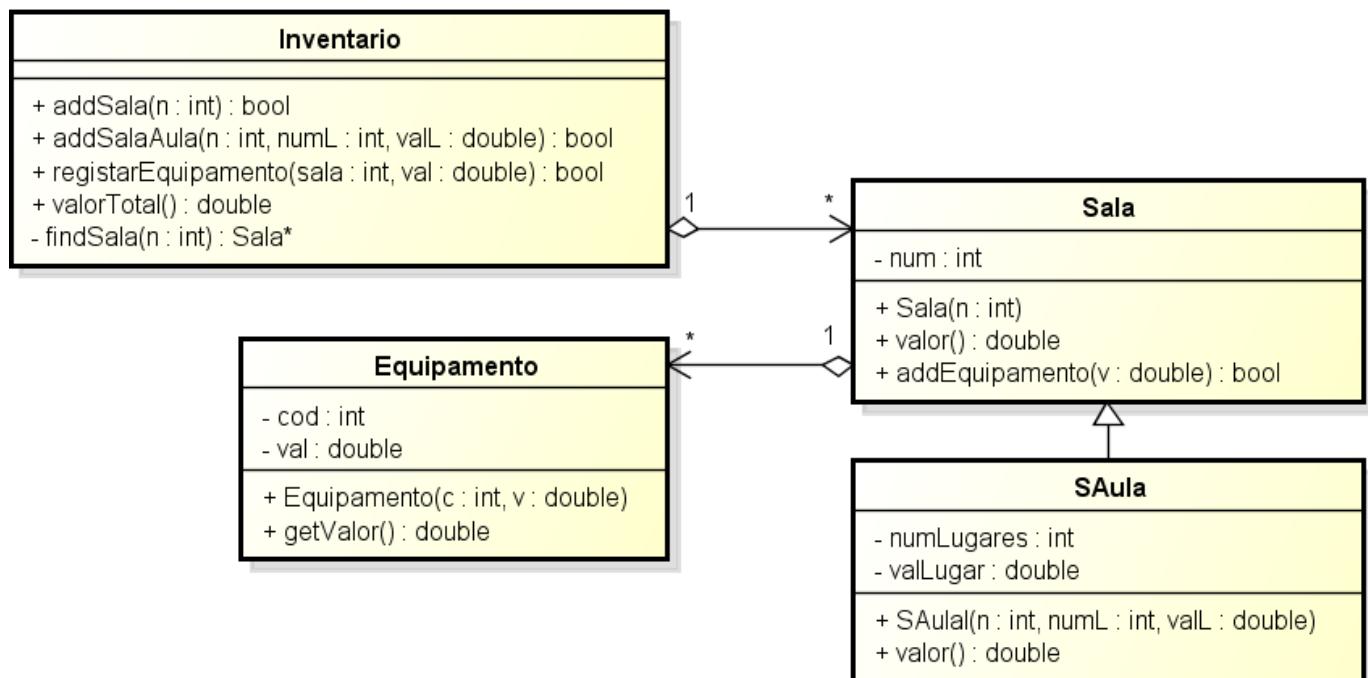


Notas importantes:

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
[15 valores]

Com o diagrama UML que se segue pretende-se modelar um sistema simplificado de gestão do inventário de uma escola. O sistema de gestão em causa é responsável por manter o registo dos diferentes bens (equipamentos) alocados às salas, cada um com um valor financeiro estimado. Tratando-se de uma sala normal, o valor financeiro que representa será simplesmente o somatório do valor dos bens que integra; mas caso se trate de uma sala de aula, a esse montante deverá ser ainda adicionado o valor estimado para o mobiliário que equipa todos os lugares de que dispõe. Cada sala de aula tem um determinado número de lugares e um valor financeiro médio estimado para equipar cada lugar. A aplicação terá, como principal funcionalidade, o cálculo automático do valor financeiro de todo o inventário da escola.



- 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 código do equipamento deverá ser automaticamente atribuído pelo sistema e ser único dentro de cada sala, usando o código 1 para o 1º equipamento, o 2 para o 2º, e assim sucessivamente;
 - As coleções deverão ser implementadas com base nos templates 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 listar na saída standard o inventário de uma dada sala, com indicação discriminada do valor de cada um dos seus equipamentos e, caso faça sentido, o valor total estimado para o mobiliário que equipa os lugares. [3.6 val.]

Grupo II
[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 quais são os métodos construtores (definidos explicitamente ou não) que estão presentes em cada uma das classes do problema. [0.6 val.]
- b) Apresente o resultado que será visualizado na saída standard com a execução do programa. [2.8 val.]
- c) Diga que consequências é que teria, naquilo que é visualizado, caso o método print() da classe A fosse virtual. [0.4 val.]
- d) Dê uma breve explicação do erro cometido quando se acrescenta ao main do programa do Anexo A uma última linha com cada um dos seguintes códigos: [1.2 val.]
 - (1) cout << objeto << endl;
 - (2) A vetor[3];
 - (3) A *p = new B();
 - (4) C *p = new C(objeto); delete p; objeto.print();

ANEXO A

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

class A {
    int val;
public:
    A(int v) :val(v) { cout << "Criado A(" << val << ')' << endl; }
    void print() { cout << "A(" << val << ")"; }
    ~A() { cout << "Destruido A" << endl; }
};

class B {
    A obj;
    static int cont;
public:
    B(): obj(++cont) {cout << "Criado " << 'B' << endl;}
    void print() { cout << "B:"; obj.print(); }
    ~B() { cout << "Destruido B" << endl; }
};

int B::cont=0;

class C {
    B *vetor;
    int nbs;
public:
    C(int n=1) {
        nbs = n;
        vetor = new B[nbs];
        cout << "Criado um C com " << nbs << " Bs" << endl;
    }
    void print() {
        for (int i = 0; i < nbs; i++)
            { cout << "C:"; vetor[i].print(); cout << endl; }
    }
    ~C() { cout << "Destruido C" << endl; delete[] vetor; }
};

void main() {
    C objeto(3);
    cout << "-1-" << endl;
    objeto.print();
    cout << "-2-" << 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();
};

};
```

Grupo I (15 valores)

- a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos ... [11.4 val.]

```

class Inventario{
    ColecaoHibrida<Sala*> salas;      ④
public:
    bool addSala(int n) {                ②
        Sala *s = new Sala(n);
        return salas.insert(s);
    }
    bool addSalaAula(int n, int numL, double valL) { ②
        Sala *sa = new SAula(n, numL, valL);
        return salas.insert(sa);
    }
    bool registarEquipamento(int sala, double val) { ③
        Sala *s = findSala(sala);
        if (s != NULL)
            return s->addEquipamento(val);
        else return false;
    }
    double valorTotal() {                ③
        double tot = 0.0;
        ColecaoHibrida<Sala*>::iterator it;
        for (it = salas.begin(); it != salas.end(); it++)
            tot += (*it)->valor();
        return tot;
    }
private:
    Sala *findSala(int n){             ②
        Sala s(n);
        return salas.find(&s);
    }
};

class Equipamento{                  ①
    int cod;
    double val;
public:
    Equipamento(int c, double v){   ①
        cod = c; val = v;
    }
    double getValor() const { return val;} ①

    bool operator<(const Equipamento &outro) const { ①
        return cod < outro.cod;
    }
};

```

```

class Sala{
    int num;
    Colecao<Equipamento> equipamentos;
public:
    Sala(int n) { num = n; } 1
    virtual double valor() { 2
        double tot=0.0;
        Colecao<Equipamento>::iterator it;
        for (it = equipamentos.begin(); it != equipamentos.end(); it++)
            tot += it->getValor();
        return tot;
    }
    bool addEquipamento(double v) { 3
        int cod = equipamentos.size() + 1;
        Equipamento e(cod, v);
        return equipamentos.insert(e);
    }
    bool operator<(const Sala &outra) const { 1
        return num < outra.num;
    }
};

class SAula: public Sala { 2
    int numLugares;
    double valLugar;
public:
    SAula(int n, int numL, double valL):Sala(n){numLugares=numL; valLugar=valL;} 2
    double valor() {return Sala::valor() + numLugares * valLugar;} 3
};

```

b) Acrescente ao problema os métodos que permitam apresentar na saída [3.6 val.]

```

bool Inventario::listarSala(int sala) { 3
    Sala *s = findSala(sala);
    if (s != NULL) {
        s->listar();
        return true;
    }
    else return false;
}
virtual void Sala::listar() const{ 4
    Colecao<Equipamento>::iterator it;
    for (it = equipamentos.begin(); it != equipamentos.end(); it++)
        it->print();
}
void SAula::listar() const{ 3
    Sala::listar();
    cout<<"<Equipamento dos lugares> Valor estimado: "<<numLugares*valLugar
        << " euros." << endl;
}
void Equipamento::print() const{ 2
    cout<<"Equipamento "<<cod<< " Valor estimado: " << val << " euros." << endl;
}

```

Grupo II (5 valores)

- a) Diga quais são os métodos construtores (definidos explicitamente ou não) que... [0.6 val.]
*Nas classes A e C estão presentes quer o construtor de conversão quer o de cópia;
Já na classe B estão presentes o construtor por defeito e o de cópia.*
- b) Apresente o resultado que será visualizado na saída standard com a execução do programa. [2.8 val.]

Criado A(1)	2
Criado B	1
Criado A(2)	2
Criado B	1
Criado A(3)	2
Criado B	1
Criado um C com 3 Bs	4 (2 se na pos. errada)
-1-	
C:B:A(1)	2
C:B:A(2)	2
C:B:A(3)	2
-2-	
Destruido C	3 (1 se na pos. errada)
Destruido B	1
Destruido A	1
Destruido B	1
Destruido A	1
Destruido B	1
Destruido A	1

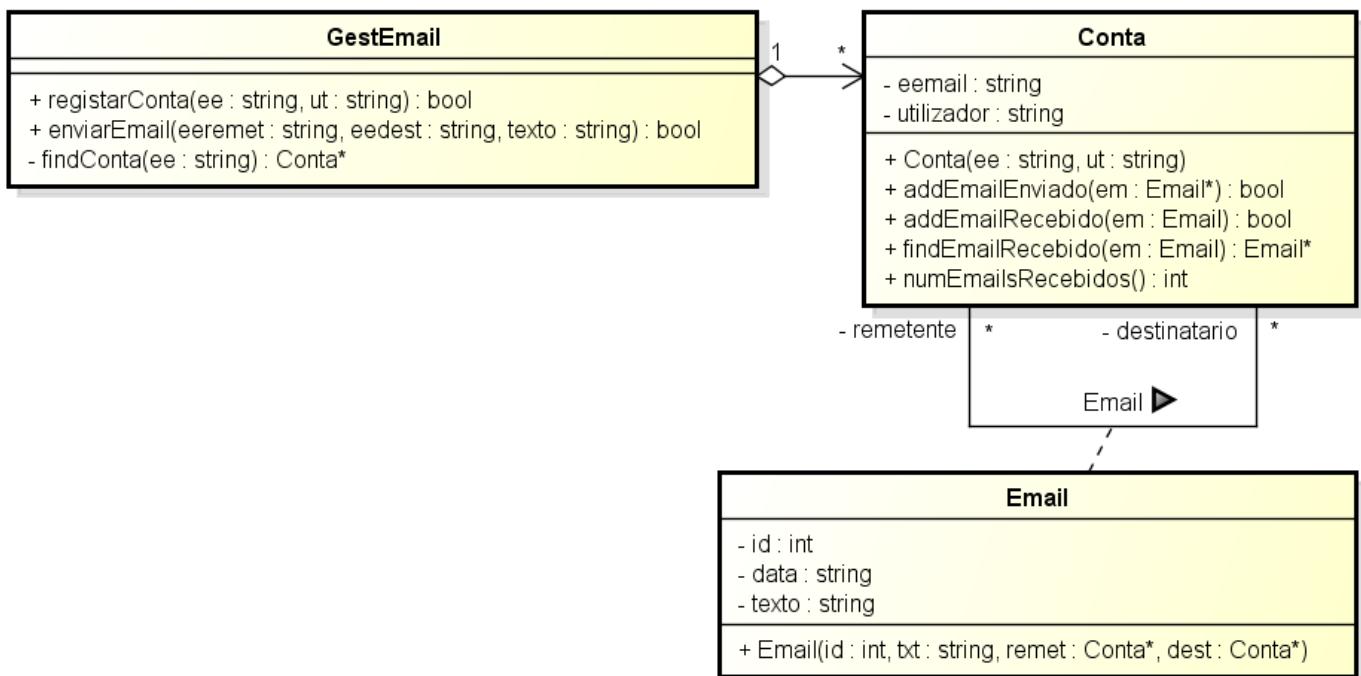
- c) Diga que consequências é que teria, naquilo que é visualizado, o método `print()` ... [0.4 val.]
Nada se alteraria (estamos na presença de relações de agregação, não de herança).
- d) Dê uma breve explicação do erro cometido quando se acrescenta ao `main` do programa: [1.2 val.]
- (1) `cout << objeto << endl;`
O operador insensor (<<) não se encontra definido para operandos do tipo C.
- (2) `A vetor[3];`
A classe A não dispõe de construtor por defeito.
- (3) `A *p = new B();`
Está-se a tentar fazer uma conversão ascendente entre apontadores, só possível, se B derivasse de A.
- (4) `C *p = new C(objeto); delete p; objeto.print();`
É criado dinamicamente um objeto da classe C, por cópia de um outro que já existe, ficando os dois objetos a partilhar o mesmo vetor de objetos B; esse array é depois eliminado com a eliminação do objeto criado; logo, o método `print()` do objeto inicial vai tentar aceder a um array que já não existe.

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("Bob");
    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();
};
```

