SIMULADOR DE TRÁFEGO EM MALHA VIÁRIA

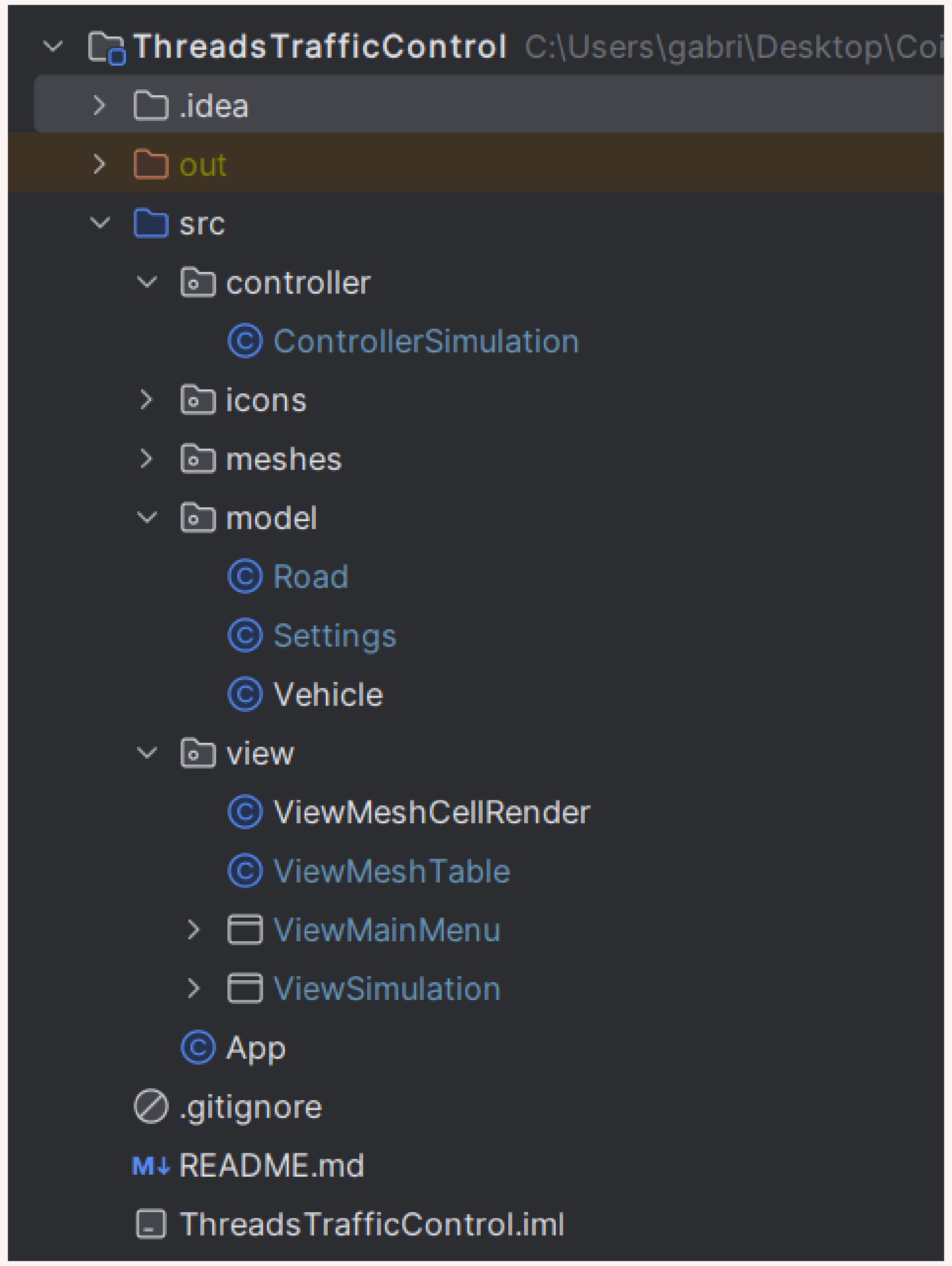
Trabalho 2

Aluno: Gabriel Fernando Doege

