

Trabalho de implementação 1

Tema: Simulação orientada a eventos  
Data de entrega: 23 de outubro de 2008  
Grupos: 2 (mínimo) a 4 (máximo)

### Introdução

Simulação é um processo computacional que permite reproduzir o comportamento de um sistema real – construído ou em projeto – para avaliar suas funcionalidades e/ou seu desempenho no transcorrer de sua operação. Existem diversas estratégias de simulação, entre elas a simulação orientada a eventos (*event-driven*). A simulação orientada a eventos é um método essencialmente discreto, ou seja, o tempo não evolui de forma contínua: um relógio global indica o momento em que o evento em tratamento foi disparado.

Neste contexto, um evento descreve uma situação que deve ser tratada em um determinado instante de tempo. Assim, um evento é composto por, pelo menos, três informações: **data**, **tipo** e **duração**. A data se refere ao tempo relativo de simulação no qual o evento deve ser disparado. O tipo informa qual situação do modelo simulado que o referido evento se refere – note que tipo corresponde a descrição da *atividade* que deve ser simulada. A duração indica a duração deste evento/atividade. Note que tanto a data como a duração correspondem a informações de tempo relativas ao relógio de simulação. Datas e duração podem ser, por exemplo, representadas por unidades de tempo (u.t.). Desta forma, duração pode ser uma informação discreta (tal 10 u.t.), e data pode evoluir de forma discreta (como  $\text{horaAtual} = \text{horaAtual} + \text{evento.duracao}$ ).

Note que os tempos de duração dos eventos em uma simulação depende, fortemente, do tipo de atividade que o evento está representado. Em geral, não é informado um tempo fixo de duração para cada tipo de evento, mas sim uma “faixa de tempo” necessária para tratamento. Assim, costuma-se informar o tempo médio de duração da atividade, bem como o desvio padrão a esta média, que é aplicado para mais ou para menos.

Além do **relógio global**, outra estrutura de dados é fundamental na simulação orientada a eventos: uma **lista de eventos**. Nesta lista são armazenados os eventos a serem tratados. Importante notar que esta lista deve ser mantida ordenada, de sorte que o próximo evento a ser executado (aquele com menor data) esteja no início da lista. Assim, o evento de maior prioridade deve ser aquele cujo tempo de simulação é o mais próximo do relógio global.

Importante notar que, como em sistemas reais, na simulação, a ocorrência de um evento pode gerar novos eventos que devem vir a ocorrer em tempos futuros de simulação. Esta situação é, na verdade, bastante comum e é a principal responsável por fazer com que a simulação evolua. Por exemplo, simulando a operação de uma central de *call center*. Considere que esta central seja composta por  $n$  atendentes. Para simulação desta central, podem ser considerada a existência dos seguintes eventos (considere  $rg$  o relógio global indicando a data atual da simulação e  $d$  um valor gerado randomicamente):

- **Recebimento de Chamada**
  - Tratamento: Se atendente livre, cria novo evento **Atendimento** na data  $rg + d$ . Se atendente não livre, descarta ligação (como se o cliente ouvisse uma mensagem: “todas nossas linhas estão ocupadas, ligue mais tarde”) e cria novo evento **Recebimento de Chamada** na data  $rg + d$ .
- **Atendimento**
  - Tratamento: Decrementa número de atendentes livres, cria novo evento **Fim de Atendimento** na data  $rg + d$ .
- **Fim de Atendimento**
  - Tratamento: Incrementa número de atendentes livre, cria novo evento **Recebimento de Chamada** na data  $rg + d$ .

Note que diversas situações não estão sendo consideradas nesta simulação, como a saída de um atendente para o almoço e seu retorno. Quanto mais detalhados os eventos, mais próxima a simulação é da realidade.

A evolução da simulação se dá através do tratamento da sucessão de eventos. O motor de execução de um simulador orientado a eventos pode ser exemplificado como segue.

```
Evento e; // Evento em curso
ListaEventos le; // Lista de eventos ordenada por tempo
RelogioGlobal rg = 0; // Relógio global inicializado com 0
int atendentes = 10; // call center tem 10 atendentes

e.tipo = RecebimentoChamada;
e.data = rg + 0;
e.duracao = 1; // Tempo que necessita para passar ao atendimento (caso haja atendente livre)
le.insere( e );
e.tipo = FimSimulacao;
e.data = 100; // Simula 100 u.t.
e.duracao = 0; // duração não relevante neste caso
le.insere( e );

for( e = le.getProximo() ; e.tipo != FimSimulacao ; e = le.getProximo() ) {
    switch(e.tipo) {
        case RecebimentoChamada: se (atendentes > 0) ... else ...
                                ContabilizaEstatistica();
                                break;
        case ...
    }
}
ApresentaEstatistica();
```

Como dito anteriormente, o objetivo da simulação é avaliar o comportamento de um sistema. No caso do exemplo da central telefônica, poderia ser avaliada a satisfação do cliente, contabilizando o número de ligações atendidas/perdidas em função do número de atendentes ou da duração das chamadas. Esta situação está ilustrada no exemplo do motor de simulação.

### Definição

Deve ser implementado um simulador orientado a eventos para um sistema real. Cabe ao grupo definir o sistema a ser simulado, bem como os dados estatísticos a serem oferecidos como saída. Como entrada da simulação, o usuário deve ser capaz de configurar dados da simulação.

### Avaliação

O trabalho será avaliado quanto a documentação entregue e qualidade do software desenvolvido. Em particular será considerado o uso de recursos da programação orientada a objetos (encapsulamento, herança, especialização).

### Material a ser entregue

- Documento contendo uma **descrição do sistema real**. Esta descrição deve conter, pelo menos:
  - Caracterização do sistema (descrição textual do modelo real).
  - Objetivos da simulação (o que será avaliado).
  - Modelo da simulação (atividades que estão sendo considerado na simulação).
  - Descrição dos parâmetros de entrada para execução
  - Descrição das saídas obtidas.
- Manual do usuário, informando como o simulador deve ser utilizado e um exemplo de execução.
- Manual de implementação, apresentando a hierarquia de classes (esquema UML será apreciado), bibliotecas e serviços da API Java utilizados, bem como explicação das estratégias aplicadas.
- Programa fonte em Java e seus respectivos bytecodes em meio magnético para cópia pelo professor. Os programas serão apresentados em laboratório, certifique-se que o programa encontra-se operacional.