Computação Distribuída

Licenciatura em Engenharia Informática

universidade de aveiro

Trabalho prático nº1
Cenas da Vida do Pai Natal

Grupo:

Catarina Vinagre – 67828

Diogo Reis – 64231

17/04/2015

Índice

Contextualização

Este trabalho tem como objetivo a simulação do ciclo de vida do Pai Natal usando os modelos de sincronização e de comunicação entre processos, dos quais usámos semáforos.

Esta simulação contém as seguintes classes:
- Main
- Santa
- Elf
- Reindeer
- SantaHouse
- Factory
- Stable
- TripWithSanta
- Logger
- States

Antes da construção do código, tivemos em atenção as situações de bloqueio nas classes Santa, Elf e Reindeer.
No caso da classe Santa vai bloquear nas funções rest(), harnessReindeers() e inviteIn(). Também bloquei a si próprio na função travelAround().
Em relação à classe Elf, vai bloquear nas funções needAdvice() e talk().
E em relação à classe Reindeer vai bloquear nas funções goBackStable(), groupAtTheSledge() e followSantaDirections().

Diagramas de Interação

Diagrama visto da main

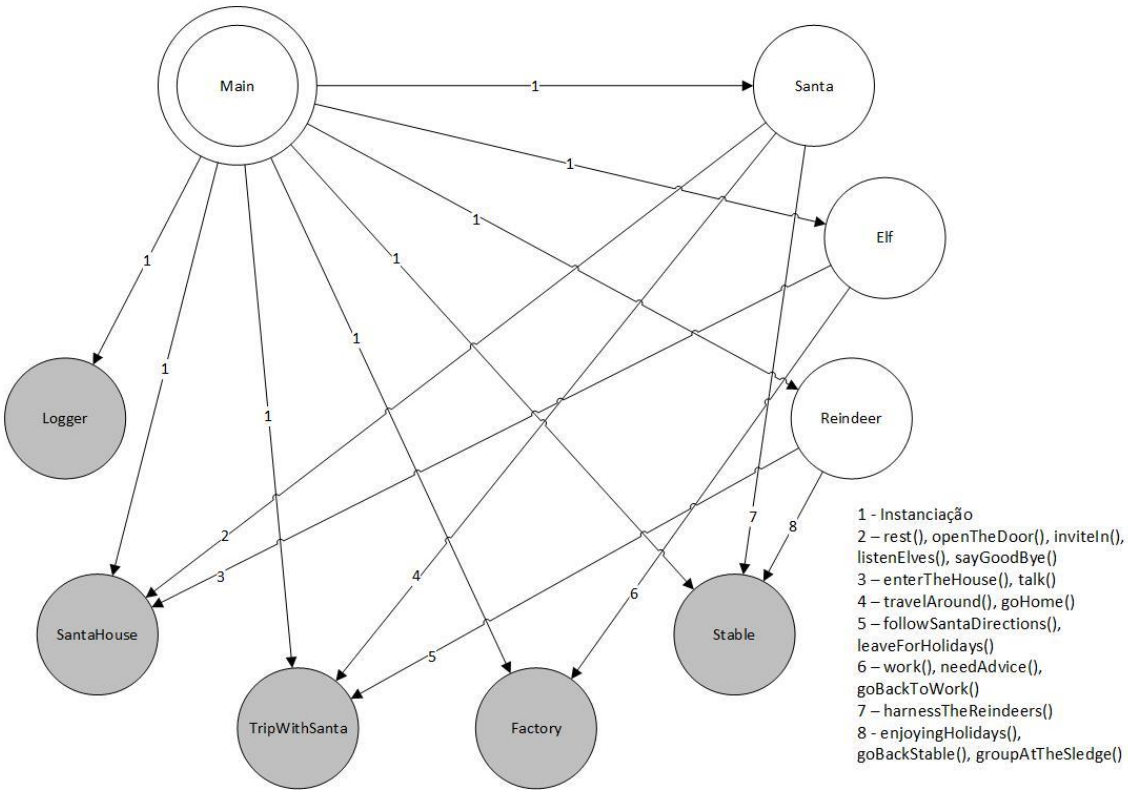


Diagrama Visto do Logger

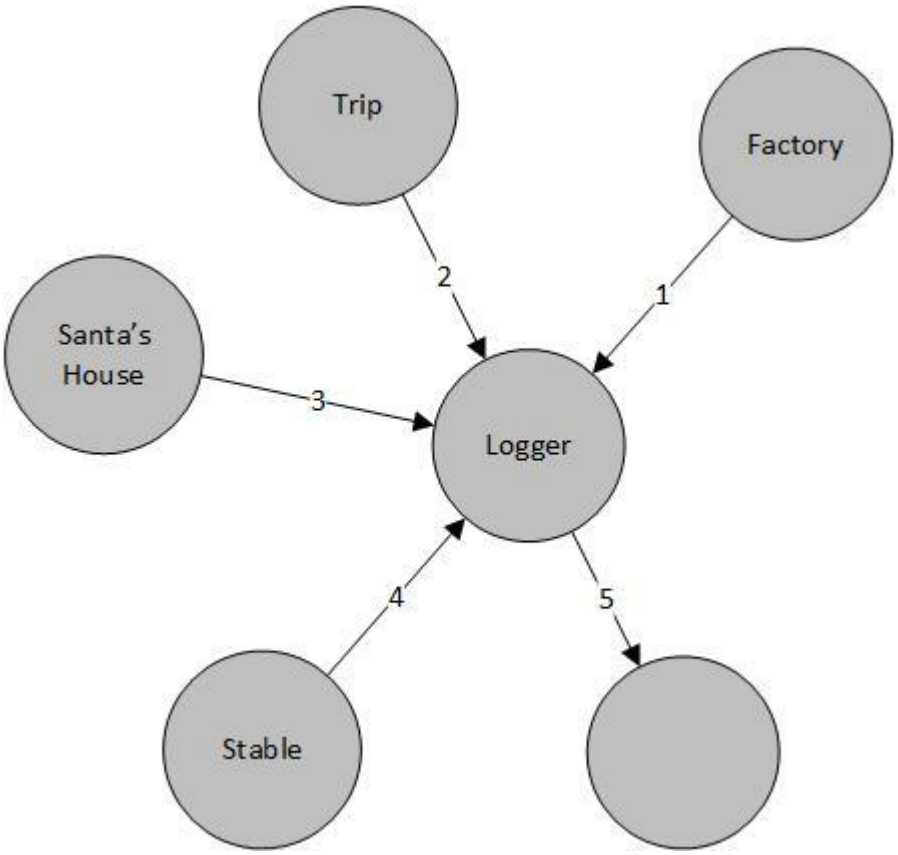
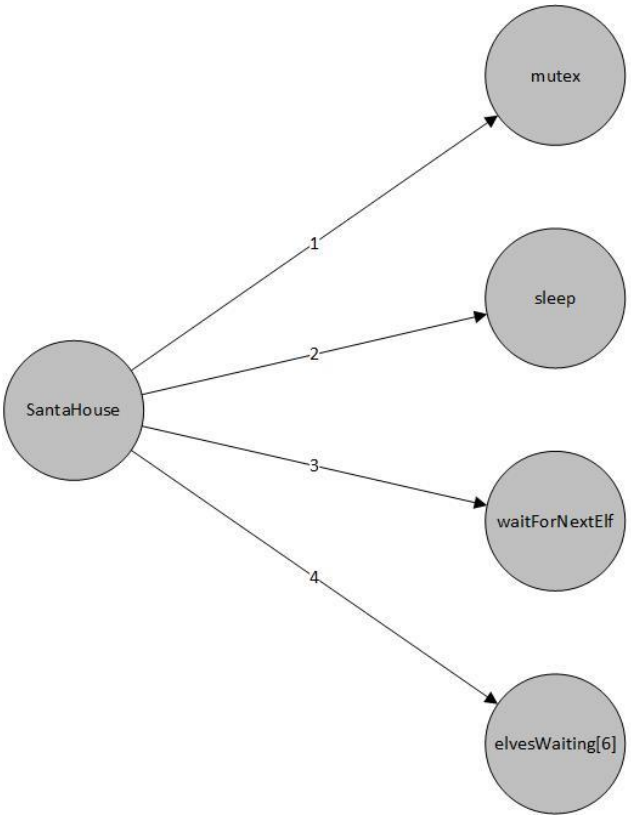
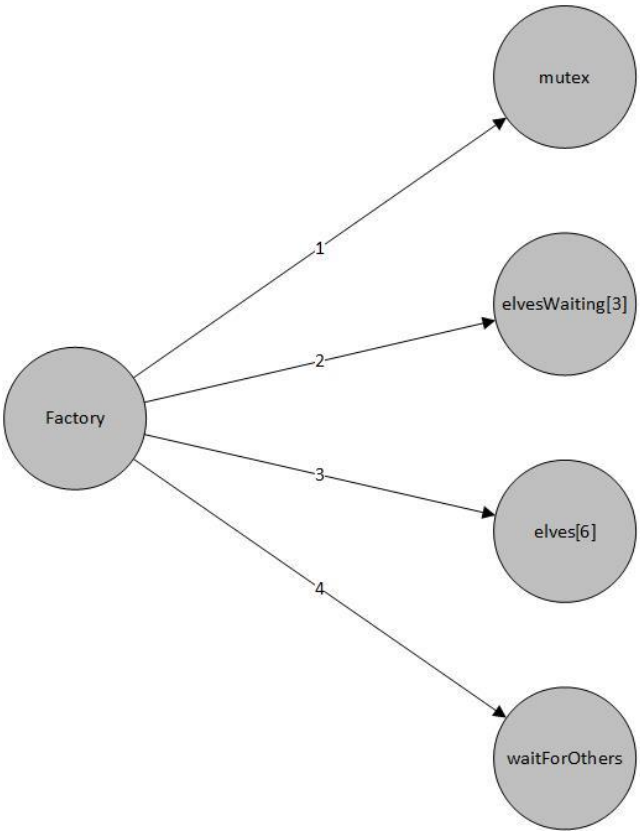


