UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

PROGRAMA DE GRADUAÇÃO EM REDES DE COMPUTADORES

JOÃO FERREIRA LIMA NETO

VINICIUS GABRIEL RODRIGUES DO NASCIMENTO

RELATÓRIO DA PRIMEIRA ENTREGA DO TRABALHO FINAL DE SISTEMAS DISTRIBUÍDOS COM O TEMA DE “LOJA DE PEÇAS DE VEÍCULOS”

QUIXADÁ

QUESTÕES 1 E 2

1. Introdução

O código é uma aplicação em Java que demonstra a manipulação de fluxos de entrada e saída personalizados para serializar e desserializar objetos de uma classe chamada Peca. Ele também apresenta a interação com arquivos e comunicação em rede via protocolo TCP.

2. Funcionalidades

Definição da Classe Peca e Suas Subclasses:

Peca: Classe base representando peças com atributos de nome, código e quantidade.
Subclasses: Amortecedor, Motor e Pneu
Manipulação de Fluxos Personalizados:

PecasInputStream: Lê objetos Peca de um fluxo de entrada.
PecasOutputStream: Escreve objetos Peca em um fluxo de saída.
Interação com o Usuário:

- Solicita ao usuário que insira dados de peças manualmente via console.

Escrita e Leitura em Arquivos:

- Os dados das peças são gravados em um arquivo binário (pecas.dat) e posteriormente lidos.

- Comunicação via Sockets:

- Implementa comunicação cliente-servidor para enviar e receber objetos Peca usando sockets TCP.

3. Estrutura do Código

Definição das Classes:

Peca e suas subclasses contêm os atributos e métodos necessários.
PecasInputStream e PecasOutputStream extendem InputStream e OutputStream respectivamente, permitindo a serialização e desserialização personalizada dos objetos Peca.
Execução Principal (main):

Coleta de Dados: Lê peças do console e armazena em uma lista.
Escrita e Leitura em Arquivo: Os objetos são gravados no arquivo binário pecas.dat.
O arquivo é lido e os dados são apresentados no console.
Comunicação Cliente-Servidor: Um servidor TCP é criado para receber os dados das peças enviados por um cliente TCP.

Fluxo de Execução: Interação com o usuário → Escrita em arquivo → Leitura de arquivo → Comunicação via rede.

QUESTÃO 4

Relatório sobre a Aplicação de Votação

1 - Descrição Geral

A aplicação consiste em um sistema de votação cliente-servidor implementado em Java, utilizando sockets para comunicação. O servidor gerencia os candidatos, os votos e autentica os usuários, enquanto o cliente fornece uma interface para os usuários interagirem com o servidor.

2 - Componentes e Funcionalidades do Servidor (VotingServer)

Porta e Inicialização: O servidor é inicializado em uma porta definida (8080) e suporta múltiplas conexões concorrentes.
Um cronômetro limita o período de votação a 10 minutos; após isso, apenas o administrador pode logar.

Autenticação: Usuários e senhas são armazenados em um ConcurrentHashMap para evitar condições de corrida.

Dois tipos de usuários:
***Administrador*:** Pode adicionar, remover candidatos e visualizar resultados.
Eleitor: Pode visualizar a lista de candidatos e votar.

Gerenciamento de Candidatos: Os candidatos são armazenados em uma lista sincronizada. O administrador pode adicionar e remover candidatos.

Contagem de Votos: A contagem de votos é gerida por um HashMap, associando cada candidato ao número de votos recebidos.

Comunicação Cliente-Servidor: Cada cliente é gerenciado por uma thread separada (ClientHandler). Protocolo de comunicação com mensagens trocadas via DataInputStream e DataOutputStream.

3 - Ações do Servidor:

Eleitor:
Recebe a lista de candidatos.

Envia o nome do candidato escolhido.
Incrementa o número de votos no servidor.

Administrador:
Adicionar ou remover candidatos.
Consultar resultados da votação.
Cliente (VotingClient)

4 - Conexão ao Servidor:

O cliente conecta-se ao servidor usando um endereço e porta configuráveis.
Protocolo de comunicação similar ao servidor (DataInputStream e DataOutputStream).
Autenticação e Login:

Usuário insere nome e senha.
Caso as credenciais sejam inválidas ou o período de votação tenha expirado (para eleitores), o acesso é negado.
Fluxo de Interação:

Administrador:
Apresenta um menu para adicionar, remover candidatos ou visualizar resultados.
Permite interagir dinamicamente com o servidor para gerenciar o processo de votação.

Eleitor:
Exibe a lista de candidatos.
Solicita a seleção de um candidato e envia o voto ao servidor.
Retorna confirmação da validação e registro do voto.
Destaques Técnicos

Uso de Threads:
Cada conexão de cliente é gerenciada em uma thread separada, permitindo suporte a múltiplos usuários simultâneos.

Sincronização e Confiabilidade:

Uso de ConcurrentHashMap para gerenciar usuários e evitar problemas de concorrência.
O servidor usa ScheduledExecutorService para programar o término do período de votação.
Modularidade:
O servidor e o cliente são organizados em métodos bem definidos, facilitando manutenção e extensibilidade.

Comunicação Simples e Direta:
Comunicação eficiente com DataInputStream e DataOutputStream.
Testes e Cenários de Uso
Cenário 1 - Votação Simples:

Usuário "user1" conecta ao servidor e vota em "Candidato1".
O voto é registrado e confirmado pelo servidor.
Cenário 2 - Ações do Administrador:

Usuário "admin" conecta ao servidor.
Adiciona um novo candidato "Candidato4".
Remove o candidato "Candidato3".
Consulta os resultados.
Cenário 3 - Limite de Tempo:

Após 10 minutos, o servidor bloqueia acessos de eleitores.
Apenas o administrador pode conectar e consultar resultados.