TRABALHO 1
(em duplas)

Quadro de revisões do enunciado:

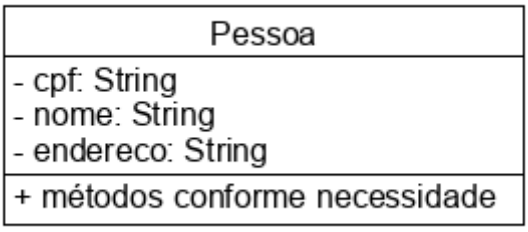
Data	Revisão
26/2/2024	Versão inicial, apresentada nas aulas de 26/2.

O objetivo do trabalho é praticar o desenvolvimento de sistemas distribuídos **usando Sockets**. Será desenvolvido um sistema distribuído composto por uma aplicação servidor e outra cliente para manter dados de objetos (CRUD).

Cenário Base: CRUD de Pessoas

Aplicação Servidor

A aplicação servidor mantém dados de pessoas, com cpf, nome, endereço, conforme diagrama de classes abaixo. A aplicação servidor deve oferecer operações para que os clientes manipulem estes dados remotamente. Os dados são mantidos em memória, mas a equipe pode persisti-los.



A manipulação dos dados é realizada a partir de **mensagens** recebidas do(s) cliente(s), **via Socket**. A mensagem é uma String. Essa String é formada por diversos **campos**, que podem ser separados por algum caractere à critério da equipe (por exemplo ';'). O primeiro campo sempre deve ser a operação, e os demais são todos os dados requeridos pela operação.

As operações suportadas pelo servidor e o conteúdo da mensagem são apresentadas a seguir.

INSERÇÃO DE REGISTRO

Conteúdo da Mensagem	**Campos**	**Conteúdo**
	`operação`	`"INSERT"`
	`cpf`	`O cpf da pessoa`
	`nome`	`O nome da pessoa`
	`endereço`	`O endereço da pessoa`
Descrição	Insere o registro da pessoa	
Retorno	Não há	

ATUALIZAÇÃO DE REGISTRO

Conteúdo da Mensagem	**Campos**	**Conteúdo**
	`operação`	`"UPDATE"`
	`cpf`	`O cpf da pessoa (não pode ser alterado)`
	`nome`	`O nome da pessoa`
	`endereço`	`O endereço da pessoa`
Descrição	Atualiza o registro da pessoa	
Retorno	Se a pessoa for atualizada, então retorna `"Pessoa atualizada com sucesso"`. Se a pessoa não existir, então retorna `"Pessoa não encontrada"`.	

OBTENÇÃO DE REGISTRO

Conteúdo da Mensagem	**Campos**	**Conteúdo**
	`operação`	`"GET"`
	`cpf`	`O cpf da pessoa para obter os demais dados`
Descrição	Busca os dados da pessoa com o CPF informado.	
Retorno	Retorna uma String com os dados da pessoa no formato: `cpf;nome;endereço`. Se a pessoa não existir, então retorna `"Pessoa não encontrada"`. Se não houver nenhuma pessoa cadastrada, então retorna "`Sem pessoas cadastradas`"	

REMOÇÃO DE REGISTRO

Conteúdo da Mensagem	**Campos**	**Conteúdo**
	`operação`	`"DELETE"`
	`cpf`	`O cpf da pessoa que será removida.`
Descrição	Remove o registro da pessoa com o CPF informado.	
Retorno	Se a pessoa for removida, então retorna `"Pessoa removida com sucesso"`. Se a pessoa não existir, então retorna `"Pessoa não encontrada"`. Se não houver nenhuma pessoa cadastrada, então retorna "`Sem pessoas cadastradas`"	

OBTENÇÃO DE TODOS OS REGISTROS

Conteúdo da Mensagem	**Campo**	**Conteúdo**
	`operação`	`"LIST"`
Descrição	Busca todas as pessoas cadastradas para retornar.	

Retorno	A String de retorno é formada pela quantidade de registros existentes seguido pelos dados das pessoas cadastradas, com quebra de linha entre eles. Exemplos:
	
	Se existissem 3 pessoas, o retorno seria:
	`03`
	`01234567891;José;Rua X`
	`23456789012;Maria;Rua Y`
	`34567890123;Pedro;Rua Z`
	
	Se não existirem pessoas cadastradas, o retorno seria apenas o valor 0 (zero).	

