Nome: _____ **Número:** _____

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
UNIVERSIDADE DO MINHO

Sistemas Distribuídos

Exame[1]

1 de Fevereiro de 2018 Duração: 2h00m

I

1 Considere as primitivas de controlo de concorrência que estudou `lock` e `wait`. Exemplifique **detalhadamente** um caso em que a utilização de `wait` não é dispensável.

Um caso em que a utilização do wait é necessária é no caso do produtor/consumidor, uma vez que em caso de falha de recursos no produtor, o consumidor vê-se necessário a efetuar uma espera, levando a um deadlock.

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 Admita que há vários algoritmos distribuídos para a resolução de um determinado problema (eg. centralizado, descentralizado, em anel, etc.). Que aspectos consideraria se fosse chamado a compará-los?

Um algoritmo centralizado é um algoritmo simples, baseado num coordenador, que envolve pouca troca de mensagens para aceder a um recurso, no entanto apresenta pouca tolerância a falhas e é muito pouco escalável.
Um algoritmo descentralizado é mais complexo e envolve troca de bastantes mensagens para aceder a um recurso, mas mais escalável. Pode ocorrer falhas quando algum recurso falhar.
Um algoritmo em anel, baseia-se na transmissão de um token entre recursos. A perda desse token provoca a falha do sistema.

[1] Cotação — 10+10

II

Considere um serviço ao qual clientes se ligam por TCP, para participarem num jogo em que cada partida pode ter entre 20 a 30 jogadores: uma partida terá de preferência 30 jogadores, mas se algum jogador estiver à espera para jogar há mais de um minuto a partida poderá começar com um número par de jogadores de pelo menos 20. Cada jogador (cliente) envia o seu nome ao servidor, ficando à espera de a partida poder começar, devendo nessa altura ser informado dos nomes dos jogadores da partida.

1 Apresente uma classe que implementa a interface abaixo, tendo em conta que os seus métodos serão invocados num ambiente multi-threaded.

```
interface Jogo {
  List<String> inscrever(String nome);
}
```

O método `inscrever` bloqueia até a partida poder começar, devolvendo os nomes dos jogadores.

2 Implemente um servidor para o serviço de jogo, usando threads, sockets TCP, e a classe desenvolvida na pergunta anterior.