PMR2300 - Computação para Automação

Exercício Programa 2 - 2014

Filas: Um Simulador para Sistemas de Eventos Discretos

Prof. Dr. Fabio Gagliardi Cozman
Prof. Dr. Newton Maruyama

1 Introdução

Nesse exercício programa estruturas de dados do tipo Filas com implmentação dinâmica serão utilizadas para a implementação de um simulador de eventos discretos.

2 Sistemas de Eventos Discretos

Qualquer processo que desejamos simular pode ser considerado como um sistema. Vamos aqui definir sistema como sendo um grupo de objetos interagindo no sentido de se produzir algum resultado. Por exemplo, uma fábrica é um grupo de pessoas e máquinas trabalhando juntos para a produção de algum produto. Um sistema pode ser discreto ou contínuo, dependendo da natureza de suas variáveis. Um sistema contínuo possui variáveis que podem assumir qualquer valor real em um dado intervalo (por exemplo, medidas de temperatura e de pressão). Um sistema discreto possui variáveis que podem assumir apenas um número limitado de valores ou estados (por exemplo, uma máquina pode assumir os estados ligado ou desligado).

Um sistema também pode ser determinístico ou estocástico dependendo da relação entre as suas entradas e saídas. Num sistema determínistico, é possível determinar o estado final do sistema a partir do estado inicial. Muitos sistemas contêm tanto variáveis determinísticas como variáveis estocásticas.

Para que um sistema possa ser simulado, um modelo do sistema deve ser criado. Um modelo é um conjunto de informações que representam o sistema de uma maneira simplificada, evidenciando apenas os aspectos que se deseja estudar. Para se determinar a estrutura do modelo devemos definir as entidades, os atributos, e as atividades do sistema.

As entidades representam os componentes e a força de trabalho do sistema. Os atributos se referem às características das entidades. O estado do sistema em qualquer instante é especificado pelos atributos das entidades do sistema e pelas relações entre as entidades naquele instante. Uma atividade é um processo que causa uma mudança no estado do sistema. Um evento é a ocorrência de uma atividade num determinado instante de tempo.

Finalmente, podemos realizar a simulação dirigida pelo tempo (*Time-driven simulation*) ou dirigida por eventos (*Event-driven simulation*). Numa simulação dirigida pelo tempo um relógio principal é utilizado para contar o tempo de simulação. O relógio é incrementado com uma unidade de tempo. A cada incremento de

tempo o programa deve examinar cada evento para verificar se o evento deve ocorrer naquele instante. Se o evento deve ser processado a atividade associada começa a ser realizada.

No caso da simulação ser dirigida por eventos, o programa deve examinar todos os eventos para descobrir qual será o próximo evento que vai ocorrer. Para tal, é necessário que cada evento tenha o seu próprio contador de tempo. Após a atualização do relógio principal, a atividade associada ao evento é realizada e o relógio do evento é atualizado para a próxima ocorrência do evento.

3 Teoria de Filas

Um dos modelos de sistemas de eventos discretos comumente utilizados são Sistemas de Filas (*Queueing System*, *Waiting Line*). Para tal, um corpo teórico, denominado Teoria de Filas, foi desenvolvido [1,2].

A teoria de filas é utilizada em áreas diversas como: sistemas de atendimento a clientes (agências bancárias, supermercados, etc.) e sistemas de manufatura. Veja Figura 1.

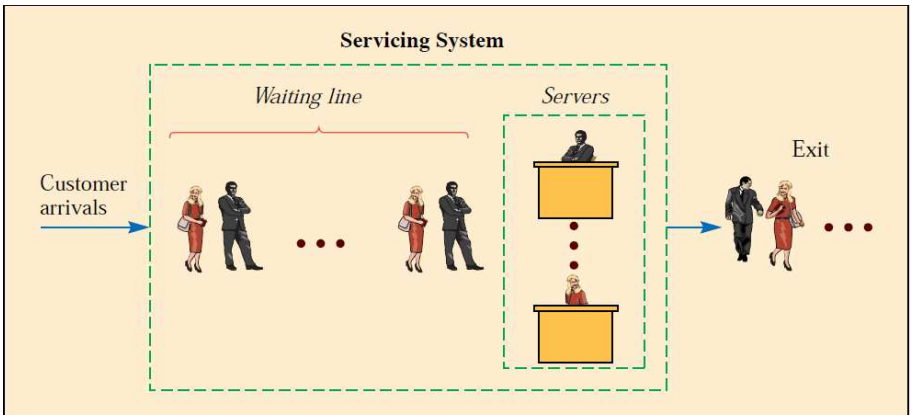

Figura 1: Sistemas de fila de atendimento a clientes.

