



Simulador de Tráfego em Malha Viária

Danton Kriek Mohr
Ediane Lara Loch





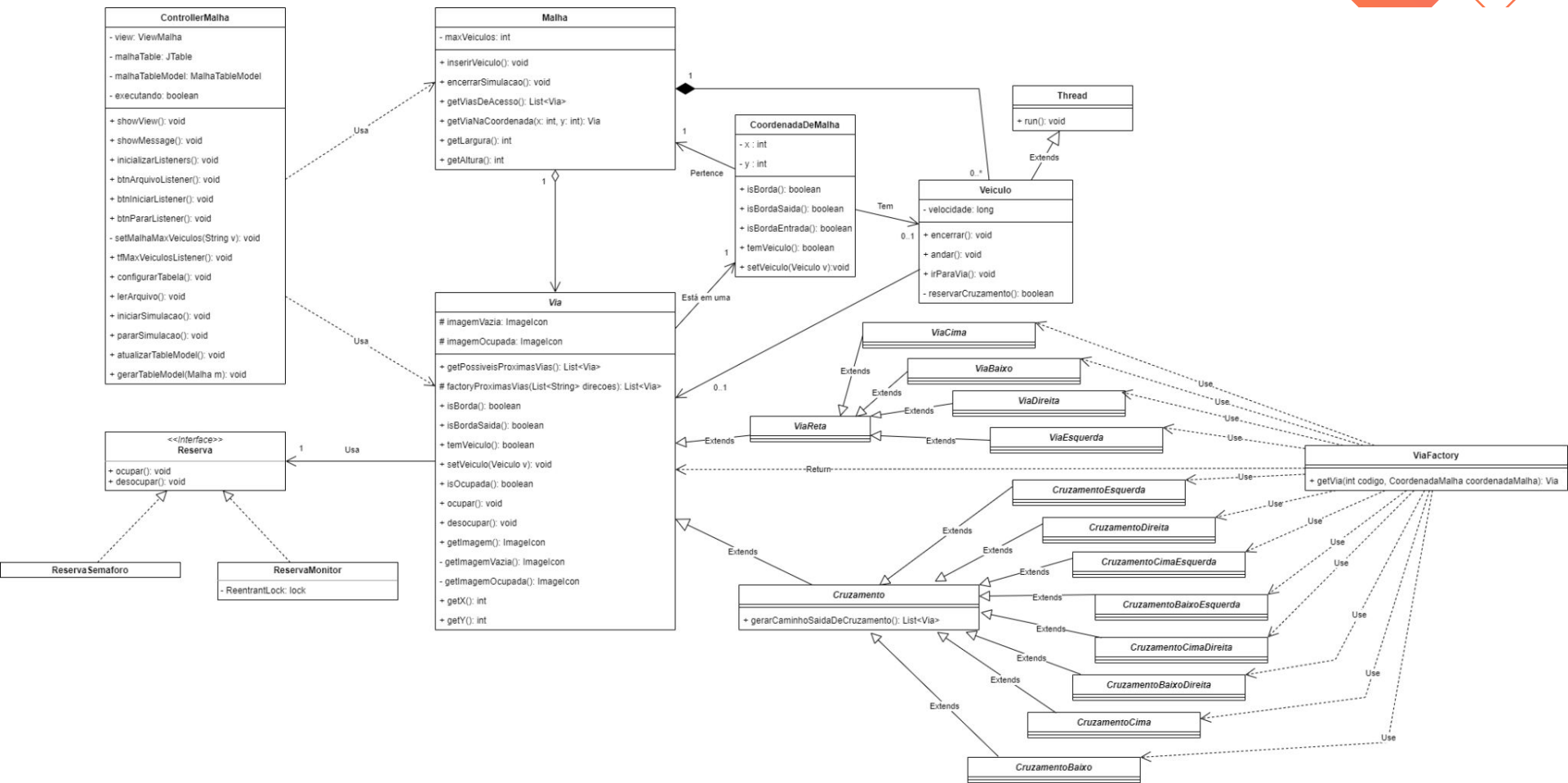
Roteiro