Grupo I (14 valores)

- a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos ... [11.4 val.]

```
class Email{
    int id;
    string data;
    string texto;
    Conta* remetente;
    Conta* destinatario;
public:
    Email(int id, string txt, Conta *remet, Conta *dest): texto(txt){
        this->id = id; remetente = remet; destinatario = dest;
        data = agora();
    }

    bool operator<(const Email &outro) const {
        return id < outro.id;
    }
};

class Conta{
    string eemail;
    string utilizador;
    Colecao<Email> inbox;
    Colecao<Email*> enviadas;
public:
    Conta(string ee, string ut) : eemail(ee), utilizador(ut) {}
    bool addEmailEnviado(Email *em) {
        return enviadas.insert(em);
    }
    bool addEmailRecebido(const Email &em) {
        return inbox.insert(em);
    }
    Email *findEmailRecebido(const Email &em){
        return inbox.find(em);
    }
    int numEmailsRecebidos() const { return inbox.size(); }
    bool operator<(const Conta &outra) const {return eemail<outra.eemail;}
};

class GestEmail{
    Colecao<Conta> contas;
public:
    bool registrarConta(string ee, string ut){
        Conta c(ee, ut);
        return contas.insert(c);
    }
}
```

```

bool enviarEmail(string eeremet, string eedest, string texto) {
    Conta *rem = findConta(eeremet);
    if (rem != NULL) {
        Conta *dest = findConta(eedest);
        if (dest != NULL) {
            int id = dest->numEmailsRecebidos()+1;
            Email em(id, texto, rem, dest);
            if (dest->addEmailRecebido(em))
                return rem->addEmailEnviado(dest->findEmailRecebido(em));
            else return false;
        } else return false;
    } else return false;
}
private:
    Conta *findConta(string ee){
        Conta c(ee, "");
        return contas.find(c);
    }
};

```

b) Acrescente ao problema os métodos que permitam apresentar na saída [1.8 val.]

```

void GestEmail::mostrarEmailsRecebidos(string ee) {
    Conta *c = findConta(ee);
    if (c != NULL) c->mostrarEmailsRecebidos();
}
void Conta::mostrarEmailsRecebidos() {
    Colecao<Email>::iterator it;
    for (it = inbox.begin(); it != inbox.end(); it++)
        it->print();
}
void Email::print() const {
    cout << "De: " << remetente->getEEEmail() << " Data: " << data;
    cout << texto << endl;
}
string Conta::getEEEmail() const { return eemail; }

```

c) Implemente um pequeno *main* que faça uso de todas as funcionalidades ... [0.8 val.]

```

void main(){
    GestEmail ge;
    ge.registarConta("arita@ipb.pt", "Ana Rita");
    ge.registarConta("asofia@ipb.pt", "Ana Sofia");
    ge.enviarEmail("arita@ipb.pt", "asofia@ipb.pt", "Ola!");
    ge.mostrarEmailsRecebidos("asofia@ipb.pt");
}

```

Grupo II (6 valores)

a) Quais das classes dispõem do construtor por defeito? E quais delas dispõem do construtor de cópia?

[0.8 val.]

Apenas a classe Animal dispõe do construtor por defeito, e em todas as classes está presente o construtor por cópia.

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

```
Novo Animal de nome Boby
Novo Domestico
-1-
Novo Animal de nome Lacy
-2-
Novo Animal de nome Kitty
Novo Domestico
-3-
->Bicho mimado->Boby
-4-
->Lacy
-5-
->Bicho mimado->Kitty
-6-
->Kaputt->Xau Boby->Menos um animal
-7-
->Xau Lacy->Menos um animal
-8-
->Kaputt->Xau Kitty->Menos um animal
```

c) Apresente o resultado que seria visualizado na saída standard se o método print() ... [0.8 val.]

```
-3-
->Indiferenciado
-4-
```

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");`

A classe Animal não dispõe do construtor de conversão que está a ser usado na sua instanciação.

(2) `Animal *a = new Domestico(); Domestico *d; d = a;`

A conversão descendente (downcast) que se está a tentar fazer na atribuição `d = a` só seria possível na forma explícita, usando um operador de conversão.

(3) `Domestico d("Farrusco"); DeEstimacao &de = d;`

Também aqui se está a tentar fazer uma conversão descendente, só possível, como dissemos, na forma explícita.

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::DeEstimacao(): Domestico("") {}
```

Outra opção válida, seria:

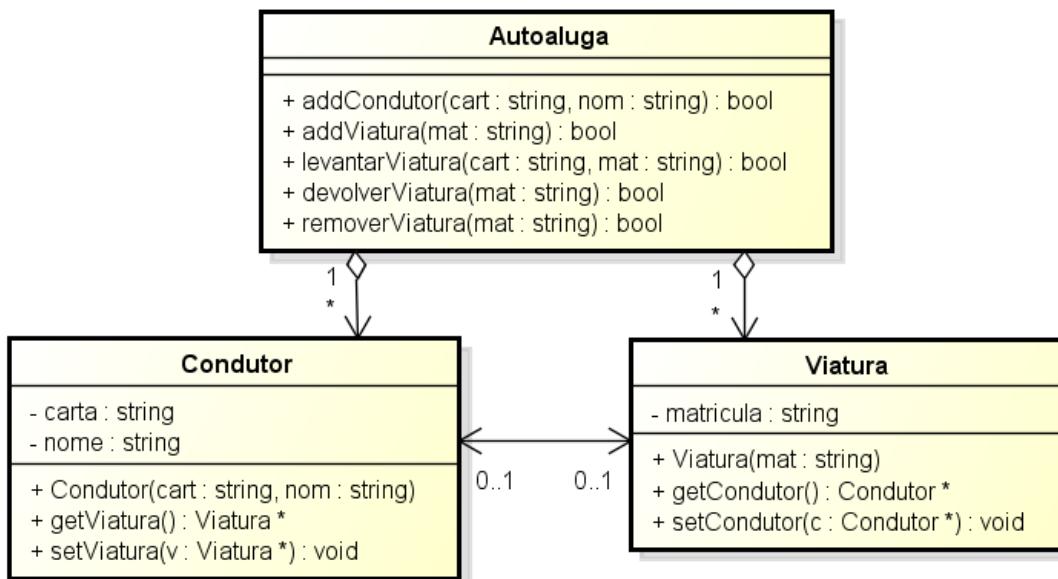
```
DeEstimacao::DeEstimacao(){}
Domestico::Domestico(): nome("") {}
```

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
[15 valores]

A Autoaluga é uma empresa de aluguer de automóveis que, encontrando-se em franca expansão, necessita urgentemente de uma aplicação que lhe permita, a todo o momento, monitorizar as viaturas que se encontrem alugadas. 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 e um ou outro método auxiliar de que precise, definindo-os, nesse caso, como privados. [11 val.]

NOTA 1: Um condutor só poderá levantar uma viatura se já estiver registado na aplicação, não estiver já na posse de outra viatura e se aquela que pretenda estiver disponível;

NOTA 2: Uma viatura só poderá ser removida do sistema se a mesma não estiver associada a nenhum condutor;

NOTA 3: 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 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 diretivas #include.

- b) Acrescente ao problema os métodos que permitam apresentar na saída standard as viaturas que estejam a ser utilizadas (alugadas), mostrando para cada uma delas a respetiva matrícula, e o nome e carta de condução de quem a está a utilizar. [4 val.]

Grupo II

[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) Desenhe o diagrama de classes UML do problema, indicando somente as classes, relações e respetivos atributos. Não inclua, portanto, qualquer método. [1.0 val.]
- b) Indique três métodos (construtores ou não) que, embora não definidos explicitamente, vão estar presentes em ambas as classes do problema. [0.9 val.]
- c) Apresente o resultado que será visualizado na saída standard após a execução do programa. [2.4 val.]
- d) Apresente o resultado que seria visualizado na saída standard se nenhum dos métodos da classe base fosse virtual. [0.7 val.]

ANEXO A

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

class Pessoa{
    string nome;
public:
    Pessoa(const string &n):nome(n) {cout << "Criado " << nome << endl; }
    string getNome() { return nome; }
    virtual void print() { cout << "Solteiro"; }
};

class PCasada: public Pessoa {
    Pessoa *conjugue;
    PCasada(const string &n, PCasada *c):Pessoa(n) {
        conjugue = c;
        cout << "Conjuge 1" << endl;
    }
public:
    PCasada(const string &n, const string &nc):Pessoa(n){
        conjugue = new PCasada(nc,this);
        cout << "Conjuge 2" << endl;
    }
    void operator+=(PCasada &outra) { conjugue = &outra; }
    void print() { cout << getNome() << " + " << conjugue->getNome() << endl; }
};

void main() {
    cout << "-0-" << endl;
    PCasada *p1=new PCasada("Manuel","Maria");
    cout << "-1-" << endl;
    p1->print();
    cout << "-2-" << endl;
    PCasada p2("Ana", "Luis");
    cout << "-3-" << endl;
    p2.print();
    *p1 += p2;
    p2 += *p1;
    cout << "-4-" << endl;
    p1->print();
    cout << "-5-" << endl;
    p2.print();
}
```

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();
};
```

Grupo I (15 valores)

a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e ... [11 val.]

-44-

```
class Condutor{      -6-          1
    string carta, nome;
    Viatura *viatura;          1
public:
    Condutor(string cart, string nom) :carta(cart), nome(nom){viatura=NULL;} 1
    Viatura *getViatura(){return viatura;} 1
    void setViatura(Viatura *v){viatura=v;} 1
    bool operator<(const Condutor &outro) const { return carta < outro.carta;} 1
};

class Viatura{      -6-          1
    string matricula;
    Condutor *condutor;          1
public:
    Viatura(string mat) :matricula(mat){ condutor = NULL; }
    Condutor *getCondutor() const { return condutor; } 1
    void setCondutor(Condutor *c){ condutor=c;} 1
    bool operator<(const Viatura &out) const{return matricula<out.matricula;} 1
};

class Autoaluga{      -33-          2
    Colecao<Viatura> viaturas;
    Colecao<Condutor> condutores;

    Viatura *findViatura(string mat) {          2
        Viatura v(mat);
        return viaturas.find(v);
    }

    Condutor *findCondutor(string cart) {          2
        Condutor c(cart, "");
        return condutores.find(c);
    }

public:
    bool addCondutor(string cart, string nom){          2
        Condutor c(cart,nom);
        return condutores.insert(c);
    }
```

```
bool addViatura(string mat){ 2
    Viatura v(mat);
    return viaturas.insert(v);
}

bool levantarViatura(string cart, string mat){ 8
    Condutor *c = findCondutor(cart);
    if (c == NULL) return false; //Condutor inexistente!
    else
        if (c->getViatura() != NULL) return false; //Conduz outra viatura!
        else{
            Viatura *v = findViatura(mat);
            if (v == NULL) return false; //Viatura inexistente!
            else
                if (v->getCondutor() != NULL) return false; //V. já com condutor!
                else{
                    c->setViatura(v);
                    v->setCondutor(c);
                    return true;
                }
        }
    }

bool devolverViatura(string mat){ 6
    Viatura *v = findViatura(mat);
    if (v == NULL) return false; //Viatura inexistente!
    else
        if (v->getCondutor() == NULL) return false; //V. não alugada
        else{
            v->getCondutor()->setViatura(NULL);
            v->setCondutor(NULL);
            return true;
        }
    }

bool removerViatura(string mat) { 6
    Viatura *v = findViatura(mat);
    if (v != NULL && v->getCondutor() == NULL){//Existe e não está alugada
        viaturas.erase(Viatura(mat));
        return true;
    }
    else return false;
}