Diagrama Visto do SantaHouse



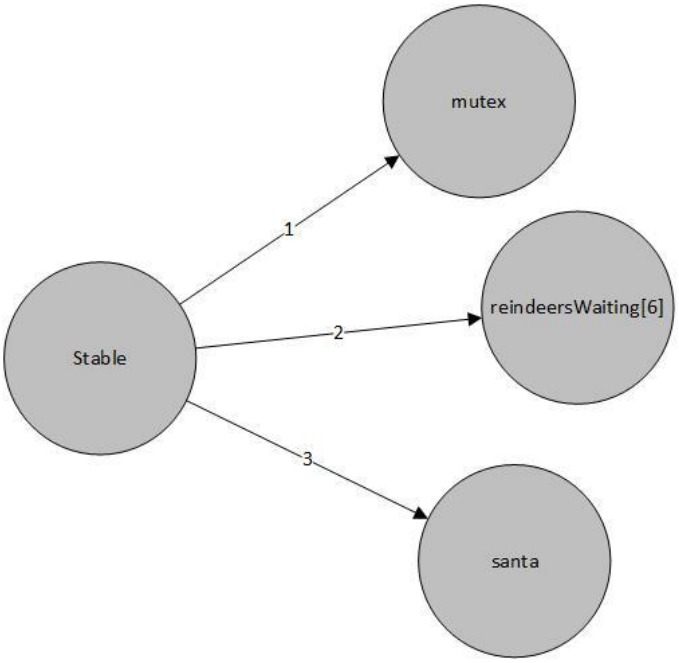
1 - rest(), openTheDoor(), inviteIn(), sayGoodBye(), enterTheHouse()
2 - rest(), enterTheHouse()
3 - inviteIn(), Talk()
4 - inviteIn(), sayGoodBye(), enterTheHouse(), talk()

Diagrama Visto do Factory



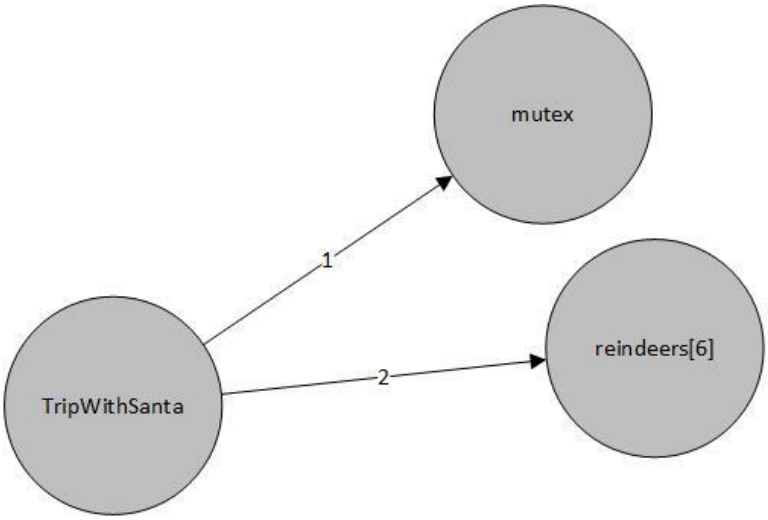
1 - needAdvice()
2 - goBackToWork()
3 - needAdvice()
4 - needAdvice(), goBackToWork()

Diagrama Visto do Stable



1 - goBackStable(), groupAtTheSledge(), leaveForHolidays()
2 - goBackStable(), groupAtTheSledge(), harnessTheReindeers()
3 - groupAtTheSledge(), harnessTheReindeers()

Diagrama Visto do TripWithSanta



1 - followSantaDirections()
2 - followSantaDirections()

Descrição do código

Zonas de memória partilhada:

Fabrica:

```
public void needAdvice(int id) {
    try {
        elvesWaiting.acquire();
        mutex.acquire();
        log.writeElfState(id, States.WISH);
        cont++;
        log.nElvesInGroup(1);
        log.queueAdd(id);

        if (cont == 3) {
            mutex.release();
            //gnoIds.remove();
            for (int i = 0; i < cont-1; i++) {
                elves[gnoIds.remove()].release();
            }
        } else {
            gnoIds.add(id);
            mutex.release();
            elves[id].acquire();
        }

    } catch (InterruptedException ex) {
    }
}
```

“elvesWaiting.acquire();” – zona de memoria partilhada, este semáforo é inicializado com 3 permits, isto vai fazer com que os gnomos quando acabam de trabalhar e passam para o método “needAdvice” passam 3 e o 4º (e próximos) ficam bloqueados até terem permissão para continuar.

“mutex.acquire();” – zona de memoria partilhada, este semáforo é inicializado com 1 permit, assim sempre que se dá um acquire() apenas uma entidade consegue aceder aos dados partilhado.

“elves[id].acquire();” – este semáforo vai fazer com que o 1º e o 2º gnomo fiquem bloqueados até a chegada do 3º, este entra na condição e desbloqueia os colegas com o “elves[gnoIds.remove()].release();”.