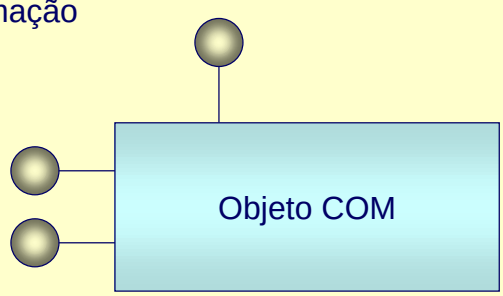
Um Objeto COM

- Tem métodos e estado
- Pode ser implementado por um ou mais objetos de uma linguagem de programação como C++ ou Java
  - Não é a mesma coisa que um objeto da linguagem de programação



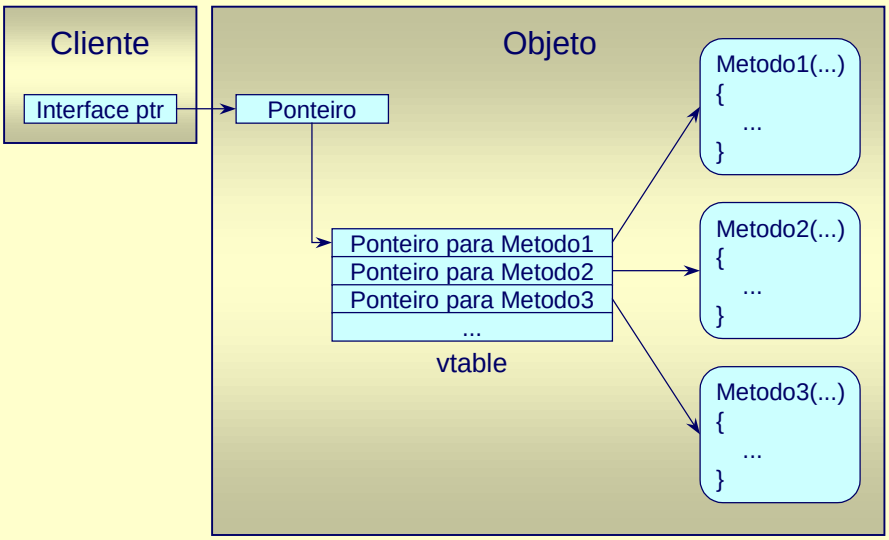
Interfaces de Objetos COM

- Cada objeto COM suporta um certo número de interfaces (geralmente mais de uma)
  - Nomes de interfaces geralmente começam com "**`I`**" (exemplo: **`IUnknown`**)
- Cada interface define um certo número de métodos
- Um cliente do objeto só precisa saber:
  - Quais são esses métodos
  - Como invocá-los
- Interfaces são imutáveis
  - Elas não tem números de versão

Interfaces de Objetos COM (cont.)

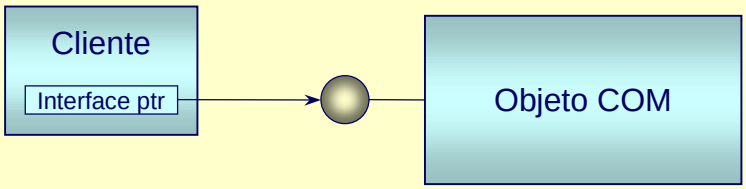
- COM define um padrão binário para interfaces
  - Em princípio qualquer linguagem que possa suportar esse padrão binário pode ser usada para implementar objetos COM
- COM define também uma linguagem para especificação de interfaces chamada Interface Definition Language (IDL)
  - A IDL do COM é diferente da IDL de CORBA
  - A grosso modo ela é um superconjunto da IDL do OSF DCE
  - Não é obrigatório que uma interface COM seja definida em IDL

Anatomia de uma Interface COM



Anatomia de uma Interface COM (cont.)