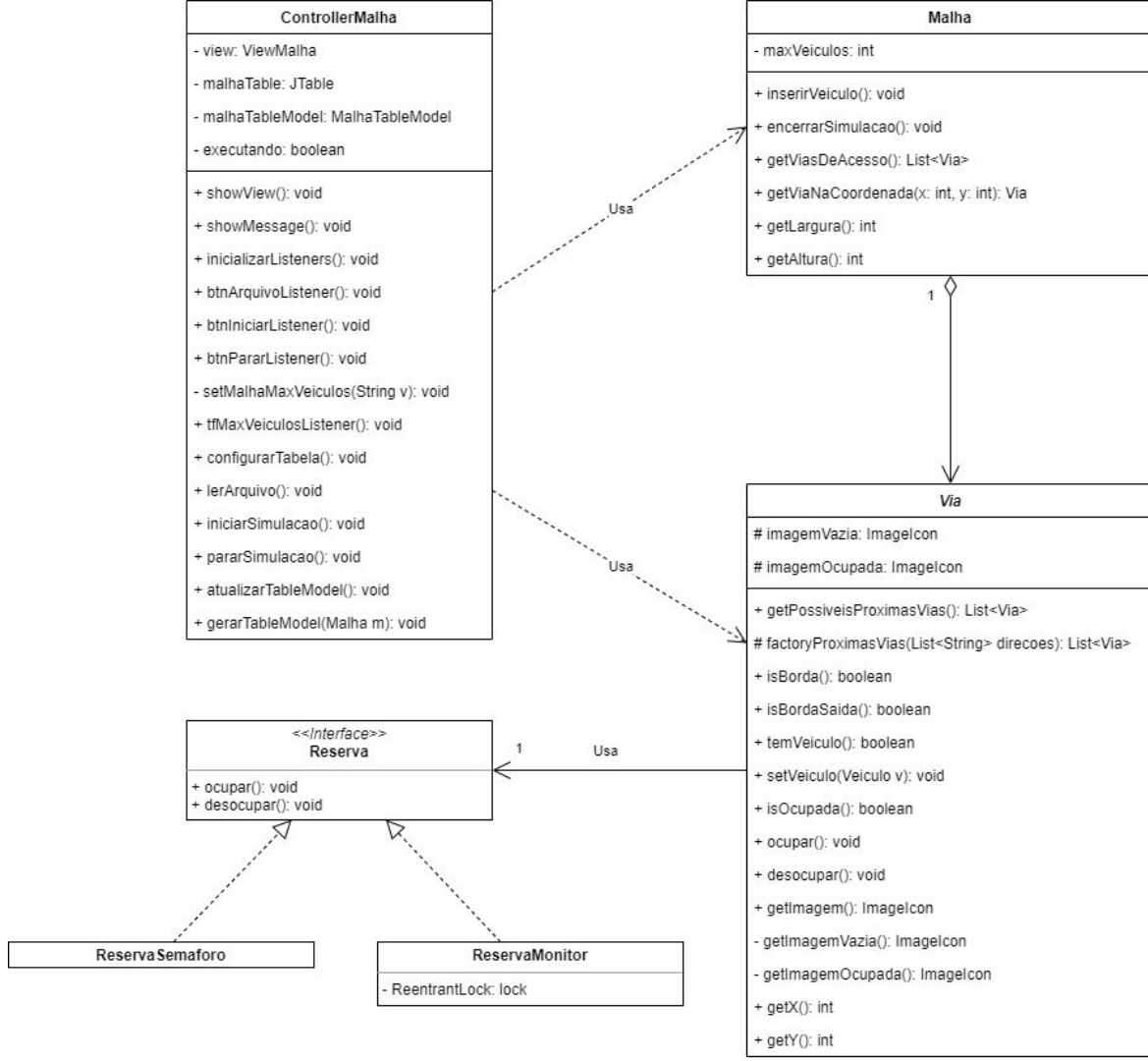
- Padrões utilizados
- Diagrama
- Estrutura
- Implementação
- Demonstração do sistema
- Dificuldades e soluções