O sistemas de filas são classificados em diversos tipos de acordo com sua arquitetura (Veja Figura 2):

- Servidor único, fase única (*Single-server*, *single-phase*)
- Servidor único, múltiplas fases (*Single-server*, *multiphase*)
- Servidor múltiplo, fila única, fase única (*Mutiserver*, *single-line*, *single-phase*)
- Servidor múltiplo, múltiplas filas, fase única (*Multiserver*, *multiline*, *single-phase*)
- Servidor múltiplo, múltiplas fases (*Multiserver*, *multiphase*)

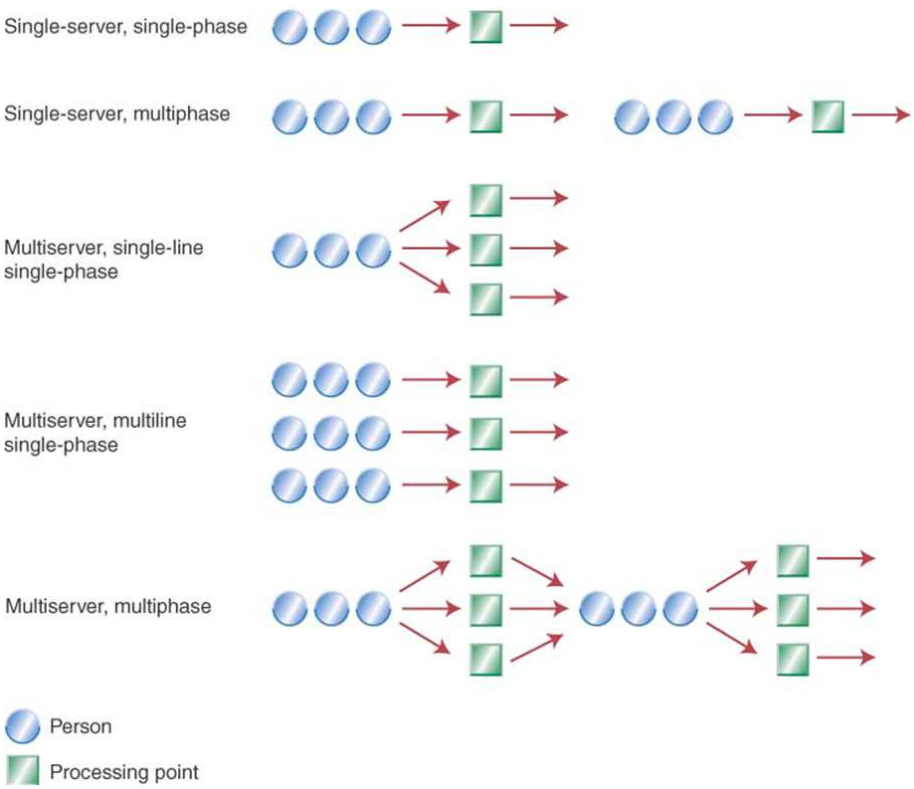

Figura 2: Diferentes tipos de sistemas de filas. Retirado de [2]

Para muitos sistemas é possível estabelecer cálculos analíticos que permitem estimar grandezas como o tempo médio que um cliente permanece na fila, qual a probabilidade de um cliente permanecer na fila mais que 1 hora, etc. Entretanto, muitas vezes os sistemas possuem alto nível de complexidade e as análises do sistema são conduzidas através de simulações. Exemplos de software dessa categoria seriam por exemplo, o ProModel da ProModel Corporation [3] e o Arena da Rockwell Automation [4].

