Sistemas Distribuídos

shayra.kelly@estudante.ifgoiano.edu.br
Mudar de conta

Rascunho salvo.

* Indica uma pergunta obrigatória

Enviar por e-mail *

- [x] Registrar **shayra.kelly@estudante.ifgoiano.edu.br** como o e-mail a ser incluído na minha resposta

Qual método da classe ServerSocket é responsável por bloquear a execução até que um cliente se conecte? * 1 ponto

- ( ) close()
- ( ) getLocalPort()
- ( ) bind()
- (•) accept()

O método close() na classe ServerSocket em Java é utilizado para: * 1 ponto

- ( ) Interromper apenas os fluxos de saída (OutputStream).
- (•) Fechar o servidor, liberando a porta utilizada.
- ( ) Pausar temporariamente a comunicação entre cliente e servidor.
- ( ) Encerrar a execução da aplicação Java

Por que a serialização e desserialização são importantes na comunicação com sockets? * 1 ponto

- ○ Porque o TCP exige sempre o uso de buffers de armazenamento.
- ○ Porque os sockets transmitem apenas pacotes em JSON.
- ◉ Porque objetos e dados precisam ser convertidos em bytes para trafegar na rede e reconstruídos do outro lado.
- ○ Porque o Java não suporta envio de Strings diretamente.

O socket pode ser entendido como: * 1 ponto

- ○ Um protocolo de rede exclusivo para sistemas distribuídos.
- ○ Um hardware que conecta cabos de rede diretamente à máquina.
- ◉ Uma abstração de software que encapsula um endpoint e fornece uma API para comunicação.
- ○ Uma biblioteca externa que precisa ser instalada à parte em Java.

No uso de PrintWriter com sockets em Java, o parâmetro true no construtor indica que: * 1 ponto

- ◉ O envio dos dados será imediato, sem ficar retido no buffer.
- ○ O socket fechará automaticamente após o envio da mensagem.
- ○ O envio será criptografado por padrão.
- ○ O servidor receberá confirmação automática de entrega.

Em relação à comunicação via sockets, serialização é: * 1 ponto

- (●) O processo de transformar um objeto em uma sequência de bytes para transmissão pela rede.
- ( ) A capacidade de transmitir apenas Strings pela rede.
- ( ) O processo de transformar bytes em caracteres no momento da recepção.
- ( ) A técnica de criptografar dados sensíveis em trânsito.

Em qual cenário os sockets UDP são mais indicados? * 1 ponto

- (●) Para comunicações rápidas onde a perda ocasional de pacotes não compromete, como jogos online e streaming.
- ( ) Para transferência de arquivos que não pode perder nenhuma parte.
- ( ) Para sistemas bancários que exigem alta confiabilidade de dados.
- ( ) Quando é essencial garantir que todos os pacotes cheguem na ordem correta.

Quando um cliente e um servidor estabelecem comunicação via Socket, quais streams são criados? * 1 ponto

- ( ) Apenas OutputStream, já que o envio de dados é suficiente.
- ( ) Apenas DatagramStream, que abstrai TCP e UDP.
- (●) Fluxos de entrada e saída: InputStream e OutputStream.
- ( ) Apenas BufferedReader, que cuida tanto da entrada quanto da saída.

Um serviço em execução em uma máquina é identificado de forma única por: * 1 ponto

- ○ Apenas pelo endereço MAC da placa de rede.
- ○ Nome do protocolo e versão do sistema operacional.
- ◉ Endereço IP e número da porta
- ○ Nome do servidor e senha de acesso.

Um exemplo de aplicação onde o uso de sockets UDP é mais apropriado seria: * 1 ponto

- ○ Acesso a páginas web, que exigem confiabilidade total na transmissão.
- ◉ Jogos online, onde a prioridade é a baixa latência mesmo com perda ocasional de pacotes.
- ○ Transferência de arquivos bancários que não podem sofrer perdas.
- ○ Envio de e-mails, que requer garantias de entrega.

