MAUÁ	PREENCHA SEUS DADOS NOS CAMPOS AMARELOS	PROVA P4
Disc.: **ECM251 – LINGUAGEM DE PROGRAMAÇÃO I**		
Curso: **ENGENHARIA DA COMPUTAÇÃO**		

Aluno: **Pedro Wilian Palumbo Bevilacqua**		Test Code: **MCJQP**
Curso: **ECM** Série **3ª** Período: **Matutino**		
São Caetano do Sul, 18 de novembro de 2025.	RA: **23.01307-9**	
Assinatura:	Nota:	

Instruções da Prova

1. Esta prova é individual e prática, devendo ser realizada nos computadores do IMT, sendo permitido ao aluno, se assim desejar, utilizar seu próprio computador, sob sua inteira responsabilidade;
2. Não poderá haver acesso à Internet, sob nenhuma circunstância, exceto ao CANVAS LMS do próprio aluno e, mesmo assim, somente em duas etapas: para receber ("baixar") as questões de sua prova na máquina em uso; e para entregar ("subir") as resoluções das questões de sua prova no devido local de entrega, na mesma plataforma.

   IMPORTANTE: O aluno deverá informar ao professor quando fará os dois acessos permitidos a ele ao CANVAS LMS, pela Internet;
3. Poderá haver consulta a qualquer material do próprio aluno, seja ele físico (livros, artigos, material de aula etc.) ou virtual (livros, artigos, material de aula, exercícios, resoluções etc.), desde que esse material esteja previamente armazenado em seu computador ou em qualquer dispositivo de armazenamento externo de sua posse (*pendrive*, *hd* externo etc.).

   IMPORTANTE: Não será permitido o acesso pela Internet a pastas compartilhadas (*Google Drive*, *OneDrive* etc.), nem a repositórios virtuais (*GitHub*, *GitLab*, *BitBucket* etc.), mesmo sendo de posse e administração do próprio aluno;
4. O aluno deverá responder às questões da prova no próprio arquivo da prova (*.docx*) devidamente identificado (RA e Nome completo do Aluno) e quando for o caso, gerando os códigos necessários, somente na linguagem de programação JAVA e utilizando sempre e somente o paradigma da Programação Orientada à Objetos visto nas aulas;
5. Para realizar a entrega da prova na plataforma CANVAS LMS, o aluno deverá gerar e entregar um único arquivo compactado (*.rar* ou *.zip*), tendo seu RA e seu NOME completo como nome desse arquivo. Nesse arquivo compactado o aluno deverá fornecer, obrigatoriamente, os seus dados acadêmicos preenchidos na prova, bem como a resolução das questões nela solicitadas (*.docx*), além dos arquivos e códigos gerados em suas resoluções, uma pasta para cada questão, contendo as suas classes e demais arquivos que possibilitem sua posterior execução pelo professor, durante a resolução.
   IMPORTANTE: Na correção, para executar o código gerado pelo aluno em sua prova, o professor seguirá exatamente as instruções fornecidas pelos alunos e contidas nas resoluções das provas! Caso não obtenha sucesso, a questão será considerada errada.
6. Não poderá haver troca de informações, nem de materiais, sejam físicos ou virtuais, entre os alunos durante a prova;
7. Não é permitido ao estudante se ausentar da sala antes da entrega da prova;
8. Celulares, *Smartphones*, *Smartwatches* e outros equipamentos eletrônicos devem permanecer desligados enquanto o estudante estiver na sala;
9. O tempo limite para realização da prova é de **90** minutos;
10. Mantenha sobre a carteira apenas um documento com foto, caneta, lápis e borracha;
11. O entendimento das questões faz parte da avaliação;
12. O tempo mínimo de permanência na sala é de **30** minutos;
13. O estudante que chegar atrasado em até **30** minutos do início da prova poderá fazê-la.

<u>Questão 1</u>: (*1 ponto*)

Stream, em Inglês, significa corrente, fluxo ou riacho e, a partir do *Java 8*, seu conceito foi ampliado para se referir a um novo recurso da linguagem, utilizado para realizar o processamento dos dados de forma sequencial (serial) ou simultânea (paralela), podendo se referir ao fluxo de entrada e saída de dados, ou ao processamento de dados sequencial ou simultâneo.

Levando-se em consideração esse contexto, a bibliografia adotada e o material de aula utilizado, avalie as seguintes asserções e a relação proposta entre elas:

I. Os *streams* são os canais de comunicação entre o cliente e o servidor, ou seja, os *sockets*;

PORQUE

II. Um *stream* é usado em programação para permitir imprimir, ler e gravar dados de/em dispositivos e entrada e saída e, ao usá-lo, pode-se enviar ou receber os dados por meio de um fluxo.

