Estudo de Loja de Comunicações

SCC

João Montenegro nº

João Mendes nº 2016230975

# Arquitetura do Simulador

O objetivo deste simulador é criar uma simulação verídica do atendimento numa loja de comunicações, em que existem 2 filas, uma geral e uma empresarial.

Para que isto aconteça, foi necessário criar os 2 serviços existentes, para isto, colocou-se na classe serviço, um booleano, em que, se fosse *true*, simbolizava o atendimento geral e se fosse *false* o atendimento empresarial.

Como existem 2 tipos de serviços, também existem dois tipos de clientes, distinguidos também, por um booleano, tal e qual o serviço, este normalmente iriam ser atendidos no seu serviço inicial, no entanto, em casos específicos, isto pode não acontecer.

Em termos de eventos, existem 2, a chegada e a saída dos clientes, estes, usam a aleatoriedade (exponencial ou normal) para que o cliente chegue ou saia a um tempo especifico.

Já a aleatoriedade, se for de caso normal, apenas se usa a média do tempo de chegada de cada cliente, no entanto, se for exponencial, já se usa a media, o desvio padrão e uma stream, para o fator aleatório.

# Descrição da abordagem

Para a realização deste trabalho, dividimos o trabalho em 3 partes.

A primeira foi colocar, ambos os serviços em vez de um único, para isto, como foi dito anteriormente, foi colocado um booleano no serviço e no cliente, isto para se conseguir distinguir os tipos, também foi necessário colocar duas chegadas, na lista de eventos, para simular a chegada de clientes.

A segunda parte foi dedicada à aleatoriedade, ou seja, criar na classe “Aleatorio”, uma função, que ao receber uma média, um desvio padrão, e uma stream, devolva 2 números, em que um é usado de seguida enquanto que outro é guardado num array previamente implementado, para que possa ser usado caso o número devolvido pela stream se repita.

A parte final e a que exigiu mais trabalho, foi implementar os casos específicos, em que os clientes possam sair da sua fila, para isto, decidimos executar estas mudanças na classe serviço, usando uma função presente na classe simulador para que devolve o serviço empresarial e uma variável do tipo serviço presente na classe, que devolve sempre o serviço contrario a este. No caso em que é preciso uma interrupção da fila, foi necessário cancelar o evento de saída do cliente, para isto, cada cliente possui uma variável, que guarda o evento, para que seja possível retirar o evento referente a esse cliente.

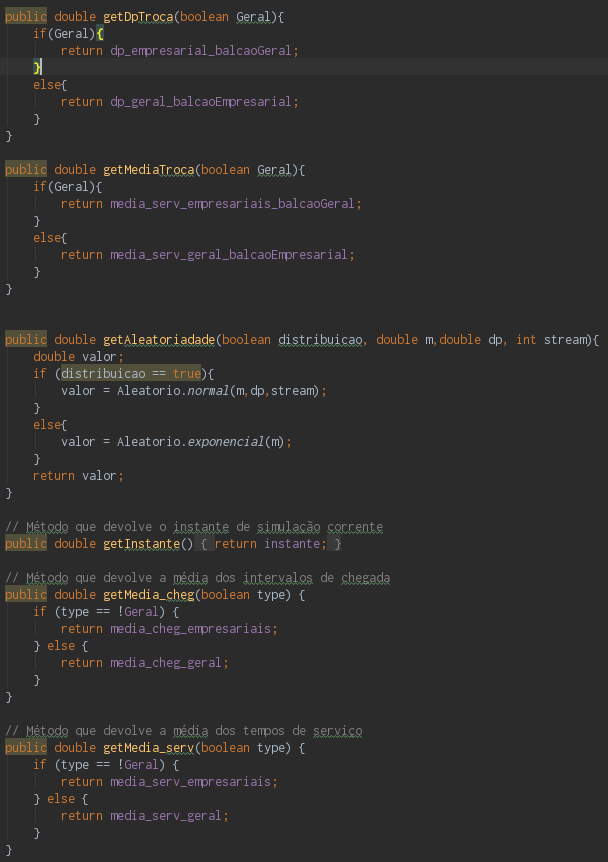
# Alterações do código

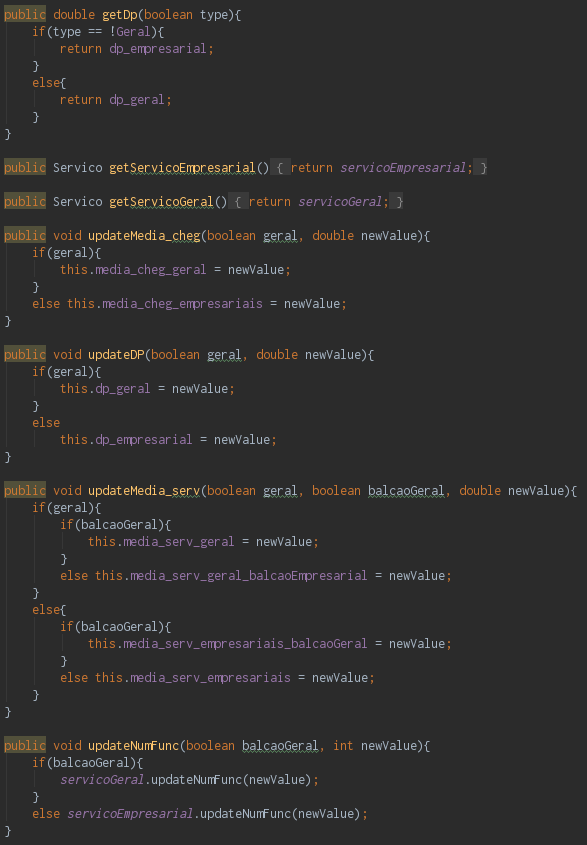
Para representar a situação descrita no enunciado foi necessário alterar as classes *Simulador*, *Serviço*, *Aleatorio*, *Cliente* e *ListaEventos*, estas duas ultimas sendo as menos modificadas.

