Trabalho 3

Cenas da vida do Pai Natal *

O Pai Natal mantém-se quase sempre na sua casa do Pólo Norte e, face à sua idade avançada, passa boa parte do tempo a descansar. Só intervém em duas situações

- quando as renas, que foram passar férias ao Pacífico Sul, regressam na véspera de Natal para puxar o trenó na viagem de distribuição de presentes às crianças de todo o mundo
- quando alguns dos gnomos que trabalham na fábrica de brinquedos ao lado, têm algum problema e precisam do seu conselho.

Para que o Pai Natal não esteja continuamente a ser incomodado, os gnomos decidiram só lhe bater à porta quando houver exactamente três que pretendam ouvir a sua opinião. Havendo mais gnomos nessa situação, os restantes têm que aguardar pelo regresso dos primeiros à fábrica antes de poderem avançar (sempre em grupos de três).

Se a última rena já tiver voltado dos trópicos, o Pai Natal sabe que chegou o momento de aparelhar o trenó e de dar início à viagem porque é Véspera de Natal. Qualquer grupo de gnomos que esteja nesse momento à espera de ser atendido, terá que esperar pelo seu regresso, dado que a viagem é prioritária.

Supõe-se ainda que as renas regressam de férias uma de cada vez e de uma forma independente umas das outras e que aguardam no estábulo o início da viagem. A última, porém, ao constatar esse facto, vai também bater à porta da casa do Pai Natal para o alertar que as renas já estão todas reunidas. Quando a viagem termina, as renas partem imediatamente para férias para recuperar forças do esforço despendido.

Construa uma simulação das actividades de cenas da vida do Pai Natal baseada no modelo cliente-servidor, com replicação de servidor, em que o Pai Natal, os gnomos e as renas são os *clientes* e as regiões de interacção que tenha estabelecido representam os serviços que lhes são prestados pelos *servidores*. A comunicação deve ser baseada em Strings cujo conteúdo usa XML.

A solução deve ser implementada em Java, ser passível de execução em Linux sobre *sockets* TCP, poder ser executada de uma forma concentrada (numa única máquina), ou de uma forma distribuída (até oito máquinas diferentes), e terminar (deve contemplar a possibilidade de *shutdown* do serviço). A extracção da informação das Strings XML deve seguir o modelo DOM, disponível em Java através da utilização das classes DocumentBuilder, DocumentBuilder, etc. e das interfaces Document, Node, etc.

