Trabalho prático
Invocação de Métodos Remotos (RMI)

Objetivo

O objetivo deste trabalho é colocar em prática o projeto e implementação de aplicações distribuídas em conformidade com o modelo cliente-servidor que se comunicam remotamente por meio de invocação de métodos remotos (RMI - *remote method invocation*).

Problemas

Problema 1: Servidor de Autenticação

Implementar um servidor de autenticação que permita a clientes cadastrarem-se utilizando *login* e senha e realizar sua autenticação. O servidor deverá também gravar um arquivo de texto (*log*) com informações acerca de acessos realizados, tais como endereço IP, *login* e data/hora de acesso.

Problema 2: Jogo da Velha Distribuído

Implementar um Jogo da Velha (*tic-tac-toe*) distribuído que permita que dois clientes possam jogá-lo, cada um realizando um movimento por vez. A cada rodada, o programa deverá apresentar um conjunto de opções de movimentos que podem ser feitos. Detalhes sobre o jogo em si podem ser encontrados em https://en.wikipedia.org/wiki/Tic-tac-toe.

Problema 3: Comunicação em Grupo

Implementar um servidor de grupos que permita a criação de grupos e possibilite a clientes entrarem, saírem e enviem mensagens para o grupo para que todos os clientes as recebam.

Problema 4: Sistema de Informações Meteorológicas

Implementar um servidor que forneça informações acerca do tempo (por exemplo, temperatura, umidade relativa do ar, etc.) de várias cidades, conforme requisição de clientes. Essas informações podem ser simuladas ou reais, obtidas a partir de algum serviço desse tipo disponível na Internet.

Problema 5: Cálculos Numéricos

Implementar um servidor que forneça as operações para (i) verificar se um dado número informado pelo cliente é primo, (ii) calcular o fatorial de um número informado pelo cliente e (iii) determinar a série de Fibonacci com o número de termos informado pelo cliente.

Problema 6: Aplicação Bancária

Implementar uma aplicação bancária simples que permita realizar o cadastro de contas bancárias de clientes, bem como as operações de depósito, saque, transferência e emissão de extrato.

Problema 7: Comandos Remotos

Implementar um invocador de comandos remotos através do qual o cliente informa que comando (do *prompt* de comando do sistema Windows ou do terminal do sistema Linux) deverá ser executado no lado servidor e este executa o comando invocado pelo cliente.

Problema 8: Dicionário Remoto

Implementar um dicionário remoto que fornece o significado de palavras (armazenadas em um arquivo de texto no lado servidor) informadas pelos clientes. Caso a palavra requisitada pelo cliente não esteja no dicionário, o servidor informa que esta palavra não pôde ser encontrada e possibilita ao cliente que este adicione a palavra (e seu respectivo significado) ao dicionário remoto.

Problema 9: Votação Eletrônica

Implementar um sistema de votação eletrônica remota no qual clientes informam o seu número de votação (similar ao número do seu título de eleitor, que é único) e o nome do candidato no qual desejam votar. O servidor deverá computar os votos, informando quantos votos cada candidato recebeu até o momento.

Problema 10: Sistema de Estacionamento

Implementar um sistema de estacionamento no qual cada vaga contém sensores remotos que sinalizam se uma determinada vaga está disponível ou não. O sistema deve conter um controlador que armazena quantas vagas estão disponíveis no estacionamento, quais as vagas e quanto tempo o carro está estacionado em uma determinada vaga. O sistema deve levar em consideração que o estacionamento pode ter vários andares.

Problema 11: Sistema de Reserva de Hotel

Implementar um sistema para o gerenciamento de reservas de quarto de um hotel, que possui cinco tipos de quartos com diferentes valores de diária (à escolha). O sistema de gerenciamento das reservas dos quartos do hotel deve receber requisições provenientes de clientes informando o tipo de quarto a ser reservado e o nome do cliente. Após cada reserva, que deve ser devidamente confirmada ao cliente, o sistema deve listar a quantidade de quartos disponíveis para cada tipo. Para fins de simplicidade, considerar-se-á que os clientes estão fazendo reservas

para um período específico, ou seja, não haverá a opção de os clientes escolherem as datas para a reserva.