- O “padrão binário” para interfaces COM é baseado no lay-out de uma *vtable* C++
  - Mais precisamente: é baseado no lay-out das vtables geradas pelo Visual C++ da Microsoft
  - Uma interface COM é uma vtable
  - Um *interface pointer* é um ponteiro para uma vtable
- Representação simplificada:



Invocando Operações

- O cliente que tiver um um *interface pointer* pode usá-lo para invocar métodos da interface
- Suponha que **`pIExemplo`** é um *interface pointer* para uma implementação da interface **`IExemplo`**, implementação essa que faz parte de um certo objeto
  - Em C++, os métodos de **`IExemplo`** poderiam ser invocados assim:

```
pIExemplo->Metodo1(arg1, &arg2);
pIExemplo->Metodo2(arg3);
```

Exemplo de Interface em IDL

```
[ object,
  uuid(E7CD0D00-1827-11CF-9946-444553540000) ]
interface ISpellChecker : IUnknown {
    import ”unknown.idl”;
    HRESULT LookUpWord([in] OLECHAR word[31],
                       [out] boolean *found);
    HRESULT AddToDictionary([in] OLECHAR
                                 word[31]);
    HRESULT RemoveFromDictionary([in] OLECHAR
                                      word[31]);
}
```


UUIDS / GUIDS

- Em COM várias coisas recebem um identificador globalmente único (*globally unique identifier*, ou GUID)
  - GUIDS são também conhecidos como UUIDs (*universally unique identifiers*)
  - Eles são equivalentes aos UUIDs do OSF DCE
  - Um *interface identifier* (IID) é o GUID (ou UUID) de uma interface COM
- Novos GUIDS podem ser obtidos:
  - Rodando um utilitário gerador de GUIDs, ou
  - Contatando a Microsoft

Herança de Interfaces

- Interfaces podem ser derivadas de outras interfaces
  - COM só suporta herança simples (não suporta herança múltipla)
  - Somente a interface é herdada (não há herança de implementação)
- Herança de interfaces é relativamente pouco usada no ambiente COM
  - O mais comum é um objeto simplesmente suportar como interfaces separadas cada uma das interfaces requeridas

Classes

- Cada instância de objeto COM pertence a alguma classe
  - Cada classe é geralmente identificada por um GUID denominado identificador de classe (*class identifier*, ou CLSID)
  - Objetos da mesma classe normalmente suportam as mesmas interfaces
    - Embora isso não seja obrigatório
  - Para saber exatamente que interfaces uma classe suporta, um cliente pode examinar as *component categories* dessa classe
    - Cada categoria é identificada por um GUID denominado *category identifier* (CATID)

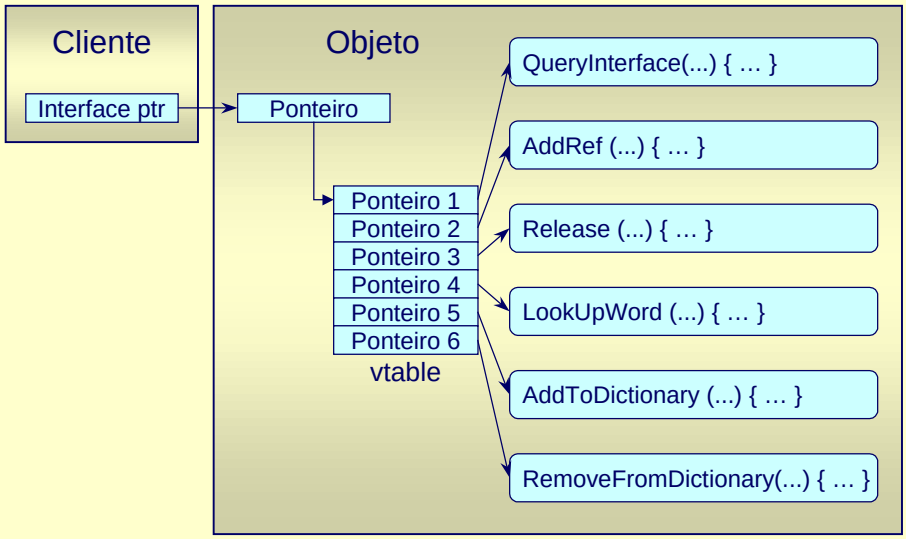
IUnknown, a Interface Fundamental

- Todo objeto COM precisa suportar a interface **IUnknown**
- Toda interface COM é derivada de **IUnknown**
- Os métodos de **IUnknown** permitem que um cliente:
  - Obtenha *interface pointers* para as outras interfaces suportadas pelo objeto
  - Controle o tempo de vida do objeto através de um contador de referências