Filas de eventos usualmente obedecem a Processos de Poisson. A chegada de clientes numa agência bancária por exemplo pode ser expressa através de uma distribuição de Poisson. A distribuição de Poisson descreve a probabilidade da chegada de n clientes durante um período de tempo T e é expressa por:

$$P_T(n) = \frac{(\lambda T)^n \exp^{-\lambda T}}{n!} \tag{1}$$

onde λ é a média da taxa de chegada de clientes no tempo T.

Por exemplo, se a taxa média de chegada de clientes é de 3 clientes por minuto ($\lambda = 3$) e deseja-se saber qual a probabilidade da chegada de 5 clientes por minuto ($n = 5, T = 1$), utilizando a equação acima obtemos:

$$P_1(5) = 0.101 \tag{2}$$

A Figura 3 ilustra uma distribuição de Poisson para $\lambda = 3$. Tanto a média como a variância da distribuição de Poisson são iguais a λ.

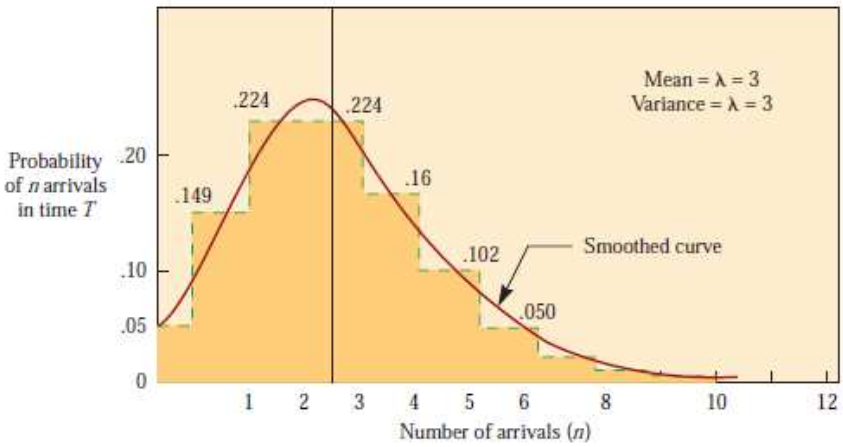

Figura 3: Distribuição de Poisson para $\lambda = 3$. Retirado de [2]

Durante o processo de simulação necessitamos da distribuição de probabilidade do intervalo de tempo entre eventos, por exemplo, se um cliente chega agora qual o intervalo de tempo que demorará para o próximo cliente chegar.

Se a chegada de clientes segue um processo de Poisson então a função densidade de probabilidade exponencial modela a distribuição de intervalos de tempo entre eventos:

$$f(t) = \lambda \exp^{-\lambda t}, \tag{3}$$

onde λ é o número médio de chegadas por unidade de tempo.

Obviamente, λ da Equação 3 é o mesmo da Equação 1. A média da distribuição exponencial é $1/\lambda$ e sua variância $1/\lambda^2$. Se existe uma média de λ chegadas por unidade de tempo é natural imaginar que na média uma chegada demora $1/\lambda$.

Obviamente o mesmo raciocínio vale para o modelo para a função densidade de probabilidade do intervalo de tempo que o servidor demora para finalizar uma tarefa (Taxa de serviço). Por exemplo, quanto tempo o caixa demora para atender um cliente. Podemos utilizar também uma distribuição de Poisson. A Figura 4 ilustra uma função densidade de probabilidade exponencial.

