**Nome:** _____ **Número:** _____

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
UNIVERSIDADE DO MINHO

# Sistemas Distribuídos

*Exame*[1]

1 de Fevereiro de 2016 Duração: 2h00m

---

## I

**1** Explique como utiliza a operação de *wait(cond, lock)* e por que razão é necessário associar o argumento *lock*.

A operação de wait utiliza-se em problemas de ordem de execução, isto é quando uma thread não pode avançar enquanto um determinado predicado, associado a um região de memória partilhada, se verifique. Uma vez que se trata de memória partilhada é necessário que se controle a concorrência de threads no acesso a tal região do código, motivo pelo qual cada variável de condição está associada a um lock. Esta operação é utilizada da seguinte forma:

```
lock.lock()
while(!PREDICADO){
        wait(cond,lock);
}
//(FAZER ALGO)
lock.unlock();
```

Antes de verificar o predicado é necessário obter o lock.Por esta razão e de forma a que mais threads possam verificar o predicado enquanto estamos bloqueados, é necessário passar o lock ao wait. A operação de wait inclui pois as operações de libertações e re-obtenção do lock.

**2** Diga o que entende por middleware orientado a mensagens identificando sucintamente os seus principais componentes.

O middleware orientado a mensagens é um método de comunicação assíncrona e persistente baseado em trocas de mensagens entre a componente do software num sistema distribuído. Cada um dos componentes inclui duas filas de mensagens que armazenam as mensagens a enviar e as mensagens recebidas. As mensagens são enviadas para um gestor de mensagens, o "Message broker", que gere um conjunto de filas de mensagens e realiza as trocas de mensagens entre filas. A transmissão de mensagens ocorre através de canais estabelecidos por "Message Channel Agents"(MCA's) que empacotam/desempacotam as mensagens a receber/enviar pelo canal de rede.

**3** Descreva sucintamente as funcionalidades de um servidor de objectos em sistemas de objetos distribuídos.

A principal funcionalidade de um servidor de objetos é a gestão de um conjunto de objetos e intermediar a realização de pedidos aos objetos. Os objetos que controla são detentores dos variados serviços que oferece. Este é responsável por verificar se um determinado pedido se destina a um dos objetos que possui, por assegurar que tal objeto se encontre carregado, por realizar o pedido ao objeto e por retornar a resposta ao cliente. Este deve ainda remover da cache objetos que já não são utilizados há muito tempo.

[1] Cotação — 10+10

## II

**1**

Considere uma biblioteca de apoio a um concurso para ser usada num ambiente multi-threaded. Esta deve permitir serem adicionadas questões (pares pergunta-resposta), e oferecer a possibilidade de threads competirem para tentarem responder. Devem ser implementadas as seguintes interfaces:

```
interface Controlador {
  void adiciona(String pergunta, String resposta);
  Questao obtem(int id);
}
interface Questao {
  String responde(String resposta);
  int id();
}
```

A operação `adiciona` introduz um novo par pergunta-resposta, criando um objecto `Questao`, etiquetado por um *id* numérico crescente (1, 2, 3, ...). A operação `obtem` devolve um objecto que representa uma questão que tiver sido previamente adicionada, com `id` maior do que o argumento, e que se encontre ainda *disponível*: não tenha ainda sido respondida correctamente nem tenha sido sujeita a mais de 10 tentativas de resposta. Caso não exista nenhuma, deverá bloquear até tal ser possível. O método `responde` deverá devolver "R", "C", ou "E", conforme a questão já tiver sido previamente respondida correctamente, a resposta esteja certa, ou a resposta esteja errada, respectivamente. Tenha cuidado para evitar o uso continuamente crescente e desnecessário de memória (*memory leak*).

**2**

Escreva um programa servidor que usando threads, sockets TCP e a biblioteca acima, permita que clientes remotos tentem responder a perguntas. Cada cliente ligado, até fechar a ligação, poderá em ciclo: enviar "Pergunta", esperar pelo enunciado de uma pergunta, enviar a resposta e esperar pelo resultado, que deverá ser "Respondida", "Certa", ou "Errada". O servidor não deve devolver ao mesmo cliente perguntas repetidas. O servidor deverá adicionar uma nova pergunta por minuto, utilizando um método `Util.novaPergunta()`, que devolve um array com duas strings: pergunta e resposta correspondente.