*Sistemas Distribuídos*
Teste 1

9 de Junho de 2004

## Parte A (12 valores)

1. Explique o que é um sistema distribuído e indique dois dos aspectos que têm que ser ponderados na sua concepção.
2. Caracterize o modelo cliente-servidor e indique qual é o papel aí desempenhado por um servidor de 'proxies'.
3. Descreva o modelo de programação providenciado pelo protocolo TCP, quer do lado do servidor, quer do lado do cliente.
4. Explique como é que o *RMI* (Remote Method of Invocation) em Java lida com a implementação da transparência de acesso e da transparência de localização. Que limitações apresenta?

## Parte B (8 valores)

```
public class Producer extends Thread
{
   private Store st;
   private int number;
   public Producer (Store s, int number)
   {
     st = s;
     this.number = number;
   }
   public void run ()
   {
     int val;
     for (int i = 0; i < 5; i++)
     { val = 10*number+i;
       st.put (number, val);
       try
       { sleep ((int) (Math.random () * 100));
       } catch (InterruptedException e) { }
     }
   }
}

public class Consumer extends Thread
{
  private Store st;
  private int number;
  public Consumer (Store s, int number)
  {
    st = s;
    this.number = number;
  }
  public void run ()
  {
    int value;
    while (true)
      value = st.get (number);
  }
}

public class Store
{
   private int [] contents = new int[3];
   private boolean full = false;
   private int in = 0,
               out = 0;
```

```
    public synchronized int get (int number)
    {
      int val;
      while (!full && (in == out))
      { try
        { wait ();
        } catch (InterruptedException e) { }
      }
      val = contents[out];
      out = (out+1) % 3;
      System.out.println("Consumidor #" + number + " consome: " + val);
      full = false;
      notify ();
      return val;
    }
    public synchronized void put (int number, int val)
    {
      while (full)
      { try
        { wait ();
        } catch (InterruptedException e) { }
      }
      contents[in] = val;
      in = (in+1) % 3;
      System.out.println("Produtor #" + number + " produz: " + val);
      full = (in == out);
      notify ();
    }
}
```

```
public class ProducerConsumer
{
   public static void main (String[] args)
   {
     Store s = new Store();
     Producer [] p = new Producer[2];
     Consumer [] c = new Consumer[3];
     for (int i = 0; i < 3; i++)
     { c[i] = new Consumer (s, i);
       c[i].start ();
     }
      for (int i = 0; i < 2; i++)
     { p[i] = new Producer (s, i);
       p[i].start ();
     }
  }
}
```

1. Que valores são impressos no écran do monitor vídeo quando este programa é executado?
2. Faça um diagrama que descreva a interacção entre os diferentes 'threads' envolvidos neste programa. Use círculos para representar os 'threads' e rectângulos para identificar as regiões de memória partilhada e identifique claramente cada objecto colocado no diagrama com o nome da variável associada. Explique seguidamente por uma frase curta o papel desem-penhado por cada objecto na interacção.
3. Altere a classe `Store` de modo a que o mecanismo de sincronização seja unicamente baseado em semáforos. Para isso, admita que tem disponível a classe seguinte

```
public class Semaphore
{
  private int val = 0,
              numbBlockThreads = 0;
```

```
    public synchronized void down ()
    {
      if (val == 0)
         { numbBlockThreads += 1;
           try
           { wait ();
           }

    catch (InterruptedException e) {}
         }
         else val -= 1;
    }
    public synchronized void up ()
    {
      if (numbBlockThreads != 0)
         { numbBlockThreads -= 1;
           notify ();
         }
         else val += 1;
    }
}
```