

# Simulador de Coleta de Lixo

## Características Práticas do Código



# Objetivo do Sistema

## Descrição do Simulador

O simulador representa uma abordagem prática para a gestão de resíduos, permitindo a criação de rotas eficientes, cálculos de coleta e a análise do fluxo de resíduos através de áreas urbanas.



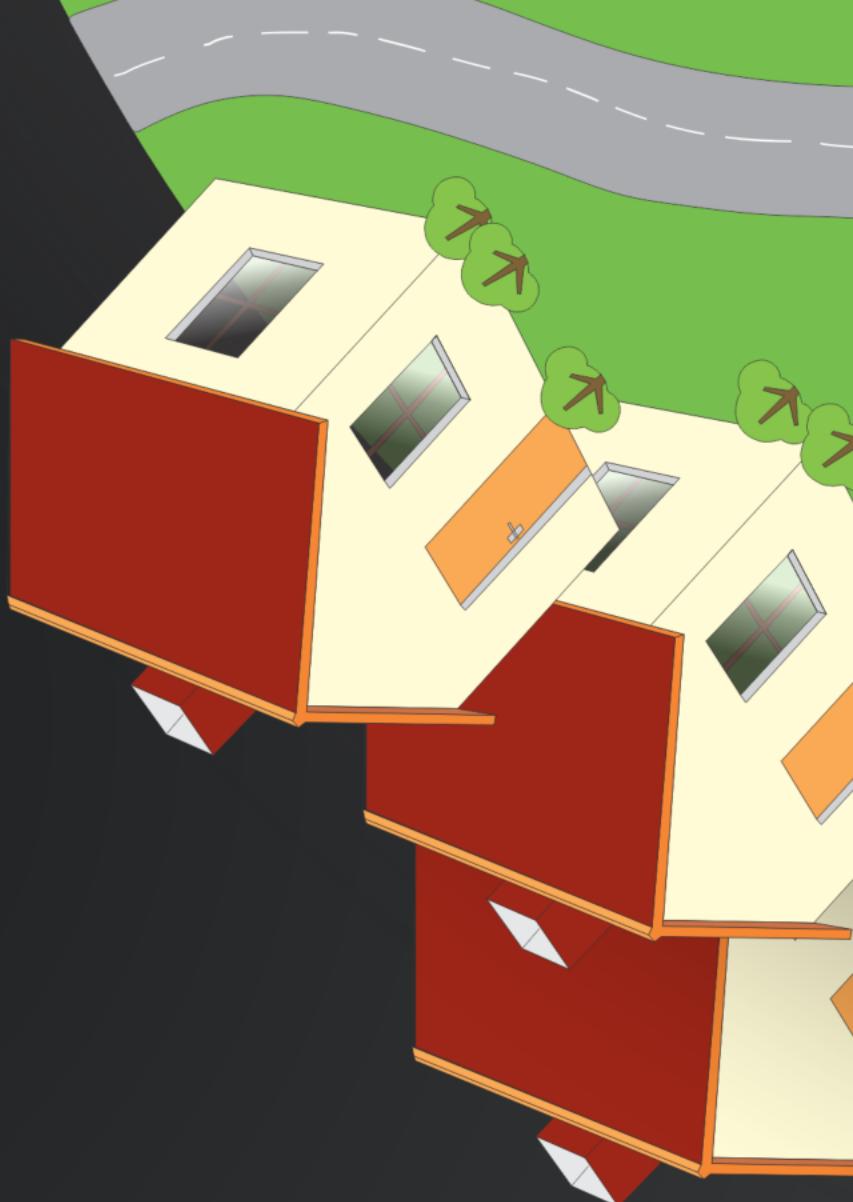
## Função de coleta e transporte

O sistema simula a coleta de resíduos de várias zonas, utilizando caminhões de diferentes tamanhos, assegurando um planejamento eficiente das rotas e estações de transferência, refletindo a realidade da logística de coleta urbana.



## Representação do fluxo de coleta

A simulação abrange o fluxo completo, desde as zonas de coleta até suas áreas de descarte.



# Descrição do Simulador

O simulador representa uma abordagem prática para a gestão de resíduos, permitindo a simulação de diferentes cenários de coleta e a análise do fluxo de resíduos através de áreas urbanas.



# Função de coleta e transporte

O sistema simula a coleta de resíduos de várias zonas, utilizando caminhões de diferentes tamanhos, assegurando um transporte eficiente até as estações de transferência, refletindo a realidade da logística de coleta urbana.



