Documento Técnico do Sistema Coleta Lixo Teresina (SCLT)

Introdução

Este documento técnico descreve o processo da modelagem do Simulador de Coleta de Lixo da cidade de Teresina (SCLT), onde caminhões pequenos fazem a coleta de lixo diariamente em zonas e caminhões grandes recebem o lixo e faz o transporte para o aterro. O processo foi feito inteiramente em Java sem uso de bibliotecas para filas, listas e pensado em lógica configurável para poder maximizar a eficiência do sistema.

Contextualização do Sistema de Coletas:

Há 5 zonas na cidade de Teresina (Norte, Sul, Centro, Leste e Sudeste) e toneladas de lixo precisam ser coletadas todos os dias.
Para a coleta de lixo nas zonas, varios caminhões pequenos são encarregados de coletar lixo das zonas, porém tem capacidades limitadas e número maximo de viagens coletando lixo que são feitas no dia.

Após o numero de viagens atingir o maximo ou caso sua capacidade de carga tenha enchido, eles vão em direção a uma de duas estações da cidade. Quando chegados a estação, eles depositam seus lixos em caminhões grandes com maior capacidade que estão esperando nas estações para poder processar o lixo e leva-lo ao aterro.

No entanto, há certas dificuldades durante o tempo de coleta, pois os horários de pico interfere na locomoção dos caminhões pequenos entre as zonas, e além disso, os caminhões grandes tem um tempo de espera por caminhões pequenos para poder ir ao aterro, caso não há carga, eles toleram por certo tempo até receber carga e partir.

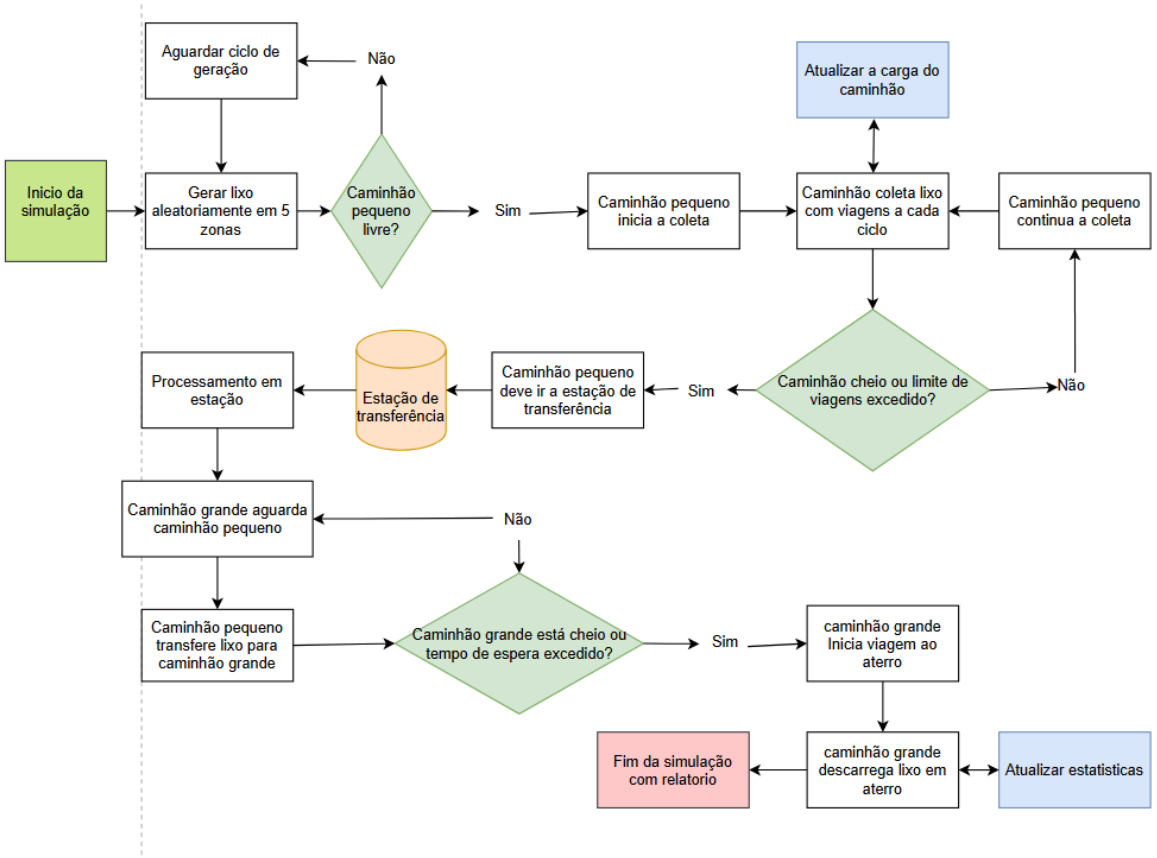
O sistema simula não apenas a coleta de lixo como considera todos esses fatores; tempo de espera, horário de pico e viagens limitadas que podem atrapalhar na eficiência da coleta. Com relatórios por hora que indicam as atividades dos caminhões e relatório final do dia com cálculo sobre a eficiência do sistema utilizado.