};
```

b) Acrescente ao problema os métodos que permitam apresentar na saída... [4 val.]

-16-

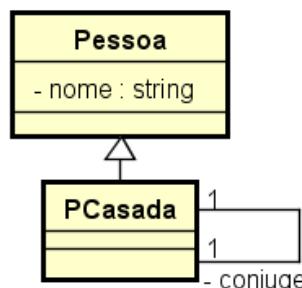
```
void Condutor::print() const{cout << nome << " - carta " << carta;} 3

string Viatura::getMatricula()const { return matricula;} 2

void Autoaluga::mostrarViaturasAlugadas() const{ 1
    Colecao<Viatura>::iterator it; 1
    for(it=viaturas.begin(); it!=viaturas.end(); it++) 2
        if (it->getCondutor() != NULL) { 3
            cout << "Viatura " << it->getMatricula() << " conduzida por "; 1
            it->getCondutor()->print(); 3
            cout << endl;
        }
}
```

Grupo II (5 valores)

a) Desenhe o diagrama de classes UML do problema... [1 val.]



b) Indique três métodos (construtores ou não) que, embora não definidos explicitamente. [.9 val.]

Construtor de cópia, destrutor e operador afetação.

c) Apresente o resultado que será visualizado na saída standard após a execução do programa. [2.4 val.]

-0-

Criado Manuel

Criado Maria

Conjuge 1

Conjuge 2

-1-

Manuel + Maria

-2-

Criado Ana

Criado Luis

Conjuge 1

Conjuge 2

-3-

Ana + Luis

-4-

Manuel + Ana

-5-

Ana + Manuel

d) Apresente o resultado que seria visualizado na saída standard se nenhum dos métodos. [0.7 val.]

O mesmo.

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.

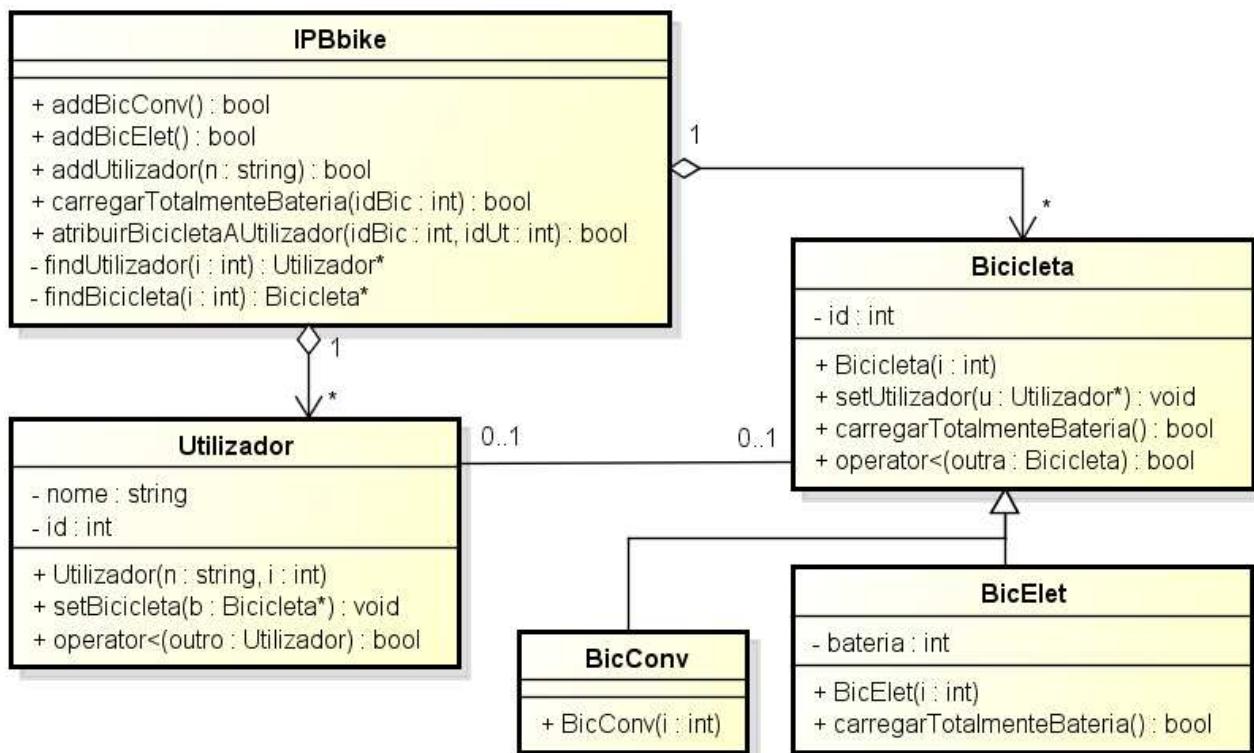
Grupo I

[15 valores]

O Projeto U-Bike Portugal é uma iniciativa de âmbito nacional (coordenada pelo IMT e enquadrando-se nos apoios do Portugal 2020), que visa incentivar a adoção de hábitos de mobilidade mais sustentáveis nas comunidades académicas das Instituições de Ensino Superior públicas, através da disponibilização de bicicletas elétricas e convencionais, estimando-se que, com essa medida, se venham a percorrer mais de 2 milhões de quilómetros a pedalar, o equivalente a mais de 60 voltas ao mundo.

O IPB viu a sua candidatura ser aprovada e, como tal, propomos-lhe que desenvolva uma pequena aplicação que venha no futuro a permitir gerir facilmente a utilização do parque de bicicletas que vierem a ser disponibilizadas ao IPB no âmbito desse projeto. Pretende-se, essencialmente, saber a cada momento que utilizador é que detém cada bicicleta que esteja a ser utilizada. O programa prevê a disponibilização de um conjunto considerável de bicicletas, quer elétricas, quer convencionais, todas elas identificadas por um código inteiro e, no caso das elétricas, também caracterizadas por possuírem um indicador que revele a todo o instante o estado da sua bateria.

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. [12 val.]

NOTA 1: Os ids, quer dos utilizadores, quer das bicicletas, devem ser atribuídos de forma automática pelo sistema, garantindo que ao 1º elemento é atribuído o id 1, ao 2º o id 2, e assim sucessivamente;

NOTA 2: As bicicletas elétricas são adicionadas ao sistema com a sua bateria completamente descarregada (0%);

NOTA 3: 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 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 diretivas #include.

- b) Acrescente ao problema os métodos que permitam mostrar na saída standard as bicicletas que estejam a ser utilizadas, mostrando para cada uma delas o respetivo id, se se trata de uma bicicleta elétrica ou convencional, e o nome de quem a está a utilizar. [3 val.]

Grupo II
[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) Apresente o resultado que será visualizado na saída standard após a execução do programa. [2 val.]
- b) Apresente o resultado que seria visualizado na saída standard se nenhum dos métodos da classe Base 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.]
- c) Por fim, para descomprimir, uma questão meramente teórica, que nada tem a ver com o pequeno programa tratado nesta secção.

Indique, no contexto da POO, um sinónimo para cada um dos seguintes conceitos: [2.0 val.]

- (A) – Instância de uma classe;
- (B) – Instância de uma template de classes;
- (C) – Método virtual;
- (D) – Método virtual puro;
- (E) – Membro variável de uma classe;
- (F) – Membro função de uma classe;
- (G) – Superclasse;
- (H) – Subclasse;
- (I) – Derivação de classes;
- (J) – Atributo estático;

ANEXO A

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

class Base{
public:
    Base() {cout << "Base()" << endl;}
    Base(const string &str) {cout << "Base(" << str << ")" << endl; }
    virtual string toString() { return "classe generica "; }
    virtual ~Base() {cout << "~Base()" << endl; }
};

class Derivada: public Base {
public:
    Derivada(){ cout << "Derivada()" << endl; }
    Derivada(const string &str): Base(str){ cout << "Derivada(" << str << ")" << endl; }
    string toString() { return "classe especializada "; }
    ~Derivada() { cout << "~Derivada()" << endl; }
};

void main() {
    Base *v[3];
    cout << "OUTPUT 0:" << endl;
    Base b;
    v[0] = &b;
    cout << "OUTPUT 1:" << endl;
    Derivada d("Ola");
    v[1] = &d;
    cout << "OUTPUT 2:" << endl;
    v[2] = new Derivada();
    cout << "OUTPUT 3:" << endl;
    cout << v[0]->toString() << v[1]->toString() << v[2]->toString() << endl;
    cout << "OUTPUT 4:" << endl;
    delete v[2];
    cout << "OUTPUT 5:" << 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();
};
```

Grupo I (15 valores)

- a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e ... [12 val.]

```
class Utilizador{ (5) 1
    string nome;
    int id;
    Bicicleta *bicileta;
public:
    Utilizador(const string &n, int i): nome(n), id(i){bicileta=NULL;} 1
    void setBicicleta(Bicicleta *b) {bicileta=b;} 1
    bool operator<(const Utilizador &outro) const {return id<outro.id;} 1
};

class Bicicleta{ (7) 1
    int id;
    Utilizador *utilizador;
public:
    Bicicleta(int i) { id = i; utilizador = NULL; } 1
    void setUtilizador(Utilizador *u) { utilizador=u;} 1
    virtual bool carregarTotalmenteBateria() { return false; } 2
    bool operator<(const Bicicleta &outra) const{return id<outra.id;} 1
};

class BicConv: public Bicicleta{ (3) 2
public:
    BicConv(int i): Bicicleta(i){}
};

class BicElet: public Bicicleta{ (6) 2
    int bateria;
public:
    BicElet(int i): Bicicleta(i){bateria=0;} 2
    bool carregarTotalmenteBateria() { bateria = 100; return true; } 2
};

class IPBbike{ (19) 2
    Coleccao<Utilizador> utilizadores;
    ColeccaoHibrida<Bicicleta*> bicicletas;
public:
    bool addBicConv(){
        int id = bicicletas.size() + 1;
        BicConv *b = new BicConv(id);
        return bicicletas.insert(b);
    }
}
```

```

bool addBicElet(){
    int id = bicicletas.size() + 1;
    BicElet *b = new BicElet(id);
    return bicicletas.insert(b);
}

bool addUtilizador(const string &n){
    int id = utilizadores.size() + 1;
    Utilizador u(n,id);
    return utilizadores.insert(u);
}

bool carregarTotalmenteBateria(int idBic){
    Bicicleta *pb=findBicicleta(idBic);
    if(pb==NULL){
        cout<<"Bicicleta inexistente!";
        return false;
    }else return pb->carregarTotalmenteBateria();
}

bool atribuirBicicletaAUtilizador(int idBic, int idUt){
    Bicicleta *pb=findBicicleta(idBic);
    if(pb==NULL) {cout<<"Bicicleta inexistente!\n"; return false;}
    else{
        Utilizador *pu=findUtilizador(idUt);
        if(pu==NULL) {cout<<"Utilizador inexistente!\n"; return false;}
        else{
            pb->setUtilizador(pu);
            pu->setBicicleta(pb);
            return true;
        }
    }
}

private:
    Utilizador *findUtilizador(int i){
        Utilizador u("",i);
        return utilizadores.find(u);
    }
    Bicicleta *findBicicleta(int i){
        Bicicleta b(i);
        return bicicletas.find(&b);
    }
};


```

- b) Acrescente ao problema os métodos que permitam mostrar no ecrã as bicicletas que estejam a ser utilizadas, mostrando para cada uma delas o respetivo id, se se trata de uma bicicleta elétrica ou convencional, e o nome de quem a está a utilizar. [3 val.]

```
void IPBbike::listarBicicletasEmUso()const{ 3
    ColeccaoHibrida<Bicicleta*>::iterator it;
    for(it=bicicletas.begin(); it!=bicicletas.end(); it++)
        if((*it)->getUtilizador()!=NULL)
            (*it)->print();
}

Utilizador *Bicicleta::getUtilizador() const{return utilizador;} 1

string Utilizador::getNome() const{return nome;} 1

virtual void Bicicleta::print()const { 2
    cout << " " << id << " conduzida por " << utilizador->getNome()<<endl;
}

void BicConv::print()const { 2
    cout << "Bicicleta Convencional";
    Bicicleta::print();
}

void BicElet::print()const { 1
    cout << "Bicicleta Eletrica";
    Bicicleta::print();
}
```

Grupo II
(5 valores)

- a) Apresente o resultado que será visualizado na saída standard após a execução do programa. [2 val.]

OUTPUT 0:

Base()

OUTPUT 1:

Base(Ola)

Derivada(Ola)

OUTPUT 2:

Base()

Derivada()

OUTPUT 3:

classe generica classe especializada classe especializada

OUTPUT 4:

~Derivada()

~Base()

OUTPUT 5:

~Derivada()

~Base()

~Base()

- b) Apresente o resultado que seria visualizado na saída standard se nenhum dos métodos. [1.0 val.]

OUTPUT 3:

classe generica classe generica classe generica

OUTPUT 4:

~Derivada()

~Base()

f) Indique, no contexto da POO, um sinónimo para cada um dos seguintes conceitos: [2.0 val.]

- | | |
|---|--------------------|
| (A) – Instância de uma classe; | objeto |
| (B) – Instância de uma template de classes; | classe template |
| (C) – Método virtual; | método polimórfico |
| (D) – Método virtual puro; | método abstrato |
| (E) – Membro variável de uma classe; | atributo |
| (F) – Membro função de uma classe; | método |
| (G) – Superclasse; | classe base |
| (H) – Subclasse; | classe derivada |
| (I) – Derivação de classes; | herança |
| (J) – Atributo estático; | atributo de classe |

Notas importantes:

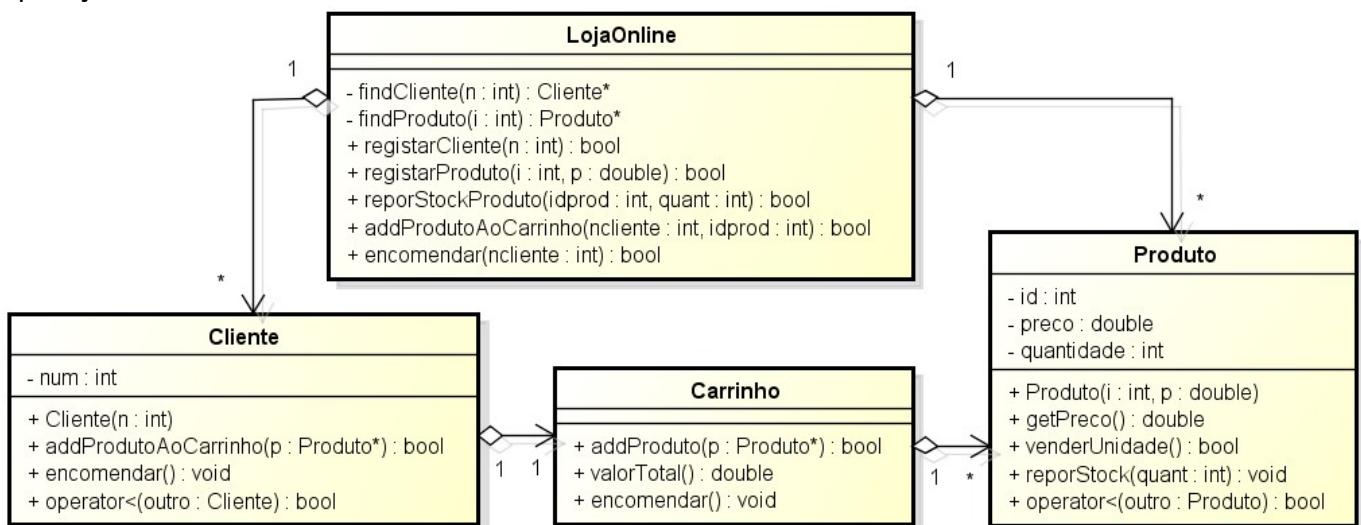
1. A duração da prova é de 1 hora e 30 minutos.
2. Comece por preencher a zona reservada à sua identificação na folha de exame.
3. Resolva os dois grupos em folhas de exame separadas.