# Representação do fluxo de coleta

A simulação abrange o fluxo completo, desde as zonas de coleta até suas áreas de descarte.



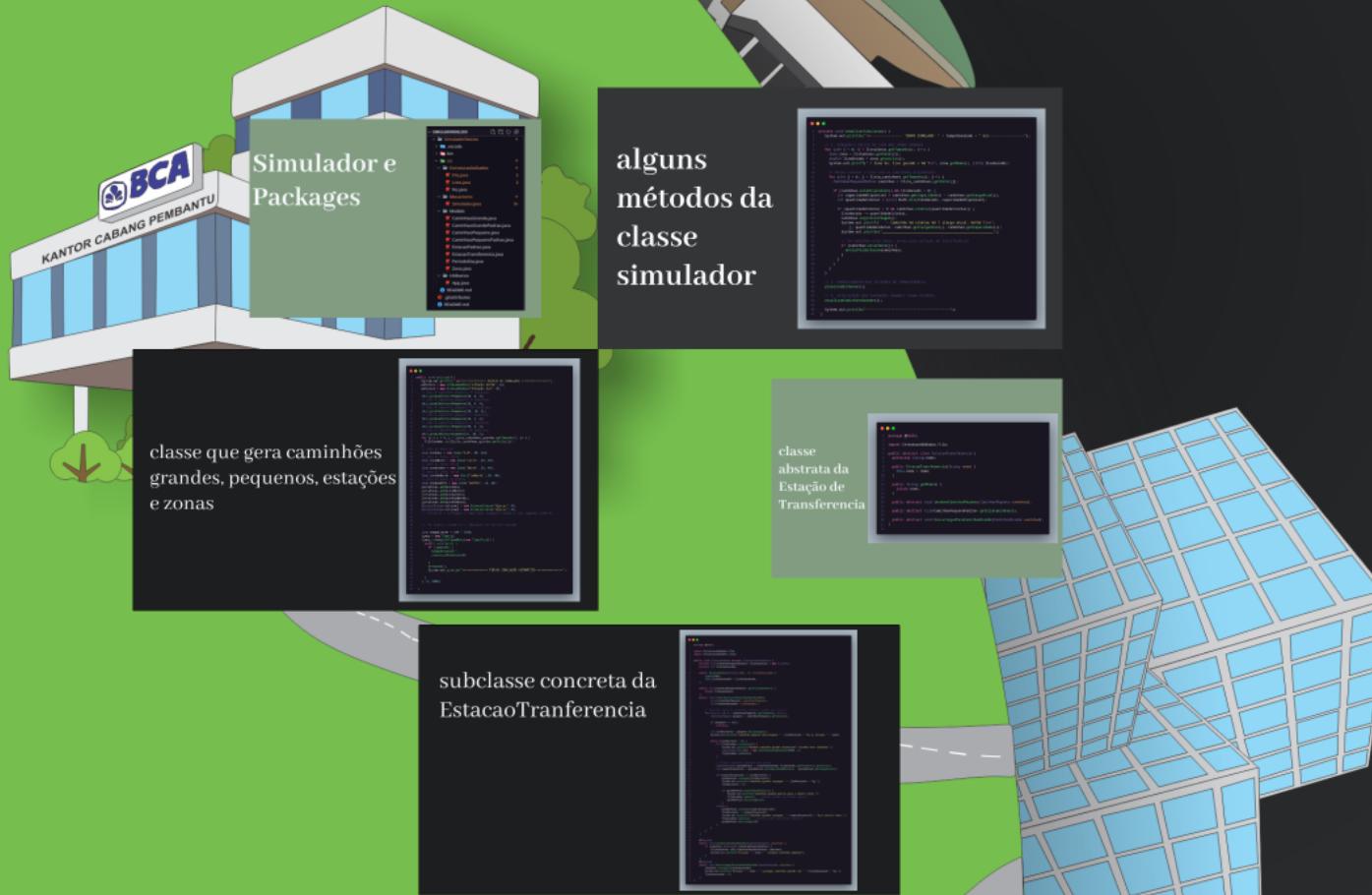
# Simulador de Coleta de Lixo

## Características Práticas do Código

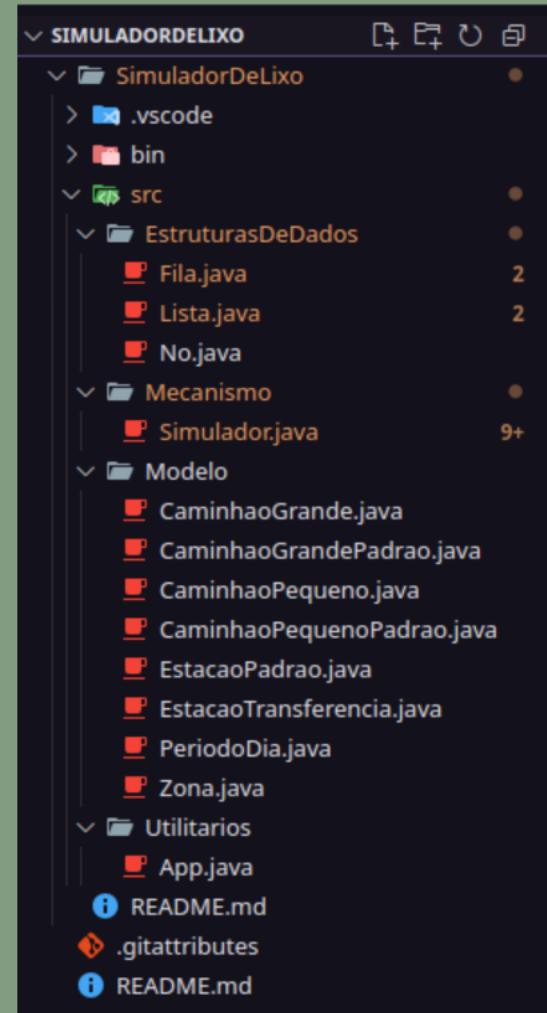


# Cara

## Estrutura do Código



# Simulador e Packages



```
SIMULADORDELIXO
├── SimuladorDeLixo
│   ├── .vscode
│   ├── bin
│   └── src
│       ├── EstruturasDeDados
│       │   ├── Fila.java
│       │   ├── Lista.java
│       │   └── No.java
│       ├── Mecanismo
│       │   └── Simulador.java
│       └── Modelo
│           ├── CaminhaoGrande.java
│           ├── CaminhaoGrandePadrao.java
│           ├── CaminhaoPequeno.java
│           ├── CaminhaoPequenoPadrao.java
│           ├── EstacaoPadrao.java
│           ├── EstacaoTransferencia.java
│           ├── PeriodoDia.java
│           └── Zona.java
└── Utilitarios
    ├── App.java
    ├── README.md
    ├── .gitattributes
    └── README.md
```

# alguns métodos da classe simulador

```
1  private void atualizarSimulacao() {
2      System.out.println("\n----- TEMPO SIMULADO: " + tempoSimulado + " min -----");
3
4      // 1. GERAÇÃO E COLETA DE LIXO NAS ZONAS URBANAS
5      for (int i = 0; i < listaZonas.getTamanho(); i++) {
6          Zona zona = listaZonas.getValor(i);
7          double lixoGerado = zona.gerarLixo();