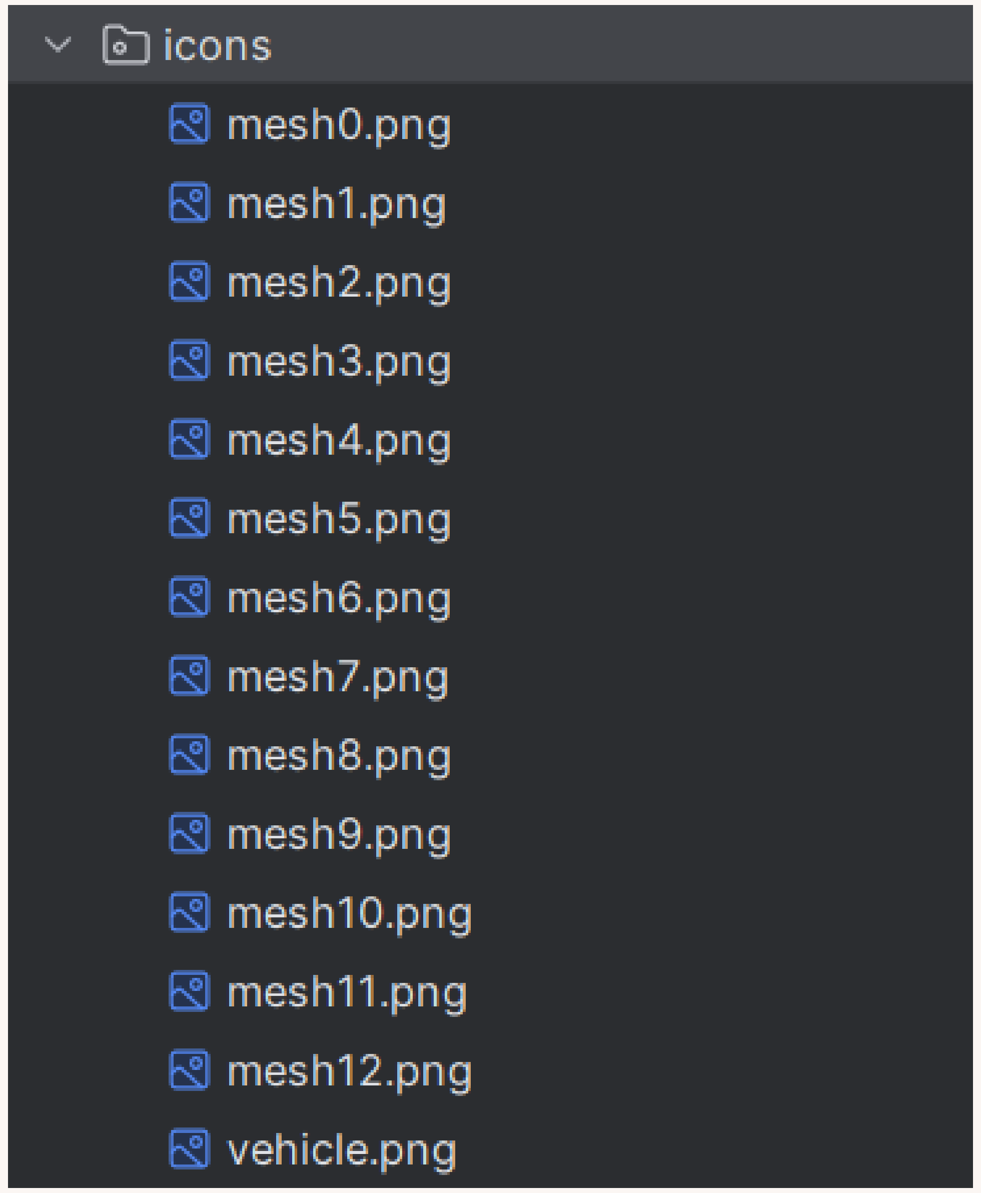
PADRÕES E TECNOLOGIAS UTILIZADAS

- MVC;
- Java;
- Swing.