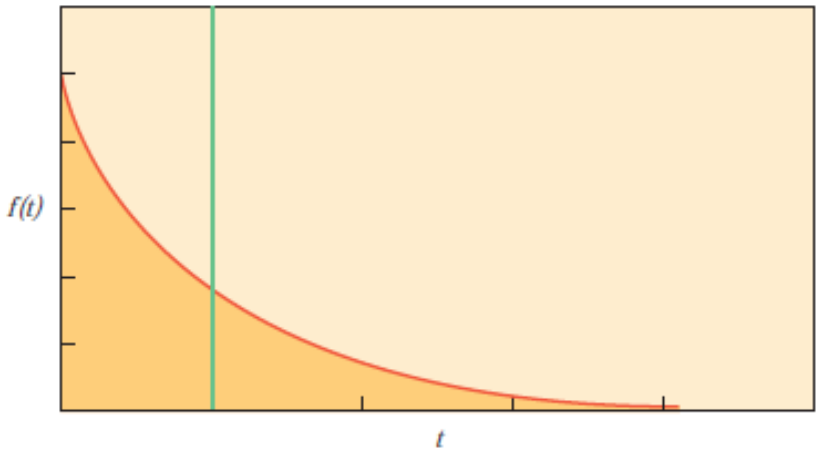

Figura 4: função densidade de probabilidade exponencial para $\lambda = 3$. Retirado de [2]

4 Algoritmo para Simulação

Uma maneira genérica para simular sistemas de eventos discretos consiste em se criar uma Fila de Eventos (FE). Tal fila contêm elementos cujo conteúdo é a data de início de um determinado evento.

A cada novo elemento inserido deve-se ordenar a Fila de Eventos na ordem cronológica crescente (O primeiro evento a ser retirado da fila é o que acontecerá mais proximamente). Ou seja, o comportamento dessa fila não segue a disciplina FIFO (primeiro a entrar primeiro a sair). Na verdade trata-se uma fila com prioridades onde a prioridade é o instante de tempo em que o evento irá ocorrer.

Num sistema de atendimento a clientes os seguintes eventos são relevantes:

- Evento 1: cliente chega ao sistema e escolhe uma fila,
- Evento 2: começa o atendimento do servidor,
- Evento 3: termina o atendimento do servidor.

Os instantes de tempo em que ocorrem os Eventos 1 e 3 dependem da amostragem da função densidade de probabilidade Exponencial. O Evento 1 depende da taxa de chegada de clientes λ enquanto que o Evento 3 depende da taxa de atendimento μ do servidor. O Evento 2 ocorre quando o servidor fica livre e existem clientes na Fila de Clientes.

Um possível algoritmo para a simulação de um sistema de eventos discretos do tipo sistema de atendimento a clientes é apresentado no Algoritmo 1. Trata-se de um algoritmo de simulação para um sistema de atendimento a clientes de agências bancárias. O sistema possui cinco filas e cinco servidores (Múltiplos servidores, Múltiplas Filas, Fase Única).

Inicialmente, o instante de chegada de um cliente na agência é amostrado, gerando um Evento 1, que então é colocado na Fila de Eventos. Este primeiro evento será o desencadeador de todos os outros eventos da simulação.

O ciclo de simulação começa verificando se existem caixas livres e se para os caixas livres existem clientes na Fila de Clientes. Se a resposta for afirmativa um Evento 2 é gerado e colocado na Fila de Eventos para cada um dos clientes que passará a ser atendido.

Posteriormente, é verificado qual o primeiro evento da Fila de Eventos. O relógio de simulação, `clock` é atualizado para o instante de tempo do evento.

São três as situações possíveis:

- Se for um Evento 1, este evento é colcado na Fila de Clientes com o menor tamanho. Neste instante o Evento 1 fica associado a uma fila e caixa específicos. Um novo Evento 1 é gerado e colocado na Fila de Eventos.
- Se for um Evento 2, um Evento 3 correspondente é gerado amostrando-se função densidade de probabilidade correspondente ao intervalo de tempo de duração do atendimento. O Evento 3 é colocado na Fila de Eventos.
- Se for um Evento 3, identifica qual o caixa associado e marca este como livre.

Ao final de cada passo da simulação é necessário atualizar as estimativas das grandezas estatísticas que estão sendo consideradas.

Como exemplo de um instante específico da simulação de um sistema, com duas Filas de clientes e dois caixas, apresentamos a Figura 4 contendo o estado da Fila de Eventos e a Figura 4 contendo o estado das Filas de Clientes.

Os eventos são identificados da seguinte forma:

- Ex:tttt:i,
- onde Ex: E1, E2, E3 indicam a identidade do evento,
- tttt indica o instante de tempo do evento em minutos,
- e i o caixa e fila associados.