Qual das alternativas descreve corretamente o papel das portas em uma comunicação via socket? * 1 ponto

- ○ Servem apenas para identificar a velocidade da conexão.
- ○ Substituem a necessidade de usar endereços IP.
- ◉ Definem qual processo ou serviço dentro da máquina vai receber a comunicação.
- ○ São usadas apenas em UDP, não em TCP.

Qual é o ciclo correto de comunicação entre cliente e servidor utilizando sockets TCP? * 1 ponto

- ( ) Cliente cria accept(), servidor cria Socket, estabelecem conexão, trocam dados e encerram.
- (•) Servidor cria ServerSocket, cliente cria Socket, estabelecem conexão, trocam dados e encerram.
- ( ) Cliente cria ServerSocket, servidor usa Socket, ambos trocam dados e encerram.
- ( ) Servidor e cliente utilizam apenas OutputStream, pois InputStream não é necessário.

Sobre os tipos de sockets, assinale a alternativa correta: * 1 ponto

- (•) Sockets TCP são orientados à conexão e garantem entrega confiável das mensagens.
- ( ) Sockets TCP não garantem a ordem de entrega dos dados.
- ( ) Sockets UDP retransmitem pacotes em caso de falha, garantindo integridade.
- ( ) Sockets UDP são orientados à conexão e ideais para aplicações críticas.

O handshake no protocolo TCP é responsável por: * 1 ponto

- (•) Estabelecer uma conexão confiável entre cliente e servidor antes da troca de dados
- ( ) Escolher o tamanho do buffer padrão de cada conexão.
- ( ) Fechar a conexão de forma ordenada.
- ( ) Converter automaticamente caracteres em bytes.

Por que usamos classes como InputStreamReader e OutputStreamWriter ao trabalhar com sockets em Java? * 1 ponto

- ○ Para criptografar automaticamente os dados em trânsito.
- ○ Para evitar que o cliente precise usar PrintWriter.
- ○ Para reduzir a latência de rede em conexões TCP.
- ◉ Para converter entre caracteres e bytes durante a transmissão.

O método readLine() de um BufferedReader baseado em um InputStream associado a um socket é chamado de: * 1 ponto

- ◉ Método bloqueante, que aguarda até que uma linha de dados seja recebida.
- ○ Método exclusivo para uso em conexões UDP.
- ○ Método não bloqueante, que retorna imediatamente mesmo sem dados.
- ○ Método de escrita, que envia dados ao servidor.

Um socket em sistemas distribuídos pode ser definido como: * 1 ponto

- ○ Um protocolo usado exclusivamente para envio de pacotes UDP.
- ○ Um número de porta associado unicamente a um endereço IP.
- ◉ Um ponto de comunicação bidirecional entre dois processos em rede.
- ○ Uma classe específica do Java utilizada apenas para transferência de arquivos.

Em termos práticos, um endpoint em comunicação de rede é: * 1 ponto

- ○ A classe Java utilizada para enviar pacotes pela rede.
- ◉ A combinação de um endereço IP e uma porta que identifica um serviço específico em uma máquina.
- ○ O número total de conexões que um servidor aceita simultaneamente.
- ○ O fluxo de dados associado a um objeto Socket.

Qual alternativa descreve corretamente a diferença entre socket e endpoint? * 1 ponto

- ○ O socket é sempre físico (hardware), enquanto o endpoint é lógico (software).
- ◉ O socket é uma abstração de software para comunicação, enquanto o endpoint identifica precisamente o endereço IP e porta de destino.
- ○ Não há diferença entre socket e endpoint, pois são sinônimos.
- ○ O endpoint é um protocolo de comunicação, e o socket é a implementação em Java.

No ciclo de comunicação TCP, qual é a ordem correta dos eventos? * 1 ponto

- ◉ Servidor cria ServerSocket, cliente cria Socket, conexão é estabelecida, ocorre a troca de dados e, ao fim, há o encerramento.
- ○ Cliente envia dados sem precisar de conexão, servidor responde com confirmação e encerra.
- ○ Servidor e cliente utilizam Datagramas para garantir ordem de entrega.
- ○ Cliente cria ServerSocket, servidor cria Socket, troca de dados e encerramento.

Enviar

Limpar formulário