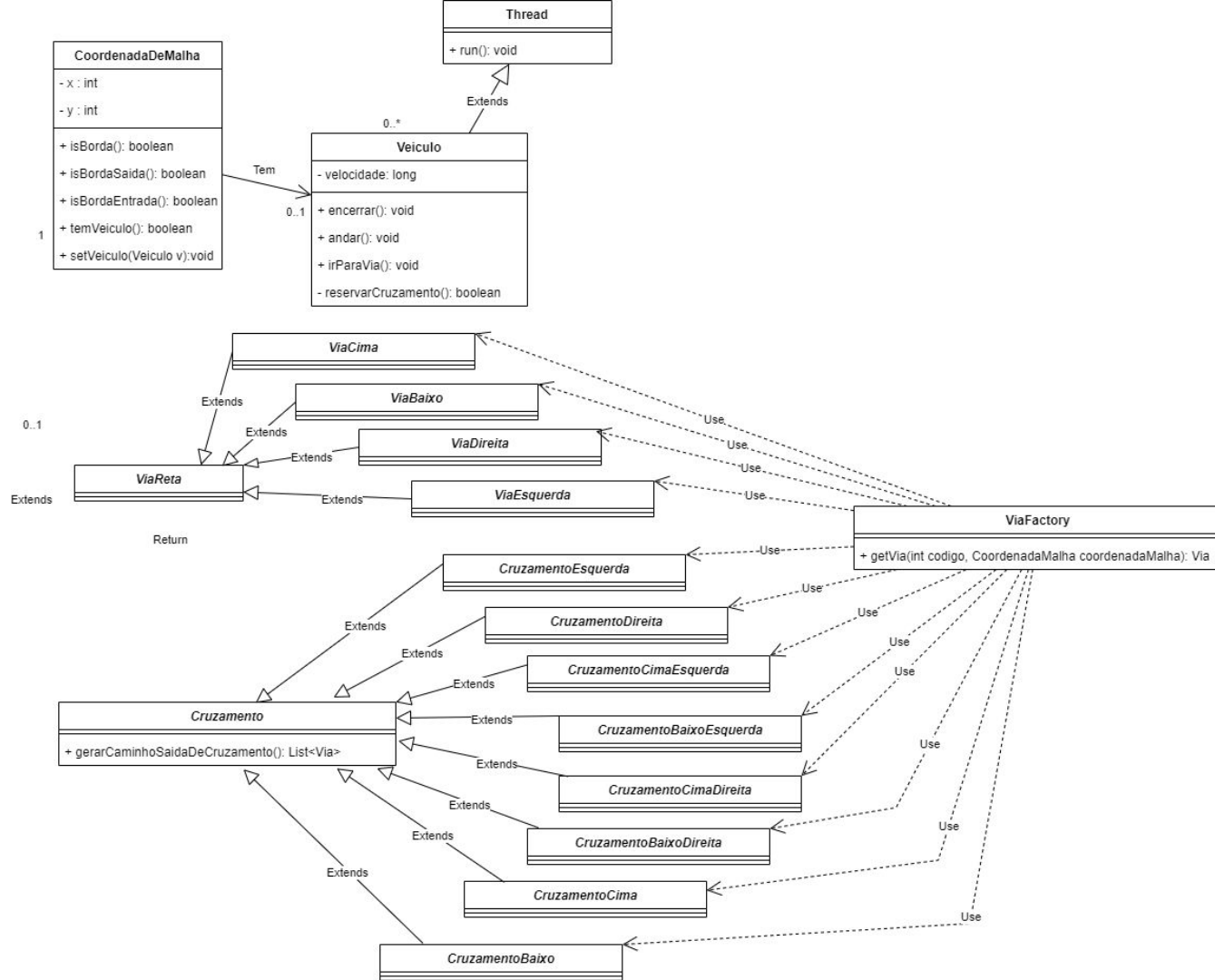
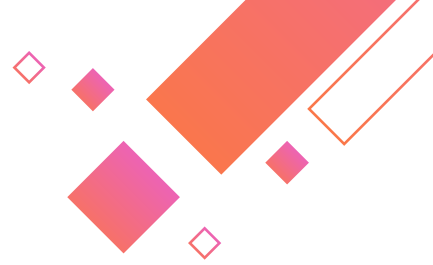
Padrões utilizados

- MVC
- Factory Method
- Abstract Factory

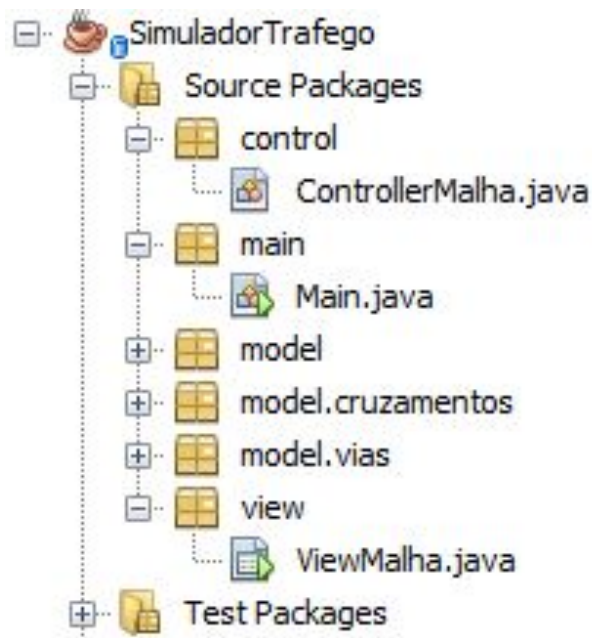
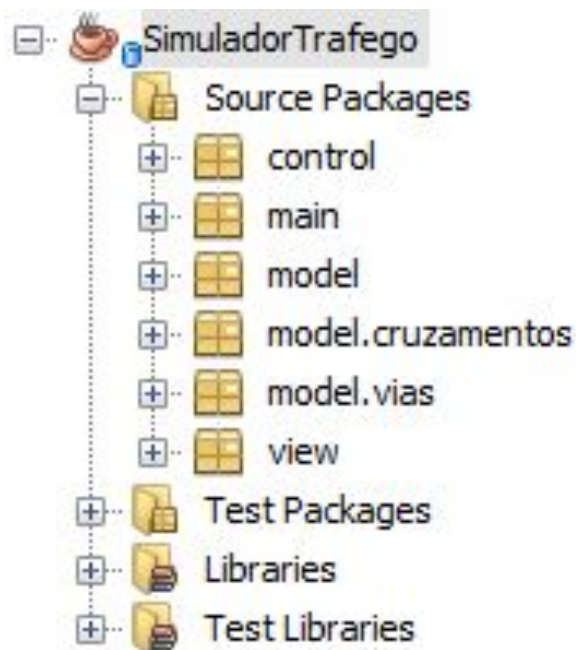