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();
};

};
```

Grupo I (13.2 valores)

- a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e ... [9.3 val.]

```
class Produto { [1]
    int id;
    double preco;
    int quantidade;
public:
    Produto(int i, double p) { quantidade = 0; id = i; preco = p; }
    double getPreco() const { return preco; }
    bool venderUnidade(){ [1]
        if (quantidade > 0) { quantidade--; return true; }
        else return false;
    }
    void reporStock(int quant) {quantidade += quant;}
    bool operator<(const Produto &outro) const { return id<outro.id; } [1]
};

class Carrinho { [1]
    Colecao<Produto *> produtos;
public:
    bool addProduto(Produto *p) { [2]
        if (p->venderUnidade()) return produtos.insert(p);
        else return false;
    }
    double valorTotal() const { [3]
        double soma = 0;
        Colecao<Produto *>::iterator it;
        for (it = produtos.begin(); it != produtos.end(); it++)
            soma += (*it)->getPreco();
        return soma;
    }
    void encomendar() { produtos.clear(); }
};

class Cliente{ [1]
    int num;
    Carrinho carrinho;
public:
    Cliente(int n){ num = n; }
    bool addProdutoAoCarrinho(Produto *p){return carrinho.addProduto(p);}
    void encomendar() { carrinho.encomendar(); }
    bool operator<(const Cliente &outro) const { return num<outro.num; } [1]
};
```

```
class LojaOnline{
    Colecao<Cliente> clientes;
    Colecao<Produto> produtos;

    Cliente *findCliente(int n) {
        Cliente cl(n);
        return clientes.find(cl);
    }
    Produto *findProduto(int i) {
        Produto p(i,0.0);
        return produtos.find(p);
    }
public:
    bool registarCliente(int n) {
        Cliente cl(n);
        return clientes.insert(cl);
    }
    bool registarProduto(int i, double p) {
        Produto pr(i, p);
        return produtos.insert(pr);
    }
    bool reporStockProduto(int idprod, int quant) {
        Produto *p = findProduto(idprod);
        if (p != NULL) { p->reporStock(quant); return true; }
        else return false;
    }
    bool addProdutoAoCarrinho(int ncliente, int idprod) {
        Produto *p = findProduto(idprod);
        if (p == NULL) return false;
        else {
            Cliente *c = findCliente(ncliente);
            if (c == NULL) return false;
            else return c->addProdutoAoCarrinho(p);
        }
    }
    bool encomendar(int ncliente) {
        Cliente *c = findCliente(ncliente);
        if (c != NULL) { c->encomendar(); return true; }
        else return false;
    }
};
```

b) Acrescente ao problema os métodos que permitam mostrar o recheio do carrinho... [2.1 val.]

```

void LojaOnline::mostrarCarrinho(int ncliente) { [2]
    Cliente *c = findCliente(ncliente);
    if (c != NULL) c->mostrarCarrinho();
}

void Cliente::mostrarCarrinho() { carrinho.mostrar(); } [1]

void Carrinho::mostrar() const { [3]
    cout << "<Carrinho de compras>" << endl;
    Colecao<Produto *>::iterator it;
    for (it = produtos.begin(); it != produtos.end(); it++)
        (*it)->print();
    cout << "Total: " << valorTotal() << " EUROS." << endl;
}

void Produto::print() const { [1]
    cout << '\t' << "Produto " << id << ' ' << preco << " euros." << endl;
}

```

c) Acrescente ao problema os métodos que permitam excluir um dos produtos do carrinho... [1.8 val.]

```

bool LojaOnline::remProdutodoCarrinho(int ncliente, int idprod) { [3]
    Produto *p = findProduto(idprod);
    if (p == NULL) return false;
    else {
        Cliente *c = findCliente(ncliente);
        if (c == NULL) return false;
        else return c->remProdutoDoCarrinho(p);
    }
}

bool Cliente::remProdutoDoCarrinho(Produto *p){ [1]
    return carrinho.remProduto(p);
}

bool Carrinho::remProduto(Produto *p) { [2]
    if (produtos.find(p) != NULL) {
        produtos.erase(p);
        p->reporStock(1);
        return true;
    }else return false;
}

```

Grupo II
(6.8 valores)

- a) Quantos métodos construtores estarão presentes em cada uma das classes do problema. [0.3 val.]
3.
- b) Alguma das classes do problema é abstrata? Diga o que entende por classe abstrata. [1.0 val.]
Nenhuma das classes do problema é abstrata. Uma classe abstrata é uma classe não instanciável, normalmente caracterizada em C++ por conter um ou mais métodos virtuais puros.
- c) Considerando as seguintes instanciações: A a1("um"), a2("dois"); diga, justificando ... [0.7 val.]
A instrução não é válida dado não estarem definidos, nem como métodos das respectivas classes, nem como funções globais, os operadores < (menor) e << (insessor) para os operandos usados. (Também não será válida devido a um erro não intencional: identificador c1 não declarado)
- d) Apresente o resultado que será visualizado na saída standard após a execução do programa. [3.8 val.]

-----1 [3]
A(str)
B(str,str)
C(str,str,str)
-----2 [3]
A()
B()
C()
-----3 [1]
A()
B()
C()
-----4
Area com sub-regioes com subsolo impermeavel [3]
objeto da classe C com subobjeto da classe B com subobjeto da classe A [3]
objeto da classe C com subobjeto da classe B com subobjeto da classe A [1]
-----5 [3]
~C()
~B()
~A()
-----6 [1]
~C()
~B()
~A()
-----7 [1]
~C()
~B()
~A()

e) Apresente o resultado que seria visualizado na saída standard se nenhum dos métodos... [1.0 val.]

-----4

...

~~objeto da classe C com subobjeto da classe B com subobjeto da classe A~~

[.6]

-----5

[.4]

~~~C()~~

~~~B()~~

~~~A()~~

-----6

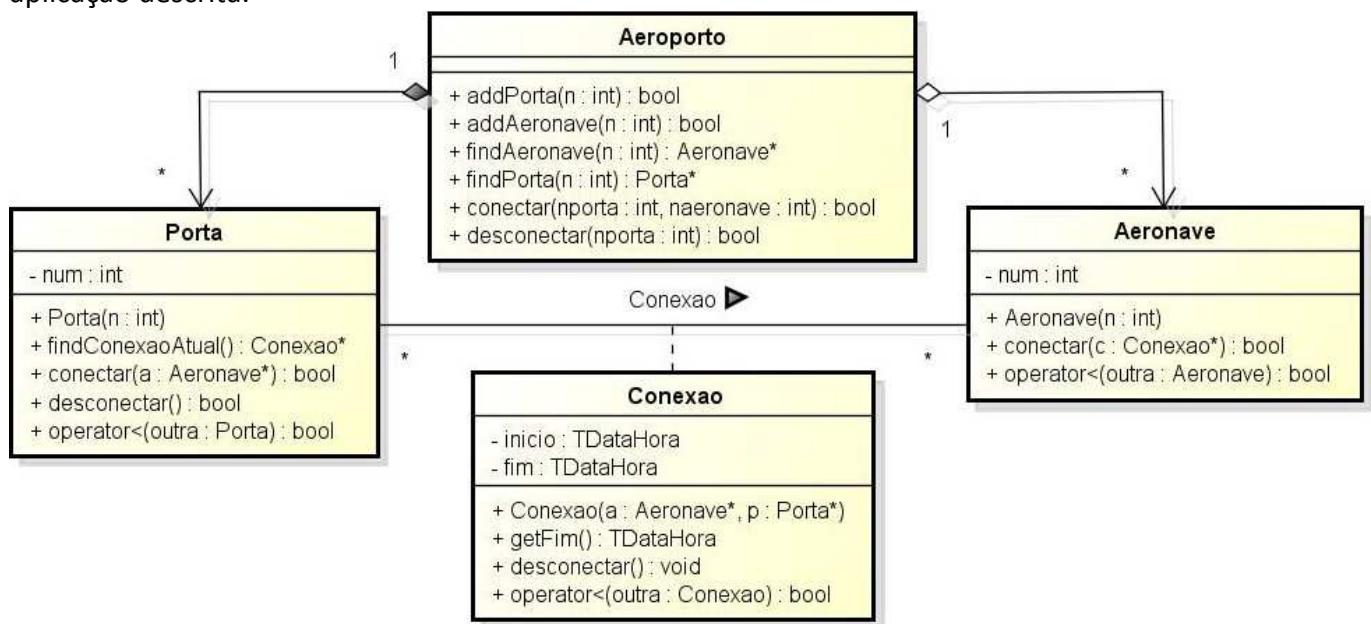
**Notas importantes:**

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**  
[12.6 valores]

Um importante aeroporto necessita de uma pequena aplicação que lhe permita manter o registo de todas as conexões de aeronaves às suas portas de embarque/desembarque (*gates*). A informação a reter relativamente a cada conexão é a data hora a que a mesma tem início e a data hora a que termina. No momento de criação de uma conexão (operação *conectar*) deve ser atribuída ao seu atributo *inicio* a data hora corrente e ao atributo *fim* a data hora "1/1/3000 0:0:0". A operação *desconectar* deverá simplesmente traduzir-se na atribuição da data hora corrente ao atributo *fim*.

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 implemente o método *findConexaoAtual()* da classe *Porta*; assuma que o mesmo já se encontra implementado e que devolve NULL caso a porta não esteja conectada (i.e., todas as suas conexões já terminaram – atributo *fim* com data hora anterior à corrente). [9.0 val.]

*NOTA 1: Para os tipos data hora é usada a classe TDataHora, estudada nas aulas da UC, e da qual se apresentam, no Anexo B deste enunciado, alguns métodos e operadores;*

*NOTA 2: Imagine que a aplicação está correr em tempo real. Por isso, para a hora e data das conexões devem ser consideradas a hora e data correntes. Use, para o efeito, as funcionalidades da classe TDataHora;*

*NOTA 3: 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 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 diretivas #include.*

- b) Acrescente ao problema os métodos que permitam mostrar todas as portas que se encontram conectadas (as que contêm uma conexão atual, i.e., que ainda não terminou), com indicação, para cada uma delas, do número da porta, do número da aeronave e da data hora a que iniciou a respetiva conexão. [2.4 val.]
- c) Implemente um pequeno main que faça uso de todas as funcionalidades da aplicação. [1.2 val.]

**Grupo II**  
[7.4 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) Alguma das classes é abstrata? Se sim, diga qual e o que a torna abstrata. [0.5 val.]
- b) Como pode constatar, apenas o método print() é um método constante. Por que razão se definiu esse método como constante? [0.5 val.]
- c) Como sabe, para além dos métodos que são definidos de forma explícita em cada classe, existirão outros definidos automaticamente pelo C++. Diga quais são os métodos implícitos que estarão presentes em cada uma das duas classes do problema, ainda que não se vejam. [1.2 val.]
- d) Apresente o resultado que será visualizado na saída standard após a execução do programa. [3.2 val.]
- e) Apresente agora o resultado que seria visualizado na saída standard se nenhum dos métodos da classe Funcionario fosse virtual. [2.0 val.]

