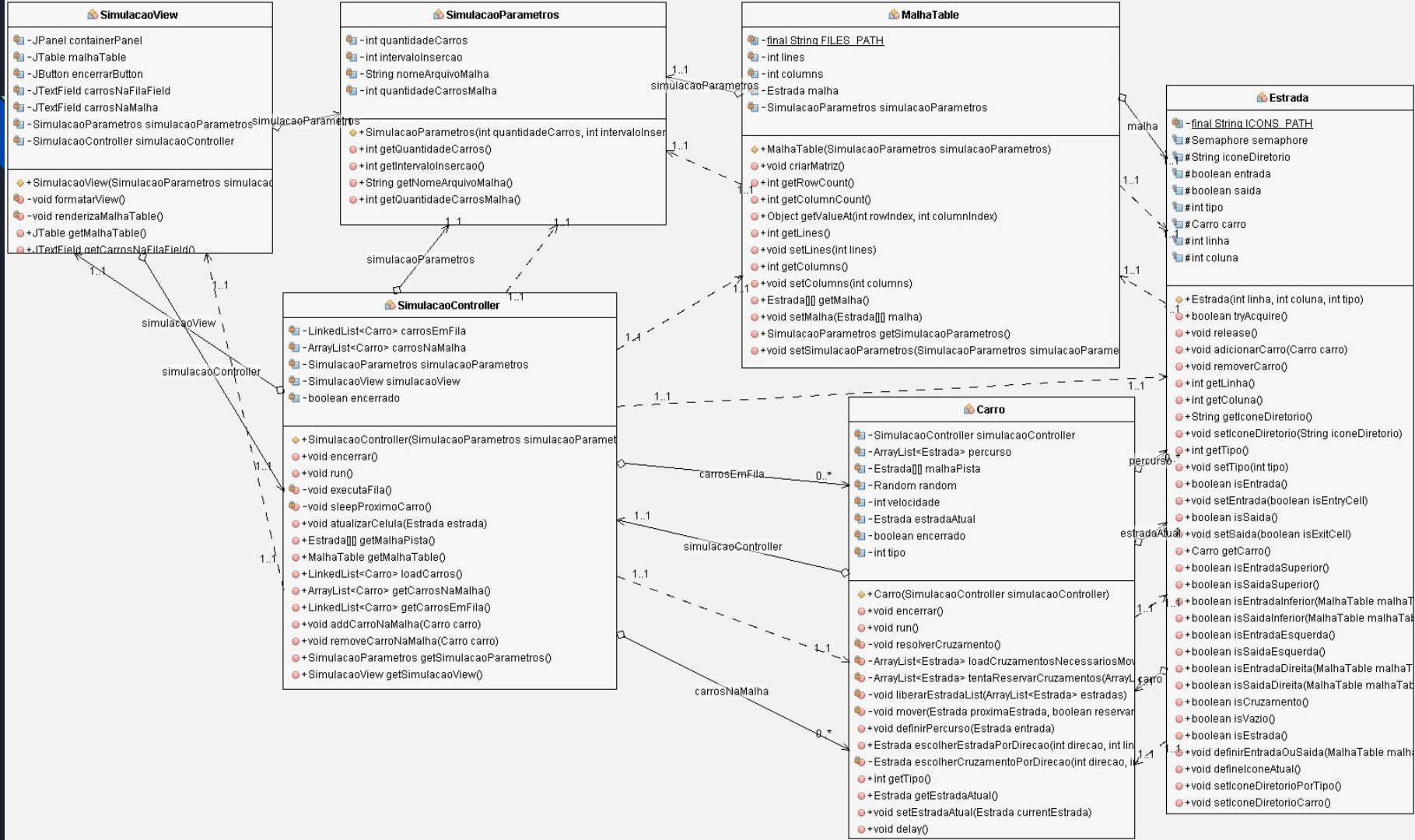




Trabalho 2 DSD

Estudantes: André Cristen e Lucas Levi Gonçalves





Bem-vindo(a) ao Simulador de Tráfego em Malha Viária



Intervalo de inserção de veículos (s):

0

Quantidade de carros:

1000

Quantidade máxima de carros simultâneos:

100

☐ Malha 1

☒ Malha 2

☐ Malha 3

Simular

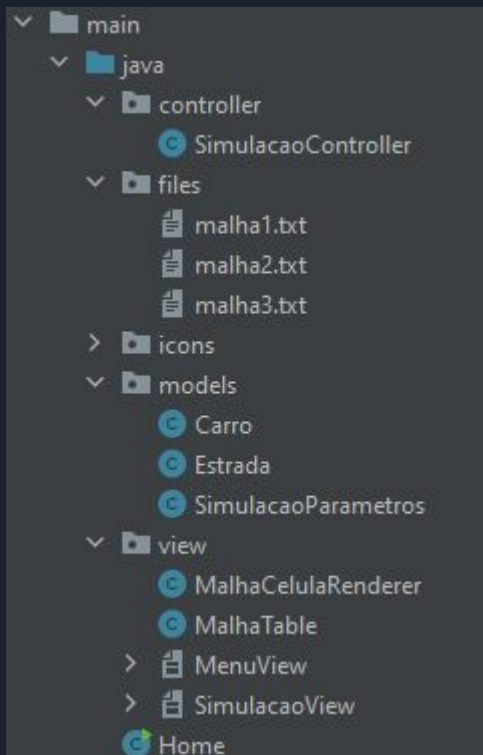


Carros a serem adicionados: 909

Carros rodando: 59

Encerrar


Estrutura do projeto



Carro.java

definirPercurso()

```
public void definirPercurso(Estrada entrada) throws Exception {
    boolean saidaEncontrada = false;
    Estrada proximaEstrada = entrada;
    percurso.add(proximaEstrada);
    //Controla os cruzamentos encontrados para não ocorrer de andar em círculos
    int cruzamentosEncontrados = 0;
    while (!saidaEncontrada) {
        int direcao = proximaEstrada.getTipo();
        boolean estradaDirecaoUnica = direcao <= 4;
        if (estradaDirecaoUnica) {
            proximaEstrada = this.escolherEstradaPorDirecao(direcao, proximaEstrada.getLinha(), proximaEstrada.getColuna());
        } else {
            proximaEstrada = this.escolherCruzamentoPorDirecao(direcao, proximaEstrada.getLinha(), proximaEstrada.getColuna(), cruzamentosEncontrados);
            if (proximaEstrada.isCruzamento()) {
                cruzamentosEncontrados++;
            } else {
                cruzamentosEncontrados = 0;
            }
        }
        percurso.add(proximaEstrada);
        saidaEncontrada = proximaEstrada.isSaida();
    }
}
```



Carro.java

run()

```
@Override
public void run() {
    while (!this.encerrado) {
        while (!percurso.isEmpty()) {
            int proximaExtradaIndex = 0;
            if (percurso.get(proximaExtradaIndex).isCruzamento()) {
                //Se for um cruzamento precisamos saber para que lado ir
                resolverCruzamento();
            } else {
                //Se for só ma estrada apenas move o veículo
                Estrada estrada = this.percurso.remove(proximaExtradaIndex);
                this.mover(estrada, reservar: true);
            }
        }
        //Chegou ao fim do percurso?
        // - Remove o carro
        this.getEstradaAtual().removerCarro();
        // - Libera a estrada
        this.getEstradaAtual().release();
        // - Tira da malha pois saiu da tela
        this.simulacaoController.removeCarroNaMalha(this);
        // - Atualização gráfica
        this.simulacaoController.atualizarCelula(this.getEstradaAtual());
        // - Fim da thread
        this.encerrar();
    }
}
```




Carro.java resolverCruzamento()

```
private void resolverCruzamento() {  
    this.delay();  
    ArrayList<Estrada> cruzamentosReservar = this.loadCruzamentosNecessariosMovimento();  
    ArrayList<Estrada> cruzamentosReservados = this.tentaReservarCruzamentos(cruzamentosReservar);  
    //Tem todos os cruzamentos para passar?  
    if (cruzamentosReservados.size() == cruzamentosReservar.size()) {  
        //Move pelo cruzamento  
        for (Estrada cruzamentoReservado : cruzamentosReservados) {  
            this.percurso.remove(cruzamentoReservado);  
            this.mover(cruzamentoReservado, reservar: false);  
        }  
    }  
}
```




Carro.java

resolverCruzamento()

```
private ArrayList<Estrada> tentaReservarCruzamentos(ArrayList<Estrada> cruzamentosReservar) {  
    //Tenta reservar todos os cruzamentos necessários  
    ArrayList<Estrada> cruzamentosReservados = new ArrayList<>();  
    for (Estrada cruzamentoTentaReservar : cruzamentosReservar) {  
        if (cruzamentoTentaReservar.tryAcquire()) {  
            cruzamentosReservados.add(cruzamentoTentaReservar);  
        } else {  
            //Não conseguiu reservar todos os cruzamentos para passar?  
            //Vamos liberar os que tínhamos conseguido reservar  
            this.liberarEstradaList(cruzamentosReservados);  
            break;  
        }  
    }  
    return cruzamentosReservados;  
}
```



Carro.java

loadCruzamentosNecessariosMovimento()

```
private ArrayList<Estrada> loadCruzamentosNecessariosMovimento() {  
    ArrayList<Estrada> cruzamentosReservar = new ArrayList<>();  
    for (int i = 0; i < this.percurso.size(); i++) {  
        Estrada estrada = this.percurso.get(i);  
        cruzamentosReservar.add(estrada);  
        if (!estrada.isCruzamento()) {  
            break;  
        }  
    }  
    return cruzamentosReservar;  
}
```



Carro.java mover()

```
private void mover(Estrada proximaEstrada, boolean reservar) {  
    if (proximaEstrada.isVazio()) {  
        boolean reservado = false;  
        if (reservar) {  
            do {  
                //Tenta "reservar/adquirir" a estrada  
                if (proximaEstrada.tryAcquire()) {  
                    reservado = true;  
                }  
            } while (!reservado);  
        }  
        //Somente quando conseguiu a estrada, adiciona o carro na posição  
        proximaEstrada.adicionarCarro(this);  
        Estrada estradaAnterior = this.getEstradaAtual();  
        if (estradaAnterior != null) {  
            //Tira o carro da estrada que ele estava  
            estradaAnterior.removerCarro();  
            //Libera a estrada  
            estradaAnterior.release();  
        }  
        //Diz em qual estrada o carro está agora  
        this.setEstradaAtual(proximaEstrada);  
        //Atualização gráfica  
        this.simulacaoController.atualizarCelula(proximaEstrada);  
        this.delay();  
    }  
}
```



Trabalho 2 DSD

Estudantes: André Cristen e Lucas Levi Gonçalves