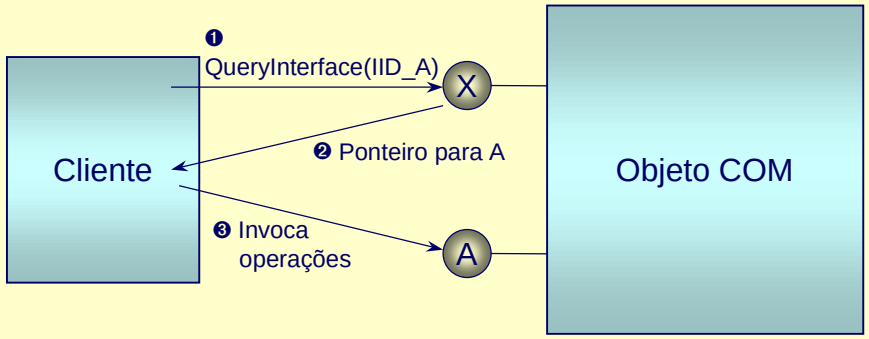
Os Métodos da Interface **IUnknown**

- **QueryInterface**
  - Método chamado para se obter ponteiros para as outras interfaces suportadas por um objeto
- **AddRef**
  - Incrementa o contador de referências do objeto
- **Release**
  - Decrementa o contador de referências do objeto

Vtable da Interface `ISpellChecker`



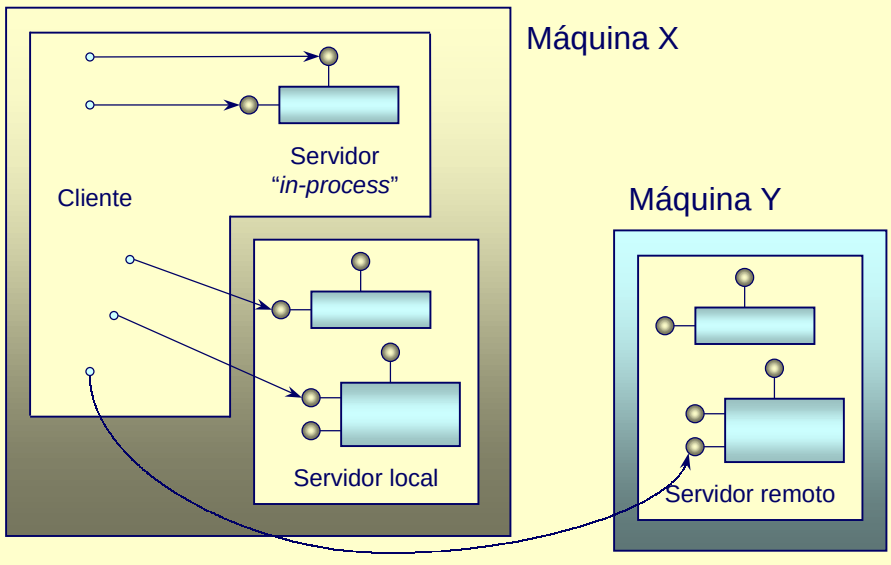
Uso de `QueryInterface`



Contagem de Referências

- Objetos são criados e destruídos dinamicamente
  - Vários clientes podem ter referências para o mesmo objeto
  - Cada objeto mantém um contador indicando quantas referências apontam para o objeto
  - O objeto tipicamente se auto-destrói quando sua contagem de referências chega a zero
- As regras do jogo:
  - Um cliente deve chamar **`AddRef`** num *interface pointer* antes de passar esse ponteiro a outro cliente
  - Antes de liberar um *interface pointer*, um cliente precisa chamar **`Release`** no ponteiro