A respeito dessas asserções, assinale a única opção correta:

a) As asserções I e II são proposições falsas;

b) As asserções I e II são proposições verdadeiras e a II é uma justificativa da I;

c) As asserções I e II são proposições verdadeiras e a I é uma justificativa da II;

d) As asserções I e II são proposições verdadeiras e não são justificativas uma da outra;

e) A asserção I é uma proposição falsa e a II é uma proposição verdadeira.

Questão 2: (*1 ponto*)

Na Engenharia de *Software*, a Arquitetura *Cliente-Servidor* é um modelo fundamental que descreve a distribuição de funções e responsabilidades em um sistema de *software*. Nesse modelo, o sistema é dividido em duas partes principais, *Cliente* e *Servidor*, que interagem entre si, via rede de comunicação e seus protocolos de comunicação, para realizar tarefas específicas através de transações e interações.

Sendo assim, com base na referida bibliografia e no material de aula utilizados, avalie as afirmações a seguir:

I. As principais características da Arquitetura *Cliente-Servidor* são: cliente; servidor; banco de dados; comunicação; distribuição de tarefas; e escalabilidade.

II. Aplicações *Java* para a arquitetura *Cliente-Servidor* possuem uma única aplicação *Java Servidor*, responsável por gerenciar as aplicações dos clientes e intermediar a comunicação entre eles pela rede.

III. Aplicações *Java* para a arquitetura *Cliente-Servidor* possuem sempre duas aplicações *Java Cliente*, cada uma delas conectada com a outra, sempre através do servidor, podendo utilizar tipos diferentes de dispositivos em rede.

IV. Um *socket* representa um canal de comunicação bidirecional entre duas aplicações, com um *socket* de um lado conectado ao *socket* do outro, possibilitando o tráfego de dados entre eles e entre suas aplicações.

É correto, apenas, o que se afirma em:

a) I e II;
b) I e III;
c) II e III;
d) II e IV;
e) III e IV.

Questão 3: (*1 ponto*)

No contexto de aplicações *Java* baseadas na arquitetura *Cliente-Servidor*, especialmente aquelas que envolvem múltiplas conexões simultâneas, o servidor precisa lidar com operações de entrada e saída que são, em sua maioria, bloqueantes. Em cenários como aplicativos de bate-papo, nos quais cada cliente mantém um fluxo contínuo e independente de comunicação com o servidor, torna-se necessário adotar mecanismos que permitam o tratamento concorrente dessas interações.

Levando-se em consideração esse contexto, a bibliografia adotada e o material de aula utilizado, avalie as seguintes asserções e a relação proposta entre elas:

I. Um servidor de bate-papo implementado em Java *Sockets* precisa criar um *thread* separado para cada cliente conectado, permitindo assim comunicação simultânea e independente.

PORQUE

II. A criação de um *thread* separada para cada cliente conectado a um servidor de bate-papo, implementado em Java *Sockets*, é necessária porque o *ServerSocket* não possui capacidade nativa de diferenciar fluxos de entrada provenientes de clientes distintos sem o uso de *multithreading*.

A respeito dessas asserções, assinale a única opção correta:

a) As asserções I e II são proposições falsas;

b) As asserções I e II são proposições verdadeiras e a II é uma justificativa da I;

c) As asserções I e II são proposições verdadeiras e a I é uma justificativa da II;

d) As asserções I e II são proposições verdadeiras e não são justificativas uma da outra;

e) A asserção I é uma proposição verdadeira e a II é uma proposição falsa.

Questão 4: (*1 ponto*)

Na Engenharia de *Software*, a Arquitetura *Cliente-Servidor* é um modelo fundamental que descreve a distribuição de funções e responsabilidades em um sistema de *software*. Nesse modelo, o sistema é dividido em duas partes principais, *Cliente* e *Servidor*, que interagem entre si, via rede de comunicação e seus protocolos de comunicação, para realizar tarefas específicas através de transações e interações.

Sendo assim, sobre *multithreading* no servidor de *chat*, com base na referida bibliografia e no material de aula utilizados, avalie as afirmações a seguir:

I. Cada *thread* criado pelo servidor para um cliente gerencia leitura e escrita simultâneas daquele cliente.
II. O limite de clientes simultâneos depende, entre outros fatores, do limite de *threads* permitido pelo sistema operacional.
III. Se nenhum *thread* adicional for usado, o servidor atenderá todos os clientes, porém com maior latência.
IV. *Threads* de um mesmo processo *Java* compartilham a mesma memória, podendo gerar condições de corrida.