## ANEXO A

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

class Funcionario{
protected:
    string nome;
public:
    Funcionario(const string &n) :nome(n){}
    virtual void print()const{ cout << "Funcionario " << nome << endl;}
    virtual bool addSubordinado(Funcionario *f){
        cout << "Nao pode ter subordinados!" << endl;
        return false;
    }
    virtual ~Funcionario(){cout<<"Funcionario "<<nome<<" dispensado"<<endl;}
};

class Diretor: public Funcionario{
    static const int MaxSub = 5;
    Funcionario *subordinados[MaxSub];
    int nSub;
public:
    Diretor(const string &n): Funcionario(n){
        nSub = 0;
        cout << "Criado Diretor " << nome << endl;
    }
    void print()const{ cout << "Diretor " << nome << endl; }
    bool addSubordinado(Funcionario *f){
        if (nSub == MaxSub) return false;
        else { subordinados[nSub++] = f; return true; }
    }
    ~Diretor(){
        cout<<"Diretor dispensado com os seus "<< nSub << " subordinados"<<endl;
    }
};

void main(){
    cout << "####Criacao###" << endl;
    Diretor rui("Rui Manuel");
    Funcionario *p = &rui, f1("Pedro"), f2("Jose");
    cout << "####Adicao###" << endl;
    rui.addSubordinado(&f1);
    p->addSubordinado(&f2);
    cout << "####Print1###" << endl;
    rui.print();
    cout << "####Print2###" << endl;
    p->print();
    cout << "####Destruicao###" << 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();
};

};
```

### Classe TDataHora

```
#include "TData.h"
#include "THora.h"

class TDataHora: public TData, public THora{
public:
    TDataHora();
    TDataHora(const char *dt); //dt: string no formato "x/x/xxxx h:m:s"
    bool operator<(const TDataHora &outra) const;
    ...
    static TDataHora hoje_agora(); //devolve data e hora corrente
};
ostream &operator<<(ostream &os, const TDataHora &dt);
```

**Grupo II**  
(7.4 valores)

- a) Alguma das classes é abstrata? Se sim, diga qual e o que a torna abstrata. [0.5 val.]

Nenhuma das classes é abstrata.

- b) Apenas método `print()` é constante. Por que razão se definiu esse método como constante? [0.5 val.]

Porque não afeta o estado do objeto. Acede ao atributo nome apenas para o consultar, não para o alterar.

- c) Diga quais são os métodos implícitos ... [1.2 val.]

Os métodos implícitos presentes em ambas as classes são o construtor de cópia e o operador afetação.

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

```
####Criacao###  
Criado Diretor Rui Manuel  
####Adicao###  
####Print1###  
Diretor Rui Manuel  
####Print2###  
Diretor Rui Manuel  
####Destruicao###  
Funcionario Jose dispensado  
Funcionario Pedro dispensado  
Diretor dispensado com os seus 2 subordinados  
Funcionario Rui Manuel dispensado
```

- e) Apresente agora o resultado se nenhum dos métodos da classe Funcionario fosse virtual ... [2.0 val.]

```
####Criacao###  
Criado Diretor Rui Manuel  
####Adicao###  
Nao pode ter subordinados!  
####Print1###  
Diretor Rui Manuel  
####Print2###  
Funcionario Rui Manuel  
####Destruicao###  
Funcionario Jose dispensado  
Funcionario Pedro dispensado  
Diretor dispensado com os seus 1 subordinados  
Funcionario Rui Manuel dispensado
```

**Grupo I**  
(12.6 valores)

a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e ... [9.0 val.]

```
class Aeroporto{           [3.3 val.]
    Colecao<Aeronave> aeronaves;          [1]
    Colecao<Porta> portas;                 [1]
public:
    bool Aeroporto::addPorta(int n){        [1]
        Porta p(n);
        return portas.insert(p);
    }
    bool Aeroporto::addAeronave(int n){       [1]
        Aeronave a(n);
        return aeronaves.insert(a);
    }

    Aeronave *findAeronave(int n){           [1]
        Aeronave a(n);
        return aeronaves.find(a);
    }
    Porta *findPorta(int n){                [1]
        Porta p(n);
        return portas.find(p);
    }

    bool conectar(int nporta, int naeronave){ [3]
        Aeronave *a = findAeronave(naeronave);
        if (a != NULL){
            Porta *p = findPorta(nporta);
            if (p != NULL) return p->conectar(a);
            else return false;
        }
        else return false;
    }

    bool desconectar(int nporta){            [2]
        Porta *p = findPorta(nporta);
        if (p != NULL) return p->desconectar();
        else return false;
    }
};
```

```

class Conexao{           [2.1 val.]
    TDataHora inicio, fim;
    Aeronave* aeronave;      [2]
    Porta* porta;
public:
    Conexao(Aeronave* a, Porta* p){      [2]
        inicio = TDataHora::hoje_agora();
        fim = TDataHora("1/1/3000 0:0:0");
        aeronave = a;
        porta=p;
    }
    TDataHora getFim()const{ return fim; }          [1]
    void desconectar(){fim=TDataHora::hoje_agora();}      [1]      [1]
    bool operator<(const Conexao &outra)const {return inicio<outra.inicio;}
};

class Aeronave{           [1.2 val.]
    int num;
    Colecao<Conexao *> conexoes;            [1]
public:
    Aeronave(int n){ num = n; }             [1]
    bool conectar(Conexao *c){ return conexoes.insert(c); }      [1]
    bool operator<(const Aeronave &outra)const { return num<outra.num; } [1]
};

class Porta{             [2.4 val.]
    int num;
    Colecao<Conexao> conexoes;            [1]
public:
    Porta(int n){num=n;}           [1]
    bool conectar(Aeronave *a){       [3]
        if (findConexaoAtual()!=NULL) return false;
        else{Conexao c(a, this);
            if (conexoes.insert(c)) return a->conectar(conexoes.find(c));
            else return false;
        }
    }
    bool desconectar(){           [2]
        Conexao *c = findConexaoAtual();
        if (c == NULL) return false;
        else {c->desconectar(); return true;}
    }
    bool operator<(const Porta &outra)const {return num<outra.num;}      [1]
}

```

```
//Conexao *findConexaoAtual()const{
//    if (conexoes.empty()) return NULL;
//    else
//        if (conexoes.rbegin()->getFim()<=TDataHora::hoje_agora()) return NULL;
//        else return (Conexao*)(conexoes.rbegin().operator->());
//}
};
```

b) Acrescente ao problema os métodos que permitam mostrar... [2.4 val.]

```
void Aeroporto::printPortasConectadas(){           [1.2 val.]
    cout << "Conexoes atuais:" << endl;
    Colecao<Porta>::iterator it;
    for (it = portas.begin(); it != portas.end(); it++){
        Conexao *c = it->findConexaoAtual();
        if (c != NULL) c->print();
    }
}
```

```
void Conexao::print()const{                      [.6 val.]
    cout<< "A Aeronave "<< aeronave->getNum()
        << " encontra-se conectada a porta "
        << porta->getNum()<< " desde " << inicio << endl;
};
```

```
int Aeronave::getNum()const { return num; }          [.3 val.]
```

```
int Porta::getNum()const { return num; }           [.3 val.]
```

c) Implemente um pequeno main que faça uso de todas as funcionalidades da aplicação. [1.2 val.]

```
void main(){
    Aeroporto ae;
    ae.addPorta(5);
    ae.addPorta(1);
    ae.addAeronave(2000);
    ae.addAeronave(7000);
    ae.conectar(5, 7000);
    ae.conectar(1, 2000);
    ae.desconectar(5);
    ae.printPortasConectadas();
}
```

Engenharia Informática  
Informática de Gestão

**Época Normal – 20 de janeiro de 2014**

---

**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**  
(8 valores)

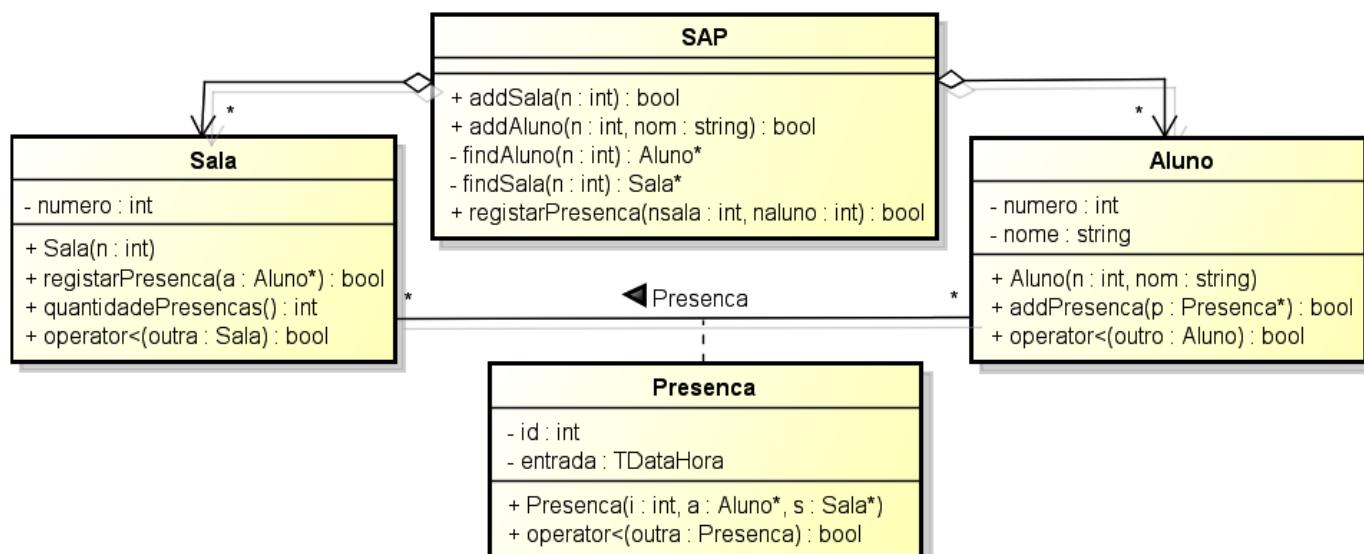
No Anexo A do presente enunciado encontra-se o código de um pequeno programa em C++. Depois de o analisar com a devida atenção, responda às seguintes questões.

- a) Identifique o tipo de relação existente entre Documento e Frase.
- b) Como sabe, para além dos métodos que são definidos de forma explícita em cada classe, existirão outros definidos automaticamente pelo C++. Diga quais são os métodos implícitos que estarão presentes em cada uma das duas classes do problema, ainda que não se vejam.
- c) O método construtor Documento::Documento(int m) tem um problema grave. Identifique-o e proponha uma solução adequada, sem alterar o método em questão (correções que deverá levar em conta na resolução das restantes alíneas).
- d) Apresente o resultado que será visualizado na saída standard após a execução do programa.
- e) Considere agora o seguinte código substituto para a função main(). O programa apresenta um erro sério de execução. Identifique-o e resolva-o convenientemente, alterando apenas o código da classe Documento.

```
void main(){
    Documento *d= new Documento(5);
    d->addFrase("Exame ");
    d->addFrase("de POO");
    Documento doc(*d);
    delete d;
    doc.print();
}
```

**Grupo II**  
 (12 valores)

Suponha que, para efetuar a implementação do atual Sistema Automático de Presenças (SAP), a ESTiG começou por solicitar a criação de um protótipo, o qual, deveria permitir registar, para além das presenças, os alunos e as salas de aula. Nesse protótipo, não deveriam ser ainda consideradas as unidades curriculares, as turmas, as salas, nem tão pouco as aulas lecionadas; apenas o registo do dia e da hora de entrada dos alunos nas salas, sem preocupações com a validação da hora e data de entrada. Para evitar a definição de chaves múltiplas, sugeriu-se considerar um identificador na presença, automaticamente incrementado. Segue-se o diagrama de classes UML, para descrever de forma precisa a solução que se pretende para a aplicação.



- a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e atributos descritos no diagrama. Não implemente quaisquer outros métodos ou atributos. [8.5 val.]

*NOTA 1: Para os tipos data hora é usada a classe TDataHora, estudada nas aulas da UC, e da qual se apresentam, no Anexo B deste enunciado, alguns métodos e operadores;*

*NOTA 2: Imagine que a aplicação está correr em tempo real. Por isso, para a hora e data das presenças devem ser consideradas a hora e data correntes. Use, para o efeito, as funcionalidades da classe TDataHora;*

*NOTA 3: 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 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 diretivas #include.*

- b) Acrescente ao problema os métodos que permitem mostrar todas as presenças de um dado aluno, com a indicação do número da presença e da respetiva data hora de entrada. [2.5 val.]

- c) Implemente um pequeno main que faça uso de todas as funcionalidades da aplicação. [1.0 val.]

## ANEXO A

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

class Frase{
    string conteudo;
public:
    Frase(const string &c):conteudo(c){
        cout << "Criada a frase: " << conteudo << endl;
    }

    void print(){cout << conteudo;}

    ~Frase(){cout << "Frase <" << conteudo << "> destruida" << endl;}
};

class Documento{
    Frase *frases;
    int n, max;
public:
    Documento(int m){
        max=m; n=0;
        frases=new Frase[max];
        cout << "Criado um documento com um maximo de "
            << max << " frases" << endl;
    }

    void addFrase(const string &c){
        if(n<max){
            Frase f(c);
            frases[n++]=f;
        }
    }

    void print(){
        for(int i=0;i<n;i++) frases[i].print();
        cout << endl;
    }

    ~Documento(){
        delete []frases;
        cout << "Documento com " << n << " frases destruido" << endl;
    }
};

void main(){
    Documento doc(5);
    doc.addFrase("Exame ");
    doc.addFrase("de POO");
    doc.print();
}
```

## 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();
};

};
```

### Classe TDataHora

```
#include "TData.h"
#include "THora.h"

class TDataHora: public TData, public THora{
public:
    TDataHora();
    string getStr() const; //devolve uma string c/ data e tempo ("x/x/xx h:m:s")
    ...
    static TDataHora hoje_agora(); //devolve data e hora corrente
};

ostream &operator<<(ostream &os, const TDataHora &dt);
istream &operator>>(istream &is, TDataHora &dt);
```

---

**Grupo I**  
(8 valores)

No Anexo A do presente enunciado encontra-se o código de um pequeno programa em C++. ...

- a) Identifique o tipo de relação existente entre Documento e Frase.

Agregação. Documento agrupa Frase numa relação de 1 para n.

- b) Como sabe, para além dos métodos que são definidos de forma explícita em cada classe, ...

Os métodos implícitos que estão presentes em ambas as classes são o construtor de cópia e o operador afetação.

- c) O método construtor Documento::Documento(int m) tem um problema grave. ...

O método em causa tenta criar um vetor de objetos da classe Frase. Essa operação é impossível dado que a classe Frase não dispõe do construtor por defeito. A solução passaria por definir explicitamente esse construtor: Frase::Frase(){}.

- d) Apresente o resultado que será visualizado na saída standard após a execução do programa.

Criado um documento com um maximo de 5 frases

Criada a frase: Exame

Frase <Exame > destruida

Criada a frase: de P00

Frase <de P00> destruida

Exame de P00

Frase <> destruida

Frase <> destruida

Frase <> destruida

Frase <de P00> destruida

Frase <Exame > destruida

Documento com 2 frases destruido

- e) Considere agora o seguinte código substituto para a função main(). O programa apresenta um ...

Como o documento doc é construído por cópia do documento apontado por d, através do construtor de cópia implícito, os dois documentos passam, através do apontador frases, a apontar para o mesmo vetor de frases. O problema surge então quando, na última instrução do main, se tenta mostrar as frases de doc: embora o apontador frases ainda exista no doc, o vetor de frases por ele apontado terá já sido desalocado aquando da eliminação do documento apontado por d (delete d).

A solução passaria por definir explicitamente o construtor de cópia da classe Documento tal como se segue:

```
Documento::Documento(const Documento &outro){  
    max=outro.max;  
    n=outro.n;  
    frases= new Frase[max];  
    for(int i=0; i<n; i++) frases[i]=outro.frases[i];  
}
```

**Grupo II**  
(12 valores)

a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e ... [8.5 val.]

