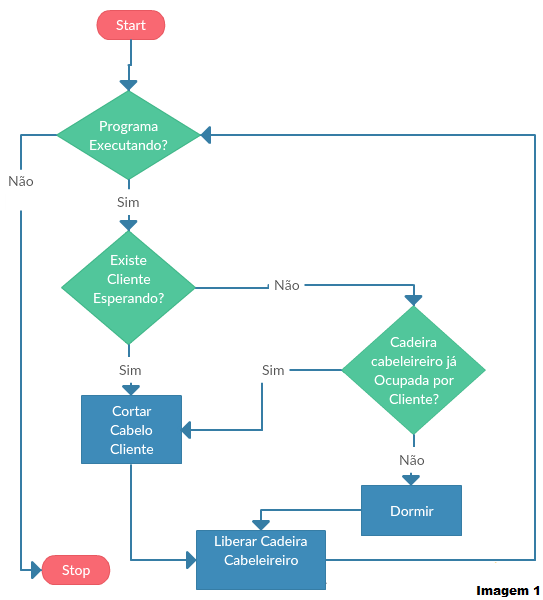
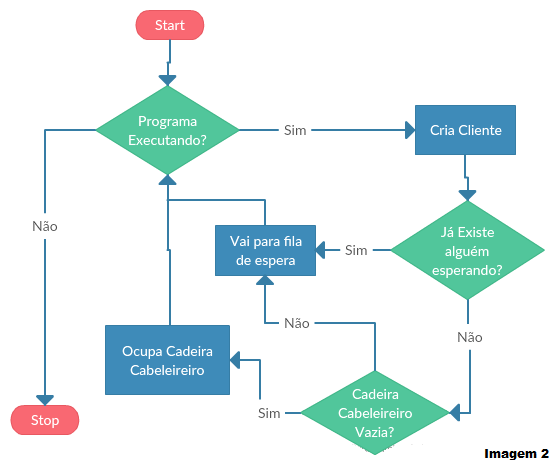
1. **O Problema do Barbeiro Dorminhoco**

O problema se resume a uma barbearia normal dos dias de hoje, onde se tem uma cadeira em que o barbeiro usa para cortar o cabelo dos clientes e outras cadeiras para os demais que chegarem esperarem a sua vez caso já tenha alguém sendo atendido. Porém no caso aqui retratado quando não houver ninguém para atendimento o barbeiro que irá ocupar a cadeira destina ao corte para dormir, sendo que de tempo em tempo ele acordará e irá verificar se há algum cliente esperando atendimento.

Abstraindo a ideia para a programação, foi trabalhado com duas threads, uma representando o cabeleireiro e outra simulando a chegada de clientes, onde elas compartilham algumas informações entre si, como a fila de espera e se a cadeira de atendimento está ocupada ou não.

O diagrama básico de ação, válido paras as threads, Cabeleireiro e Cliente, poderia ser descrito como na **Imagem 1 e Imagem 2** respectivamente representadas abaixo:

****

**O Algoritmo**

O algoritmo desse problema como descrito antes foi criado com duas classes Thread distintas, Cabeleireiro e Cliente, que tem sua execuções exemplificadas pelos seguintes passos:

**- Cabeleireiro:**

1. Verifica se o programa esta executando;
2. Verifica se há algum cliente esperando atendimento nas cadeiras de espera;
3. Se tiver alguém esperando chama o próximo da fila e atende o mesmo;
4. Senão tiver ninguém esperando verifica se já não tem alguém que chegou e ocupou a cadeira para cortar o seu cabelo, se sim atende o mesmo;
5. Caso não tenha ninguém esperando ou já em sua cadeira para atendimento, o cabeleireiro dorme por um tempo;
6. Depois de atender o cliente ou dormir libera a cadeira de uso do cabeleireiro para começar a verificação novamente;