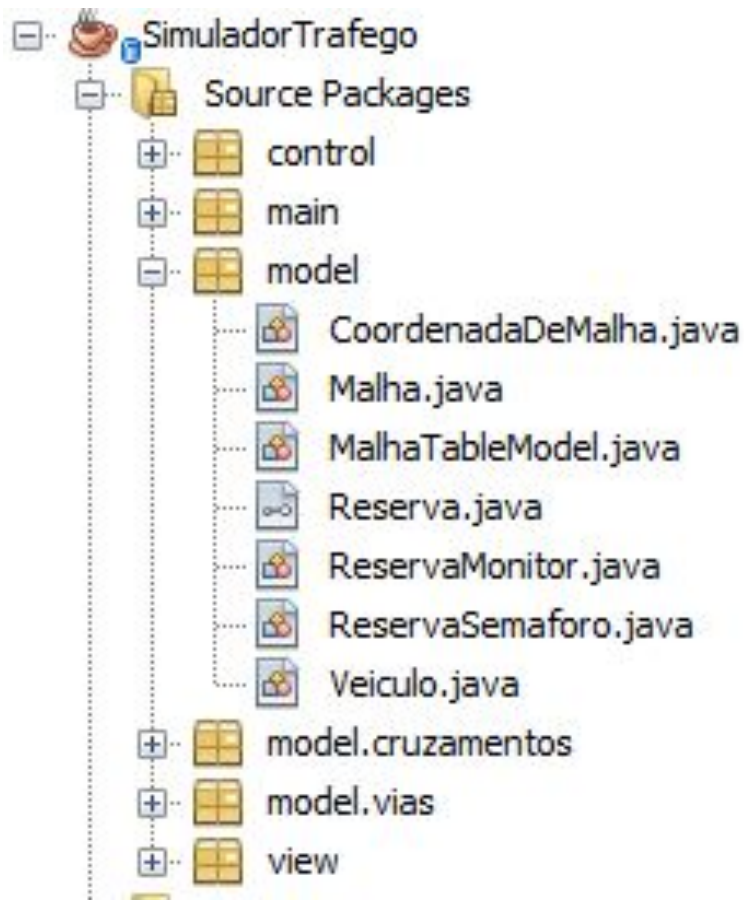




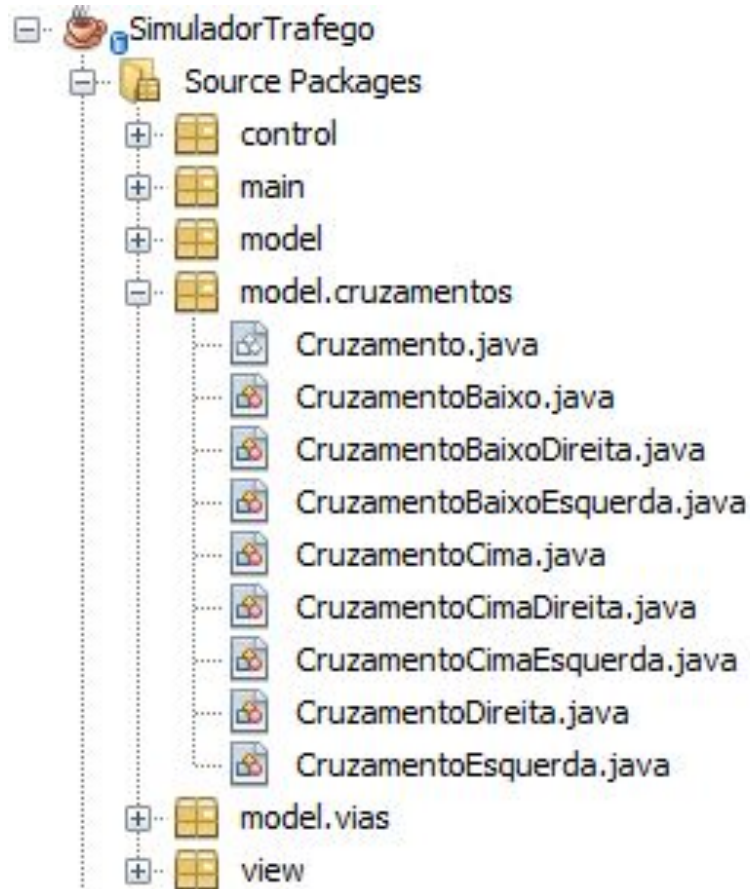
Estrutura



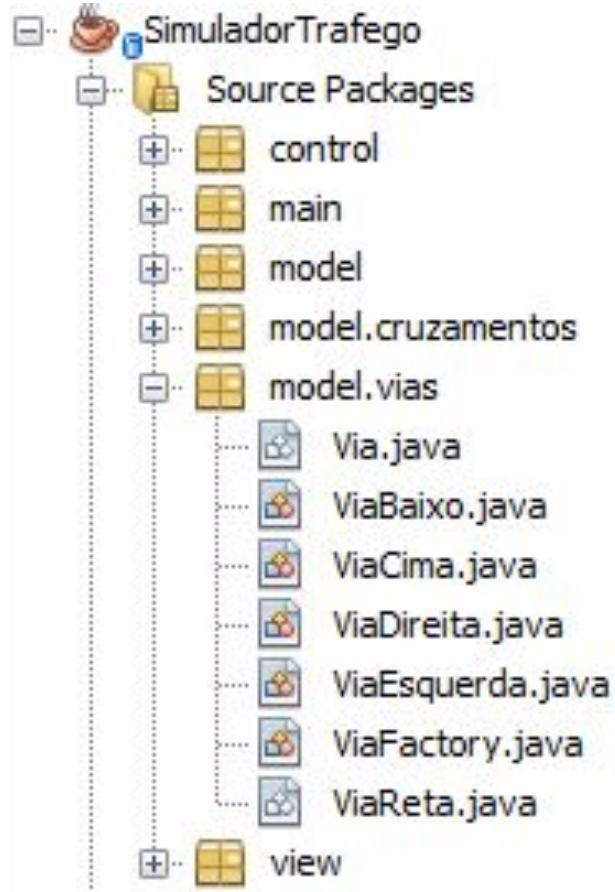
Estrutura - model



Estrutura - model > cruzamentos



Estrutura - model > vias



Estrutura - model > vias

```
26 public abstract class Via {
27
28     public abstract List<Via> getProximasVias();
29
30     protected CoordenadaDeMalha coordenadaDeMalha;
31     protected Reserva reserva;
32     protected ImageIcon imagemVazia;
33     protected ImageIcon imagemOcupada;
34
35     protected Via(CoordenadaDeMalha coordenadaDeMalha) {
36         this.coordenadaDeMalha = coordenadaDeMalha;
37         this.reserva = new ReservaSemaforo();
38     }
39
40     public void transformaTipoReservaEm(String classs) {
41         if (classs.equals("Semaforo")) {
42             trocaTipoReservaParaSemaforo();
43         } else {
44             trocaTipoReservaParaMonitor();
45         }
46     }
47
48     private void trocaTipoReservaParaMonitor() {
49         this.reserva = new ReservaMonitor();
50     }
51
52     private void trocaTipoReservaParaSemaforo() {
53         this.reserva = new ReservaSemaforo();
54     }
55
56     public CoordenadaDeMalha getCoordenadaDeMalha() {
57         return coordenadaDeMalha;
58     }
59
60     public boolean isBorda() {
61         return coordenadaDeMalha.isBorda();
62     }
```

```
63     public boolean isBordaSaida() {
64         return coordenadaDeMalha.isBordaSaida();
65     }
66
67     public boolean temVeiculo() {
68         return coordenadaDeMalha.temVeiculo();
69     }
70
71     public void setVeiculo(Veiculo v) {
72         coordenadaDeMalha.setVeiculo(v);
73     }
74
75     public void ocupar() throws InterruptedException {
76         reserva.ocupar();
77     }
78
79     public void desocupar() throws InterruptedException {
80         reserva.desocupar();
81     }
82
83     public ImageIcon getImagem() {
84         return temVeiculo() ? getImagemOcupada() : getImagemVazia();
85     }
86
87     private ImageIcon getImagemVazia() {