```
class SAP{
    Colecao<Aluno> alunos;
    Colecao<Sala> salas;
public:
    bool SAP::addSala(int n){
        Sala s(n);
        return salas.insert(s);
    }

    bool SAP::addAluno(int n,const string &nom){
        Aluno a(n,nom);
        return alunos.insert(a);
    }

    Aluno *findAluno(int n){
        Aluno a(n,"");
        return alunos.find(a);
    }

    Sala *findSala(int n){
        Sala s(n);
        return salas.find(s);
    }

    bool registrarPresenca(int nsala, int naluno){
        Aluno *pa=findAluno(naluno);
        if(pa!=NULL){
            Sala *ps=findSala(nsala);
            if(ps!=NULL) return ps->registrarPresenca(pa);
            else return false;
        }else return false;
    }
};

class Sala{
    int numero;
    Colecao<Presenca> presencias;
public:
    Sala(int n){numero=n;}

    bool registrarPresenca(Aluno *a){
        Presenca s(quantidadePresencas()+1,a,this);
        if (presencias.insert(s)) return a->addPresenca(presencias.find(s));
        else return false;
    }

    int quantidadePresencas()const {return presencias.size();}

    bool operator<(const Sala &outra)const {return numero<outra.numero;}
};
```

```

class Aluno{
    int numero;
    string nome;
    Colecao<Presenca *> presencias;
public:
    Aluno(int n,const string &nom): nome(nom){numero=n;}

    bool addPresenca(Presenca *p){return presencias.insert(p);}

    bool operator<(const Aluno &outro) const {return numero<outro.numero;}
};

class Presenca{
    int id;
    TDataHora entrada;
    Aluno* aluno;
    Sala* sala;
public:
    Presenca(int i, Aluno* a, Sala* s){
        entrada=TDataHora::hoje_agora();
        id=i;
        aluno=a;
        sala=s;
    }

    bool operator<(const Presenca &outra) const {return id<outra.id;}
};

```

- b) Acrescente ao problema os métodos que permitam mostrar todas as presenças de um ... [2.5 val.]

```

void SAP::printPresenciasAluno(int n){
    Aluno *pa=findAluno(n);
    if(pa!=NULL) pa->printPresencias();
    else cout<<"Aluno nao existe!"<<endl;
}

void Aluno::printPresencias() const{
    cout<<"Presencias do aluno "<< nome <<": "<<endl;
    Colecao<Presenca *>::iterator it;
    for(it=presencias.begin(); it!=presencias.end(); it++)
        (*it)->print();
}

void Presenca::print() const{
    cout<<"N."<<id<<" Entrada a "<<entrada<<endl;
}

```

- c) Implemente um pequeno main que faça uso de todas as funcionalidades da aplicação. [1.0 val.]

```

void main(){
    SAP sap;
    sap.addSala(5);
    sap.addAluno(123, "Ana Rita");
    sap.registarPresenca(5,123);
    sap.printPresenciasAluno(123);
}

```

Engenharia Informática  
Informática de Gestão

**Época Normal – 21 de janeiro de 2013**

---

**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.
- 

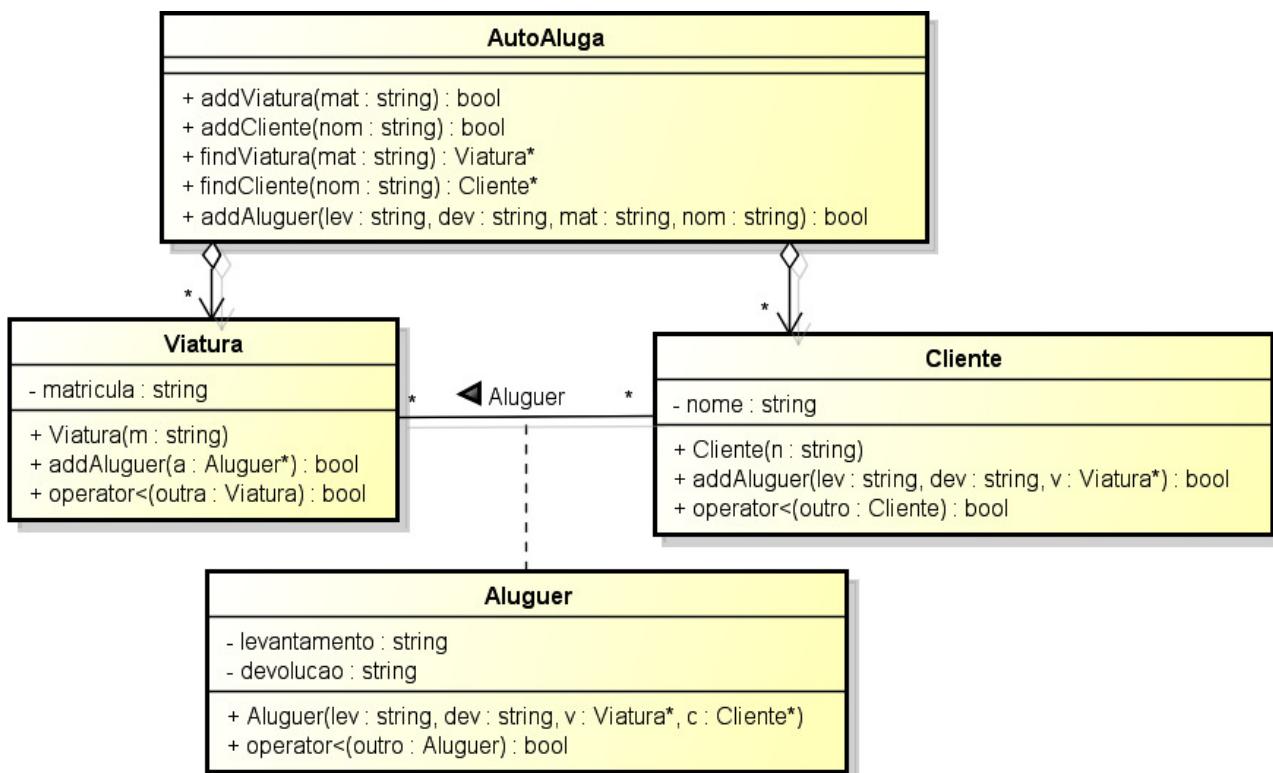
**Grupo I**  
(8 valores)

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) Propositadamente, todos os métodos da classe AveRara contêm um erro de compilação ou de execução. Identifique-os e proponha, para cada um deles, uma solução adequada (correções que deverá levar em conta na resolução das restantes alíneas). [1.6 val.]
- b) Alguma das classes do problema é abstrata? Se sim, diga qual e por que motivo a considera abstrata. [0.4 val.]
- c) Como sabe, para além dos métodos que são definidos de forma explícita em cada classe, existirão outros definidos automaticamente pelo C++. Diga quais são os métodos implícitos que estarão presentes em cada uma das duas classes do problema, ainda que não se vejam. [1.0 val.]
- d) Apresente o resultado que será visualizado na saída standard após a execução do programa. [2.6]
- e) O que seria visualizado se o método *mostrar* da classe Ave não fosse virtual? Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea anterior. [0.8 val.]
- f) E o que seria visualizado se fosse o método *acrescentar* a não ser virtual? Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea d). [0.8 val.]
- g) Considerando as seguintes declarações: AveRara a; Ave \*p=&a; diga, justificando, se a instrução (\*p)++; é ou não válida no problema considerado. Caso não seja, introduza as alterações necessárias nas classes do problema para que a instrução passe a ser válida. [0.8 val.]

**Grupo II**  
(12 valores)

A Auto Aluga é uma empresa de aluguer de automóveis que necessita de uma pequena aplicação que lhe permita registar todos os alugueres das suas viaturas. Dispõe de uma frota considerável de viaturas de modo a satisfazer todas as solicitações, que não param de aumentar. Relativamente a cada aluguer, pretende que sejam apenas registados, para além da viatura e do cliente envolvidos, a data-hora (apenas uma string) em que a viatura foi levantada e a data-hora em que a mesma terá sido devolvida. Como na Auto Aluga há alguém a perceber um pouco de informática, disponibilizou já o diagrama de classes UML que se segue, para descrever de forma precisa a solução que pretende para a sua aplicação.



- a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e atributos descritos no diagrama. Não implemente quaisquer outros métodos ou atributos. [8.5 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 necessários que permitam obter o número total de alugueres da Auto Aluga. [2.5 val.]
- c) Implemente um pequeno main que faça uso de todas as funcionalidades da aplicação. [1.0 val.]

## ANEXO A

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

class Ave{
    int num_penas;
public:
    Ave(int n): num_penas(n){cout<<"nova ave com "<<num_penas<<" penas\n";}
    virtual void acrescentar(int n){num_penas+=n;}
    virtual void mostrar(void) const{
        cout<<"tenho "<< num_penas << " penas\n";
    }
    virtual ~Ave(){cout<<"uma ave a menos\n";}
};

class AveRara: public Ave{
    int penas_regeitadas;
public:
    AveRara(){
        penas_regeitadas=0;
        cout<<"nova ave rara\n";
    }
    void acrescentar(int n) const{
        penas_regeitadas+=n;
        cout<<"tenho pena de nao ter penas!\n";
    }
    void mostrar(void) const{
        cout<<"regeitei "<< penas_regeitadas << " penas\n";
        Ave.mostrar();
    }
    ~AveRara(){cout<<"menos uma ave rara com "<<num_penas<<" penas\n";}
};

void main(){
    AveRara a0;
    Ave &a1=a0;
    Ave a2(500);
    a0.acrescentar(3);
    cout<<"ACRESCENTAR:\n";
    a1.acrescentar(10); a2.acrescentar(20);
    cout<<"MOSTRAR:\n";
    a0.mostrar(); a1.mostrar(); a2.mostrar();
    cout<<"ABATER:\n";
}
```

## 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();
};

};
```

**Grupo I**

a) Propositadamente, todos os métodos da classe AveRara contêm um erro de compilação ou de execução. Identifique-os e proponha, para cada um deles, uma solução adequada (correções que deverá levar em conta na resolução das restantes alíneas). [1.6 val.]

```
AveRara():Ave(0) {...  
void acrescentar(int n) const{...  
void mostrar(void) const{  
...  
    Ave::mostrar();  
}  
~AveRara(){cout<<"menos uma ave rara com "<<num_penas<<" penas\n";}  
    Declara-se o atributo como protected na classe Ave  
};
```

b) Alguma das classes do problema é abstrata? Se sim, diga qual ... motivo a considera abstrata. [0.4 val.]

Não

c) Como sabe, para além dos métodos que são definidos de forma explícita em cada classe, existirão outros definidos automaticamente pelo C++. Diga quais são os métodos implícitos que estarão presentes em cada uma das duas classes do problema. [1.0 val.]

Na classe Ave:

Construtor de cópia  
Operador afetação

Na classe AveRara:

Construtor de cópia  
Operador afetação

d) Apresente o resultado que será visualizado na saída standard após a execução do programa. [2.6]

```
nova ave com 0 penas  
nova ave rara  
nova ave com 500 penas  
tenho pena de nao ter penas!  
ACRESCENTAR:  
tenho pena de nao ter penas!  
MOSTRAR:  
regeitei 13 penas  
tenho 0 penas  
regeitei 13 penas  
tenho 0 penas  
tenho 520 penas  
ABATER:  
uma ave a menos  
menos uma ave rara com 0 penas  
uma ave a menos
```

e) O que seria visualizado se o método *mostrar* da classe Ave não fosse virtual? Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea anterior. [0.8 val.]

MOSTRAR:

regeitei 13 penas

tenho 0 penas

~~regeitei 13 penas~~

tenho 0 penas

tenho 520 penas

ABATER:

f) E o que seria visualizado se fosse o método *acrescentar* a não ser virtual? Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea d). [0.8 val.]

ACRESCENTAR:

~~tenho pena de nao ter penas!~~

MOSTRAR:

regeitei 13 3 penas

tenho 0 10 penas

regeitei 13 3 penas

tenho 0 10 penas

tenho 520 penas

ABATER:

uma ave a menos

menos uma ave rara com 0 10 penas

uma ave a menos

g) Considerando as seguintes declarações: AveRara a; Ave \*p=&a; diga, justificando, se a instrução (\*p)++; é ou não válida no problema considerado. Caso não seja, introduza as alterações necessárias nas classes do problema para que a instrução passe a ser válida. [0.8 val.]

