Documento Técnico - Estrutura de Dados:

Simulador de Coleta de Lixo para Teresina

Aluno: João Gabriel SIlva Rabelo

Turma: Shaw

1. Introdução

1.1 Objetivo

Esta Descrição de Projeto de Software (SDD) fornece uma visão abrangente do projeto e implementação de um simulador de coleta de lixo para a cidade de Teresina, Brasil, conforme especificado nos requisitos do projeto. O simulador modela o processo de coleta de resíduos, integrando caminhões pequenos e grandes, estações de transferência e um aterro sanitário, com foco na sustentabilidade ambiental e na eficiência da gestão de resíduos. Este documento descreve a arquitetura do sistema, estruturas de dados, algoritmos e detalhes de implementação para atender aos requisitos funcionais e não funcionais definidos na especificação do projeto.

1.2 Escopo

O simulador modela o processo de coleta de lixo em Teresina, dividido em cinco zonas urbanas (Norte, Sul, Leste, Sudeste e Centro). Ele simula caminhões pequenos (capacidades de 2, 4, 8 e 10 toneladas) coletando resíduos e transferindo-os para caminhões grandes (20 toneladas) em duas estações de transferência, que então transportam os resíduos para um aterro sanitário. O sistema utiliza estruturas de dados personalizadas (listas e filas) para gerenciar eventos, rastrear estatísticas operacionais e suporta parâmetros configuráveis, como taxas de geração de lixo, tempos de viagem, tolerâncias de espera e o nome do arquivo para salvar logs de eventos, além de exibi-los em tempo real no console. O objetivo principal é analisar o número mínimo de caminhões grandes necessários para atender à demanda de gestão de resíduos da cidade.

1.3 Definições, Siglas e Abreviações

- **TAD**: Tipo Abstrato de Dados (estruturas de dados personalizadas implementadas para o projeto).

- **Caminhão Pequeno**: Caminhões com capacidades de 2, 4, 8 ou 10 toneladas, responsáveis por coletar resíduos das zonas urbanas.

- **Caminhão Grande**: Caminhões com capacidade de 20 toneladas, usados para transportar resíduos das estações de transferência até o aterro.

- **Estação de Transferência**: Instalação onde os caminhões pequenos descarregam resíduos, seja em armazenamento ou diretamente em caminhões grandes.

- **Aterro Sanitário**: Destino final para a disposição dos resíduos.

- **Horário de Pico**: Períodos de tráfego intenso que afetam os tempos de viagem (7:00–8:59, 12:00–12:59, 17:00–18:59).

- **ZonaStats**: Classe interna para rastrear estatísticas específicas de cada zona durante a simulação.

1.4 Referências

- Especificação do Projeto: “Simulador de Coleta de Lixo para Teresina - Gestão de Resíduos Sólidos e Sustentabilidade Ambiental” (PDF fornecido).

2. Visão Geral do Sistema

O simulador de coleta de lixo é uma aplicação baseada em Java projetada para modelar o processo de gestão de resíduos em Teresina. A cidade é dividida em cinco zonas, cada uma gerando quantidades configuráveis de resíduos diariamente. Caminhões pequenos coletam resíduos dessas zonas e os entregam a uma das duas estações de transferência, onde os resíduos são armazenados ou carregados em caminhões grandes para transporte até o aterro. O sistema oferece alocação dinâmica de caminhões, gerenciamento de filas e registro em tempo real de eventos, com estatísticas para avaliar a eficiência operacional.

2.1 Objetivos do Sistema

- Simular a coleta e transporte de resíduos em cinco zonas urbanas.

- Gerenciar operações dos caminhões (pequenos e grandes) usando estruturas de dados personalizadas (listas e filas).

- Fornecer parâmetros configuráveis para geração de resíduos, capacidades dos caminhões, tempos de viagem e tolerâncias de espera.

- Gerar estatísticas detalhadas, incluindo resíduos coletados, tempos de espera e número de caminhões grandes utilizados.

- Determinar o número mínimo de caminhões grandes necessário para atender à demanda de resíduos de Teresina.

2.2 Restrições do Sistema

- Não é permitido o uso das estruturas de dados nativas do Java (por exemplo, `ArrayList`, `LinkedList`); TADs personalizados devem ser implementados.

- A simulação deve lidar com variações na geração de resíduos e nas condições de tráfego (horários de pico e fora de pico).