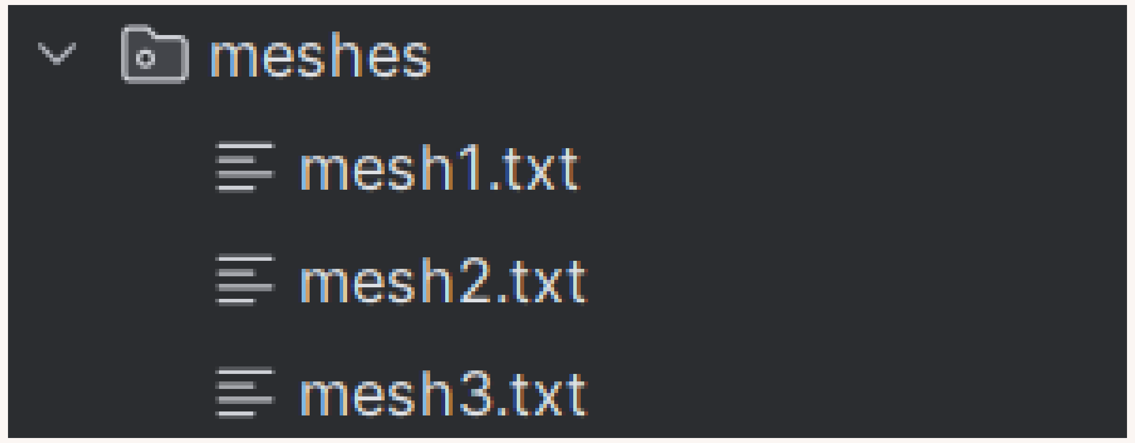
ESTRUTURA GERAL



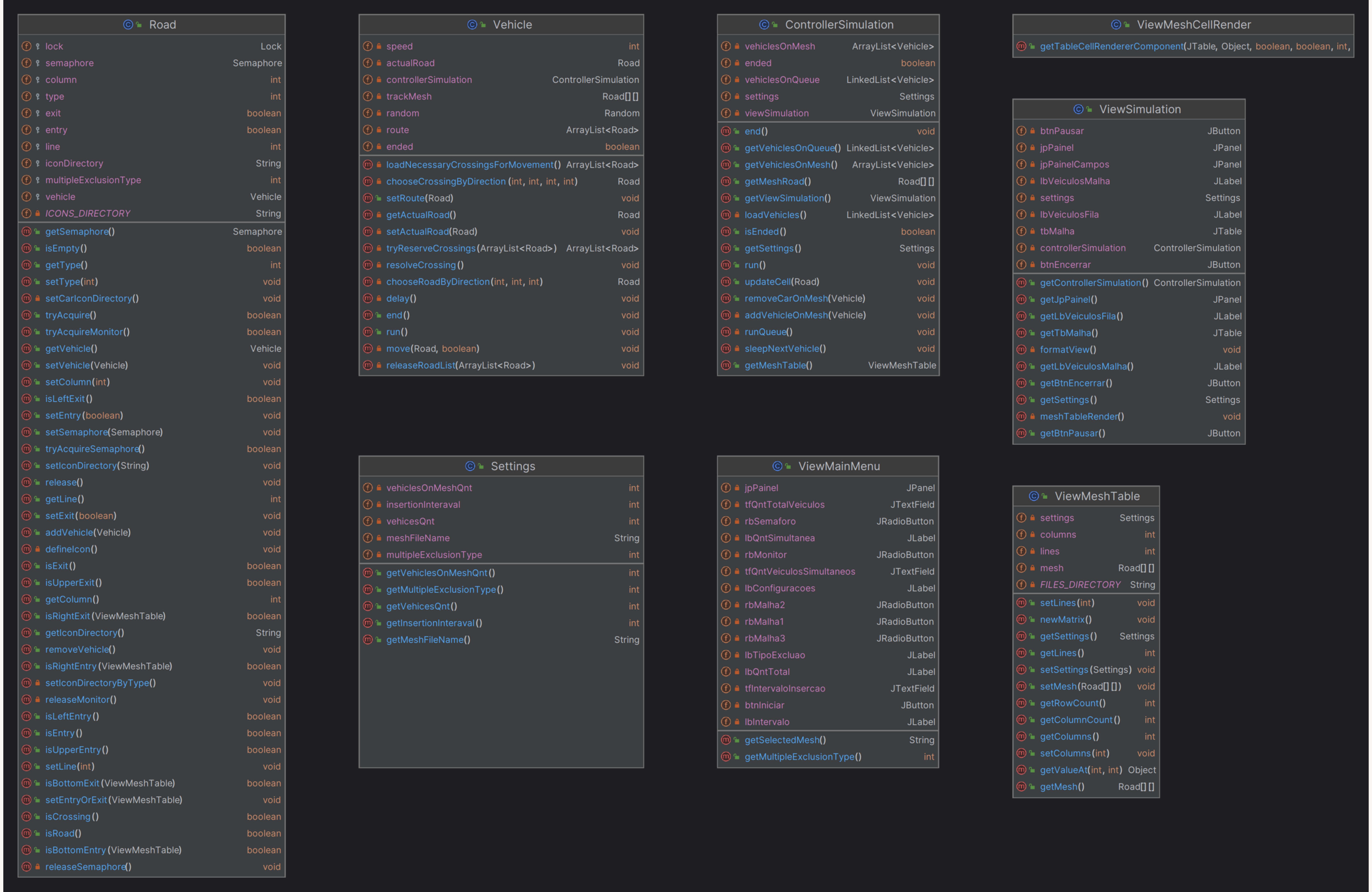
```
ThreadsTrafficControl  C:\Users\gabri\Desktop\Co
  .idea
  out
  src
    controller
      ControllerSimulation
    icons
    meshes
    model
      Road
      Settings
      Vehicle
    view
      ViewMeshCellRender
      ViewMeshTable
      ViewMainMenu
      ViewSimulation
    App
  .gitignore
  README.md
  ThreadsTrafficControl.iml
```



```
icons
  mesh0.png
  mesh1.png
  mesh2.png
  mesh3.png
  mesh4.png
  mesh5.png
  mesh6.png
  mesh7.png
  mesh8.png
  mesh9.png
  mesh10.png
  mesh11.png
  mesh12.png
  vehicle.png
```



```
meshes
  mesh1.txt
  mesh2.txt
  mesh3.txt
```



Road	
lock	Lock
semaphore	Semaphore
column	int
type	int
exit	boolean
entry	boolean
line	int
iconDirectory	String
multipleExclusionType	int
vehicle	Vehicle
ICONS_DIRECTORY	String
getSemaphore()	Semaphore
isEmpty()	boolean
getType()	int
setType(int)	void
setCarIconDirectory()	void
tryAcquire()	boolean
tryAcquireMonitor()	boolean
getVehicle()	Vehicle
setVehicle(Vehicle)	void
setColumn(int)	void
isLeftExit()	boolean
setEntry(boolean)	void
setSemaphore(Semaphore)	void
tryAcquireSemaphore()	boolean
setIconDirectory(String)	void
release()	void
getLine()	int
setExit(boolean)	void
addVehicle(Vehicle)	void
defineIcon()	void
isExit()	boolean
isUpperExit()	boolean
getColumn()	int
isRightExit(ViewMeshTable)	boolean
getIconDirectory()	String
removeVehicle()	void
isRightEntry(ViewMeshTable)	boolean
setIconDirectoryByType()	void
releaseMonitor()	void
isLeftEntry()	boolean
isEntry()	boolean
isUpperEntry()	boolean
setLine(int)	void
isBottomExit(ViewMeshTable)	boolean
setEntryOrExit(ViewMeshTable)	void
isCrossing()	boolean
isRoad()	boolean
isBottomEntry(ViewMeshTable)	boolean
releaseSemaphore()	void