```
void Ave::operator++(int){  
    num_penas++;  
}
```

a)

```

class Viatura{
    string matricula;
    Colecao<Aluguer *> alugueres;
public:
    Viatura(const string &m): matricula(m){}

    bool addAluguer(Aluguer *a){return alugueres.insert(a);}

    bool operator<(const Viatura &outra) const {return matricula<outra.matricula;}
};

class Aluguer{
    string levantamento, devolucao;
    Viatura* viatura;
    Cliente* cliente;
public:
    Aluguer(const string &lev, const string &dev, Viatura* v, Cliente* c):
        levantamento(lev), devolucao(dev){
        viatura=v;
        cliente=c;
    }

    bool operator<(const Aluguer &outro) const {
        return levantamento<outro.levantamento;
    }
};

class Cliente{
    string nome;
    Colecao<Aluguer> alugueres;
public:
    Cliente(string n){nome=n;}

    bool addAluguer(string lev, string dev, Viatura *v){
        Aluguer a(lev,dev,v,this);
        if (alugueres.insert(a)) return v->addAluguer(alugueres.find(a));
        else return false;
    }

    bool operator<(const Cliente &outro) const {return nome<outro.nome;}
};

```

```

class AutoAluga{
    Colecao<Viatura> viaturas;
    Colecao<Cliente> clientes;
public:
    bool addCliente(const string &nom){
        Cliente cl(nom);
        return clientes.insert(cl);
    }

    bool addViatura(const string &mat){
        Viatura v(mat);
        return viaturas.insert(v);
    }

    Viatura *findViatura(const string &mat){
        Viatura v(mat);
        return viaturas.find(v);
    }

    Cliente *findCliente(const string &nom){
        Cliente cl(nom);
        return clientes.find(cl);
    }

    bool addAluguer(string lev, string dev, string mat, string nom){
        Viatura *pv=findViatura(mat);
        if(pv!=NULL){
            Cliente *pc=findCliente(nom);
            if(pc!=NULL) return pc->addAluguer(lev,dev,pv);
            else return false;
        }else return false;
    }
};


```

b)

```

int AutoAluga::totalAlugueres()const{
    int tot=0;
    Colecao<Cliente>::iterator it;
    for(it=clientes.begin(); it!=clientes.end(); it++)
        tot+=it->quantidadeAlugueres();
    return tot;
}

int Cliente::quantidadeAlugueres()const {return alugueres.size();}

```

c)

```

void main(){
    AutoAluga aa;
    aa.addCliente("Ana Rita");
    aa.addViatura("11-22-AA");
    aa.addAluguer("1/1/2012 <19:25>", "3/1/2012 <19:05>", "11-22-AA", "Ana Rita");
    cout<<"Total de alugueres: "<<aa.totalAlugueres()<<endl;
}

```

Departamento de Informática e Comunicações

Engenharia Informática / Informática de Gestão

2º Ano

Época Normal

Programação Orientada por Objectos  
1º Semestre

Data: 23/01/2012  
Duração: 1h30

Resolva os 2 grupos em folhas de exame separadas

Tolerância: 0h15

**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 representação de uma equipa de futebol, formada por onze jogadores, um dos quais necessariamente guarda-redes. Depois de o analisar com a devida atenção, responda às seguintes questões.

- a) Identifique o tipo de relação existente entre Jogador e GuardaRedes, entre Equipa e Jogador, e entre Equipa e GuardaRedes. [0.5 val.]
- b) Diga, para cada uma das classes, quais os métodos que deveriam ser definidos como constantes. [1.0 val.]
- c) Diga o que entende por classe abstracta, se alguma das classes do problema é abstracta e de que forma é que se definem classes abstractas em C++? [1.5 val.]
- d) Numa das classes o destrutor não tem a implementação mais adequada. Diga qual é esse destrutor e dê-lhe a implementação desejada. [1.5 val.]
- e) Apresente o resultado que será visualizado na saída standard após a execução do programa e depois de resolvida a inconformidade mencionada na alínea anterior. [3.0]
- f) E o que seria visualizado se o método Jogador::mostrar() não fosse virtual? Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea anterior. [1 val.]
- g) Considere agora a classe adicional e o código substituto para a função main() que a seguir se apresentam. Qual o resultado que será visualizado na saída standard. [1,5 v.]

```
class EquipaComNome: public Equipa{
    string nome;
public:
    EquipaComNome(const string &name): nome(name){
        cout << nome << endl;
    }
    ~EquipaComNome(){cout << nome << " excluido" << endl;}
};
```

```
void main(){
    Equipa *pe1=new EquipaComNome("slb");
    EquipaComNome e2("fcp");
    delete pe1;
}
```

**Grupo II**  
(10 valores)

Uma importante Companhia Aérea (CA) necessita de uma pequena aplicação que lhe permita manter registadas todas as escalas realizadas pelas suas aeronaves nos diferentes aeroportos em que opera (entenda-se por escala o período de permanência de uma aeronave num dado aeroporto). A companhia dispõe de uma frota considerável de aeronaves, identificadas por um código inteiro. Já os aeroportos por elas visitados são identificados pelo seu nome. Relativamente a cada escala, pretende-se que fique registado, para além código inteiro que identifique a escala, as datas de chegada e de partida da aeronave.

(Pode utilizar o tipo string da STL para o tipo de dados a usar nas datas, e considere que essas strings vão conter informação quer relativamente à data quer relativamente à hora.)

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 colecciónáveis. [3 val.]

Não indique quaisquer outros métodos.

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

*NOTA1: 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 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 directivas #include.*

c) Implemente os métodos necessários para adicionar à aplicação um novo aeroporto e uma nova aeronave. [1 val.]

d) Implemente o(s) método(s) necessário(s) para registrar uma nova escala. [3 val.]

```

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

class Jogador{
protected:
    int num;
public:
    Jogador(int n): num(n){
        cout << "Novo Jogador " << endl;
    }
    int getNum(){return num;}
    virtual void mostrar(){cout << "Jogador " << num << endl;}
    virtual ~Jogador(){cout << "Jogador " << num << " excluido" << endl;}
};

class GuardaRedes: public Jogador{
public:
    GuardaRedes(): Jogador(1){ cout << "Novo Guarda-redes" << endl;}
    void mostrar(){
        cout << "<Guarda-redes> ";
        Jogador::mostrar();
    }
    ~GuardaRedes(){cout << "Guarda-redes excluido" << endl;}
};

class Equipa{
    static const int MAXJOG=11;
    Jogador *jogadores[MAXJOG];
    int njog;
public:
    Equipa():njog(0){cout << "Criada Equipa" << endl;}
    bool jaTemGuardaRedes(){
        int i;
        for(i=0; i<njog && jogadores[i]->getNum()!=1; i++);
        return i<njog;
    }
    void addJogador(int n){
        if(njog<MAXJOG-1 || (njog==MAXJOG-1 && jaTemGuardaRedes()))
            jogadores[njog++]=new Jogador(n);
    }
    void addGuardaRedes(){
        if(!jaTemGuardaRedes()) jogadores[njog++]=new GuardaRedes();
    }
    void mostrar(){
        cout << "Jogadores da equipa:" << endl;
        for(int i=0; i<njog; i++) jogadores[i]->mostrar();
    }
    ~Equipa(){}
};

void main(){
    Equipa slb;
    slb.addJogador(10);
    slb.addGuardaRedes();
    slb.addJogador(5);
    slb.mostrar();
}

```

**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();
};

};
```

**Grupo I**

a) Identifique ... Jogador e GuardaRedes, entre Equipa e Jogador, e entre Equipa e GuardaRedes. [0.5 val.]  
**Herança, Agregação, Agregação**

b) Diga, para cada uma das classes, quais os métodos que deveriam ser definidos como constantes. [1.0 val.]  
**Jogador: getNum(), mostrar()**  
**GuardaRedes: mostrar()**  
**Equipa: jaTemGuardaRedes(), mostrar()**

c) Diga o que entende por classe abstracta, se alguma é abstracta e com se definem em C++? [1.5 val.]  
**Classe abstracta é uma classe não instanciável**  
**Nenhuma classe é abstracta**  
**Em C++ uma classe abstracta é uma classe que tem um ou mais métodos virtuais puros**

d) Um destrutor não tem a implementação mais adequada. Diga qual é e dê-lhe a implement... [1.5 val.]  
**Destruitor da classe Equipa**  
**`~Equipa(){`**  
 **`for(int i=0; i<njog; i++) delete jogadores[i];`**  
**`}`**

e) Apresente o resultado que será visualizado na saída standard. [3.0]  
**Criada Equipa**  
**Novo Jogador**  
**Novo Jogador**  
**Novo Guarda-redes**  
**Novo Jogador**  
**Jogadores da equipa:**  
**Jogador 10**  
**<Guarda-redes> Jogador 1**  
**Jogador 5**  
**Jogador 10 excluido**  
**Guarda-redes excluido**  
**Jogador 1 excluido**  
**Jogador 5 excluido**

f) E o que seria visualizado se o método `Jogador::mostrar()` não fosse virtual? Refira-se apenas... [1 val.]  
**<Guarda-redes> Jogador 1 passaria a Jogador 1**

g) Considere agora ... Qual o resultado que será visualizado na saída standard. [1,5 v.]  
**Criada Equipa**  
**slb**  
**Criada Equipa**  
**fcp**  
**fcp excluido**

b)

```

class CompAerea{
    Coleccao<Aeroporto> aeroportos;
    Coleccao<Aeronave> aeronaves;
public:
};

class Aeronave{
    int id;
    Coleccao<Escala> escalas;
public:
    Aeronave(int i){id=i;}
    virtual bool operator<(const Aeronave &outra) const {return id<outra.id;}
};

class Aeroporto{
    string nome;
    Coleccao<Escala *> escalas;
public:
    Aeroporto(const string &n): nome(n){}
    bool operator<(const Aeroporto &outro) const {return nome<outro.nome;}
};

class Escala{
    int id;
    string chegada, partida;
    Aeroporto* aporto;
    Aeronave* anave;
public:
    Escala(int i, const string &cheg, const string &part,Aeroporto* ap,Aeronave* an):
        chegada(cheg), partida(part){
            id=i;
            aporto=ap;
            anave=an;
    }
    virtual bool operator<(const Escala &outra) const {return id<outra.id;}
};

```

c)

```

classe CompAerea:
    bool addAeronave(int i){
        Aeronave an(i);
        return aeronaves.insert(an);
    }
    Bool addAeroporto(const string &n){
        Aeroporto ap(n);
        return aeroportos.insert(ap);
    }