**- Cliente:**

1. Verifica se o programa esta executando;
2. Sorteia um numero entre 0 e 1, se cair 1 “insere” um novo cliente (feito esse sorteio para simular a entrada aleatória de clientes como no mundo real);
3. Se for inserir um novo cliente verifica se a fila de espera esta vazia;
4. Se tiver alguém já esperando na fila ou a cadeira do cabeleireiro estiver ocupada, seja por cliente ou pelo próprio cabeleireiro dormindo, toma o seu lugar, se houver cadeira disponível, na fila de espera, senão “vai embora”;
5. Se estiver vazia a fila de espera e a cadeira do cabeleireiro não estiver ocupada, ocupa a mesma;
6. Depois de simular a chegada de um novo cliente, ficando este na cadeira já para ser atendido, ou na fila de espera, ou até mesmo indo embora, a Thread retoma ao passo 1;

**O Programa explicado em código**

O programa, assim como o primeiro problema passado, foi desenvolvido em Delphi, mais especificamente na versão Xe7, que também possui suporte nativo a Threads.

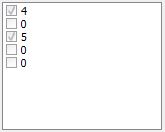
**Tela Principal:**

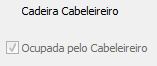
Nessa aplicação temos um Form principal onde antes de inicializar os processos pode-se configurar algumas informações que serão usados pelas Threads como o tempo para chegada de um novo cliente, quantidade de cadeiras para a fila de espera, tempo para o corte do cabelo e o tempo da soneca do cabeleireiro, sendo que os tempos são relativos a segundos.

Após inicializar os processos essas configurações são bloqueadas e só liberadas quando ao termino de tudo.



No Form principal ainda temos um CheckListBox, que seria a nossa fila de espera, onde para cada cadeira cria-se um CheckBox. Para controlar a ordem da fila e se as cadeiras estão ocupadas ou não, a aplicação percorre essa lista e para cada CheckBox desmarcado é uma cadeira vaga, os que estão marcados seriam as cadeiras ocupas, e junto consigo, no Caption de cada um, guarda-se a ordem da fila de espera, onde o menor numero das cadeiras marcadas seria o próximo a ser atendido. Para as cadeiras vagas, seta-se o Caption como 0.



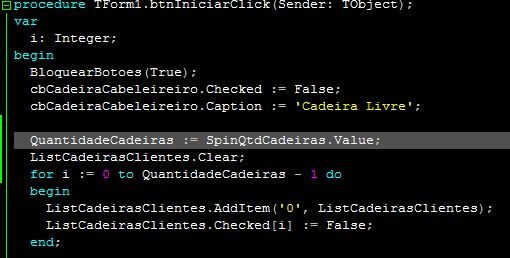
Temos também a cadeira do cabeleireiro representada por um CheckBox, que estando marcado representa que está ocupada. Para esse componente o Caption acompanha o status da cadeira como “Ocupada por cliente”, “Ocupada pelo Cabeleireiro” e “Cadeira Livre”.

Por fim nesse mesmo Form temos dois botões, um para iniciar e outro parar a aplicação, que são liberados e bloqueados para click de acordo com o necessário.



Depois de mostrar os componentes usados e a maneira como cada um se comporta, vamos inicializar a aplicação.

Após configurar as informações conforme necessário clica-se no botão Iniciar.

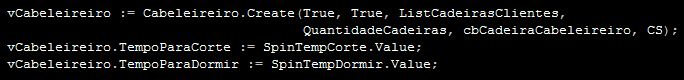
Iniciada a aplicação a primeira coisa que ela ira fazer é bloquear os botões, deixando liberado nesse caso apenas o botão de Parar, logo em seguida irá setar a cadeira do cabeleireiro como livre, além de inicializar a fila de espera deixando todas as cadeiras liberadas e com prioridade 0, já que cadeiras livres não tem prioridade nessa fila.

Logo após criamos as duas Threds que serão usadas (Cabeleireiro e Cliente), mandando na criação delas alguns parâmetros, de acordo com cada uma.