8          System.out.printf(" > Zona %s: lixo gerado = %d T\n", zona.getNome(), (int) lixoGerado);
9
10     // Tenta coletar o lixo com os caminhões disponíveis
11     for (int j = 0; j < lista_caminhoes.getTamanho(); j++) {
12         CaminhaoPequenoPadrao caminhao = lista_caminhoes.getValor(j);
13
14         if (caminhao.estaDisponivel() && lixoGerado > 0) {
15             int capacidadeDisponivel = caminhao.getCapacidade() - caminhao.getCargaAtual();
16             int quantidadeColetar = (int) Math.min(lixoGerado, capacidadeDisponivel);
17
18             if (quantidadeColetar > 0 && caminhao.coletar(quantidadeColetar)) {
19                 lixoGerado -= quantidadeColetar;
20                 caminhao.registrarViagem();
21                 System.out.printf(" - Caminhão %d coletou %d T (Carga atual: %d/%d T)\n",
22                                 j, quantidadeColetar, caminhao.getCargaAtual(), caminhao.getCapacidade());
23                 System.out.println("-----");
24
25             // Se caminhão está cheio, envia para estação de transferência
26             if (caminhao.estaCheio()) {
27                 enviarParaEstacao(caminhao);
28             }
29         }
30     }
31 }
32
33 // 2. PROCESSAMENTO DAS ESTAÇÕES DE TRANSFERÊNCIA
34 processarEstacoes();
35
36 // 3. ATUALIZAÇÃO DOS CAMINHÕES GRANDES (PARA ATERROS)
37 atualizarCaminhoesGrandes();
38
39 System.out.println("-----");
40 }
41 }
```