Modelagem do sistema:

- **`Config`**: Classe de armazenamento para configuração das instâncias presentes na simulação, como carga máxima dos caminhões, tempo de espera, tempo de simulação, etc.
- **`CaminhaoPequeno`**: Classe representante de caminhões pequenos que coletam lixo pelas zonas, com limite de viagens e substituição.
- **`CaminhaoGrande`**: Recebe a carga de lixo dos caminhões pequenos, com tempo de espera, e faz transporte do lixo para o aterro.
- **`EstacaoTransferencia`**: Duas estações de processamento de lixo antes de ir ao aterro, com fila de caminhões grandes e pequenos.
- **`Fila`**: Classe de fila encadeada para gerenciamento dos caminhões e das estações.
- **`SistemaColetaLixo`**: Classe principal do SCLT, montando a simulação e gerando relatórios por hora e um relatório final com calculo da eficiência do sistema.

Diagrama de fluxo do sistema

Um diagrama de fluxo demonstrando como é feito o processo de simulação do sistema a cada relatório.

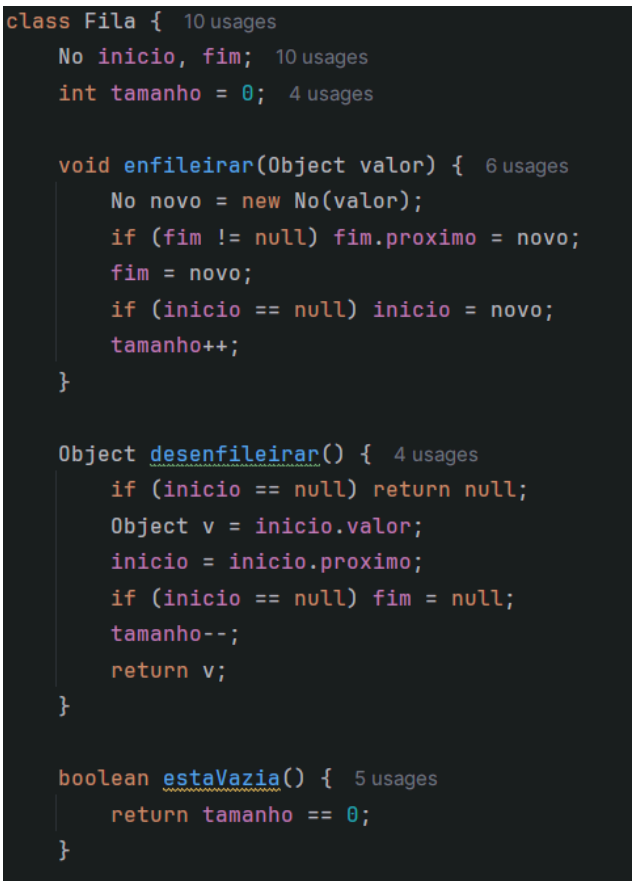


Estrutura de Dados

Utilizamos lista encadeada com nós através das classes No.java e Fila.java, cada nó armazena um elemento para montar uma fila.

Na classe fila há três funções importantes:

- `enfileirar()`:Onde um elemento é adicionado no final da fila.
- `desenfileirar()`:Onde o primeiro elemento é removido da fila
- `estaVazia()`:Onde verifica se a fila tem algum elemento e está vazia.



```
class Fila {  10 usages
    No inicio, fim;  10 usages
    int tamanho = 0;  4 usages

    void enfileirar(Object valor) {  6 usages
        No novo = new No(valor);
        if (fim != null) fim.proximo = novo;
        fim = novo;
        if (inicio == null) inicio = novo;
        tamanho++;
    }

    Object desenfileirar() {  4 usages
        if (inicio == null) return null;
        Object v = inicio.valor;
        inicio = inicio.proximo;
        if (inicio == null) fim = null;
        tamanho--;
        return v;
    }

    boolean estaVazia() {  5 usages
        return tamanho == 0;
    }
```

