Iniciado em	quarta, 1 Jun 2022, 08:18
Estado	Finalizada
Concluída em	quarta, 1 Jun 2022, 09:24
Tempo empregado	1 hora 5 minutos
Avaliar	**8,05** de um máximo de 10,00(**81**%)

Questão **1**

Correto

Atingiu 0,75 de 0,75

Criar uma nova thread dentro de um processo existente é computacionalmente mais oneroso do que criar um novo processo.

Escolha uma opção:

- Verdadeiro
- Falso ✔

A resposta correta é 'Falso'.

Questão **2**

Correto

Atingiu 0,75 de 0,75

Multithreading é quando múltiplas threads são executadas no mesmo processo.

Escolha uma opção:

- Verdadeiro ✔
- Falso

A resposta correta é 'Verdadeiro'.

Questão **3**

Incorreto

Atingiu 0,00 de 0,75

As threads dentro de um processo são isoladas umas das outras, não compartilhando dados ou outros recursos entre si.

Escolha uma opção:

- Verdadeiro ✖
- Falso

A resposta correta é 'Falso'.

Questão **4**

Correto

Atingiu 0,75 de 0,75

Threads podem ser processadas de forma paralela e concorrente.

Escolha uma opção:

- Verdadeiro ✔
- Falso

A resposta correta é 'Verdadeiro'.

Questão **5**

Completo

Atingiu 0,90 de 1,00

Explique de forma resumida o que é uma "condição de corrida" ou "condição de disputa" e como tal situação pode ser tratada.

Condições de corrida é o nome dado quando duas threads tentam acessar uma região crítica e modifica-la, mas apenas uma pode ter acesso. São definidos meios para que apenas uma tenha acesso ao recurso, conhecidos como exclusão mútua. Busy waiting: cria-se um loop, onde a thread que aguarda acesso retorna a cada intervalo de tempo programado para verificar a disponibilidade. Gera muito gasto da CPU. Sleep/Wake up: A thread que aguarda o acesso entra em modo sleep. Quando o recurso é liberado é enviado um sinal de wake up e então a thread pode utilizar o recurso. Esse método libera a CPU e reduz gasto de recursos. Semáforos: Utiliza uma variável para controlar o acesso, podendo ser do tipo binário (1 e 0) e operações de UP and DOWN. Monitor: O controle é feito pelo compilador e utiliza-se a palavra SYNCHRONIZED para implementar o método (na linguagem JAVA). Mensagem: Cria-se uma simulação de sincronização. Ela pode ser feita de modo síncrono (os recursos são bloqueados ate que seja efetuado todo envio e resposta), semi síncrono (o recurso fica livre após o envio da mensagem e bloqueia novamente quando é enviado a resposta) e assincrono (o bloqueio ocorre apenas durante o envio e recebimento, ficando desbloqueado depois)

Comentário:

Conseguiu explicar bem o conceito, faltou:

- Citar que o controle de tal situação é feito na região crítica, isto é, o trecho do código que acessa os recursos compartilhados.
- Existem outras estratégias bloqueantes além da exclusão mútua, tal qual o uso de variáveis atômicas, onde a thread em execução não é retirada de execução enquanto não concluir a operação

Questão **6**

Completo

Atingiu 0,90 de 1,00

Qual a vantagem da utilização das primitivas futures / promises e async / await?

A principal vantagem é que o bloqueio do recurso ocorre apenas no momento do envio e recebimento da mensagem, deixando o processador livre e gerando menos gastos de processamento.

Comentário: A conclusão poderia ficar um pouco melhor, pois esse desbloqueio é feito em um contexto assíncrono, facilitando que o restante do programa execute em paralelo à espera.

Questão **7**

Completo

Atingiu 1,00 de 1,00

Relacione os conceitos de imutabilidade e de "condições de disputa".

Imutabilidade está relacionada a funções puras, que não não produzem efeitos colaterais ou imprevisíveis. Uma variável, após ter seu valor declarado, ele não irá mudar nunca, podendo ser considerado uma constante. Uma string também terá seu nome definido e que não poderá ser alterado. Ao lidarmos com imutabilidade, deixa-se de falar sobre semáforos ou lock, pois o local de memória ocupado não pode ser alterado durante o processamento.

Comentário:

Questão **8**

Correto

Atingiu 1,00 de 1,00

Relacione os tipos das interfaces funcionais com as funções lambda correspondentes:

UnaryOperator	(nome) -> "Hello " + nome ✔
Supplier	() -> "Hello world!" ✔
Predicate	(idade) -> idade >= 18 ✔
BiConsumer	(num1,num2) -> System.out.println(num1*num2) ✔
Function	(text) -> Integer.parseInt(text) ✔
Consumer	(text) -> System.out.println(text) ✔

Sua resposta está correta.

A resposta correta é:
UnaryOperator → (nome) -> "Hello " + nome,

Supplier → () -> "Hello world!",

Predicate → (idade) -> idade >= 18,

BiConsumer → (num1,num2) -> System.out.println(num1*num2),

Function → (text) -> Integer.parseInt(text),

Consumer → (text) -> System.out.println(text).

Questão **9**

Correto

Atingiu 1,00 de 1,00

Qual é a saída das seguintes linhas de código Java:

```
List<Integer> lista = Arrays.asList(1, 5, 8, 7, 4, 2, 3, 2, 1, 8, 5, 7, 4);
lista.stream().filter(e -> e % 2 == 0).forEach(System.out::print);
```

a. **153157**

b. **158742**

c. **1587423218574**

d. **3218574**

e. **842284** ✔

Sua resposta está correta.

A resposta correta é:
842284

Questão **10**

Completo

Atingiu 1,00 de 1,00

O que são Exceptions verificadas e não-verificadas? Quando é recomendado utilizá-las?

Exceptions verificadas são aquelas que podem ser previstas pelo programador e tratadas, como por exemplo quando um dado inserido pelo usuário não é o correto. Já as não verificadas são aquelas provenientes de erros durante a criação do código, chamados de Runtime Exceptions. Deve-se utilizar as Exceptions quando o erro encontrado pode expor dados sigilosos do sistema, então utiliza uma mensagem de erro amigavel para o usuário e esconde-se os dados sigilosos da empresa. Já para Exceptions que podem ser verificadas, como inserção de dado errado, feito pelo usuário é melhor utilizar outros métodos para trata-lo, visto que o uso de Exceptional é pesado e pode acarretar perda de desempenho do programa.

Comentário:

Questão **11**

Não respondido

Vale 1,00 ponto(s).

Relacione os conceitos de metaprogramação e reflexão.

◄ Apresentação Metaprogramação

Seguir para...

Gilded Rose ►