Vehicle	
speed	int
actualRoad	Road
controllerSimulation	ControllerSimulation
trackMesh	Road[][]
random	Random
route	ArrayList<Road>
ended	boolean
loadNecessaryCrossingsForMovement()	ArrayList<Road>
chooseCrossingByDirection(int, int, int, int)	Road
setRoute(Road)	void
getActualRoad()	Road
setActualRoad(Road)	void
tryReserveCrossings(ArrayList<Road>)	ArrayList<Road>
resolveCrossing()	void
chooseRoadByDirection(int, int, int)	Road
delay()	void
end()	void
run()	void
move(Road, boolean)	void
releaseRoadList(ArrayList<Road>)	void

ControllerSimulation	
vehiclesOnMesh	ArrayList<Vehicle>
ended	boolean
vehiclesOnQueue	LinkedList<Vehicle>
settings	Settings
viewSimulation	ViewSimulation
end()	void
getVehiclesOnQueue()	LinkedList<Vehicle>
getVehiclesOnMesh()	ArrayList<Vehicle>
getMeshRoad()	Road[][]
getViewSimulation()	ViewSimulation
loadVehicles()	LinkedList<Vehicle>
isEnded()	boolean
getSettings()	Settings
run()	void
updateCell(Road)	void
removeCarOnMesh(Vehicle)	void
addVehicleOnMesh(Vehicle)	void
runQueue()	void
sleepNextVehicle()	void
getMeshTable()	ViewMeshTable

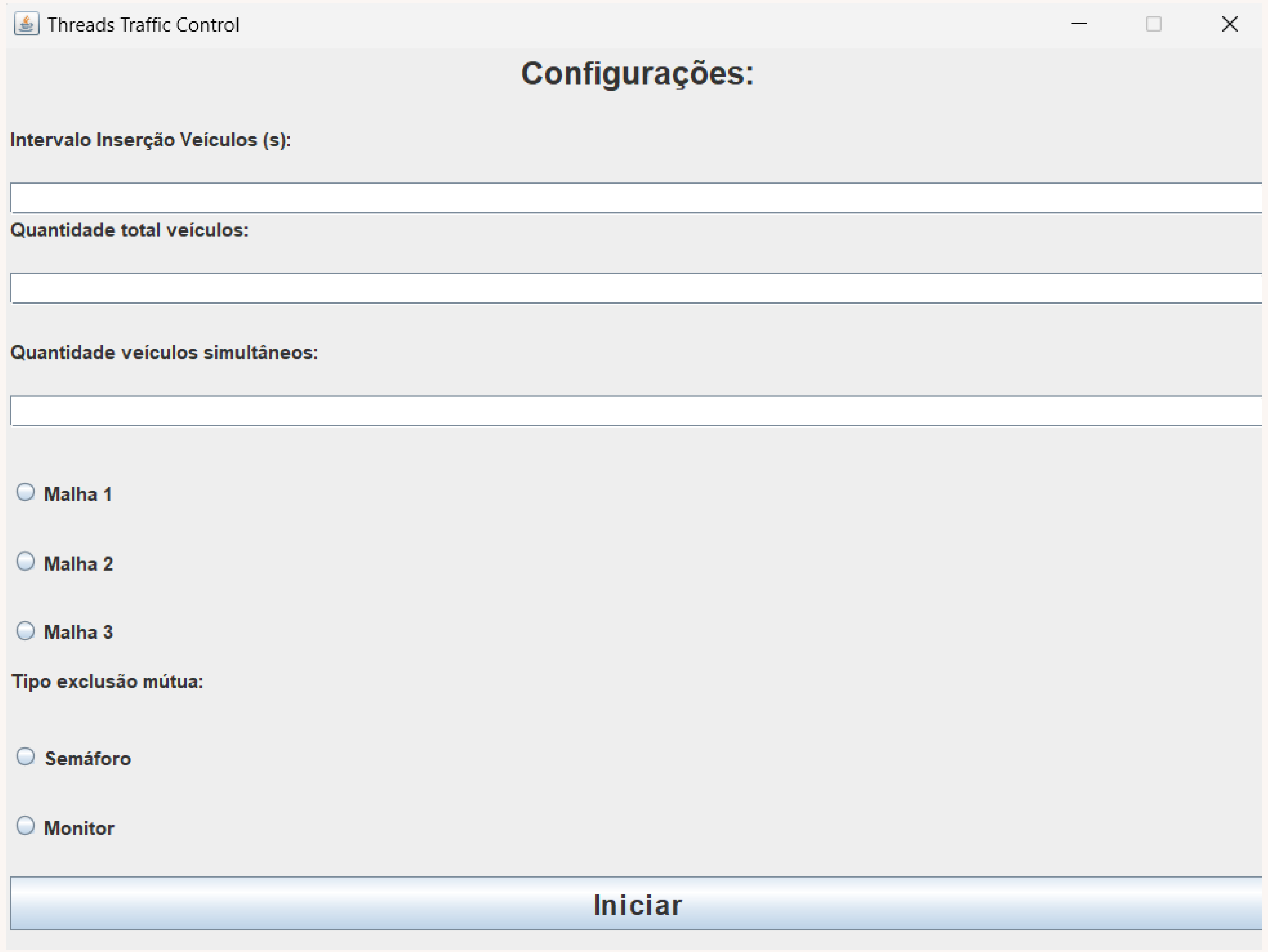
ViewMeshCellRender
getTableCellRendererComponent(JTable, Object, boolean, boolean, int,

