Departamento de Matemática — Universidade de Coimbra

Ano Lectivo de 2011/2012 Programação Orientada para os Objectos Frequência 27/04/2012

Duração da prova: 1h30

---

1. Para poder encarar a resolução de problemas de grande dimensão e complexidade é necessário usar estratégias de *Modularidade* e *Abstracção*. De que forma é que estes conceitos são tratados na *Programação Orientada para os Objectos*?

2. Qual é o significado numa Classe das componentes: `public`; `protected`; `private`?

3. Quais são os tipos básicos de relações entre classes em *C++*? Para cada um dos casos apresente um diagrama UML exemplificativo.

4. O Tipo Abstracto de Dados Árvore binária tem a seguinte especificação algébrica:

$$\begin{aligned} \text{AB} \quad = \quad & (\{\text{ABvazia:AB,(Raiz:Elemento,ABesq:AB,ABdir:AB)}\}, \\ & \quad \{\text{criaAB,insRaiz,insABesq,insABdir,obtemRaiz,} \\ & \quad\quad \text{obtemABesq,obtemABdir,destroiAB,vazia?}\} \\ &) \end{aligned}$$

   Especifique (ficheiro `.hpp`), em C++, uma classe genérica `ArvBin`, de elementos de um dado tipo genérico.

5. A loja de informática aTaKa pretende atacar o negócio da *e-U* (Universidade Electrónica), sendo que para tal pretende construir um programa para produzir prospectos entre outro material promocional. Construa um diagrama de classes apropriado à manipulação da informação respeitante a: computadores, periféricos e componentes (memória, discos, ...).

Resolução

1. **Abstracção**

   Abstracção: representação de um conjunto de objectos com propriedades comuns numa entidade que abstrai as diferenças entre os objectos e só representa aqui que é comum - **Classes em C++**.

   Instanciação: a capacidade de criar uma instância particular da entidade abstracta - **Objectos em C++**.

   Modularidade

   Módulos: com um interface público bem definido e com a capacidade de sonegar informação na sua secção privada.

   Relações entre Módulos: a possibilidade de construir o significado global através das relações entre módulos e dos seus significados parcelares.

2. **secção pública:** atributos e/ou métodos acessíveis a todos os clientes da classe;

   secção protegida: atributos e/ou métodos só acessíveis às sub-classes (relação de herança);

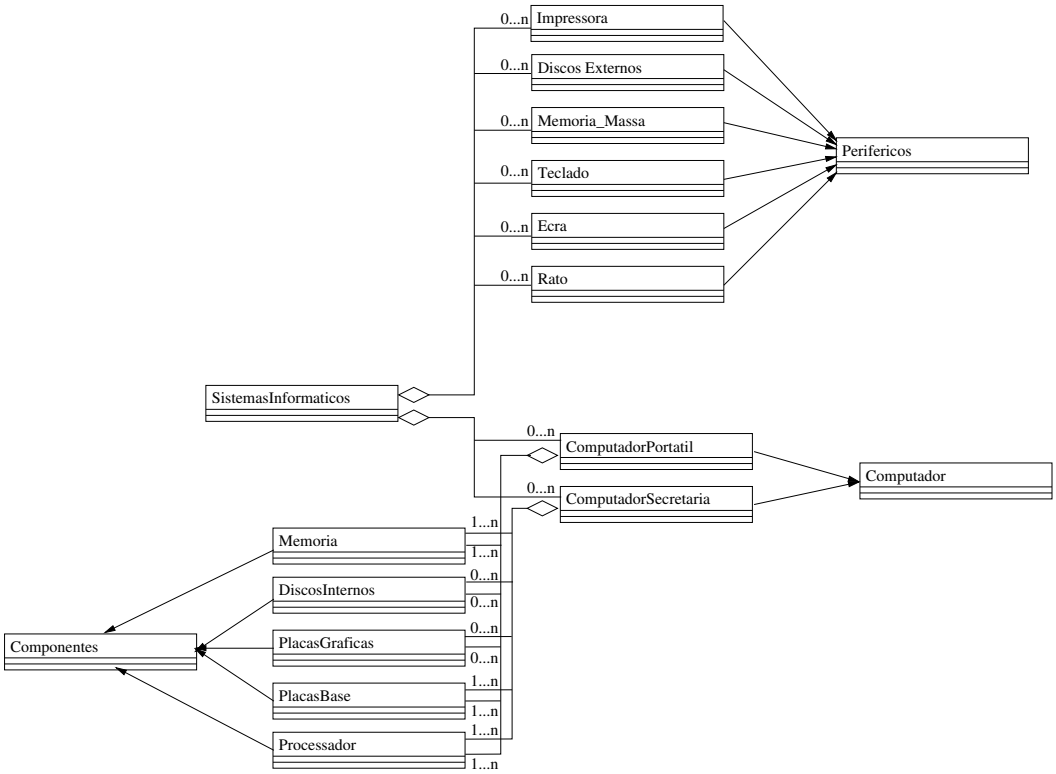
   secção privada: atributos e/ou métodos só acessíveis dentro da própria classe.

3. Faltam os diagramas UML respectivos.

   - Relação de herança (simples ou múltipla).
   - Relação de Agregação.
   - Relação de Instanciação.
   - relação de Meta-classe.

4. ```
   template <class Elementos>
   class ArvBin {
   public:
     ArvBin();
     ~ArvBin();
     void criaAB(Elementos raiz, ArvBin abesq, ArvBin abdir);
     void insRaiz(Elementos raiz);
     void insABesq(ArvBin abesq);
     void insABdir(ArvBin abdir);
     Elementos obtemRaiz();
     ArvBin obtemABesq();
     ArvBin obtemABdir();
     void destroiAB();
     bool vazia();
   private:
     struct No {
       Elementos raiz;
       No *ABesq;
       No *ABdir;
     };
     No* ptAB; // apontador para o primeiro elemento
     // define como vazios os constructores por cópia e por atribuição
     ArvBin (const ArvBin &){}
     void operator=(const ArvBin &){}
   };
   ```
   ```

5.


Impressora
Discos Externos
Memoria_Massa
Teclado
Ecra
Rato
Perifericos
0...n
SistemasInformaticos
ComputadorPortatil
ComputadorSecretaria
Computador
Memoria
DiscosInternos
PlacasGraficas
PlacasBase
Processador
Componentes
1...n