```

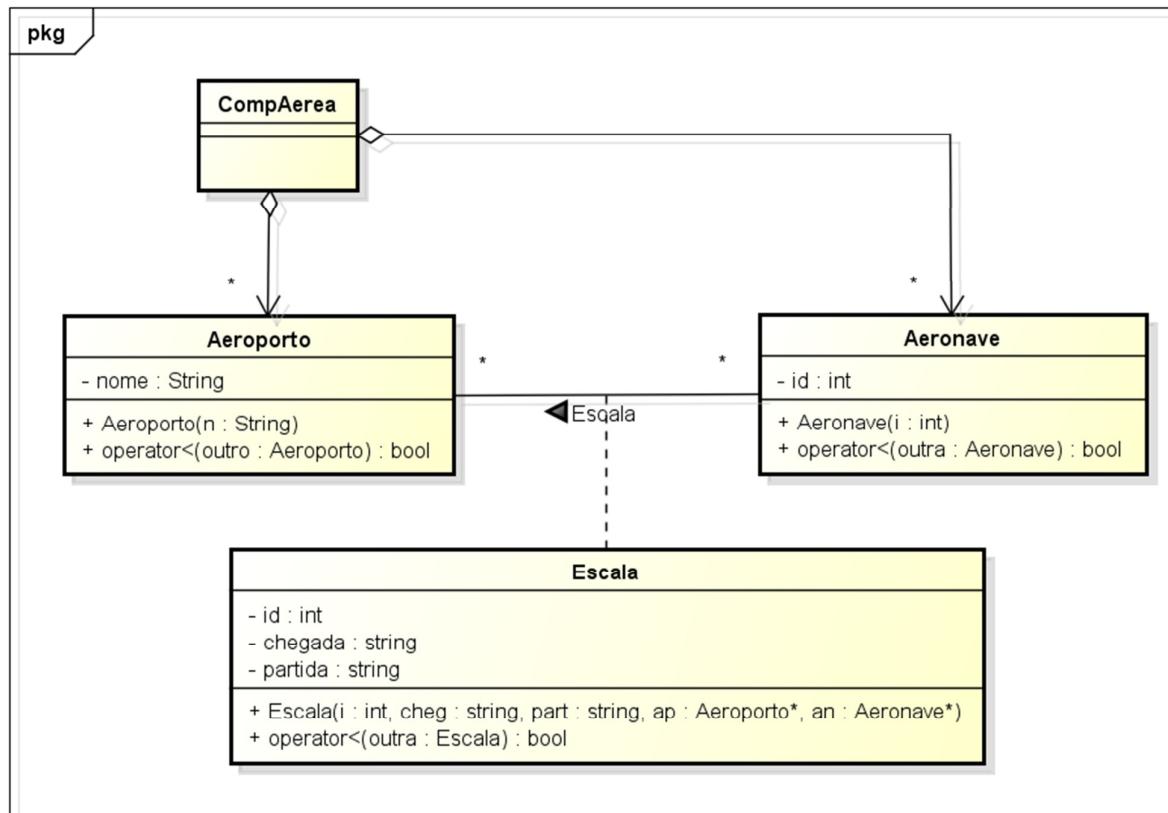
```

d)
classe CompAerea:
    bool addEscala(int es, int an, string ap, string cheg, string part){
        Aeroporto *pap=findAeroporto(ap);
        if(pap!=NULL){
            Aeronave *pan=findAeronave(an);
            if(pan!=NULL){
                return pan->addEscala(es,pap,cheg,part);
            }else return false;
        }else return false;
    }
    Aeroporto *findAeroporto(const string &n){
        Aeroporto ap(n);
        return aeroportos.find(ap);
    }
    Aeronave *findAeronave(int i){
        Aeronave ap(i);
        return aeronaves.find(ap);
    }

classe Aeronave:
    bool addEscala(int es, Aeroporto *ap, string cheg, string part){
        Escala e(es,cheg,part,ap,this);
        if (escalas.insert(e)){
            ap->add(escalas.find(e));
            return true;
        }else return false;
    }

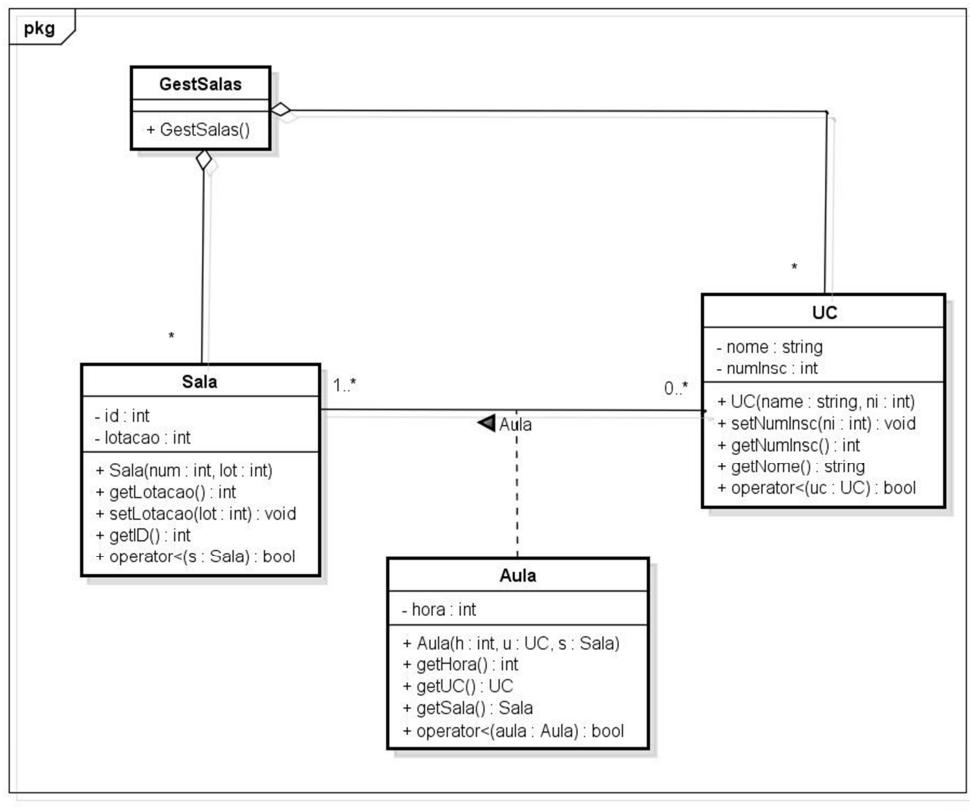
classe Aeroporto:
    bool add(Escala *pe){return escalas.insert(pe);}

```



Grupo I

a)



powered by astah® 

b)

```
class Aula{
    int hora;
    Sala *sala;
    UC *uc;
public:
    Aula(int h, UC *u, Sala *s) {
        hora=h;
        sala=s;
        uc=u;
    }
    int getHora() { return hora; }
    UC *getUC() {return uc;}
    Sala *getSala() {return sala;}
    bool operator<(const Aula& aula) const { return hora<aula.hora; }
};

class Sala{
    int id;
    int lotacao;
    Coleccao<Aula> aulas;
public:
    Sala(int num, int lot) {id=num; lotacao=lot;}
```

```

        int getLotacao() {return lotacao;}
        void setLotacao(int lot) {lotacao=lot;}
        int getID() {return id;}
        bool operator<(const Sala& s) const { return id<s.id;}
    };

    class UC{
        string nome;
        int numInsc;
        Coleccao<Aula*> aulas;
public:
    UC(string name, int ni): nome(name) {numInsc=ni;}
    void setNumInsc(int ni) {numInsc=ni;}
    int getNumInsc() {return numInsc;}
    string getNome() {return nome;}
    bool operator<(const UC& uc) const { return nome<uc.nome;}
};

    class GestSalas{
        Coleccao<Sala> salas;
        Coleccao<UC> ucs;
public:
};

c)
    bool GestSalas::addSala(int id, int lot) {
        if(findSala(id)==NULL){
            Sala s(id,lot);
            salas.insert(s);
            return true;
        }else return false;
    }
    Sala *GestSalas::findSala(int id) {
        Sala s(id,0);
        return salas.find(s);
    }

d)
    void GestSalas::listarAulasDeUmaSala(int nsala) {
        Sala *fs=findSala(nsala);
        if(fs!=NULL) fs->listarAulas();
        else cout<<"Sala "<<nsala<<" nao existe!\n";
    }
    void Sala::listarAulas() {
        Coleccao<Aula>::iterator it;
        cout<<"Aulas na Sala "<<id<<":\n";
        for(it=aulas.begin(); it!=aulas.end(); it++)
            cout<<it->getHora()<<"h - "<<it->getUC()->getNome()<<endl;
    }

e)
    bool GestSalas::alocarAula(string nomeUC, int hora, int nsala) {
        UC *fu=findUC(nomeUC);

```

```

        if(fs){
            Sala *fs=findSala(nsala);
            if(fs) return fs->alocarAula(hora,fu);
            else{
                cout << "Sala: " << nsala << " nao existe.\n";
                return false;
            }
        }else {
            cout << "Unidade Curricular: " << nomeUC << " nao existe.\n";
            return false;
        }
    }
    UC *GestSalas::findUC(string name) {
        UC u(name,0);
        return ucs.find(u);
    }
    bool Sala::alocarAula(int hora, UC* uc){
        if(disponivel(hora)){
            if(lotacao>=uc->getNumInsc()){
                Aula a(hora,uc,this);
                add(a);
                return true;
            }else cout << "A sala indicada nao tem lotacao suficiente\n";
        }else cout << "A sala indicada esta ocupada na hora pretendida\n";
        return false;
    };
    void Sala::add(Aula &a){
        if(aulas.insert(a)) a.getUC()->add(aulas.find(a));
        else cout << "Já está marcada uma aula na hora pretendida\n";
    }
}

```

## Grupo II

a) Identifique o tipo de relação existente entre *Figura* e *Circunferencia* e entre *Figura* e *Ponto*.

*Figura e Circunferencia:* HERANÇA

*Figura e Ponto:* AGREGAÇÃO

b) Na função main(), a invocação de um dos métodos resulta em erro. Identifique o erro e proponha uma solução adequada para que essa invocação passe a ser possível.

A invocação `f->ampliar(3);` requer que o método `ampliar` seja membro da classe `Figura`, uma vez que `f` está declarado como apontador para `Figura`.

Solução:

```
virtual void Figura::ampliar(int)=0;
```

c) Apresente o resultado que será visualizado na saída standard após a execução do programa e depois corrigido o erro referenciado na alínea anterior.

```
Circunferencia de raio 5 com centro em (0.1,10)
Circunferencia de raio 15 com centro em (0.1,10)
Circunferencia de raio 15 com centro em (0.2,20)
```

```
Circunferencia diz Xau
Figura diz Xau
Ponto diz Xau
Ponto diz Xau
Ponto diz Xau
```

d) Se acrescentarmos à função main() a linha de código que se segue, daí resultará algum erro na nossa aplicação? Se sim, proponha uma solução adequada para que essa linha de código deixe de dar erro.

```
Circunferencia varias[10];
```

**SIM**

```
solução: Figura::Figura(){}
Circunferencia::Circunferencia(){raio=1;}
```

e) Acrescente ao programa uma nova classe de nome Circulo que, para além de um centro e raio, tenha ainda como atributo a cor de preenchimento.

```
class Circulo: public Circunferencia{
    int cor_preench;
public:
    Circulo(){cor_preench=0;}
    Circulo(const Ponto& c, int r, int cor): Circunferencia(c,r){
        cor_preench=cor;
    }

    void mostrar() const{
        cout<<"Circulo de raio "<<raio<<" com centro em "<<posic;
        cout<<" e com a cor "<<cor_preench<<endl;
    }
~Circulo(){cout<<"Circulo diz Xau\n";}
};
```

Departamento de Informática e Comunicações

Engenharia Informática / Informática de Gestão  
2º Ano  
Época Normal

Programação Orientada por Objectos  
1º Semestre

Data: 25/01/2010  
Duração: 2h00

---

**Grupo I**  
(10 valores)

Suponha que quer utilizar os conhecimentos adquiridos em POO para desenvolver uma pequena aplicação que lhe permita fazer a gestão diária de um conjunto de salas de aula, utilizadas para a lecionação de diferentes Unidades Curriculares (UC). Cada sala é caracterizada pela sua identificação (ex: sala 12) e pela lotação, e cada UC pelo seu nome e número de alunos inscritos.

No sentido de simplificar a solução, considere que todas as aulas têm uma hora de duração, que iniciam sempre a horas certas (9h, 10h, 11h, ...), e repare que apenas se pretende gerir a alocação das salas ao longo de um único dia.

- 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 colecccionáveis).
- b) Codifique em C++ a sua solução, tal como a descreveu no diagrama da alínea anterior.

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 uma nova sala na aplicação.
- d) Implemente o(s) método(s) necessário(s) para listar as aulas alocadas a uma dada sala.
- e) Implemente o(s) método(s) necessário(s) para alojar uma aula de uma dada UC a uma sala. Se a sala indicada não se encontrar disponível à hora pretendida ou se a sua lotação não for suficiente, a operação de alocação deve ser rejeitada.  
Considere que a classe Sala já dispõe de um método, de protótipo  
`bool disponivel(int h)`, que verifica se a mesma se encontra disponível à hora `h` (i. e., entre a hora `h` e `h+1`).

**Grupo II**  
(10 valores)

Considere o seguinte programa:

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

class Ponto{
    float x, y; //coordenadas do ponto bidimensional
public:
    Ponto(){x=y=0;}
    Ponto(float a, float b){x=a; y=b;}
    float getx() const{return x;}
    float gety() const{return y;}
    ~Ponto(){cout<<"Ponto diz Xau\n";}
};

ostream &operator<<(ostream &os, const Ponto &p){
    os<<(' '<<p.getx()<<','<<p.gety()<<')';
    return os;
}

class Figura{
protected:
    Ponto posic; //localização da figura
public:
    Figura(const Ponto& p): posic(p){}
    virtual void mover(const Ponto& p){posic=p;}
    virtual void mostrar() const=0;
    virtual ~Figura(){cout<<"Figura diz Xau\n";}
};
```

```
class Circunferencia: public Figura{
protected:
    int raio;
public:
    Circunferencia(const Ponto& c, int r): Figura(c){raio=r;}
    void ampliar(int a){raio*=a;}
    void mostrar() const{
        cout<<"Circunferencia de raio "<<raio;
        cout<<" com centro em "<<posic<<endl;
    }
    ~Circunferencia(){cout<<"Circunferencia diz Xau\n";}
};

void main(){
    Ponto p1(0.1, 10), p2(0.2, 20);
    Circunferencia circ(p1, 5);
    Figura *f=&circ;
    f->mostrar();
    f->ampliar(3);
    f->mostrar();
    f->mover(p2);
    f->mostrar();
}
```

- a) Identifique o tipo de relação existente entre *Figura* e *Circunferencia* e entre *Figura* e *Ponto*.
- b) Na função main(), a invocação de um dos métodos resulta em erro. Identifique o erro e proponha uma solução adequada para que essa invocação passe a ser possível.
- c) Apresente o resultado que será visualizado na saída standard após a execução do programa e depois corrigido o erro referenciado na alínea anterior.
- d) Se acrescentarmos à função main() a linha de código que se segue, daí resultará algum erro na nossa aplicação? Se sim, proponha uma solução adequada para que essa linha de código deixe de dar erro.

```
Circunferencia varias[10];
```

- e) Acrescente ao programa uma nova classe de nome Círculo que, para além de um centro e raio, tenha ainda como atributo a cor de preenchimento.