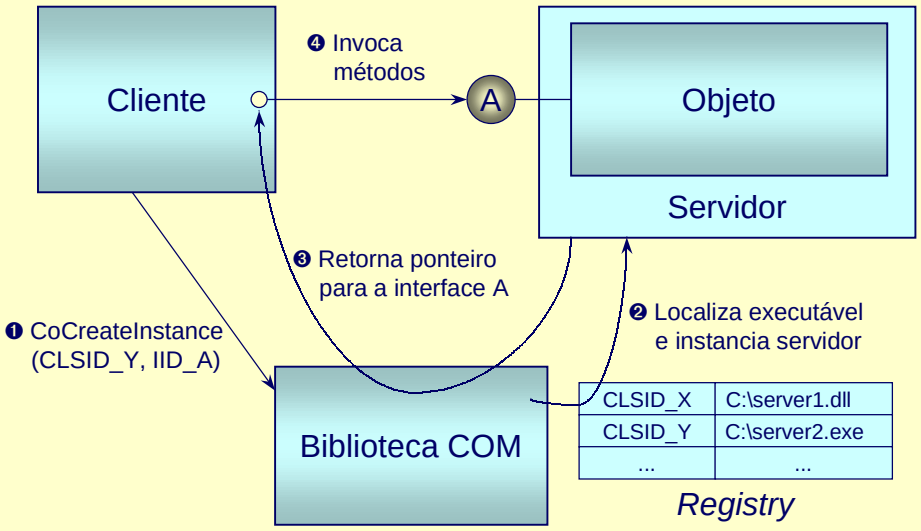
Tipos de Servidores



A Biblioteca COM

- Várias funções de biblioteca precisam ser chamadas para se usar o COM
- A biblioteca COM implementa essas funções
  - Tanto clientes como servidores usam a biblioteca COM
  - A biblioteca é acessada através de chamadas de funções
- No Windows a biblioteca COM é implementada como uma DLL
  - Os nomes das funções da biblioteca COM tem o prefixo "Co". Exemplo: **`CoCreateInstance`**

Criando um Objeto



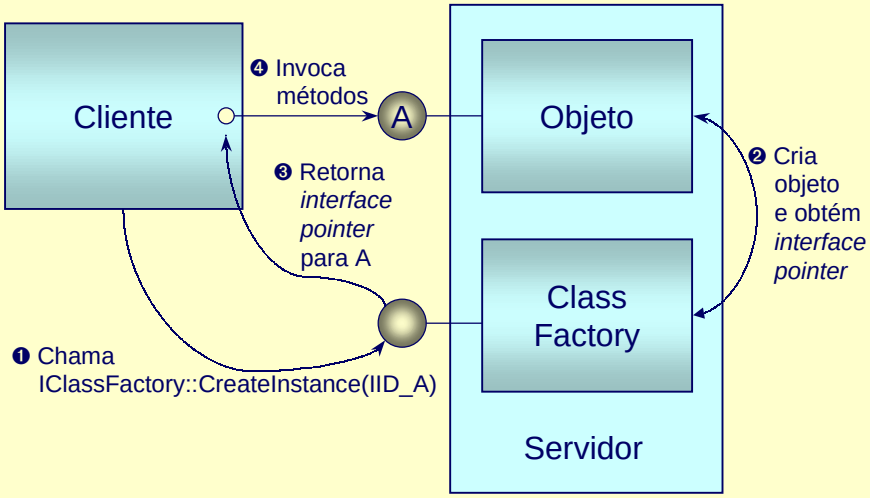
O Registry

- Todos os objetos que o COM pode instanciar tem de ser registrados aqui
- As informações de registro do objeto incluem:
  - O CLSID do objeto
  - Os tipos de servidor disponíveis
  - O nome do arquivo executável ou de outra máquina (no caso do DCOM)
- Quando uma aplicação é instalada ela tipicamente cria uma entrada no *registry*

Usando Class Factories

- Uma *class factory* é um objeto COM que suporta a interface **`IClassFactory`**
  - É útil para se criar muitos objetos da mesma classe
    - Devia se chamar “object factory”, pois fabrica objetos, não classes
- Um cliente pode obter um *interface pointer* para uma *class factory* chamando a função **`CoGetClassObject`** da biblioteca COM
  - O cliente especifica o CLSID e o IID desejados, assim como no caso de **`CoCreateInstance`**
  - **`CoCreateInstance`** é implementada usando *class factories*

Usndo Class Factories (cont.)



Inicializando Objetos

- A instanciação de um objeto com uma *class factory* cria uma instância genérica do objeto
  - Os métodos do objeto ficam disponíveis
- Em muitos casos o cliente precisa mandar o objeto se inicializar
  - Tipicamente a inicialização do objeto inclui a carga do estado persistente do objeto (a leitura de um arquivo, por exemplo)
  - Por isso os objetos geralmente suportam uma das interfaces **`IPersist*`**