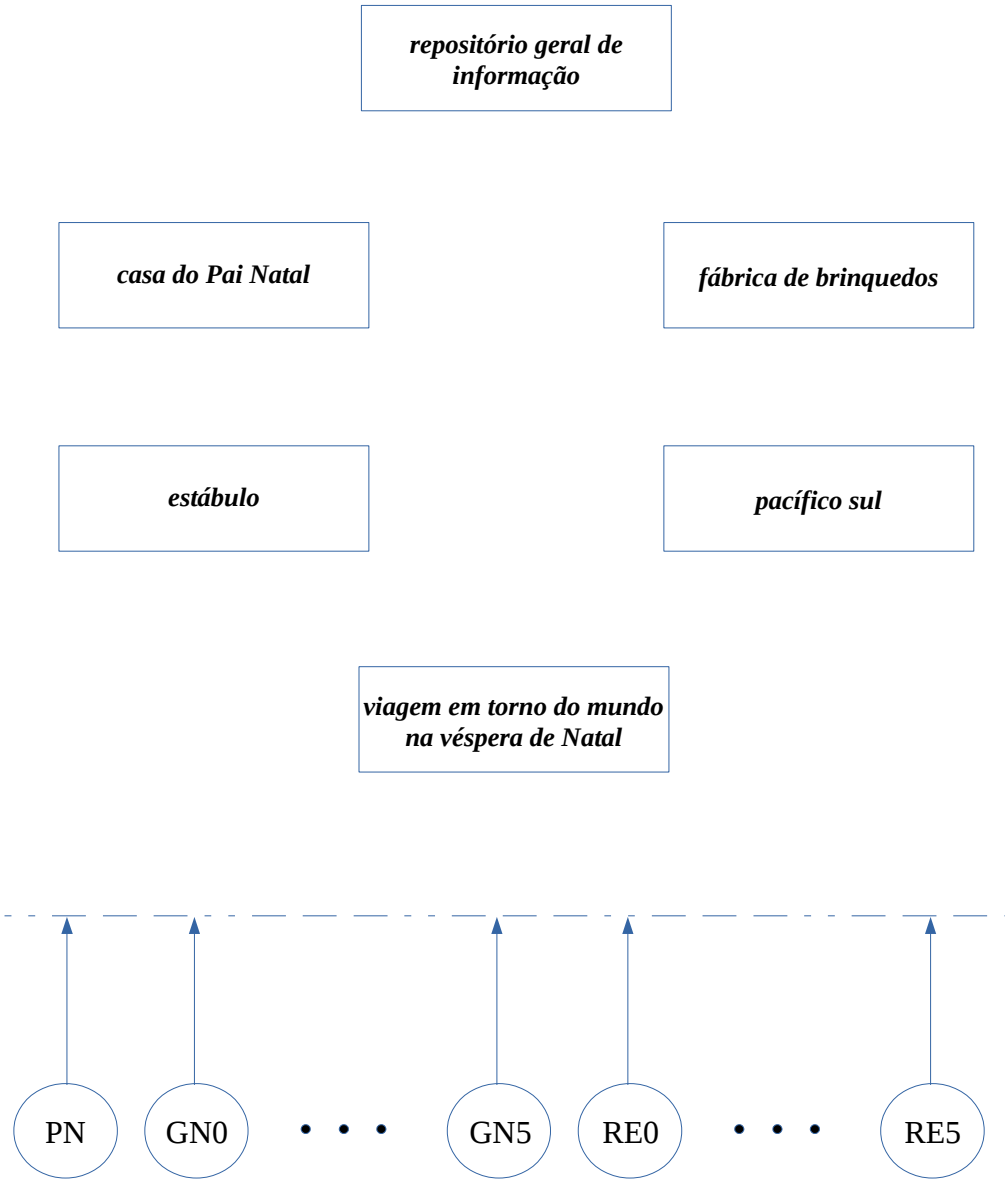
As operações a executar pelos servidores são exactamente as mesmas das da versão distribuída já implementada no Trabalho 2. A única diferença é que, enquanto no trabalho 2 o conteúdo das mensagens era baseado em tipos de dados expressamente criados para esse efeito, agora as mensagens devem usar o tipo String e o conteúdo da string deve ser XML.

Incorpore um ficheiro de *logging* que descreva de um modo conciso, mas claro, a evolução do estado interno das diversas entidades envolvidas.

---

* Adaptação de um problema descrito em Stallings, *Operating Systems*, Prentice-Hall International Inc., 4.ª Ed.

Servidores


repositório geral de informação
casa do Pai Natal
fábrica de brinquedos
estábulo
pacífico sul
viagem em torno do mundo na véspera de Natal
PN
GN0
GN5
RE0
RE5

Etapas para construção da solução

1. Especificar para cada servidor representativo de uma *região de partilha de informação* o formato das mensagens XML trocadas.
2. Proceder à sua codificação em Java dos métodos que criam as Strings XML e que extraem a informação das Strings XML (neste último caso deve ser usado o modelo DOM).

   Dica: Para criar a String XML poderão adicionar um método .toXML(), que retorna uma String, ao(s) tipo(s) de dado(s) de mensagem(ns) do Trabalho 2. Para extrair informação do XML poderão criar construtor(es) do(s) tipo(s) de dado(s) de mensagem(ns) do Trabalho 2 que aceite(m) como parâmetro uma String XML.
3. Alterar os *Stubs* e *Interfaces/Skeletons* das regiões partilhadas para usar Strings XML no conteúdo das mensagens.
4. Construir os *diagramas de interacção* que descrevem de um modo compacto, mas preciso, a dinâmica da aplicação.
5. Validar a solução efectuando execuções múltiplas e verificando para cada uma a correcção dos resultados obtidos por análise do conteúdo do ficheiro de *logging*.