# classe que gera caminhões grandes, pequenos, estações e zonas

```
● ○ ●  
1 public void iniciar() {  
2     System.out.println("\n===== INÍCIO DA SIMULAÇÃO =====");  
3     estacao1 = new EstacaoPadrao("Estação Norte", 0);  
4     estacao2 = new EstacaoPadrao("Estação Sul", 0);  
5     // Gera 4 caminhões pequenos 8 toneladas  
6     this.geraCaminhosPequenos(10, 8, 4);  
7     // Gera 4 caminhões pequenos 4 toneladas  
8     this.geraCaminhosPequenos(10, 4, 4);  
9     // Gera 4 caminhões pequenos 10 toneladas  
10    this.geraCaminhosPequenos(10, 10, 4);  
11    // Gera 4 caminhões pequenos 2 toneladas  
12    this.geraCaminhosPequenos(10, 2, 4);  
13    // Gera 2 caminhões pequenos 2 toneladas  
14    this.geraCaminhosPequenos(10, 2, 4);  
15    // Gera 4 caminhões grandes 20 toneladas  
16    this.geraCaminhosGrandes(10, 20, 5);  
17    for (int i = 0; i < lista_caminhos_grandes.getTamanho(); i++) {  
18        filaGrandes.add(lista_caminhos_grandes.getValor(i));  
19    }  
20    // gera as zona sul  
21    Zona zonaSul = new Zona("Sul", 20, 40);  
22    // gera as zonas norte  
23    Zona zonaNorte = new Zona("norte", 20, 40);  
24    // gera as zonas leste  
25    Zona zonaLeste = new Zona("leste", 20, 40);  
26    // gera as zonas oeste  
27    Zona zonaSudeste = new Zona("sudeste", 20, 40);  
28    // gera as zonas do direceu  
29    Zona zonaCentro = new Zona("centro", 30, 60);  
30    listaZonas.add(zonaSul);  
31    listaZonas.add(zonaNorte);  
32    listaZonas.add(zonaLeste);  
33    listaZonas.add(zonaSudeste);  
34    listaZonas.add(zonaCentro);  
35    EstacaoPadrao estacao1 = new EstacaoPadrao("direceu", 0);  
36    EstacaoPadrao estacao2 = new EstacaoPadrao("direceu", 0);  
37    // Instancia e configura o timer para avançar o tempo a cada segundo (1000 ms)  
38  
39    // Só avança o tempo se a simulação não estiver pausada  
40    long tempolimite = 100 * 1000;  
41    timer = new Timer();  
42    timer.scheduleAtFixedRate(new TimerTask() {  
43        public void run() {  
44            if (!pausado) {  
45                tempoSimulado++;  
46                atualizarSimulacao();  
47            }  
48        }  
49    }, 0, 1000);  
50 }
```

# classe abstrata da Estação de Transferencia

```
 1 package Modelo;
 2
 3 import EstruturasDeDados.Fila;
 4
 5 public abstract class EstacaoTransferencia {
 6     protected String nome;
 7
 8     public EstacaoTransferencia(String nome) {
 9         this.nome = nome;
10     }
11
12     public String getNome() {
13         return nome;
14     }
15
16     public abstract void receberCaminhaoPequeno(CaminhaoPequeno caminhao);
17
18     public abstract Fila<CaminhaoPequenoPadrao> getFilaCaminhoes();
19
20     public abstract void descarregarParaCaminhaoGrande(CaminhaoGrande caminhao);
21 }
22 }
```