Problema 12: Conversor de Moedas

Implementar um sistema que realiza a conversão entre moedas ao qual o usuário fornece o valor a ser convertido, a moeda de origem e, opcionalmente, a moeda para a qual o valor deverá ser convertido. Após a conversão, o sistema deverá retornar o valor convertido na moeda solicitada. Caso o usuário não informe a moeda destino da conversão, o sistema deverá exibir o valor convertido para todas as moedas por ele gerenciadas. A taxa de conversão poderá ser fixa no sistema ou obtida a partir de algum serviço disponível na Internet.

Problema 13: Busca Saúde

Implementar um sistema que gerencia informações sobre as unidades de saúde de um município considerando como informações básicas nome, endereço, bairro e especialidades disponíveis. O sistema deverá permitir cadastrar, atualizar ou excluir unidades de saúde, bem como recuperar informações sobre elas a partir de solicitações de consulta feitas pelos usuários - nesse último caso, informado o critério de busca para qualquer uma das informações.

Problema 14: Servidor de Tempo

Implementar um servidor que fornece a data e a hora correntes de acordo com um formato específico a ser informado pelo cliente em sua requisição. Esse formato deverá corresponder aos padrões especificados na classe `java.text.SimpleDateFormat` da linguagem de programação Java (https://docs.oracle.com/javase/10/docs/api/java/text/SimpleDateFormat.html).

Problema 15: Algoritmos de Ordenação

Implementar um servidor que realiza a ordenação de um conjunto de valores inteiros e o tempo despendido para a realização dessa operação. O servidor deverá prover a implementação de diferentes algoritmos de ordenação e os disponibilizar aos clientes, tais como os apresentados em https://en.wikipedia.org/wiki/Sorting_algorithm. Em sua requisição, o cliente deverá especificar o conjunto de valores a ordenar e o tipo de algoritmo que deverá ser utilizado para realizar a ordenação.

Problema 16: Processador de Markdown

Implementar um servidor que realiza a conversão de arquivos com conteúdo na linguagem HTML para arquivos de texto na linguagem Markdown (https://en.wikipedia.org/wiki/Markdown) e vice-versa, de acordo com as respectivas correspondências entre as duas linguagens. O cliente deverá enviar o arquivo que deseja converter, cujo conteúdo será processado pelo servidor e o resultado da conversão será enviado a fim de que seja materializado na forma de um arquivo no lado cliente. Caso o arquivo a ser enviado pelo cliente possua extensão `.html`, subentender-se-á que a conversão será feita de HTML para Markdown, e caso o arquivo possua extensão `.txt`, subentender-se-á que a conversão será feita de Markdown para HTML.

Problema 17: Encurtador de URLs

Implementar um servidor que gerencia o encurtamento de URLs. As URLs curtas a serem geradas pelo servidor possuirão tamanho máximo de cinco caracteres alfanuméricos (desconsiderando o endereço do servidor), a serem concatenados aleatoriamente. O servidor deverá ainda manter uma espécie de repositório para armazenar as URLs completas e suas respectivas formas curtas. Quando um cliente realizar uma requisição, deverá ser enviada a URL em sua forma completa e o servidor deverá retornar uma versão curta dessa URL. Caso a URL ainda não esteja presente no repositório interno do servidor, uma nova entrada deverá ser adicionada; caso contrário, a forma curta é simplesmente retornada.

