Documentação Técnica para o Sistema de Simulação de Coleta de Resíduos

1. Introdução

O Sistema de Simulação de Coleta de Resíduos é uma simulação de eventos discretos projetada para modelar a coleta de resíduos urbanos em uma cidade, Teresina, Brasil. O sistema simula caminhões pequenos coletando resíduos de zonas designadas, transferindo-os para estações e descarregando em caminhões grandes que são transportando resíduos para um aterro. A simulação ocorre por padrão em um período de 24 horas (1440 minutos), com parâmetros configuráveis para capacidades de caminhões, tempos de viagem e limites de filas. Este documento detalha a modelagem do sistema, as estruturas de dados utilizadas e os algoritmos implementados.

2. Modelagem do Sistema

2.1 Visão Geral

O sistema é construído usando um design orientado a objetos em Java, com classes representando entidades (zonas, caminhões, estações) e lógica de simulação. A simulação é conduzida pela classe central `Simulador`, que avança o tempo em passos de 60 minutos e coordena interações. A interface gráfica, implementada por meio de `JanelaPrincipal` e `PainelSimulacao`, oferece visualização e controles para o usuário.

2.2 Classes e Relacionamentos

- **Zona**: Representa uma zona da cidade (por exemplo, Sul, Norte) com atributos como `lixoAcumulado` (resíduos) e métodos para geração de resíduos (`gerarLixo`) e coleta (`coletarLixo`).
- **CaminhaoPequeno**: Modela caminhões pequenos com atributos como `cargaAtual`, `viagensRealizadas` e métodos para carregar (`carregar`), descarregar (`descarregar`) e mover (`moverPara`).
- **CaminhaoGrande**: Modela caminhões grandes com maior capacidade, usados para transporte ao aterro, com métodos semelhantes.
- **EstacaoTransferencia**: Gerencia a transferência de resíduos com filas para caminhões pequenos (`filaCaminhoesPequenos`) e grandes (`filaCaminhoesGrandes`), rastreando `lixoTransferido`.
- **Simulador**: Orquestra a simulação, gerenciando zonas, caminhões, estações e eventos. Usa `avancarSimulacao` para avançar o tempo e processar operações.
- **Configuracao**: Armazena parâmetros como `numCaminhoesPequenos` (10), `limiteEsperaPequeno` (30 minutos) e tempos de viagem.

- **PainelSimulacao**: Visualiza entidades (zonas como círculos verdes, caminhões como retângulos) usando Java Swing.
- **JanelaPrincipal**: Interface principal com botões para iniciar, pausar e salvar a simulação, além de um gerador de relatórios (`salvarSimulacao`).
- **Evento** : Representa um evento da simulação (por exemplo, coleta, transferência) com atributos como tipo (nome do evento), tempo (momento de ocorrência), quantidade (resíduos envolvidos) e referências a entidades (zona, trens, estação), além de métodos para configurar e acessar essas informações (setZona , getTipo , getTempoFormatado).
- **Pilha <t></t>** : Representa uma estrutura de dados genérica de pilha (LIFO) com atributos como topo (nó inicial) e tamanho, e métodos para adicionar elementos (empilhar), remover o topo (desempilhar) e verificar o estado (estaVazia , tamanho).
- **Lista <t></t>** : Representa uma lista duplamente encadeada genérica com atributos como inicio, fim e tamanho, e métodos para adicionar elementos (adicionar), remover elementos (removedorPrimeiro , removeUltimo , removedor) e acessar por índice (get).
- **Fila <t></t>** : Representa uma fila genérica (FIFO) com atributos como inicio, fim, tamanho e tempoEspera por nó, e métodos para adicionar elementos (enfileirar), remover o primeiro (desenfileirar), incrementar tempos de espera (incrementarEspera) e acessar por índice (getElemento).
- **Main** : Representa o ponto de entrada do programa com métodos para carregar configurações (Configuracao.carregarConfiguracao), inicializar o simulador (Simulador) e exibir a interface gráfica (JanelaPrincipal).
-

Relacionamentos:

- `Simulador` agrega `Zona[]`, `Fila<CaminhaoPequeno>` e `EstacaoTransferencia[]`.
- `EstacaoTransferencia` contém `Fila<CaminhaoPequeno>` e `Fila<CaminhaoGrande>`.
- `CaminhaoPequeno` e `CaminhaoGrande` interagem com `Zona` e `EstacaoTransferencia` por meio de métodos.

