Conteúdos de PCD

Preparação para a frequência

Primeira Parte - Teórica

1. O que é um livelock?
2. O que é um deadlock?
3. O que é starvation?
4. Coordenação de processos ligeiros com Barrier, CountDownLatch.
5. Diferenças entre a biblioteca NIO java (não bloqueante) e abordagem clássica de Servidor-Cliente
6. O que acontece quando se interrompe uma thread (t1.interrupt()) ? é sempre lançada uma InterruptedException como exceção?
7. O número de processos ligeiros a ser executados simultaneamente pode ser superior ao número de CPUs e núcleos do computador?
8. Podia a coordenação para o ínicio da movimentação ser feita com auxílio de um Semáforo? (Não deve sair)
9. Cite 3 possíveis vantagens em utilizar um cadeado Lock ao invés de synchronized block.
10. Explique as principais diferências entre Thread e Runnable em JAVA.
11. Qual a finalidade das interfaces Future e Callable no contexto de um ExecuterService?
12. O que faz um wait de um objeto?

Segunda Parte- Concorrência

1. Implemente na classe Cell os métodos de request e release de um objeto da classe snake tal como as variáveis locais da classe necessárias para implementar o método.
    - Não utilize blocos sinchronized.
    - Tenha conta que a Snake pode "comer" um objeto da classe Goal chamando métodos isOcuppiedByGoal() e o método eat(Snake snake) da classe Cell e já implementados.
2. Implementa a ThreadPool de workers na classe LocalBoard para o movimento dos "Obstacles". Tendo em conta que:
    - Existem 5 obstáculos em movimentação num dado instante
    - As tarefas vão ser de classe ObstacleMover já definida.

Terceira Parte - Programação em Rede

1. Implementar a Classe relativa ao Cliente Remoto Tendo em conta que:
    - Quando o cliente remoto é ativado deve ligar-se ao servidor e abrir um canal de receção de objetos (GameState) e um canal para envio dos comandos do jogador.
    - O cliente deve poder, a qq momento, receber o estado do jogo.
    - Admita que existe um método na classe para atualizar o estado gráfico do jogo. updateGUI(GameState state) que deve ser chamado sempre que recebe do canal de objetos.
2. Implementa a classe relativa ao Servidor
    - O servidor deve inicialmente estar disponível para receber ligações dos clientes remotos.

- Deve haver a máxima disponibilidade para receber novas ligações de clientes remotos, sem comprometer a disponibilidade do servidor para executar outras tarefas e lidar com outros clientes.
- Quando é recebida a ligação de um cliente remoto, o servidor deve enviar a informação sobre todos os elementos implementados no jogo até então (snakes e cells). Assume que apenas necessita de chamar o método implementado no servidor GameState getGameState() para enviar ao cliente.
- Implemente o método sendGameState() utilzado para enviar o estado do jogo a clientes atualmente ligados. É necessário implementar o método main() bem como construtores e outros métodos necessários para a implementação. Deve ser indicada o tratamento de exceções relevantes.