- O sistema deve registrar eventos em tempo real e produzir relatórios com representações gráficas da coleta de resíduos.

3. Considerações de Projeto

3.1 Suposições

- A cidade é dividida em cinco zonas fixas com distâncias predefinidas.

- A geração de resíduos ocorre diariamente dentro de intervalos configuráveis pelo usuário.

- Os tempos de viagem variam conforme o horário: 20 km/h durante horários de pico e 30 km/h fora de pico, com um fator de aleatoriedade de ±10%.

- Caminhões pequenos têm um número máximo de viagens por dia; caminhões grandes operam indefinidamente, mas possuem uma tolerância de espera.

3.2 Restrições

- Todas as operações com listas e filas devem utilizar estruturas de dados personalizadas.

- O sistema deve alocar dinamicamente caminhões grandes quando os tempos de espera forem excedidos.

3.3 Metas de Projeto

- **Modularidade**: Classes separadas para caminhões, estações, zonas e lógica de simulação, facilitando a manutenção.

- **Eficiência**: Uso otimizado de listas e filas personalizadas para gerenciar caminhões e fluxo de resíduos.

- **Configurabilidade**: Suporte a parâmetros definidos pelo usuário por meio de interface de linha de comando.

- **Transparência**:Registro detalhado de eventos em tempo real no console e armazenamento persistente em arquivo, além de relatórios estatísticos para análise de desempenho do sistema.

4. Projeto Arquitetural

4.1 Arquitetura do Sistema

O simulador segue uma arquitetura modular, orientada a objetos, com os seguintes componentes principais:

- **Main**: Ponto de entrada da aplicação, responsável por inicializar o simulador e a interface do usuário.

- **Simulador**: Núcleo do motor de simulação, gerencia o loop da simulação, distribuição dos caminhões e atualização dos estados.

- **CaminhaoPequeno**: Representa os caminhões pequenos, responsáveis pela coleta de resíduos e deslocamento até as estações de transferência.

- **CaminhaoGrande**: Representa os caminhões grandes, responsáveis pelo transporte de resíduos das estações até o aterro.

- **EstacaoTransferencia**: Modela as estações de transferência, gerenciando as filas de caminhões pequenos e a transferência de resíduos para os grandes.

- **ZonaUrbana**: Representa as zonas urbanas, onde os resíduos são gerados e coletados.

- **DistribuicaoCaminhoes**: Gerencia a alocação de caminhões pequenos para as zonas, com base na disponibilidade de resíduos e tempo de viagem.

- **Estatisticas**: Acompanha e reporta métricas da simulação (ex.: resíduos coletados, tempos de espera).

- **LoggerSimulacao**: Lida com o registro de eventos, com modos configuráveis (normal/debug), saída colorida no console e armazenamento em um arquivo de log especificado pelo usuário.

- **InterfaceSimulador**: Fornece uma interface de linha de comando para interação e configuração do usuário.

4.2 Estrutura de Pacotes

O sistema está organizado nos seguintes pacotes:

- **caminhoes**: Contém classes dos caminhões pequenos (CaminhaoPequeno) e grandes (CaminhaoGrande), geração de placas (Placa) e distribuição de caminhões (DistribuicaoCaminhoes).

- **estacoes**: Inclui a lógica das estações de transferência (EstacaoTransferencia) e o resultado do processamento das filas (ResultadoProcessamentoFila).

- **estruturas**: Define estruturas de dados personalizadas (Lista, Fila, No, MapaEventos) para listas, filas e mapeamento de cores de eventos.

- **simulacao**: Contém a lógica da simulação (Simulador), estatísticas (Estatisticas), logging (LoggerSimulacao) e a interface do usuário (InterfaceSimulador).

- **zonas**: Inclui o gerenciamento das zonas (ZonaUrbana) e suas estatísticas (ZonaEstatistica).

4.3 Fluxo de Dados

1. **Inicialização**: O usuário configura os parâmetros (número de caminhões, intervalos de geração de resíduos, tolerâncias de espera) via InterfaceSimulador.

2. **Geração de Resíduos**: As zonas urbanas (ZonaUrbana) geram resíduos diários conforme os intervalos definidos.

3. **Coleta**: Caminhões pequenos coletam os resíduos, e a classe DistribuicaoCaminhoes os aloca às zonas com base no acúmulo de resíduos e tempo de viagem.