88         return imagemVazia;
89     }
90
91     private ImageIcon getImagemOcupada() {
92         return imagemOcupada;
93     }
94
95     public void setImagemVazia(ImageIcon imagemVazia) {
96         this.imagemVazia = imagemVazia;
97     }
98
99     public void setImagemOcupada(ImageIcon imagemOcupada) {
100         this.imagemOcupada = imagemOcupada;
101     }
```

Estrutura - model > vias

```
104 protected List<Via> factoryProximasVias(List<String> direcoes) {
105     List<Via> proximas = new ArrayList<>();
106     List<Via> ultrapassagens = new ArrayList<>();
107
108     for (String direcao : direcoes) {
109         Via via = null;
110         Via ultrapassagem = null;
111         switch (direcao) {
112             case "cima": {
113                 via = coordenadaDeMalha.getMalha().getViaNaCoordenada(getX(), getY() - 1);
114                 break;
115             }
116             case "baixo": {
117                 via = coordenadaDeMalha.getMalha().getViaNaCoordenada(getX(), getY() + 1);
118                 break;
119             }
120             case "direita": {
121                 via = coordenadaDeMalha.getMalha().getViaNaCoordenada(getX() + 1, getY());
122                 break;
123             }
124             case "esquerda": {
125                 via = coordenadaDeMalha.getMalha().getViaNaCoordenada(getX() - 1, getY());
126                 break;
127             }
128             case "diagonal-baixo-esquerda": {
129                 ultrapassagem = coordenadaDeMalha.getMalha().getViaNaCoordenada(getX() - 1, getY() + 1);
130                 break;
131             }
132             case "diagonal-baixo-direita": {
133                 ultrapassagem = coordenadaDeMalha.getMalha().getViaNaCoordenada(getX() + 1, getY() + 1);
134                 break;
135             }
136             case "diagonal-cima-esquerda": {
137                 ultrapassagem = coordenadaDeMalha.getMalha().getViaNaCoordenada(getX() - 1, getY() - 1);
138                 break;
139             }
140         }
141     }
142 }
```

