Tratando Exceções no Cliente

```
try {
    char* name = quoter->name();
    Long value = quoter->value(argv[1]);
    cout << "Cotação de " << argv[1]
         << ", fornecida por " << name << ": "
         << value << endl;
} catch(UnknownStock& us) {
    cerr << "Não disponível cotação de "
         << us.name << endl;
} catch(SystemException&) {
    cerr << "Exceção de sistema!" << endl;
}
```


_ptrs e **_var**s

- O mapeamento de IDL para C++ define dois tipos de referências para objetos CORBA:
  - Dada uma interface **A**, o compilador IDL gera definições de dois tipos C++, **A_ptr** e **A_var**
  - Ambos tem semântica de ponteiros
  - Ambos podem ser usados para para invocar operações sobre objetos remotos
  - Uma referência tipo **_ptr** requer gerenciamento de memória explícito
  - Uma referência tipo **_var** faz gerenciamento de memória de modo automático (smart pointer)

Usando referências `_var`

```
ORB_var orbref = ORB_init(argc, argv);
Object_var objref =
    orbref->string_to_object(objstr.c_str());
Quoter_var quoter = Quoter::_narrow(objref);
String_var name = quoter->name();
Long value = quoter->value(argv[1]);

// não é mais preciso chamar release para
// quoter, objref ou orbref, nem chamar
// string_free para name
```


Observações

- No exemplo visto o servidor e o cliente foram escritos na mesma linguagem
- Poderiam ter sido escritos em linguagens diferentes
  - Por exemplo: servidor em C++ e cliente em Java
- CORBA provê interoperabilidade entre
  - linguagens de programação
  - sistemas operacionais
  - arquiteturas de hardware