É correto, apenas, o que se afirma em:

a) I, II e IV;
b) I e II;
c) I, II e III;
d) II, III e IV;
e) I, III e IV.

Questão 5: (*3 + 3 = 6 pontos*)

Gerado por qualquer *IDE*, pelo próprio aluno, sem o auxílio de nenhum agente de Inteligência Artificial local ou *online*, fornecer o código fonte, em *Java*, das classes necessárias (incluindo a de execução), implementadas sob o paradigma da *Programação Orientada a Objetos*, além do programa ***.jar*** que as executa diretamente, objetivando resolver o problema descrito a seguir.

ATENÇÃO:

a) O aluno deverá fornecer uma lista de instruções detalhadas para o professor executar as aplicações desenvolvidas por ele;

b) Se os programas *.jar* fornecidos não executarem automaticamente as respectivas aplicações desenvolvidas, serão subtraídos 1 (um) ponto para cada aplicação com problema no *.jar*;

c) As classes soluções desta questão (arquivos *.java*) deverão ser compactadas em um único arquivo (*.zip* ou *.rar*), em conjunto com o respectivo arquivo *.jar* funcional, além deste arquivo *.docx* da prova, contendo todas as respostas para as outras questões e a devida identificação do aluno e entregue em resposta à tarefa do *CANVAS LMS* da prova.

Problemas:

I. Utilizando-se apenas os códigos vistos e praticados em aula e os códigos desenvolvidos pelo próprio aluno para esta disciplina, através de uma aplicação implementada durante a prova, deseja-se criar uma **comunicação bidirecional** entre **dois clientes**, e somente dois clientes, através de **um servidor**, do tipo *chat*. Toda comunicação entre os clientes deverá utilizar uma *UI - User Interface* (Interface com o Usuário) baseada em Janelas (*layout*, *swing* etc.) e não em console;

II. Utilizando-se apenas os códigos vistos e praticados em aula e os códigos desenvolvidos pelo próprio aluno para esta disciplina, através de uma aplicação implementada durante a prova, deseja-se que no **console do servidor** sejam apresentadas todas as mensagens relativas às conexões dos clientes (*status*) e, além disso, todas as mensagens trocadas entre os dois clientes conectados a ele, com devida identificação dos remetentes.

Casos de Teste (sequenciais):

1. Ao executar a **Instância 1 do Cliente**, antes de executar a **Instância 1 do Servidor**, o sistema deverá informar que houve problema, que não conseguiu permanecer com o respectivo aplicativo *Cliente* aberto, pois não há servidor disponível naquele endereço/porta para conexão;

2. Ao executar a **Instância 1 do Servidor**, o sistema deverá informar que não houve problema, que conseguiu executar o respectivo aplicativo *Servidor* e que aguarda pela conexão de até dois clientes ao mesmo tempo;

3. Ao executar a **Instância 2 do Servidor**, o sistema deverá informar que houve problema, que não conseguiu executar o respectivo aplicativo *Servidor*, pois já há um servidor em funcionamento naquele endereço/porta para conexão;

4. Ao executar a **Instância 1 do Cliente**, após executar a **Instância 1 do Servidor**, o sistema deverá informar que não houve problema, que está com o aplicativo *Cliente* #1 aberto, conectado ao servidor e que está aguardando a conexão de um segundo cliente para efetuar a comunicação bidirecional com ele.

5. Ao executar a **Instância 2 do Cliente**, depois de executar **Instância 1 do Servidor** e a **Instância 1 do Cliente**, o sistema deverá informar que não houve problema, que está com o aplicativo *Cliente* #2 aberto, conectado ao servidor e que está com a comunicação bidirecional com o outro cliente ativa, aguardando pelas mensagens que serão trocadas entre eles;

6. Ao executar a **Instância 3 do Cliente**, depois de executar **Instância 1 do Servidor** e as **Instâncias 1 e 2 do Cliente**, o sistema deverá informar que houve problema, que não conseguiu permanecer com o respectivo aplicativo cliente aberto, pois já há duas instâncias de Clientes conectadas através do servidor;

7. Ao se digitar quaisquer mensagens na **Instância 1 do Cliente**, por comunicação bidirecional, essas mensagens deverão ser apresentadas na **Instância 2 do Cliente**, sem ecoar no servidor, nem na instância de origem;

8. Ao se digitar quaisquer mensagens na **Instância 2 do Cliente**, por comunicação bidirecional, essas mensagens deverão ser apresentadas na **Instância 1 do Cliente**, sem ecoar no servidor, nem na instância de origem;

9. Quando qualquer um dos dois clientes, ou o servidor, se desconectar da comunicação, o outro cliente deverá se desconectar automaticamente.