**Cabeleireiro:**

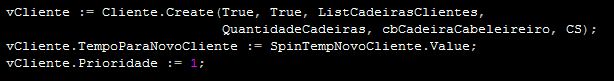
Para ela enviamos duas constantes Boolean, a primeira para identificar se ela será criada suspensa ou não, a segunda ira setar uma variável na própria Thread que irá controlar se o programa está executando ou não. Manda-se no construtor dela também a quantidade de cadeiras da fila de espera que foram criadas, além de enviar a referência a CheckListBox (Lista das cadeiras de espera) e o CheckBox (Cadeira do cabeleireiro). Para finalizar os parâmetros enviamos a referencia a uma variável do tipo TCriticalSection, sendo ela a responsável dentro da Thread de não deixar ela alterar algum campo que já esta sendo alterado por outra.

Após a criação dela, setamos outras duas variáveis que ela utiliza: Tempo utilizado para o corte dos cabelos e Tempo que o cabeleireiro irá dormir.



**Cliente:**

A criação da Thread Cliente é exatamente igual a Cabeleireiro, inclusive mandado os mesmos dados e parâmetros, a única diferença entre as duas nessa parte está nas variáveis que são setadas após a criação, onde nesse caso agora é o Tempo para criação de um novo cliente e a prioridade da fila, começando sempre com 0, pois esse número será a sequencia dos clientes que ali chegarem.

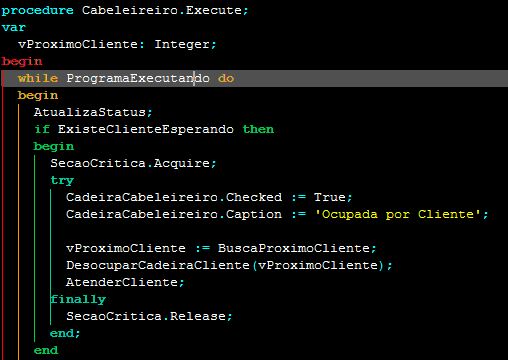


Após a criação das Threas é dado um Start em cada uma delas, tendo em vista que é cada uma por si agora, vamos analisar o processo detalhado delas.