2.3 Fluxo de Simulação

A simulação começa com `Simulador.inicializarSimulacao`, criando zonas, caminhões e estações. O método `avancarSimulacao` é executado a cada segundo, avançando o tempo em 60 minutos:

1. Gerar resíduos nas zonas.
2. Atribuir caminhões pequenos a zonas, coletar resíduos ou enfileirar nas estações.
3. Processar filas das estações, descarregando caminhões pequenos e ativando caminhões grandes, se necessário.
4. Transportar resíduos para o aterro por meio de caminhões grandes.
5. Registrar eventos e atualizar a interface gráfica.

3. Estruturas de Dados

3.1 Fila (Fila)

- **Descrição**: Uma fila FIFO genérica usada para gerenciar caminhões.
- **Usos**:
  - `caminhoesPequenos` em `Simulador`: Armazena caminhões pequenos ativos.
  - `filaCaminhoesPequenos` em `EstacaoTransferencia`: Fila de caminhões pequenos esperando para descarregar.
  - `filaCaminhoesGrandes` em `EstacaoTransferencia`: Fila de caminhões grandes para transporte ao aterro.
- **Operações**: `enfileirar` (adicionar), `desenfileirar` (remover), `getElemento` (acessar por índice), `tamanho` (tamanho), `incrementarEspera` (incrementar tempo de espera).
- **Implementação**: Provavelmente uma `List<T>` (por exemplo, `ArrayList`) com semântica FIFO.

3.2 Lista (Lista)

- **Descrição**: Uma lista que armazena eventos da simulação (por exemplo, coletas, transferências).
- **Uso**: `historicoEventos` em `Simulador` registra eventos para relatórios.
- **Operações**: Adicionar, iterar, limpar.
- **Implementação**: Lista personalizada ou `ArrayList<Evento>`.

3.5 Pilha

- **Descrição: Uma pilha LIFO (Last In, First Out) genérica usada para gerenciar elementos em uma estrutura de dados empilhada.**
- **Não utilizada diretamente no sistema de simulação de coleta de resíduos, mas pode ser aplicada em extensões que requerem processamento LIFO, como rastreamento de estados temporários ou históricos de ações.**

Operações:

- **empilhar: Adiciona um elemento ao topo da pilha.**
- **desempilhar: Remove e retorna o elemento do topo.**
- **estaVazia: Verifica se a pilha está vazia.**
- **tamanho: Retorna o número de elementos na pilha.**

3.4 Arrays

- **Zona[] zonas**: Array fixo de 5 zonas em `Simulador`.
- **double[] lixoColetadoPorZona**: Rastreia resíduos coletados por zona em `Simulador`.
- **double[] lixoPorZona**: Rastreia resíduos coletados por cada caminhão pequeno por zona em `CaminhaoPequeno`.
- **double[] capacidadesCaminhoesPequenos**: Define capacidades possíveis para caminhões pequenos em `Configuracao`.

3.5 Escalares e Objetos

- **Escalares**: `lixoAcumulado` (resíduos da zona), `cargaAtual` (carga do caminhão), `tempoSimulacao` (tempo da simulação).
- **Objetos**: `Zona`, `CaminhaoPequeno`, `CaminhaoGrande`, `EstacaoTransferencia` encapsulam estado e comportamento.

4. Algoritmos

4.1 Ciclo de Simulação (avancarSimulacao)

- **Entrada**: `deltaTempo` (60 minutos).
- **Passos**:
    1. Incrementar `tempoSimulacao`.
    2. Chamar `Zona.gerarLixo` para cada zona (distribuição Gaussiana).
    3. Para cada caminhão pequeno:
        - Se sem destino, selecionar uma zona (`escolherZonaParaColeta`).
        - Se cheio ou `tempoEmRota >= tempoMaximoRota`, enfileirar na estação.
        - Calcular tempos de viagem (`calcularTempoViagem`).
    4. Processar filas das estações:
        - Descarregar caminhões pequenos.
        - Promover para caminhão grande se o tempo de espera ≥ `limiteEsperaPequeno` (30 minutos).
        - Processar caminhões grandes para o aterro.
    5. Ativar caminhões grandes se `lixoTransferido > 20` toneladas e fila quase cheia.
    6. Atualizar interface gráfica (`PainelSimulacao.atualizar`).
- **Complexidade**: O(N + M + S), onde N é o número de caminhões pequenos, M é o número de caminhões enfileirados e S é o número de estações.