Estrutura - model > vias

```
139         case "diagonal-cima-direita": {
140             ultrapassagem = coordenadaDeMalha.getMalha().getViaNaCoordenada(getX() + 1, getY() - 1);
141             break;
142         }
143         default: {
144             via = null;
145         }
146     }
147     if (via != null && !via.temVeiculo()) {
148         proximas.add(via);
149     }
150     if (ultrapassagem != null && this.getClass() == ultrapassagem.getClass() && !ultrapassagem.temVeiculo()) {
151         ultrapassagens.add(ultrapassagem);
152     }
153 }
154 return proximas.isEmpty() ? ultrapassagens : proximas;
155 }
156
157 public int getX() {
158     return coordenadaDeMalha.getX();
159 }
160
161 public int getY() {
162     return coordenadaDeMalha.getY();
163 }
164
165 @Override
166 public String toString() {
167     return "Via{" + getX() + "," + getY() + "}";
168 }
169
170 }
```

```
17 public class ViaBaixo extends ViaReta{
18
19     public ViaBaixo(CoordenadaDeMalha coordenadaDeMalha) {
20         super(coordenadaDeMalha);
21     }
22
23     @Override
24     public List<Via> getProximasVias() {
25         List<String> direcoes = new ArrayList<>();
26         direcoes.add("baixo");
27         direcoes.add("diagonal-baixo-esquerda");
28         direcoes.add("diagonal-baixo-direita");
29
30         return super.factoryProximasVias(direcoes);
31     }
32
33 }
34
```


Estrutura - model > cruzamentos

```
22 public abstract class Cruzamento extends Via {
23
24     public Cruzamento(CoordenadaDeMalha coordenadaDeMalha) {
25         super(coordenadaDeMalha);
26     }
27
28     public List<List<Via>> calculaSaidasCruzamento() throws InterruptedException {
29         List<List<Via>> caminhos = new ArrayList<>();
30
31         List<Via> inicio = new ArrayList<>();
32         inicio.add(this);
33
34         calculaSaidasCruzamentoRecursive(caminhos, inicio, new ArrayList<Via>(), this);
35
36         return caminhos;
37     }
38
39     public void calculaSaidasCruzamentoRecursive(List<List<Via>> caminhos,
40         List<Via> caminhoPercorrido, List<Via> viasVisitadas, Via v) throws InterruptedException {
41         if (v instanceof ViaReta) {
42             List<Via> copy = new ArrayList<>(caminhoPercorrido);
43             caminhos.add(copy);
44         } else {
45             if (!viasVisitadas.contains(v)) {
46                 viasVisitadas.add(v);
47                 for (Via via : v.getProximasVias()) {
48                     List<Via> copy = new ArrayList<>(caminhoPercorrido);
49                     copy.add(via);
50                     calculaSaidasCruzamentoRecursive(caminhos, copy, viasVisitadas, via);
51                 }
52             }
53         }
54     }
55
56     public List<Via> gerarCaminhoSaidaDeCruzamento() throws InterruptedException {
57         List<List<Via>> caminhosASeguir = calculaSaidasCruzamento();
58         List<Via> caminhoASeguir = caminhosASeguir.get((int) (Math.random() * caminhosASeguir.size()));
59         return caminhoASeguir;
60     }
61 }
```


Estrutura - model > cruzamentos

```
61 @Override
62 protected List<Via> factoryProximasVias(List<String> direcoes) {
63     List<Via> proximas = new ArrayList<>();
64     for (String direcao : direcoes) {
65         Via via = null;
66         switch (direcao) {
67             case "cima": {
68                 via = coordenadaDeMalha.getMalha().getViaNaCoordenada(getX(), getY() - 1);
69                 break;
70             }
71             case "baixo": {
72                 via = coordenadaDeMalha.getMalha().getViaNaCoordenada(getX(), getY() + 1);
73                 break;
74             }
75             case "direita": {
76                 via = coordenadaDeMalha.getMalha().getViaNaCoordenada(getX() + 1, getY());
77                 break;
78             }
79             case "esquerda": {
80                 via = coordenadaDeMalha.getMalha().getViaNaCoordenada(getX() - 1, getY());
81                 break;
82             }
83             default: {
84                 via = null;
85             }
86         }
87         if (via != null) {
88             proximas.add(via);
89         }
90     }
91     return proximas;
92 }
93 }
```

