PROGRAMAÇÃO DISTRIBUÍDA - 2016/17

Exame Teórico – 20 de Janeiro de 2017

Sem consulta / *Duração: 60 minutos* / *Todas as perguntas possuem a mesma cotação* / *As respostas devem ser objectivas e sintéticas*

1. Diga, justificando, se é correcto afirmar-se que o seguinte método permite apenas copiar o conteúdo de um ficheiro para outro. Se a resposta for negativa, inclua na justificação pelo menos um contra-exemplo.

```
public static void copy(java.io.InputStream in, java.io.OutputStream out) throws java.io.IOException
{
      int c;
      while ((c = in.read()) != -1){
            out.write(c);
      }
}
```

2. Geralmente, associado a plataformas de *middleware* que oferecem a abstracção de objecto remoto/distribuído, existem aplicações que, em termos genéricos, podem ser designadas de serviços de nomeação (por exemplo, rmiregistry.exe, tnamrserv.exe e orbd.exe). Explique quais são os seus dois objectivos principais e de que forma estes são atingidos (ou seja, enumere as principais operações/passos executados desde a fase de arranque). Na resposta, que não deve incluir código, utilize, entre outros, os termos *socket*, mensagem, porto e *thread*.

3. Diga qual é o objectivo do seguinte método e descreva o significado de cada um dos campos que compõem a URL passada como argumento (*rmi*, *192.168.1.1*, *1099* e *RMILightBulb*):

   *java.rmi.Remote r = java.rmi.Naming.**lookup**("rmi://192.168.1.1:1099/RMILightBulb").*

4. O pedaço de código seguinte permite obter um recurso alojado um servidor Web através do protocolo HTTP, recorrendo a uma classe específica que encapsula esse tipo de interacção. Sabendo que o HTTP é um protocolo do nível de aplicação que recorre ao protocolo de transporte TCP e, por imissão, ao porto 80, deduza a sequência de acções/passos principais desencadeados pelo método *openStream* (*1- Estabelece… 2- … m- Envia… n- Obtém… o- Devolve…*). A resposta não deve incluir código.

```
java.net.URL myURL = new java.net.URL ("https://moodle.isec.pt/moodle/mod/folder/view.php?id=470");
java.net.InputStream in = myURL.openStream();
int b;

while(true){
      b = in.read();
      ...
}
```

5. Acrescente uma única linha de código na classe *UseMyThreads* ou *MyThread* de modo a que a *thread t2* apenas inicie depois de *t1* deixar de estar activa. Altere igualmente a declaração do método *metodo1* de modo a que, se várias *threads* possuírem uma referência para a mesma instância da classe *MyThread*, este apenas possa ser executado por uma única t*hread* em cada instante.

```
public class MyThread extends Thread{
      X x;
      public MyThread(X x){
            this.x = x;
      }
      ...
      public metodo1(){
            ...
      }
      ...
      public void run() {
            ...
      }
      ...
}
```

```
public class UseMyThreads{
      ...
      public static void main(String args[]){
            ...
            Thread t1 = new MyThread(new X()).start();
            Thread t2 = new MyThread(new X()).start();
            ...
      }
}
```