Utilizamos estas funções para montar a fila de caminhões pequenos, grandes e em viagem na classe EstacaoTransferencia. Gerenciando a ordem de atendimento dos caminhões que estão presentes nas estações

Algoritmos Principais

1. O Algoritmo de processamento da Estação `EstacaoTransferencia.processar()`

- Atualiza os tempos de viagem dos caminhões grandes
- Verifica se os caminhões grandes atingiram sua tolerância de espera
- Responsavel por gerenciar a transferencia de lixo de um caminhão pequeno para um caminhão grande

- Caso o tempo de espera exceda e os caminhões grandes forem ao aterro, novos são adicionados.

```
void processar(int tempoAtual) { //algoritmo para o processamento de caminhões pequenos e grandes em estações
    Fila novosEmViagem = new Fila();
    while (!filaGrandesEmViagem.estaVazia()) {
        CaminhaoGrande g = (CaminhaoGrande) filaGrandesEmViagem.desenfileirar();
        g.atualizarTempo();
        if (g.emViagemAterro) {
            novosEmViagem.enfileirar(g);
        } else {
            filaGrandesDisponiveis.enfileirar(g);
        }
    }
    filaGrandesEmViagem = novosEmViagem;

    if (!filaGrandesDisponiveis.estaVazia()) {
        CaminhaoGrande grande = (CaminhaoGrande) filaGrandesDisponiveis.primeiro();
        grande.tempoEspera++;

        if (grande.tempoEspera >= Config.TOLERANCIA_CAMINHAO_GRANDE) {
            if (grande.carga > 0) {
                grande.iniciarViagemAterro();
                filaGrandesEmViagem.enfileirar(grande);
                filaGrandesDisponiveis.desenfileirar();
            } else {
                grande.tempoEspera = 0;
```

2. Algoritmo de substituição dos caminhões pequenos
   substituirCaminhoesVelhos()

Caso um caminhão pequeno atinja o maximo de viagens, ele não opera mais, logo são substituidos por novos caminhões para continuar coletando lixo durante o resto do dia até eles atingirem sua quantidade maxima de viagens.

```
void substituirCaminhoesVelhos(SistemaColetaLixo sistema) { //Lógica para a substituição de caminhões pequenos
    Fila novaFila = new Fila();
    while (!filaPequenos.estaVazia()) {
        CaminhaoPequeno cp = (CaminhaoPequeno) filaPequenos.desenfileirar();
        if (cp.paraSubstituicao) {
            sistema.substituirCaminhao(cp);
        } else {
            novaFila.enfileirar(cp);
        }
    }
    filaPequenos = novaFila;
}
```

3. Algoritmo de geração dos relatórios
   SistemaColetaLixo.gerarRelatorioHora()

Exibe dados a cada 60 minutos, com cores no terminal, formando 24 relatorios ao todo. A cada relatório mostra a atividade de cada caminhão grande e pequeno, o lixo coletado por zona, a hora da

coleta e o processamento nas estações.

```
private void gerarRelatorioHora(int hora) { //logica do relatorio de hora em hora  1 usage
    String horario = String.format("%02d:00-%02d:59", hora, hora);

    System.out.println(Config.AZUL + "\n╔═══════════════════════════════════════════════════╗");
    System.out.println("║ " + Config.NEGRITO + "RELATÓRIO HORÁRIO - " + horario + Config.RESET + Config.AZUL + "
    System.out.println("╚═══════════════════════════════════════════════════╝" + Config.RESET);

    System.out.println(Config.VERDE + "\n[LIXO COLETADO POR ZONA]" + Config.RESET);
    for (int i = 0; i < lixoPorZona.length; i++) {
        System.out.printf("  %-8s: %s%4d toneladas%s\n",
                Config.NOMES_ZONAS[i],
                Config.CIANO,
                lixoPorZona[i],
                Config.RESET);
    }
    System.out.println(Config.VERDE + "\n[STATUS DOS CAMINHÕES PEQUENOS]" + Config.RESET);
    System.out.printf("%s%-4s %-8s %-10s %-14s %-24s %s\n",
            Config.AMARELO,
            "ID", "Zona", "Carga", "Status", "Atividade", "Viagens",
            Config.RESET);
```