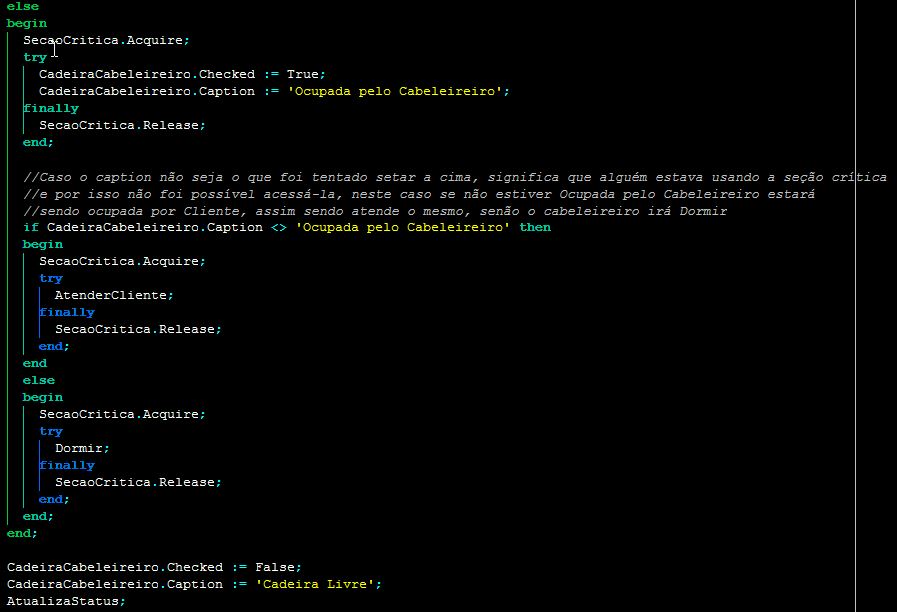
**Thread Cabeleireiro:**

Dentro do Execute dela foi criado um While que é controlado pela variável que indica se a aplicação está em execução ou não, feito isso para controlar uma possível parada solicitada pelo usuário na tela principal. Logo abaixo, dentro do While chamamos a procedure *AtualizaStatus* que é usada para manter a CheckListBox e o CheckBox atualizados com as ultimas informações, onde nela se da um Update em cada um. Na sequência chamamos a função *ExisteClienteEsperando*, que nos retorna True se houver alguma cadeira ocupada na fila de espera, se sim a Thread faz uso da Seção Crítica para reservar a cadeira do cabeleireiro, impedindo que outra tente usar essa mesma Cadeira, além de verificar qual o próximo cliente da fila a ser chamado, liberando assim a cadeira de espera que ele estava usando.

Após essas verificações a Thread chama a procedure *AtenderCliente* que dá um Sleep de acordo com o tempo para corte setado lá no Form principal.



Caso a verificação anterior, se existe alguém esperando, seja Falsa a Thread irá verificar se a Cadeira do Cabeleireiro já não esta ocupada, pois algum cliente pode, nesse meio tempo ter chegado e não encontrando ninguém esperando e a cadeira do cabeleireiro estando vazia, pode ele ter a ocupado. Nesse caso atende-se o cliente como já citado a cima.

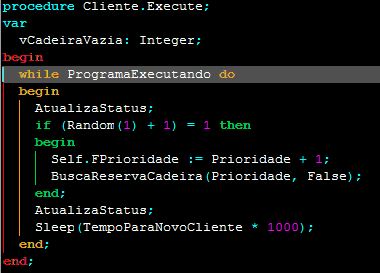
Agora caso nenhuma das verificações anteriores forem verdadeiras, algum cliente esperando e cadeira do cabeleireiro ocupada, a Thread fará uso da Seção Crítica e tentará setar como ocupada a cadeira do cabeleireiro, mas caso outra Thread já estiver usando a Seção Crítica não se conseguirá setar como Ocupada pelo Cabeleireiro, nesse caso ela estará Ocupada por Cliente, assim sendo atende-se o Cliente, senão será chamada a procedure *Dormir* que dará um Sleep, como para o corte de cabelo, de acordo com o tempo definido para dormir lá no Form principal.

Após o Cabeleireiro atender os clientes ou dormir, a Thread irá liberar a Cadeira dele, atualizar os status dos Componentes e iniciar essas verificações novamente até o programa não estar mais executando.

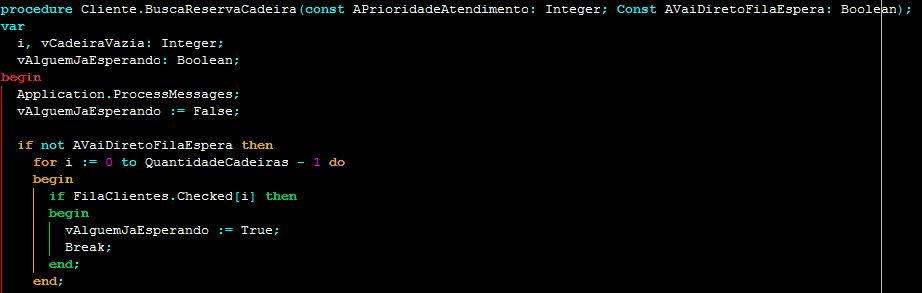
**Thread Cliente:**

Como na Thread Cabeleireiro dentro do Execute dela foi criado um While que é controlado pela variável que indica se a aplicação está em execução ou não, feito isso para controlar uma possível parada solicitada pelo usuário na tela principal. Logo abaixo, dentro do While chamamos a procedure *AtualizaStatus*, mesma função já utilizada na Thread Cabeleireiro. Na sequência sorteia-se um numero entre 0 e 1, se for um será criado um novo cliente, feito isso para simular a entrada aleatória de clientes como no mundo real.