4. **Transferência**: Caminhões pequenos cheios se dirigem a uma estação de transferência, entram em uma fila (Fila) e descarregam os resíduos em armazenamento ou diretamente em caminhões grandes.

5. **Transporte para o Aterro**: Caminhões grandes, quando cheios ou ao excederem a tolerância de espera, vão até o aterro (considerado uma zona especial - zonaAterro) e descarregam.

6. **Estatísticas e Registro**: A classe Estatisticas coleta métricas e o LoggerSimulacao registra os eventos com carimbos de tempo e cores específicas no console, salvando-os simultaneamente em um arquivo de log configurado.

5. Projeto de Dados

5.1 Estruturas de Dados

O simulador utiliza estruturas de dados personalizadas para gerenciar entidades e eventos:

- **Lista<T>**: Lista encadeada simples para armazenar caminhões, zonas e estatísticas.
    - Operações: `adicionar`, `obter`, `remover`, `limpar`.
    - Complexidade: O(1) para adicionar ao final; O(n) para acessar/remover por índice.
- **Fila<T>**: Fila circular usada para gerenciar caminhões pequenos nas estações de transferência.
    - Operações: `enfileirar`, `remover`, `primeiroDaFila`, `obter`.
    - Complexidade: O(1) para enfileirar/desenfileirar; O(n) para acesso por índice.
- **No<T>**: Classe de nó usada tanto na `Lista` quanto na `Fila`, contendo os dados e ponteiros (`prox`, `ant`).
- **MapaEventos**: Mapa personalizado baseado em lista para associar tipos de eventos a códigos de cores ANSI usados no log.
    - Complexidade: O(n) para `put` e `get` devido à busca linear.

5.2 Elementos de Dados

CaminhaoPequeno

- Atributos:
    - `id` (placa),
    - `capacidade` (2, 4, 8 ou 10 toneladas),
    - `cargaAtual`,
    - `limiteViagens`,
    - `viagensFeitas`,

    - `status` (1–6),
    - `zonaAtual`,
    - `zonaInicial`,
    - `estacaoDestino`,
    - `zonaDestino`,
    - `tempoViagemRestante`,
    - `tempoColetaRestante`,
    - `tempoEsperaFila`,
    - `quantidadeColetando`,
    - `cargaPorMinuto`.

CaminhaoGrande

- Atributos:
    - `placa`,
    - `capacidade` (20 toneladas),
    - `cargaAtual`,
    - `toleranciaEspera`,
    - `status` (0–3),
    - `tempoViagemRestante`,
    - `estacaoOrigem`,
    - `estacaoDestino`.

EstacaoTransferencia

- Atributos:

    - `nome`,
    - `lixoArmazenado`,
    - `filaPequenos` (Fila),
    - `listaGrandes` (Lista),
    - `esperaMaxPequenos`,
    - `esperaTotalPequenos`,
    - `caminhaoGrandeEsperando`,
    - `temCaminhaoGrandeEsperando`,
    - `tempoEsperaCaminhaoGrande`,
    - `tempoProcessamentoRestante`,
    - `processandoGrande`,
    - `cargaParaGrande`.

ZonaUrbana

- Atributos:
    - `nome`,
    - `lixoAcumulado`,
    - `lixoMin`,
    - `lixoMax`,
    - `variacaoPico`,
    - `variacaoNormal`,
    - `caminhoesAtivos`.

ZonaEstatistica

- Atributos:
    - `nomeZona`,
    - `lixoColetado`,
    - `lixoGerado`.

Estatisticas

- Atributos:
    - `totalLixoColetado`,
    - `totalLixoGerado`,
    - `totalLixoAterro`,
    - `totalCaminhoesGrandesUsados`,
    - `tempoTotalEsperaPequenos`,
    - `descarregamentos`,
    - `maxCaminhoesGrandesEmUso`,
    - `lixoPorZona` (Lista),
    - `tempoSimulado`,
    - `caminhoesGrandesEmUsoAtual`.

6. Projeto de Componentes

6.1 Descrição dos Componentes

6.1.1 Main

- **Propósito**: Inicializa o simulador e a interface do usuário.
- **Responsabilidades**: Cria uma instância do `Simulador` e inicia a `InterfaceSimulador`.

- **Métodos Principais**: `main` (ponto de entrada).