# subclasse concreta da EstacaoTransferencia

```
● ● ●
1 package Modelo;
2
3 import EstruturasDeDados.Fila;
4 import EstruturasDeDados.Lista;
5
6 public class EstacaoPadrao extends EstacaoTransferencia {
7     private Fila<CaminhaoPequenoPadrao> filaCaminhoes = new Fila<>();
8     private int lixoArmazenado;
9
10    public EstacaoPadrao(String nome, int lixoArmazenado) {
11        super(nome);
12        this.lixoArmazenado = lixoArmazenado;
13    }
14
15    public Fila<CaminhaoPequenoPadrao> getFilaCaminhoes() {
16        return filaCaminhoes;
17    }
18    public void transferirLixoParaCaminhosGrandes(
19        Lista<CaminhaoPequeno> caminhosPequenos,
20        Fila<CaminhaoGrande> filaGrandes) {
21
22        // Percorre todos os caminhões pequenos usando sua Lista<>
23        for (int i = 0; i < caminhosPequenos.getTamanho(); i++) {
24            CaminhaoPequeno pequeno = caminhosPequenos.getValor(i);
25
26            if (pequeno == null)
27                continue;
28
29            int lixoRestante = pequeno.descarregar();
30            System.out.println("Caminhão pequeno descarregou " + lixoRestante + "kg na estação " + nome);
31
32            while (lixoRestante > 0) {
33                if (filaGrandes.estaVazia()) {
34                    System.out.println("Nenhum caminhão grande disponível! Criando novo caminhão.");
35                    CaminhaoGrande novo = new CaminhaoGrandePadrao(20000, 1);
36                    filaGrandes.add(novo);
37                }
38
39                // Obtém o primeiro caminhão sem remover
40                CaminhaoGrande grandeAtual = (CaminhaoGrande) filaGrandes.getPrimeiro().getValor();
41                int espacoDisponivel = grandeAtual.getCapacidadeMaxima() - grandeAtual.getCargaAtual();
42
43                if (espacoDisponivel > lixoRestante) {
44                    grandeAtual.carregar(lixoRestante);
45                    System.out.println("Caminhão grande carregou " + lixoRestante + "kg.");
46                    lixoRestante = 0;
47
48                    if (grandeAtual.prontoParaPartir()) {
49                        System.out.println("Caminhão grande partiu para o aterro cheio.");
50                        filaGrandes.remove(); // Remove usando seu método remove()
51                        grandeAtual.descarregar(0);
52                    }
53                } else {
54                    grandeAtual.carregar(espacoDisponivel);
55                    lixoRestante -= espacoDisponivel;
56                    System.out.println("Caminhão grande carregou " + espacoDisponivel + "kg e partiu cheio.");
57                    filaGrandes.remove(); // Remove usando seu método remove()
58                    grandeAtual.descarregar(0);
59                }
60            }
61        }
62    }
63
64    @Override
65    public void receberCaminhaoPequeno(CaminhaoPequeno caminhao) {
66        if (caminhao instanceof CaminhaoPequenoPadrao) {
67            filaCaminhoes.add((CaminhaoPequenoPadrao) caminhao);
68            System.out.println("Estação " + nome + " recebeu caminhão pequeno");
69        }
70    }
71    @Override
72    public void descarregarParaCaminhaoGrande(CaminhaoGrande caminhao) {
73        caminhao.carregar(lixoArmazenado);
74        System.out.println("Estação " + nome + " carregou caminhão grande com " + lixoArmazenado + "kg.");
75        lixoArmazenado = 0;
76    }
77 }
```

# Simulador de Coleta de Lixo

## Características Práticas do Código



A screenshot of the Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and several icons for file operations. The title bar shows the project name "SimuladorDeLixo-main".

The Explorer sidebar on the left lists the project structure:

- OPEN EDITORS: App.java
- SIMULADORDELIXO-MAIN:
  - SimuladorDeLixo
  - .vscode
  - bin
  - src
    - EstruturasDeDados
    - Mecanismo
    - Modelo
    - utilitarios
      - App.java
  - README.md
  - .gitattributes
  - README.md

The main editor area displays the `App.java` file:

```
package Utilitarios;
import Mecanismo.Simulador;

public class App {
    public static void main(String[] args) throws Exception {
        Simulador simulador = new Simulador();
        simulador.iniciar();
    }
}
```

The bottom right corner shows the terminal output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Run App + × ⌂ ⌄ ⌁ ⌅ ⌆ ⌇ ×

Caminhão pequeno enviado para Estação Norte
Caminhão pequeno coletou 4T. Carga atual: 4
- Caminhão 10 coletou 4 T (Carga atual: 4/4 T)
Estação Estação Norte recebeu caminhão pequeno
Caminhão pequeno enviado para Estação Norte
Caminhão pequeno coletou 4T. Carga atual: 4
- Caminhão 11 coletou 4 T (Carga atual: 4/4 T)
Estação Estação Norte recebeu caminhão pequeno
Caminhão pequeno enviado para Estação Norte
Caminhão pequeno coletou 4T. Carga atual: 4
```

The bottom status bar indicates: Line 1, Col 1 | Spaces: 4 | UTF-8 | LF | Java | Go Live | Prettier.