Note que neste instante de tempo `clock=100`. Este instante de tempo coincide com o instante de tempo do Evento 2 que está em atendimento no caixa 1. Na Fila de Eventos podemos observar o Evento 3 correspondente que já foi gerado. Também podemos observar na Fila de Eventos um Evento 3 (E3:101:2) associado ao Evento 2 (E2:98:2) referente ao caixa 2.

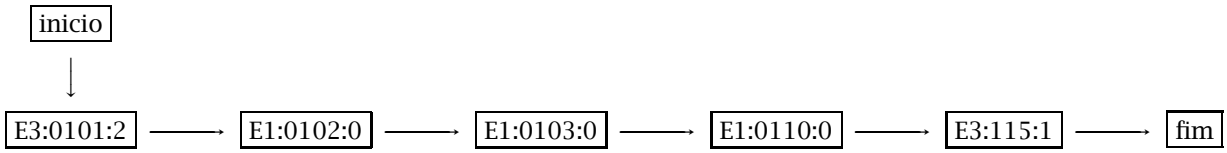

Figura 5: Fila de Eventos, $clock = 0100$.

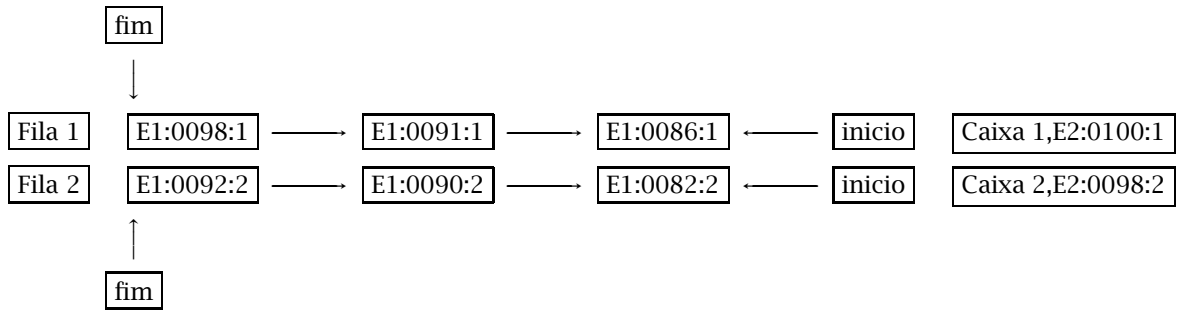

Figura 6: Fila de Clientes, $clock = 0100$.

O algoritmo apresentado simula um sistema Múltiplos Servidores, Múltiplas Filas, Fase Única mas pode facilmente ser adaptado para Múltiplos Servidores, Fila Única, Fase Única.

Algorithm 1 Algoritmo Principal

```
1: procedure Main                                   ▷ Algoritmo para simulação de sistemas de filas
2:     clock ← 0.0;                                                          ▷ Relógio da simulação
3:     Δt_evento1 ← amostrar função;        ▷ Inserir na FE a chegada de um primeiro cliente - Evento_1
4:     t_evento1 ← Δt_evento1;                                          ▷ Nesse ponto clock = 0.0
5:     inserir Evento_1(t_evento1) na FE;
6:     marcar todos os caixas (i = 1,...,5) como livres;
7:     repeat
8:         for caixas i = 1,...,5; do
9:             if caixa i livre then
10:                if existem clientes na fila i then
11:                    retirar o cliente da fila i;
12:                    t_evento2 ← clock          ▷ O atendimento começa agora - gerar um Evento_2
13:                    inserir Evento_2(t_evento2, i);
14:                    marcar o caixa i como ocupado;
15:                end if
16:            end if
17:        end for
18:        retirar um Evento da FE;
19:        atualizar clock;                ▷ clock é atualizado para o tempo contido no Evento
20:        switch Evento do
21:            case Evento_1
22:                i ← escolher a fila de menor comprimento;                          ▷ i indica a fila
23:                inserir Evento_1(t_evento1, i) na FC[i];                ▷ Cliente é colocado numa fila
24:                Δt_evento1 ← amostrar função;          ▷ Inserir um novo cliente na agência bancária
25:                t_evento1 ← clock + Δt_evento1;
26:                inserir Evento_1(t_evento1) na FE;
27:            case Evento_2
28:                i ← Evento_2(t_evento2, i);                          ▷ Descobrir o caixa associado
29:                Δt_evento3 ← amostrar função;                              ▷ Gerar um Evento_3
30:                t_evento3 ← clock + Δt_evento3;
31:                inserir Evento_3(t_evento3, i) na FE;
32:            case Evento_3
33:                i ← Evento_3(t_evento3, i);                          ▷ Descobrir o caixa associado
34:                Marcar o caixa i como livre;
35:        calcular estatísticas;
36:    until FE vazia e FC[i] (i=1,...,5) vazias e tempo máximo de simulação atingido
37: end procedure
```