Problema 18: Banco de Dados Remoto

Implementar um servidor que gerencia operações de consulta, inserção, atualização e remoção sobre um banco de dados relacional remoto ao qual ele possui acesso. O servidor deverá receber requisições de clientes contendo o nome do banco de dados e uma *query* na linguagem SQL e deverá retornar os resultados correspondentes à *query* em questão sobre esse banco de dados. Para a operação de consulta, deverão ser retornados os registros armazenados no banco de dados, enquanto que para as operações de inserção, atualização e remoção deverá ser exibida uma mensagem informando do sucesso ou falha na execução da operação. Para implementar o acesso ao banco de dados (que deverá ter sido previamente criado e populado), deverão ser utilizadas facilidades da API JDBC: http://www.oracle.com/technetwork/java/javase/jdbc.

Tarefas

Você deverá escolher **um** dos problemas descritos anteriormente, com manifestação pública em Fórum aberto na Turma Virtual do SIGAA para esse fim, e implementar uma solução distribuída para ele, em conformidade com o modelo cliente-servidor e utilizando os mecanismos de RMI providos pela linguagem de programação Java. O desenvolvimento da solução deve de antemão visar pela busca de desenvolvimento de *software* de qualidade, isto é, funcionando correta e eficientemente, exaustivamente testado, obrigatoriamente bem documentado e com tratamento adequado de eventuais exceções. É importante salientar que, para este trabalho, o uso de uma interface gráfica com o usuário **não é obrigatório**.

Além da implementação da solução para o problema escolhido, você deverá elaborar um relatório escrito simples descrevendo, pelo menos:

- como a solução foi projetada em termos de sua arquitetura, justificando inclusive decisões específicas de projeto que foram adotadas;
- a lógica de execução da solução e como as entidades que a constituem se comunicam;
- instruções para compilação e execução do programa;
- eventuais dificuldades encontradas durante o desenvolvimento.

Autoria e política de colaboração

O trabalho deverá ser feito em equipe composta de **no máximo dois estudantes**. O trabalho em cooperação entre estudantes da turma é estimulado, sendo admissível a discussão de ideias e estratégias. Contudo, tal interação não deve ser entendida como permissão para utilização de (parte de) código fonte de colegas, o que pode caracterizar situação de plágio. Trabalhos copiados em todo ou em parte de outros colegas ou da Internet serão sumariamente rejeitados e receberão nota zero.

Conforme previsto no cronograma do componente curricular, haverá duas sessões para a apresentação oral (com duração máxima de **10 minutos**) do trabalho desenvolvido por cada equipe à turma, às quais **a presença é obrigatória**. Cada equipe deverá responder, com desenvoltura, aos questionamentos que porventura forem levantados. Nessa apresentação também serão analisadas a autoria do trabalho desenvolvido e a contribuição real de cada integrante da equipe, mesmo que uma determinada parte do trabalho tenha sido conduzida por outro membro da equipe. Portanto, é possível que ocorra, após a entrevista, ajustes nas notas individuais dos estudantes de uma mesma equipe.

Entrega

O trabalho deverá ser entregue até as **23h59 do dia 29 de agosto de 2018, prazo que não será estendido**. Exclusivamente através da opção *Tarefas* da Turma Virtual do SIGAA, deverá ser submetido um único arquivo compactado contendo todos os códigos fonte referentes à implementação da solução para o problema escolhido, disponibilizados sem erros e devidamente testados e documentados, além do relatório escrito, preferencialmente em formato PDF. Se for o caso, é possível fornecer o endereço de um repositório remoto destinado ao controle de versões, porém esta opção **não exclui** a necessidade de submissão dos arquivos via SIGAA, os quais serão exclusivamente avaliados.

Avaliação

A avaliação deste trabalho será feita principalmente sobre os seguintes critérios: (i) utilização correta dos conceitos e facilidades de comunicação remota entre processos; (ii) corretude da execução do programa implementado; (iii) aplicação de boas práticas de programação, incluindo legibilidade, organização e documentação de código fonte, e; (iv) qualidade do relatório produzido. O trabalho possuirá nota máxima de 5,0 (cinco) pontos e será contabilizada para a primeira unidade da disciplina.