6.1.2 Simulador

- **Propósito**: Gerencia o loop de simulação e coordena as interações entre caminhões, estações e zonas.
- **Responsabilidades**:
    - Inicializa zonas, caminhões e estações.
    - Atualiza o estado da simulação.
    - Executa reinicializações diárias.
    - Gera estatísticas.
- **Métodos Principais**:
    - `iniciar`: Inicia a simulação em uma nova thread.
    - `atualizarSimulacao`: Avança o tempo da simulação, processa caminhões e estações, e gera relatórios horários.
    - `concluirDia`: Reinicia os caminhões e limpa as estações no fim de cada dia.
    - `processarCaminhoesPequenos`: Gerencia coleta e trânsito dos caminhões pequenos.
    - `processarEstacoes`: Processa filas e atribuições de caminhões grandes nas estações.
    - `distribuirCaminhoesDisponiveis`: Aloca caminhões pequenos às zonas.

6.1.3 CaminhaoPequeno

- **Propósito**: Modela caminhões pequenos responsáveis por coletar resíduos nas zonas.

- **Responsabilidades**:
    - Coleta de resíduos incrementalmente.
    - Deslocamento até as estações.
    - Entrada em filas.
- **Métodos Principais**:
    - `coletar`: Inicia a coleta de resíduos com alocação proporcional à capacidade.
    - `processarColeta`: Atualiza o progresso da coleta, adicionando resíduos gradualmente.
    - `processarViagem`: Monitora o deslocamento para zonas ou estações.
    - `descarregar`: Descarrega os resíduos na estação de transferência.

6.1.4 CaminhaoGrande

- **Propósito**: Modela caminhões grandes que transportam resíduos até o aterro.
- **Responsabilidades**:
    - Carregar resíduos na estação.
    - Viajar até o aterro.
    - Descarregar resíduos.
- **Métodos Principais**:
    - `carregar`: Recebe resíduos da estação (armazenamento ou caminhões pequenos).
    - `descarregar`: Descarrega no aterro e atualiza estatísticas.
    - `atualizarEstado`: Atualiza o estado do caminhão (em viagem, descarregando, retornando).

6.1.5 EstacaoTransferencia

- **Propósito**: Gerencia a transferência de resíduos de caminhões pequenos para armazenamento ou caminhões grandes.
- **Responsabilidades**:
    - Processar filas de caminhões pequenos.
    - Atribuir caminhões grandes conforme necessário.
    - Lidar com limites de armazenamento.
- **Métodos Principais**:
    - `receberCaminhaoPequeno`: Adiciona caminhões pequenos à fila.
    - `processarFila`: Processa o descarregamento de caminhões pequenos e transferência para os grandes.
    - `atribuirCaminhaoGrande`: Designa um caminhão grande para a estação.
    - `liberarCaminhaoGrandeSeNecessario`: Envia caminhões grandes ao aterro quando cheios ou ao excederem o tempo de espera.

6.1.6 ZonaUrbana

- **Propósito**: Representa zonas urbanas com geração e coleta de resíduos.
- **Responsabilidades**:
    - Gerar resíduos diariamente.
    - Acompanhar caminhões ativos.
    - Fornecer dados de distância para cálculo de tempo de viagem.
- **Métodos Principais**:
    - `gerarLixo`: Gera quantidade aleatória de resíduos conforme os intervalos configurados.
    - `coletarLixo`: Remove resíduos coletados e atualiza estatísticas.

- `getDistancia`: Retorna a distância entre zonas.

6.1.7 DistribuicaoCaminhoes

- **Propósito**: Aloca caminhões pequenos para zonas com base na quantidade de resíduos e tempo de viagem.
- **Responsabilidades**:
  - Avaliar zonas com função de pontuação.
  - Calcular tempo de viagem com variação de tráfego.
- **Métodos Principais**:
  - `distribuirCaminhoes`: Atribui caminhões disponíveis às zonas.
  - `calcularTempoViagem`: Calcula tempo de deslocamento considerando variações.
  - `calcularPontuacao`: Atribui pontuações às zonas para priorização.

6.1.8 Estatisticas

- **Propósito**: Acompanha e reporta métricas da simulação.
- **Responsabilidades**:
  - Registrar resíduos coletados e enviados ao aterro.
  - Monitorar tempos de espera e uso de caminhões.
- **Métodos Principais**:
  - `registrarColeta`: Registra resíduos coletados por zona.
  - `registrarLixoAterro`: Registra quantidade descarregada no aterro.
  - `imprimirRelatorio`: Gera relatório formatado com estatísticas e gráficos de barras em ASCII.

