Instituto Federal do Ceará
Engenharia de Computação
Programação Paralela e Distribuída
2019.1
Prof. Cidcley T. de Souza (cidcley@gmail.com)

Projeto de CORBA

Objetivo: Implementar utilizando SOMENTE CORBA um Sistema de Controle de Peças (**Parts**).

O Sistema de Controle de Peças será distribuído em um conjunto de servidores CORBA, cada qual implementando um repositório de informações sobre peças. Cada peça será representada por um objeto cuja interface é **Part**. Cada servidor implementará um objeto **PartRepository**, que é uma coleção de **Parts.**

As definições de IDL de **Part** e **PartRepository** está no final deste documento e deve ser utilizado sem modificações.

Cada objeto **Part** deve ter as seguintes informações:

- O código da peça, que deve ser gerado automaticamente
- O nome da peça
- A descrição da peça

O Servidor

O programa servidor implementará as interfaces `PartRepository` e `Part`. O programa servidor implementará um objeto `PartRepository`, mais a correspondente coleção de objetos `Part`. Isto significa que o programa servidor deve receber como argumentos, certos parâmetros que devem variar de um processo servidor para outro (o nome do servidor, por exemplo).

O Cliente

O programa cliente será usado para utilizar o sistema. Ele deve permitir que o usuário:

- estabeleça uma conexão com um (processo) servidor;
- interaja com o repositório implementado pelo servidor,
    - examinando o nome do repositório e o número de peças nele contidas,
    - listando as peças no repositório,
    - buscando uma peça (por código de peça) no repositório,
    - adicionando ao repositório novas peças (primitivas ou agregadas);
- tendo uma referência a uma peça, referência essa previamente obtida como resultado de uma busca num repositório, interaja com a peça,
    - examinando o nome e a descrição da peça,
    - obtendo o (nome do) repositório que a contém,

O cliente deve apresentar uma interface que espera comandos do usuário. Ele

aceitaria comandos como:

`bind`	Faz o cliente se conectar a outro servidor e muda o repositório corrente. Este comando recebe o nome de um repositório e obtêm do serviço de nomes uma referência para esse repositório, que passa a ser o repositório corrente.
`listp`	Lista as peças do repositório corrente.
`getp`	Busca uma peça por código. A busca é efetuada no repositório corrente. Se encontrada, a peça passa a ser a nova peça corrente.
`showp`	Mostra atributos da peça corrente.
`addp`	Adiciona uma peça ao repositório corrente.
`quit`	Encerra a execuçao do cliente.

A lista acima tem a finalidade de ilustrar como um cliente "linha de comando" poderia funcionar. É apenas uma sugestão (incompleta, por sinal), que pode ser seguida. Clientes com interfaces gráficas podem ser implementados.

Critérios de Avaliação
Interface da Aplicação (0-5) pontos
Implementação de Funcionalidades (0-15) pontos

Trabalho Individual
Data de Entrega: 25/04 (nota cheia)
Entrega até 29/04 (-2 pontos)
Depois disso o trabalho será desconsiderado.
Endereço de Envio: http://tinyurl.com/gnthrbw

Observações:

1. **TODOS** os trabalhos só serão aceitos se apresentados **pessoalmente** pelo aluno na sala de aula na data final de entrega ou, em casos excecionais, a combinar com o professor.
2. **TODOS** os trabalhos só serão recebidos através do link até às 12h da data de entrega.
3. **Não serão aceitos** trabalhos enviados de qualquer outra forma.
4. Devem ser entregues **TODOS** os códigos.
5. Deverá ser entregue, se a linguagem de programação permitir, um código executável (.jar, .exe, etc).

```
// ppdPecas.idl

module PPDCorba {

   // Forward decls:
   //
   interface Part;            // uma "peça"
   interface PartRepository;   // um repositório de informações sobre peças


   // A interface "peça"
   //
   interface Part {

      // Cada peça tem um código, um nome e uma descrição:
      //
      readonly attribute string code;         // código
      readonly attribute string name;         // nome
      readonly attribute string description;  // descrição

   };


   // Para a operação PartRepository::getPartList():
   //
   struct PartListItem {
      string code;           // código da peça
      string name;           // nome da peça
   };
   typedef sequence<PartListItem> PartList;

   // A operação PartRepository::getPartInfo() retorna esta estrutura
   // (os campos `code', `name' e `description' tem a finalidade de evitar
   // acessos remotos à peça referenciada pelo campo `part'):
   //
   struct PartInfo {
      PPDCorba::Part part;     // a peça
      string code;           // código da peça
      string name;           // nome da peça
      string description;     // descrição
   };

   // Exceção levantada pelas operações de busca de peça num repositório:
   //
   exception NoSuchPart {
      string partCode;
   };
```

```
    // A interface "repositório de informações sobre peças":
    //
    interface PartRepository {

        // Atributos:
        //
        readonly attribute string name;     // nome do repositório
        readonly attribute long numParts;   // numero de peças registradas
                                            //   neste repositório

        // Operação que adiciona ao repositório os dados de uma peça
        // e gera um novo código de peça
        //
        Part addPart(in string name, in string description);

        // Operações de busca de peça no repositório:
        //
        Part getPart(in string code) raises(NoSuchPart);
        PartInfo getPartInfo(in string code) raises(NoSuchPart);

        // Operação que lista as peças no repositório:
        //
        PartList getPartList();
    };

};
```