

Trabalho sobre o Paradigma de Programação Orientado a Objetos

Tema: Simulação orientada a eventos

Data de entrega: 2/outubro

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 o evento se refere. A duração indica a duração deste evento. Note que a data é relativa ao relógio de simulação, ela pode ser, por exemplo, representada por unidades de tempo (u.t.), evoluindo de forma discreta em uma variável inteira.

Além do relógio global, outra estrutura de dados é fundamental na simulação orientada a eventos: uma lista armazenando os eventos a serem tratados. Importante notar que esta lista deve ser mantida ordenada considerando o tempo de simulação. 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.

Por exemplo, simulando a operação de uma central de *call center* para atendimento a clientes composta por n atendentes, podemos considerar 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 uma central telefônica para atendimento a clientes. Como resultado da simulação, deve ser informado o número de ligações atendidas e perdidas durante um determinado período de tempo. O usuário do simulador deve poder informar o número de atendentes e o tempo médio de cada chamada (a duração efetiva deve ser gerada randomicamente).

Material a ser entregue

- Programa fonte em Java e seus respectivos bytecodes em meio magnético.
- Manual do usuário, informando como o simulador deve ser utilizado e um exemplo de execução.
- Manual de implementação, apresentando a hierarquia de classes, bibliotecas e serviços da API Java utilizados, bem como explicação das estratégias aplicadas.