Aplicação Cliente

A aplicação cliente deve oferecer meios para o usuário escolher qual operação quer realizar (modo texto ou GUI). Após escolher a operação, a aplicação cliente lê os dados requeridos, conecta com o servidor, envia a mensagem para realizar a respectiva operação, recebe a resposta do servidor, e exibe a resposta ao usuário.

Outros Requisitos das Aplicações Servidor/Cliente

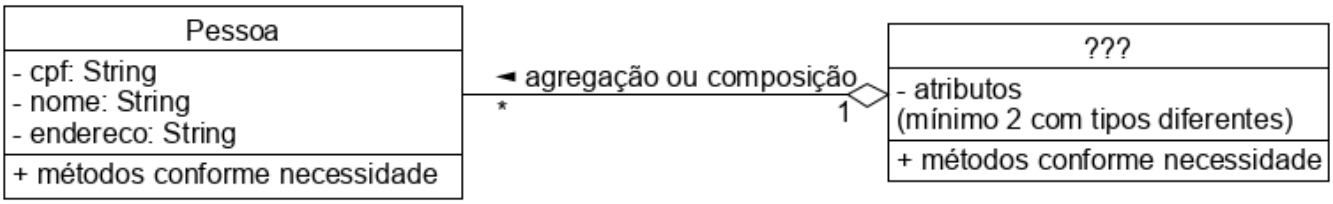
Não é permitido uso de protocolos existentes (ex: http, ws, etc).

O servidor deve permitir que diferentes clientes executem operações sem ter que reiniciá-lo. Contudo essas operações não precisam ser executadas simultaneamente. Exemplo: clienteX faz *insert* da pessoaA; depois um clienteY faz *get* da pessoaA; depois um clienteZ faz *update* da pessoaA.

Recomenda-se que as conexões não fiquem ociosas. Ou seja, que os sockets sejam abertos apenas no momento do envio de cada mensagem, e fechados logo que a resposta seja recebida.

Cenário Estendido: CRUD Pessoas e Outros Objetos

A equipe deve estender o CRUD de Pessoas para implementar um relacionamento de agregação ou composição, conforme diagrama de classes a seguir. A equipe é livre para definir qual classe "???" terá a agregação ou composição com Pessoas. Atenção para a quantidade mínima de atributos.



A equipe deve modificar a aplicação servidor e a cliente para suportar este cenário. As mesmas operações de INSERT/UPDATE/GET/DELETE/LIST devem ser implementadas para a nova classe. Este é o cenário mínimo que a equipe deve implementar, entregar e apresentar.

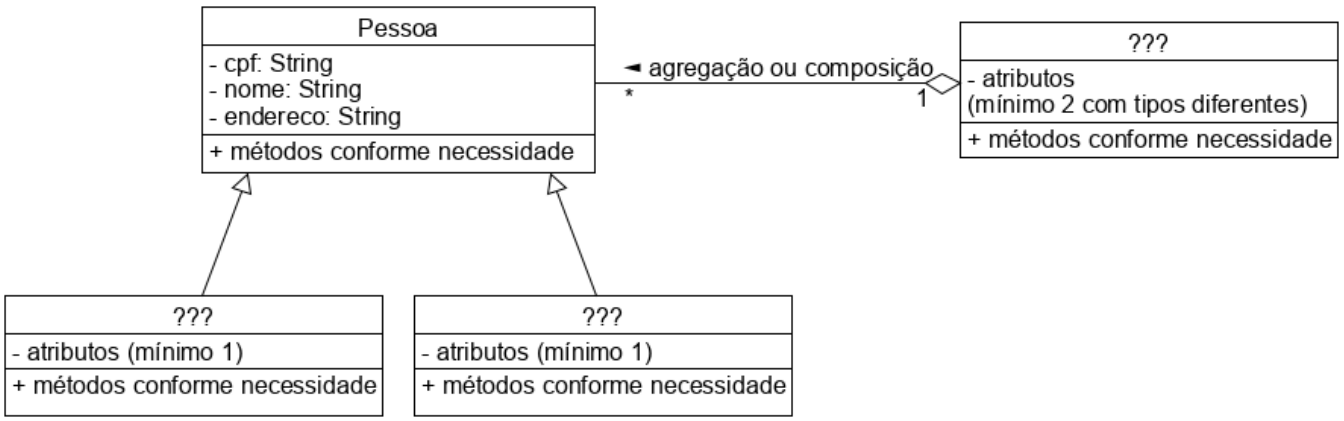
ATENÇÃO:

- Ao retornar um objeto da classe "???", deve retornar também os objetos Pessoa relacionados.
- Para ser vinculado a um objeto "???", os objetos Pessoa já devem estar inseridos no servidor.
- A nova classe (e seus atributos) deve ser DIFERENTE para cada equipe. Usar o *Fórum de escolha da classe* para postar um diagrama de classes, contendo a classe Pessoa, a agregação/composição, e a classe "???" com os atributos que sua equipe escolheu. Consultar este fórum para descobrir as classes escolhidas por outras equipes.

Cenário Extra: Apenas para os Espertos

Este cenário é opcional. Equipes que implementarem 100% funcional (avaliação tudo ou nada) receberão um **crédito de 1.0 ponto**, para utilizar na nota de qualquer trabalho (limitado a nota máxima de 10,0 em cada trabalho).

Estender o CRUD do cenário anterior (com agregação ou composição) para implementar uma **herança**, tendo Pessoa como superclasse conforme diagrama de classes abaixo. Atenção para a quantidade mínima de atributos. Modificar a aplicação servidor e cliente para suportar este cenário.



Entregáveis

Entregar o cenário base + cenário estendido. A entrega do cenário extra é opcional.
O trabalho deve ser entregue na tarefa *Entrega do T1*.
Enviar um arquivo ZIP contendo:

- O projeto do servidor, incluindo:
  - Diagrama de classes do modelo, com a classe Pessoa e as demais escolhidas (em PDF).
  - Diagrama de classes da aplicação servidor, contendo as classes e métodos que realizam as operações CRUD dos objetos do modelo (em PDF).
  - Um documento descrevendo as mensagens das novas operações: campos das mensagens, descrição, e retorno, conforme exemplo do Cenário Base acima (em PDF).
- A implementação do cliente e servidor.
- Executável do cliente e servidor (ex: arquivo "jar" para implementações Java).
- Arquivo PDF dos slides utilizados na apresentação. **É obrigatório elaborar e utilizar slides.**

Data de Entrega

25/3 até 19:00h.
Todas as equipes devem entregar tudo nesta data, independente de quando for apresentar.
A equipe perderá 2.0 pontos na nota por dia de atraso na entrega.

Critérios de Avaliação

Projeto e Implementação (peso 65%)

- Consistência na escolha da classe "???".
- Funcionalidade (deve atender todas as especificações).
- Ausência de erros e adoção de técnicas e padrões de engenharia de software quando necessário (ou seja, ausência de "gambiarras").

Apresentação (peso 35%)

- Exposição do projeto do sistema, das técnicas/padrões de projeto utilizados, dificuldades encontradas e soluções adotadas.
- Demonstração do software desenvolvido.
- Aproveitamento do tempo. O tempo que cada equipe terá para apresentar será divulgado no Moodle. A apresentação, incluindo o tempo de *setup* deve ser realizada dentro desse tempo. Sugere-se fazer o *setup* do ambiente/executáveis previamente.
- Linguagem verbal adequada, termos e expressões relacionados ao assunto.
- Equipes que não apresentaram o trabalho na data/horário agendados recebem nota zero.

Referências Bibliográficas

Tutorial Java sobre TCP (Sockets)
https://docs.oracle.com/javase/tutorial/networking/sockets/index.html

GARG, Vijay K. Concurrent and distributed computing in Java. [Piscataway, N.J.?]: IEEE Press; Hoboken, N.J.: Wiley-Interscience, c2004. Disponível em: <https://ieeexplore.ieee.org/book/5259924>. Acesso liberado via VPN da Udesc.