Nome: _____ **Número:** _____

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
UNIVERSIDADE DO MINHO

Sistemas Distribuídos

Teste[1]

11 de Janeiro de 2018 Duração: 2h00m

I

1 Distinga, em termos de objetivo e forma de utilização, as primitivas de *lock/unlock* e *wait/notify*.

As primitivas lock/unlock permitem controlar a execução de secções críticas de código, lidando com a exclusão mútua relativamente a essa secção. É um método de controlo de concorrência. As primitivas wait/notify permitem suspender a execução de threads específicas, aguardando por uma determinada alteração noutra thread, através de um ciclo while e da primitiva notify / notifyAll.

2 Caracterize, comparativamente, as arquitecturas de sistemas distribuídos **baseadas em dados** e **baseadas em eventos**.

Uma arquitetura baseada em dados, tem vários componentes a aceder a uma memória partilhada e só conseguem ler ou escrever. Em arquiteturas por eventos os diversos componentes conseguem publicar ou subscrever num dado middleware (event bus), sabendo que apenas os componentes que o subscreveram é que vão receber os eventos.

3 Descreva com detalhe o fluxo de operações e dados na realização de uma invocação remota (RPC ou RMI) entre cliente e servidor. Use, por exemplo, a invocação local de uma função `int somar(int a, int b)`.

Uma

[1] Cotação — 10+10

II

Considere um serviço simplificado de transferência de passageiros entre terminais de um aeroporto. Assuma 4 terminais e um shuttle com capacidade para 20 passageiros. Um shuttle pára numa estação durante 1 minuto para permitir saída e entrada de passageiros. O shuttle faz um percurso circular, demorando 3 minutos na viagem entre cada par de terminais. Cada passageiro utiliza uma aplicação (cliente) que interage com o servidor que controla o sistema, devendo permitir: informar o servidor que o passageiro pretende viajar entre um terminal de origem e outro de destino; informar o passageiro que pode entrar no shuttle (o shuttle chegou e há lugar disponível); e informar o passageiro que chegou ao destino.

1 Apresente uma classe (para ser usada no servidor) que implemente a interface abaixo, tendo em conta que os seus métodos serão invocados num ambiente multi-threaded.

```
interface Controlador {
  void quero_viajar(int origem, int destino);
  void quero_sair(int destino);
  void partida();
  void chegada();
}
```

O método `quero_viajar` bloqueia até o passageiro poder ser informado que pode entrar no shuttle; o método `quero_sair`, bloqueia até ao shuttle ter chegado ao terminal `destino`. Os métodos `partida` e `chegada` (para serem usados por uma thread que controla o shuttle) sinalizam a partida/chegada do shuttle.

2 Implemente o programa servidor usando threads (incluindo a thread controladora do shuttle), sockets TCP, e a classe desenvolvida na pergunta anterior.