4.2 Agendamento de Viagens (escolherZonaParaColeta)

- **Entrada**: `CaminhaoPequeno`.
- **Passos**:
    1. Iterar pelas zonas.

2. Selecionar a primeira zona com `lixoAcumulado > 0`.
3. Retornar null se nenhuma zona tiver resíduos.

- **Complexidade**: O(Z), onde Z é o número de zonas (5).

4.3 Cálculo de Tempo de Viagem (calcularTempoViagem)

- **Entrada**: Coordenadas (x1, y1, x2, y2).
- **Passos**:
  1. Calcular distância Euclidiana: `sqrt((x2 - x1)^2 + (y2 - y1)^2)`.
  2. Escalar por `distancia / 200`.
  3. Aplicar faixa de tempo com base no horário de pico (`tempoViagemPicoMin/Max` ou `tempoViagemForaPicoMin/Max`).
- **Complexidade**: O(1).

4.4 Processamento de Filas

- **Entrada**: `filaCaminhoesPequenos` para uma estação.
- **Passos**:
  1. Incrementar tempos de espera (`incrementarEspera`).
  2. Enquanto a fila não estiver vazia:
     - Se o tempo de espera do caminhão líder ≥ `limiteEsperaPequeno`, desenfileirar e ativar caminhão grande.
     - Caso contrário, desenfileirar, descarregar e reiniciar o caminhão.
- **Complexidade**: O(M), onde M é o número de caminhões enfileirados.

4.5 Geração de Resíduos (Zona.gerarLixo)

- **Entrada**: `taxaGeracao`.
- **Passos**:
  1. Adicionar resíduos aleatórios Gaussianos: `lixoAcumulado += nextGaussian() * taxaGeracao / 2 + taxaGeracao`.
  2. Garantir `lixoAcumulado >= 0`.
- **Complexidade**: O(1).

4.6 Relatórios (salvarSimulacao)

- **Entrada**: Estado da simulação.
- **Passos**:
  1. Escrever resíduos totais, estatísticas de caminhões e dados de zonas em `simulacao.txt`.
  2. Incluir resíduos acumulados por zona.
  3. Registrar eventos de `historicoEventos`.
- **Complexidade**: O(N + E), onde N é o número de caminhões e E é o número de eventos.

5. Detalhes de Implementação

5.1 Parâmetros de Configuração

- `numCaminhoesPequenos =30` , `numCaminhoesGrandesInicial = 2`.
- `limiteEsperaPequeno = 60 minutos`, `limiteFilaEstacao = 10`.
- `capacidadeCaminhaoGrande = 20 toneladas`, capacidades de caminhões pequenos: 2,4,8,10 toneladas.
- Tempos de viagem: 20–50 minutos (pico), 10–20 minutos (fora de pico).
- `tempoMaximoRota = 180 minutos`, `limiteViagensMin/Max = 5–7`.

5.2 Integração com Interface Gráfica

- `PainelSimulacao` usa Java Swing para desenhar zonas (círculos verdes), caminhões pequenos (retângulos azuis), caminhões grandes (retângulos laranjas) e estações (quadrados vermelhos).
- `JanelaPrincipal` oferece botões para iniciar, pausar e salvar a simulação, com um painel de controle e log de eventos.

6. Conclusão

O Sistema de Simulação de Coleta de Resíduos é um modelo robusto e extensível de gerenciamento de resíduos urbanos, utilizando design orientado a objetos, estruturas de dados eficientes (filas, arrays) e algoritmos bem definidos. O sistema suporta configurabilidade e visualização, tornando-o adequado para análise de estratégias de coleta de resíduos. Melhorias futuras podem incluir simulações de vários dias, roteamento dinâmico de caminhões ou ajustes de parâmetros em tempo real.

7. Referências

- Documentação do Java Swing para implementação da interface gráfica.
- Conceitos de Simulação de Eventos Discretos para o design do sistema.
-