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*.

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

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

[1] Cotação — 10+10

(Confine as suas respostas às áreas pautadas.) v.s.f.f.

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.