Estrutura - model > cruzamentos



```
18 public class CruzamentoBaixoEsquerda extends Cruzamento{
19
20     public CruzamentoBaixoEsquerda(CoordenadaDeMalha coordenadaDeMalha) {
21         super(coordenadaDeMalha);
22     }
23
24     @Override
25     public List<Via> getProximasVias() {
26         List<String> direcoes = new ArrayList<>();
27         direcoes.add("baixo");
28         direcoes.add("esquerda");
29         return super.factoryProximasVias(direcoes);
30     }
31
32 }
```


Estrutura - reserva

```
public interface Reserva {  
    public void ocupar() throws InterruptedException;  
    public void desocupar() throws InterruptedException;  
    public boolean tentaOcupar() throws InterruptedException;  
    public String toString();  
}
```

```
public class ReservaMonitor implements Reserva {
```

```
    private ReentrantLock lock = new ReentrantLock();
```

```
    @Override
```

```
    public void ocupar() throws InterruptedException {  
        lock.lock();  
    }
```

```
    @Override
```

```
    public void desocupar() throws InterruptedException {  
        lock.unlock();  
    }
```

```
    @Override
```

```
    public String toString() {  
        return "Monitor";  
    }
```

```
    @Override
```

```
    public boolean tentaOcupar() throws InterruptedException {  
        return lock.tryLock((int) (Math.random() * 300 + 100), TimeUnit.  
    }
```

```
public class ReservaSemaforo extends Semaphore implements Reserva {
```

```
    public ReservaSemaforo() {  
        super(1);  
    }
```

```
    @Override
```

```
    public void ocupar() throws InterruptedException {  
        acquire();  
    }
```

```
    @Override
```

```
    public void desocupar() throws InterruptedException {  
        release();  
    }
```

```
    @Override
```

```
    public String toString() {  
        return "Semáforo";  
    }
```

```
    @Override
```

```
    public boolean tentaOcupar() throws InterruptedException {  
        return tryAcquire((int) (Math.random() * 300 + 100), TimeUnit.  
    }
```

```
194 public void iniciarSimulacao() {
195     if (malhaTableModel.getMaxVeiculos() != 0) {
196         if (!reservaTipoAtivo.equals("")) {
197             executando = true;
198             SwingWorker atualizador = new SwingWorker() {
199                 @Override
200                 protected Void doInBackground() throws Exception {
201                     do {
202                         atualizarTableModel();
203                         Thread.sleep(10);
204                     } while (executando || malhaTableModel.ge
205                         return null;
206                 }
207             };
208             SwingWorker worker = new SwingWorker() {
209                 @Override
210                 protected Void doInBackground() throws Exception {
211                     while (executando) {
212                         malhaTableModel.getMalha().inserirVeiculo();
213                         Thread.sleep(10);
214                     }
215                     return null;
216                 }
217             };
218             atualizador.execute();
219             worker.execute();
220         } else {
221             showMessage("Você precisa trocar o tipo para Reserva");
222         }
223     } else {
224         showMessage("Você precisa definir o máximo de veículos");
225     }
226 }
```

Usado para executar
tarefas de interação
de GUI longas em
um thread de
segundo plano



```
50  public void inserirVeiculo() {  
51      List<Via> acessos = getViasDeAcesso();  
52      Via via = acessos.get((int) (Math.random() * acessos.size()));  
53      if (veiculos.size() < maxVeiculos) {  
54          Veiculo v = new Veiculo((long) (50), via);  
55          via.setVeiculo(v);  
56          veiculos.add(v);  
57          v.start();  
58      }  
59  }
```