5 Cálculo de estatísticas

Através de simulações de sistemas de filas objetiva-se a estimativa de variáveis que caracterizam o desempenho do sistema.

Em sistemas de atendimento ao cliente, por exemplo, procura-se estabelecer sistemas de filas que possam aumentar a eficiência do sistema (taxa de atendimento alta) o que consequentemente promove a satisfação dos clientes (tempo de espera na fila pequeno).

As seguintes variáveis são usualmente calculadas:

1. Número médio de clientes na fila $\overline{n}_{CF}$:

$$\overline{n}_{CF} = \frac{\int_0^{T_{Total}} n_{CF}(t)dt}{T_{Total}}, \tag{4}$$

   onde T_{Total} é o maior tempo entre o tempo máximo especificado pela simulação ou o tempo quando todos os clientes terminam de ser atendidos.

2. Tempo médio que um cliente permanece esperando na fila $\overline{\omega}_{CF}$ e sua respectiva variância σ^2_{CF}:

$$\overline{\omega}_{CF} = \frac{\int_0^{T_{Total}} n_{CF}(t)dt}{N_{Total}}, \tag{5}$$

$$\sigma^2_{CF} = \frac{\int_0^{T_{Total}} [n_{CF}(t) - \overline{n}_{CF}]^2 dt}{N_{Total}}, \tag{6}$$

6 Para você fazer

Suponha que deseja-se construir um sistema de simulação de filas para analisar alternativas de implementação de serviços de atendimento a clientes em agências bancárias.

A chegada dos clientes à(s) fila(s) é caracterizada da seguinte forma:

1. O cliente escolhe sempre a fila de menor tamanho,
2. A fila de clientes pode ter tamanho ilimitado,
3. A taxa de clientes que chegam à agência bancária segue um Processo de Poisson de média $\lambda = 44/h$.

O atendimento é caracterizado da seguinte forma:

1. A taxa de atendimento obedece a um Processo de Poisson de média μ
2. Cada caixa tem um desempenho diferente:

(a) Caixa 1: $\mu_1 = 10/h$,

(b) Caixa 2: $\mu_2 = 9/h$,

(c) Caixa 3: $\mu_3 = 8/h$,

(d) Caixa 4: $\mu_4 = 7/h$,

(e) Caixa 5: $\mu_5 = 6/h$.

Deeja-se analisar duas implementações distintas de sistema de atendimendo de agências bancárias:

1. Múltiplos Servidores, Múltiplas Filas, Fase Única (*Multiserver*, *single-line*, *single-phase*): Simulação com cinco caixas, cinco filas, fase única.
2. Múltiplos Servidores, Fila Única, Fase Única (*Multiserver*, *single line*, *fase única*): Simulação com cinco caixas e fila única.

Deseja-se monitorar as seguintes variáveis:

1. Número médio de clientes na fila.
2. Tempo médio que um cliente permanece esperando na fila e seu respectivo desvio padrão.
3. Maior tempo de permanência de um cliente na fila.

Compare as duas soluções e apresente argumentos para indicar qual a melhor solução sob o ponto de vista do cliente.

6.1 Sugestões de implementação

· Note que você deve implementar dois tipos de fila com comportamentos distintos:

  – a Fila de Eventos (FE) que é uma fila com prioridade (O tempo de ocorrência indica a prioridade do evento),

  – e a Fila de Clientes (FC) com comportamento do tipo FIFO.