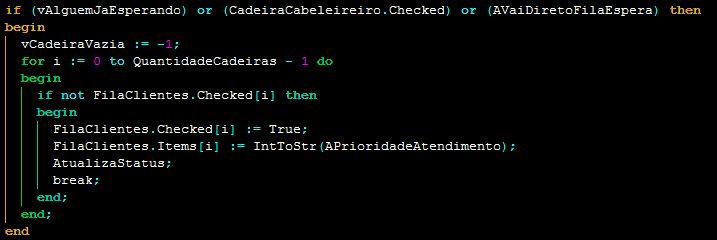
Caso for criado um novo cliente a Thread irá chamar a procedure *BuscaReservaCadeira* onde passamos por parâmetros para ela a prioridade, ou sequência que esse cliente terá na fila de espera, e se ele irá direto para a fila de espera, nesse caso nunca irá direto.



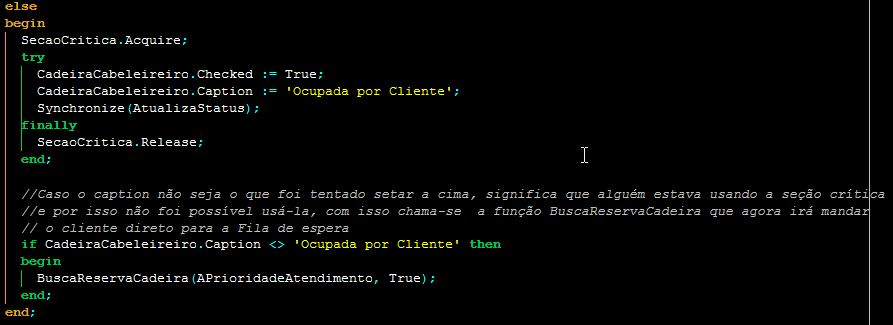
Entrando na procedure *BuscaReservaCadeira* a primeira coisa que ela irá verificar é se já tem algum cliente esperando, agora caso o parâmetro que sinaliza se o cliente irá direto para a fila de espera for verdadeiro não será feita essa verificação.



Na sequência, após a Thread saber se já há algum cliente esperando, faz-se um IF onde caso a cadeira do cabeleireiro esteja ocupada, tenha alguém esperando ou o parâmetro para enviar o cliente direto para a fila for verdadeiro, o sistema irá procurar um lugar vago, tendo um seta ele como ocupado e coloca-se no caption a prioridade do novo cliente. Entretanto, caso não tenha lugar vago a Thread não fará mais nada, pois nesse caso o cliente “irá embora”.



Agora caso não tenha ninguém esperando, a cadeira do cabeleireiro esteja livre e o parâmetro para ir direto para a fila de espera seja falso, o cliente irá ocupar direto a cadeira para cortar o cabelo. Nesse momento usa-se a variável de Seção Crítica para evitar que alguém tente alterar campos ao mesmo tempo, altera-se com isso a Cadeira do Cabeleireiro como ocupada e o Caption dela como “Ocupada por Cliente”, após isso a Thread verifica se o Caption atual da Cadeira é igual ao setado a pouco, pois outra Thread pode ter feito uso da Seção Crítica antes, podendo com isso não ter sido reservada para esse cliente. Caso isso ocorra chama-se a função *BuscaReservaCadeira* tendo como parâmetros a mesma prioridade do inicio da função atual, e a variável para enviar o cliente direto a lista de espera como verdadeira.



Após o fim da execução da procedure *BuscaReservaCadeira* a Thread volta ao Execute que atualiza os status dos componentes usados, e dá-se um Sleep para dormir a quantidade de tempo configurada para a criação de um novo Cliente.