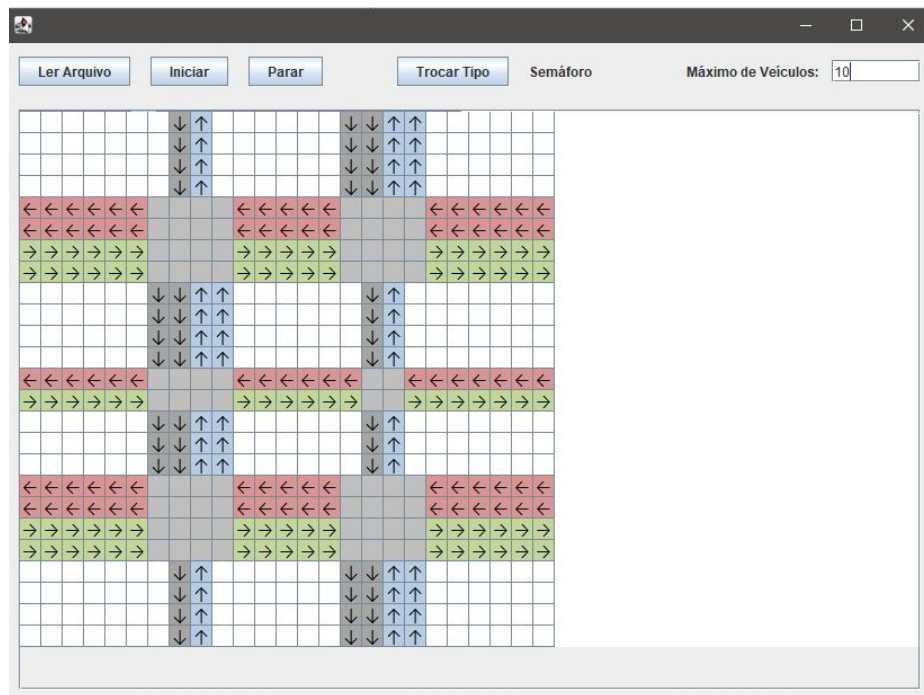
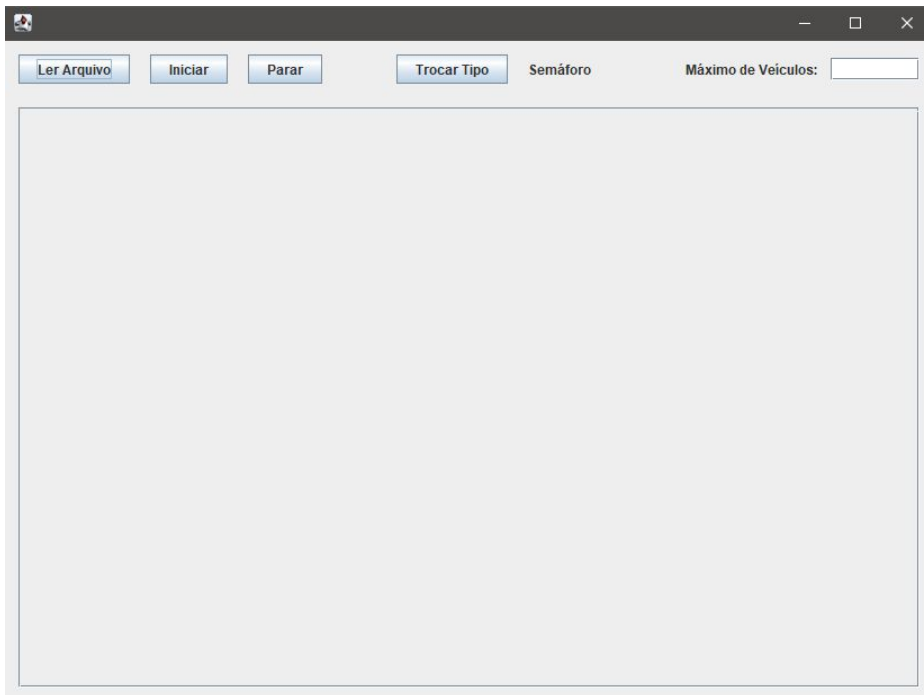


```
31  @Override
32  public void run() {
33      try {
34          via.ocupar();
35          do {
36              sleep(velocidade / 2);
37              andar();
38              sleep(velocidade / 2);
39          } while (!via.isBordaSaida());
40          encerrar();
41      } catch (InterruptedException ex) {
42          System.out.println("Carro bateu");
43      }
44  }
```

```
public void andar() throws InterruptedException {  
    List<Via> proximasVias = via.getProximasVias();  
    if (!proximasVias.isEmpty()) {  
        Via proximaVia = proximasVias.get((int) (Math.random() * proximasVias.size()));  
        if (proximaVia instanceof Cruzamento) {  
            List<Via> caminho = ((Cruzamento) proximaVia).gerarCaminhoSaidaDeCruzamento();  
            boolean tentativa = reservarCruzamento(caminho);  
            this.via.desocupar();  
            for (Via viaCaminho : caminho) {  
                sleep(velocidade / 2);  
                this.via.setVeiculo(null);  
                this.via = viaCaminho;  
                this.via.setVeiculo(this);  
                viaCaminho.desocupar();  
                sleep(velocidade / 2);  
            }  
        } else {  
            irParaVia(proximaVia);  
        }  
    }  
}
```

```
public void irParaVia(Via proximaVia) throws InterruptedException {  
    proximaVia.ocupar();  
    this.via.desocupar();  
    this.via.setVeiculo(null);  
    this.via = proximaVia;  
    this.via.setVeiculo(this);  
}
```

Demonstração do sistema





Dificuldades e soluções

- Threads para atualizar o Swing:
 - Implementação SwingWorker;
- Problema na remoção do Carro do Array de Carros na Malha:
 - **ConcurrentModificationException;**
 - Retiramos uma iteração para gerar verificar se a Via tinha um carro e vinculamos a coordenada de malha com a Via



Simulador de Tráfego em Malha Viária

Danton Krieck
Mohr
Edianeze Lara
Loch