6.1.9 LoggerSimulacao

- **Propósito**: Gerencia o registro de eventos com saída colorida no console e armazenamento em arquivo de log
- **Responsabilidades**:
    - Registrar eventos da simulação em modo normal ou debug.
    - Marcar logs com timestamp e cor por tipo de evento no console, e com timestamp e tipo de evento no arquivo.
    - "Inicializar e fechar o arquivo de log de forma segura.
- **Métodos Principais**:
    - `log`: Imprime mensagens formatadas.
    - `logRelatorio`: Imprime relatórios estatísticos.
    - `formatarTempo`: Converte o tempo da simulação para um formato legível (ex: "Dia 1, 08:30").
    - `inicializarLogArquivo:` Inicializa o arquivo de log com o nome especificado.
    - `fecharLogArquivo:`Fecha o arquivo de log ao encerrar a simulação.

6.1.10 InterfaceSimulador

- **Propósito**: Fornece interface de linha de comando para o usuário.
- **Responsabilidades**:
    - Configurar parâmetros da simulação.
    - Controlar execução: iniciar, pausar, retomar, encerrar, salvar log.
- **Métodos Principais**:
    - `iniciar`: Executa o loop principal de interação.
    - `configurarSimulador`: Coleta entradas do usuário para configurar a simulação.

- `mostrarMenuCompleto`: Exibe os comandos disponíveis.

7. Projeto de Algoritmos

7.1 Distribuição de Caminhões (DistribuicaoCaminhoes.distribuirCaminhoes)

- **Entrada**: Lista de caminhões pequenos (`Lista<CaminhaoPequeno>`), lista de zonas (`Lista<ZonaUrbana>`).
- **Saída**: Número de caminhões distribuídos.
- **Algoritmo**:
    1. Iterar sobre os caminhões pequenos.
    2. Para cada caminhão disponível (`estado == 1`), encontrar a melhor zona usando `encontrarMelhorZona`.
    3. Se nenhuma zona válida for encontrada, marcar o caminhão como ENCERRADO (`6`).
    4. Se a melhor zona for diferente da zona atual, calcular tempo de viagem (`calcularTempoViagem`) e definir estado como EM_TRANSITO (`3`).
    5. Se já estiver na zona ideal, definir estado como COLETANDO (`2`).
    6. Retornar o número de caminhões redistribuídos.

7.2 Pontuação das Zonas (DistribuicaoCaminhoes.calcularPontuacao)

- **Entrada**: Zona (`ZonaUrbana`), zona atual (`ZonaUrbana`).
- **Saída**: Pontuação (`double`).
- **Algoritmo**:
    1. Obter lixo acumulado, tempo de viagem base, e caminhões ativos na zona.
    2. Recuperar estatísticas de geração de lixo da zona (`ZonaEstatistica`).

3. Calcular proporção de lixo restante: `lixo / lixoGerado`.
4. Calcular pontuação: `(lixo * 1.0) + (proporcaoRestante * 5000) - (tempoViagem * 50)`.
5. Aplicar penalidade por excesso de caminhões: `- (caminhoesAtivos - limiteCaminhoesPorZona) * 200`.
6. Retornar a pontuação.

7.3 Cálculo do Tempo de Viagem (DistribuicaoCaminhoes.calcularTempoViagem)

- **Entrada**: Zonas de origem e destino (`ZonaUrbana`).
- **Saída**: Tempo de viagem (int, em minutos).
- **Algoritmo**:
    1. Obter distância entre zonas (`ZonaUrbana.getDistancia`).
    2. Determinar velocidade com base no horário:
        - Pico: 20 km/h.
        - Fora do pico: 30 km/h.
    3. Calcular tempo base: `(distancia / velocidadeMedia) * 60`.
    4. Adicionar variação específica da zona (pico ou normal).
    5. Aplicar aleatoriedade de ±10%: `tempoAjustado * (1.0 + random(-0.1, 0.1))`.
    6. Arredondar o tempo, garantindo mínimo de 1 minuto (ou 0 se for a mesma zona).
    7. Retornar tempo calculado.