  Uma das maneiras de se implementar a Fila de Eventos é a definição de um nó através da classe `Evento` (Veja Listagem 1). A classe Evento possui uma variável `e` do tipo enumerado TipoEvento que indica qual evento está associado ao nó.

  Define-se em seguida a classe que representa a Fila de Eventos denominada *FE* (Veja Listagem 2).

  A Fila de Clientes também pode ser construída através de nós definidos pela classe `Eventos` (Veja Listagem 3).

  Cada nó do tipo `Evento` está associado sempre a um único cliente. Ao final do ciclo completo da cliente na agência bancária o nó deve conter contêm as informações dos três eventos associados: `evento1`, `tevento2` e `tevento3`.

  Por fim, uma possível representação para o sistema de caixas está representado na Listagem 4

Listing 1: Classe Evento.

```
public enum TipoEvento {Evento1,Evento2,Evento3}; // definir no programa principal

public class Evento {
private Evento proximo;        // ponteiro para o proximo elemento da Fila de Eventos
private TipoEvento e;          // Evento1, Evento2, Evento3
private double tevento1, tevento2, tevento3; // instantes de tempo do Evento1,
                                             // Evento2 e Evento3 do mesmo cliente
private int caixa;                           // numero do caixa associado a este cliente

// constructors
...
// accessor
...
```

Listing 2: Classe FE.

```
public class FE {
private Evento inicio;       // ponteiro para o inicio da fila
private int NumeroDeEventos; // numero de eventos na FE

// constructors
...
// accessors
...
```

Listing 3: Classe FC.

```
public class FC {
private Evento inicio;       // ponteiro para o inicio da fila
private Evento fim;          // ponteiro para o fim da fila
private int NumeroDeClientes; // numero de clientes na FE

// constructors
...
// accessors
...
```

Listing 4: Classe SistemaDeCaixas.

```
public class SistemaDeCaixas {
          private boolean CaixasLivres[6];        // Posicoes 1 ate 5 representam
                                                  // os 5 caixas utilizados
          private int NumeroDeCaixasLivres;

// constructors
...
// accessors
...
```

· A natureza estocática do problema implica que cada solução deve ser simulada várias vezes para que as estimativas das grandezas convirjam para o resultado correto.

- Realize 50 simulações de cada solução. Calcule valores médios das grandezas desejadas.
- O tempo total de cada simulação é de 6 horas (Horário bancário das 10h as 16h).
- Ao finalizar o tempo de 6 horas nenhum cliente adicional pode entrar na fila mas todos os clientes que estão na fila devem ter o seu atendimento realizado.
- Tanto para a determinação do intervalo de tempo em que ocorrerá a chegada do próximo cliente quanto para a determinação do tempo de duração do próximo atendimento é necessário amostrar uma função exponencial.
- A biblioteca padrão do Java não possui geração de números aleatórios com distribuição exponencial. Aconselha-se a utilização da classe `ExponentialPower` da Biblioteca COLT desenvolvida no CERN (Sim ! aquele lugar do acelerador de partículas que descobriu o Bóson de Higgs.):
  - Página principal: `https://acs.lbl.gov/software/colt/`
  - Documentação das classes relativas a amostragem de funções densidade de probabilidade: `https://acs.lbl.gov/software/colt/api/cern/jet/random/engine/class-use/RandomEngine.html`
  - Código fonte da implementação: `https://github.com/carlsonp/Colt/blob/master/src/cern/jet/random/ExponentialPower.java`

7 Bibliografia

1. Christos G. Cassandras and Stephane Lafortune, *Introduction to Discrete Event Systems*,Springer, 2nd Edition, 2009.
2. F. Robert Jacobs and Richard B. Chase, *Operations and supply management: the core*, McGraw-Hill, 2012.
3. `http://www.promodel.com`
4. `http://www.arenasimulation`

8 Coisas importantes

- **DATA FINAL DE ENTREGA: Quarta-Feira 28/05/2014 (No horário de monitoria: 12:00h-13:00h).**
- **O exercício deve ser feito individualmente.**
- **Entregue ao monitor da disciplina**
  - **Documentação impressa descrevendo o seu projeto em uma página A4,**
  - **Código fonte, programa executável (bytecode) e documentação em um arquivo compactado.**