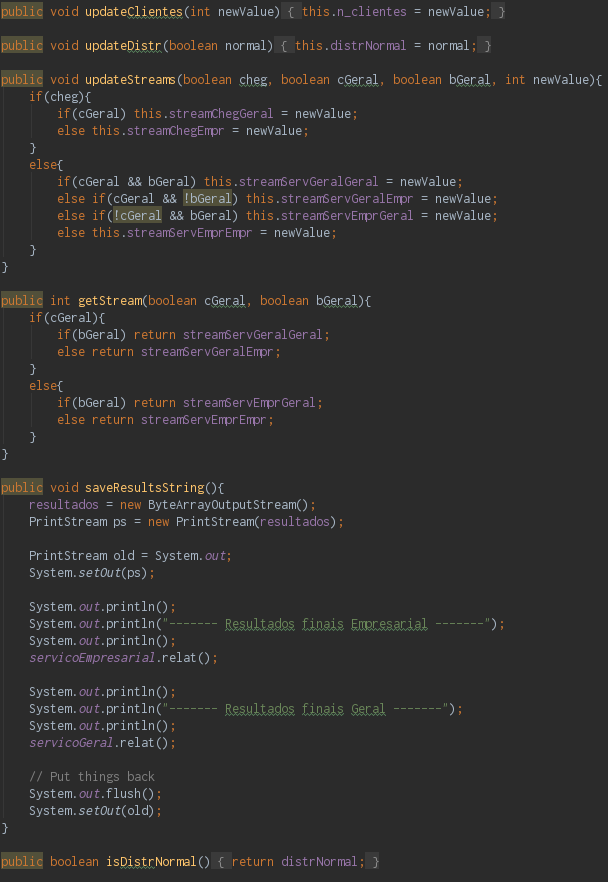
Servico

Foi necessário alterar o método *insereServico* para respeitar as condições pretendidas. Ao chegar um novo cliente é verificado se há funcionários livres, se houver, o cliente é imediatamente atendido. Se não, são verificada as condições em que o cliente e o balcão são do mesmo tipo e há funcionários livres, em que o cliente é um cliente geral e há funcionários livres no balcão empresarial, caso em que é imediatamente atendido nesse balcão, em que o cliente é um cliente empresarial e não há funcionários livres no balcão geral, neste caso existem mais duas condições, ou encontra-se um cliente geral a ser atendido, caso em que é redireccionado para o seu balcão, ou encontra-se um cliente empresarial a ser atendido, neste caso o cliente original entra na fila de espera e o último caso em que o cliente é empresarial e há funcionários livres no balcão geral, sendo aqui também imediatamente atendido.

Simulador

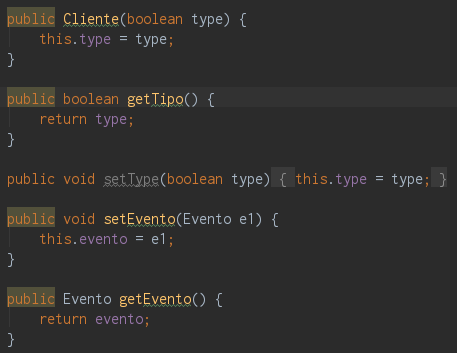






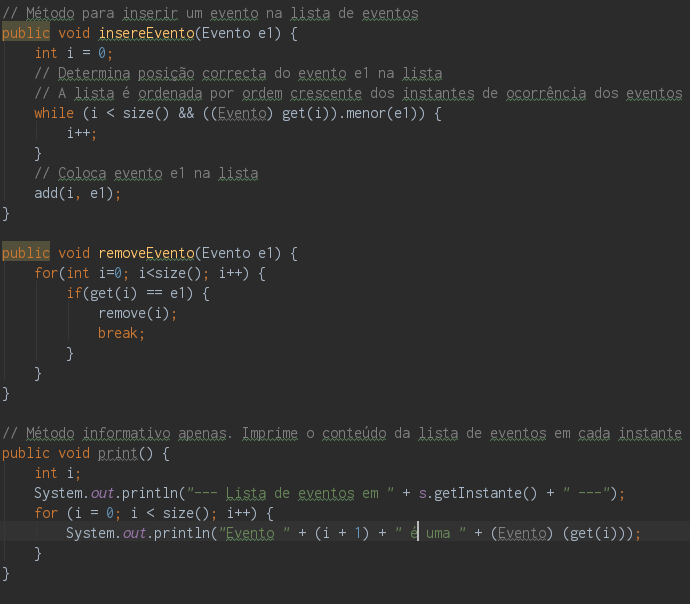
Em *Simulador* apenas foram criados métodos para devolver ou atualizar os vários parâmetros como as médias de serviço, as streams, os desvios padrão, …. Os métodos que devolvem parâmetros são usados pelo simulador em si, enquanto os que os atualizam são usados pela classe *Interface*, referente à GUI do simulador.

Cliente



Em *Cliente* apenas foi criado um método para devolver o evento de saída corresponde ao cliente atual.

ListaEventos



Em *ListaEventos* apenas foi criado um método para remover o evento correto da lista.