7.4 Processamento da Fila (EstacaoTransferencia.processarFila)

- **Entrada**: Tempo atual (`tempoAtual`).
- **Saída**: `ResultadoProcessamentoFila` (caminhão processado, tempo de espera, flag de processado).
- **Algoritmo**:
    1. Atualizar tempos de espera dos caminhões pequenos na fila.
    2. Redirecionar caminhões que excederem `esperaMaxPequenos`.
    3. Se um processo estiver em andamento (`tempoProcessamentoRestante > 0`), decrementar o contador e finalizar se concluir.
    4. Se a fila não estiver vazia, processar o primeiro caminhão pequeno:
        - Se houver caminhão grande com capacidade, descarregar diretamente.
        - Caso contrário, descarregar no armazenamento.
        - Definir `tempoProcessamentoRestante` como 5 minutos.
    5. Se não houver caminhões pequenos, mas houver lixo armazenado e caminhão grande disponível, iniciar transferência (7 minutos).
    6. Retornar o resultado do processamento.

7.5 Liberação de Caminhão Grande (EstacaoTransferencia.liberarCaminhaoGrandeSeNecessario)

- **Entrada**: Nenhuma.
- **Saída**: Caminhão grande liberado (`CaminhaoGrande`) ou `null`.
- **Algoritmo**:
    1. Verificar se há caminhão grande esperando com carga.
    2. Avaliar condições para liberação:

        - Caminhão está cheio.
        - Tolerância de espera foi excedida.
        - Não há lixo ou fila na estação.
    3. Se condições forem atendidas:
        - Definir origem e destino do caminhão.
        - Calcular tempo de viagem até o aterro.
        - Iniciar viagem para o aterro (`iniciarViagemParaAterro`).
    4. Limpar o estado de espera do caminhão na estação.
    5. Retornar o caminhão liberado.

8. Detalhes de Implementação

8.1 Linguagem de Programação

- **Java**: Escolhida por suas características de orientação a objetos e independência de plataforma.

8.2 Principais Funcionalidades da Implementação

- **Estruturas de Dados Personalizadas**:
  `Lista` e `Fila` são implementadas como estruturas de lista simplesmente encadeada e fila circular, respectivamente, para evitar o uso das coleções embutidas do Java.

- **Sistema de Logs**:
  `LoggerSimulacao` utiliza códigos de cor ANSI para representar diferentes tipos de eventos (por exemplo, verde para coleta, vermelho para erros) e suporta modos normal e debug.

- **Configurabilidade**:
  `InterfaceSimulador` permite ao usuário definir:
    - Quantidade de caminhões

    - Intervalos de geração de lixo
    - Tolerância de espera
    - Modo de logs
      Tudo via interface de linha de comando.

- **Relatórios Estatísticos**:
  `Estatisticas` gera relatórios contendo:
    - Quantidade de lixo coletado por zona
    - Total de lixo gerado
    - Tempos de espera
    - Utilização de caminhões grandes
      Os dados são apresentados com gráficos de barras em ASCII.

8.3 Tratamento de Erros

- **Entradas Inválidas do Usuário**:
  Como valores negativos, são tratados com mensagens de erro e solicitações de nova entrada na `InterfaceSimulador`.

- **Dados Nulos ou Inválidos de Zonas/Distâncias**:
  Acionam valores padrão (por exemplo, tempo de viagem de 15 minutos) para manter a execução da simulação.

- **Erros de Log**:
  Como valores inválidos de lixo, são registrados com mensagens de erro detalhadas.

10. Validação

10.1 Resultados Obtidos

- O sistema demonstrou estabilidade na simulação de até 100 ciclos diários, com parâmetros configuráveis para capacidade e número de caminhões.

- O controle das filas e os tempos de espera ficaram dentro dos limites estipulados, comprovando a eficiência da lógica de tolerância e disparo para caminhões grandes.

- Os logs gerados permitiram rastrear eventos em tempo real e detectar facilmente qualquer anomalia durante a simulação.

10.2 Limitações Identificadas

- A interface do simulador ainda é muito primiriva, idealmente uma interface em JavaFX ou Swing seria mais efetiva, porém não estava no escopo desse projeto.

11. Conclusão

O simulador de coleta de lixo modela com sucesso o sistema de gestão de resíduos da cidade de Teresina, utilizando estruturas de dados personalizadas para gerenciar caminhões, zonas e estações de transferência. Ele oferece parâmetros configuráveis, registro em tempo real e estatísticas detalhadas para analisar a eficiência operacional e determinar o número mínimo de caminhões grandes necessários. O design modular e os algoritmos eficientes garantem escalabilidade e facilidade de manutenção, contribuindo para estratégias sustentáveis de gestão de resíduos.