ViewSimulation	
btnPausar	JButton
jpPainel	JPanel
jpPainelCampos	JPanel
lbVeiculosMalha	JLabel
settings	Settings
lbVeiculosFila	JLabel
tbMalha	JTable
controllerSimulation	ControllerSimulation
btnEncerrar	JButton
getControllerSimulation()	ControllerSimulation
getJpPainel()	JPanel
getLbVeiculosFila()	JLabel
getTbMalha()	JTable
formatView()	void
getLbVeiculosMalha()	JLabel
getBtnEncerrar()	JButton
getSettings()	Settings
meshTableRender()	void
getBtnPausar()	JButton

Settings	
vehiclesOnMeshQnt	int
insertionInteraval	int
vehicesQnt	int
meshFileName	String
multipleExclusionType	int
getVehiclesOnMeshQnt()	int
getMultipleExclusionType()	int
getVehicesQnt()	int
getInsertionInteraval()	int
getMeshFileName()	String

ViewMainMenu	
jpPainel	JPanel
tfQntTotalVeiculos	JTextField
rbSemaforo	JRadioButton
lbQntSimultanea	JLabel
rbMonitor	JRadioButton
tfQntVeiculosSimultaneos	JTextField
lbConfiguracoes	JLabel
rbMalha2	JRadioButton
rbMalha1	JRadioButton
rbMalha3	JRadioButton
lbTipoExcluao	JLabel
lbQntTotal	JLabel
tfIntervaloInsercao	JTextField
btnIniciar	JButton
lbIntervalo	JLabel
getSelectedMesh()	String
getMultipleExclusionType()	int

ViewMeshTable	
settings	Settings
columns	int
lines	int
mesh	Road[][]
FILES_DIRECTORY	String
setLines(int)	void
newMatrix()	void
getSettings()	Settings
getLines()	int
setSettings(Settings)	void
setMesh(Road[][])	void
getRowCount()	int
getColumnCount()	int
getColumns()	int
setColumns(int)	void
getValueAt(int, int)	Object
getMesh()	Road[][]



Threads Traffic Control

Configurações:

Intervalo Inserção Veiculos (s):

Quantidade total veiculos:

Quantidade veiculos simultâneos:

Malha 1

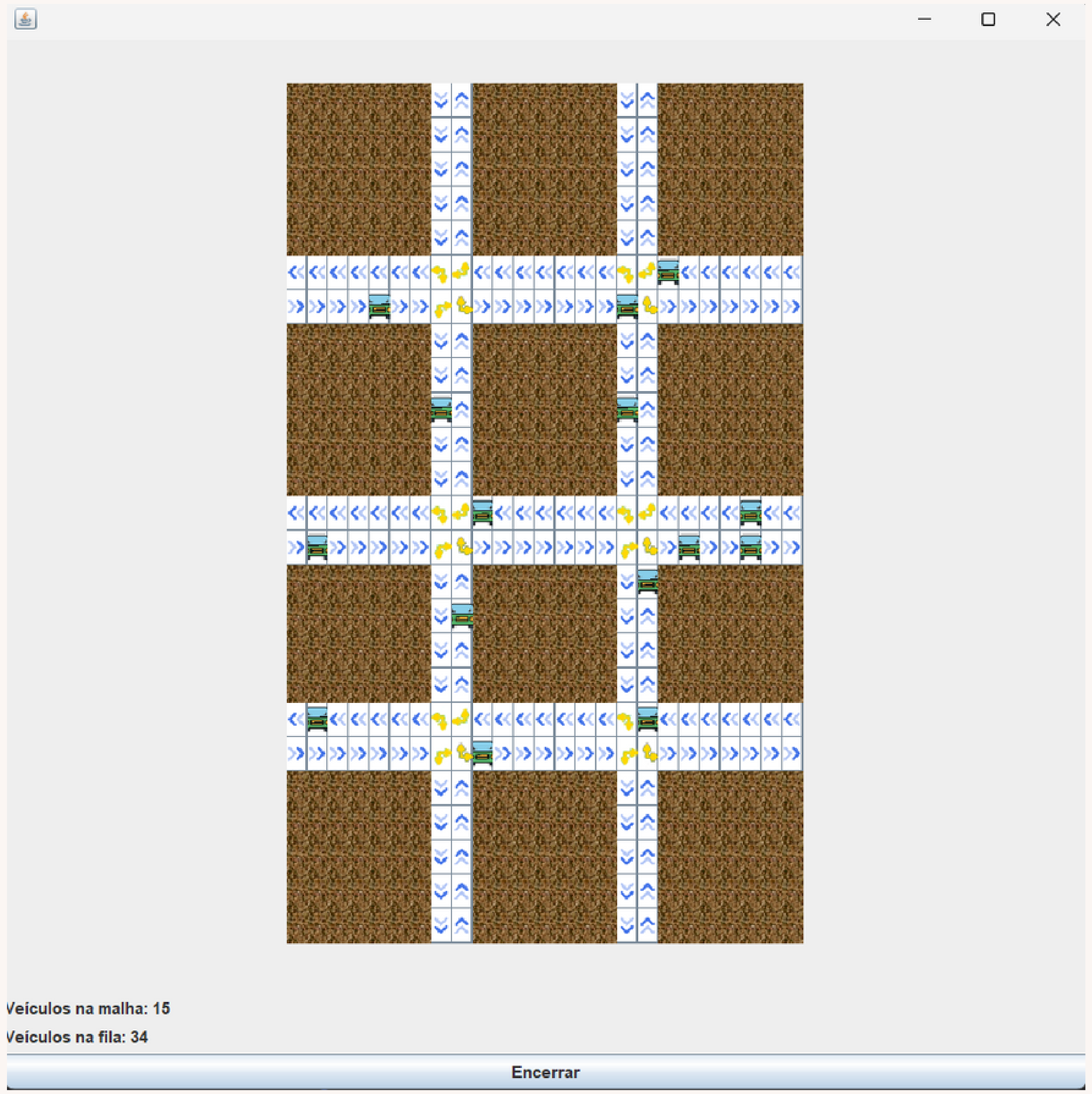
Malha 2

Malha 3

Tipo exclusão mútua:

Semáforo

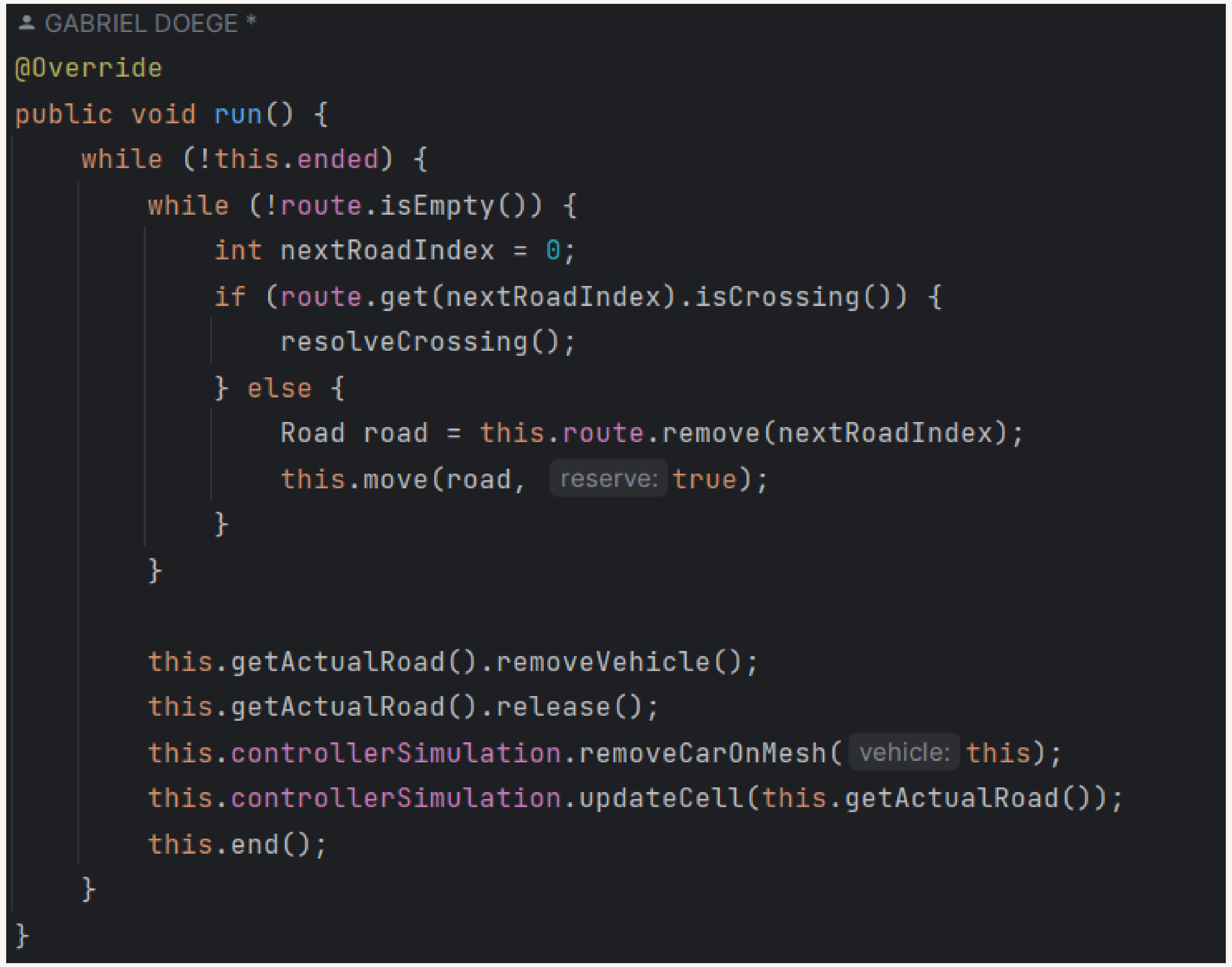
Monitor



Veiculos na malha: 15

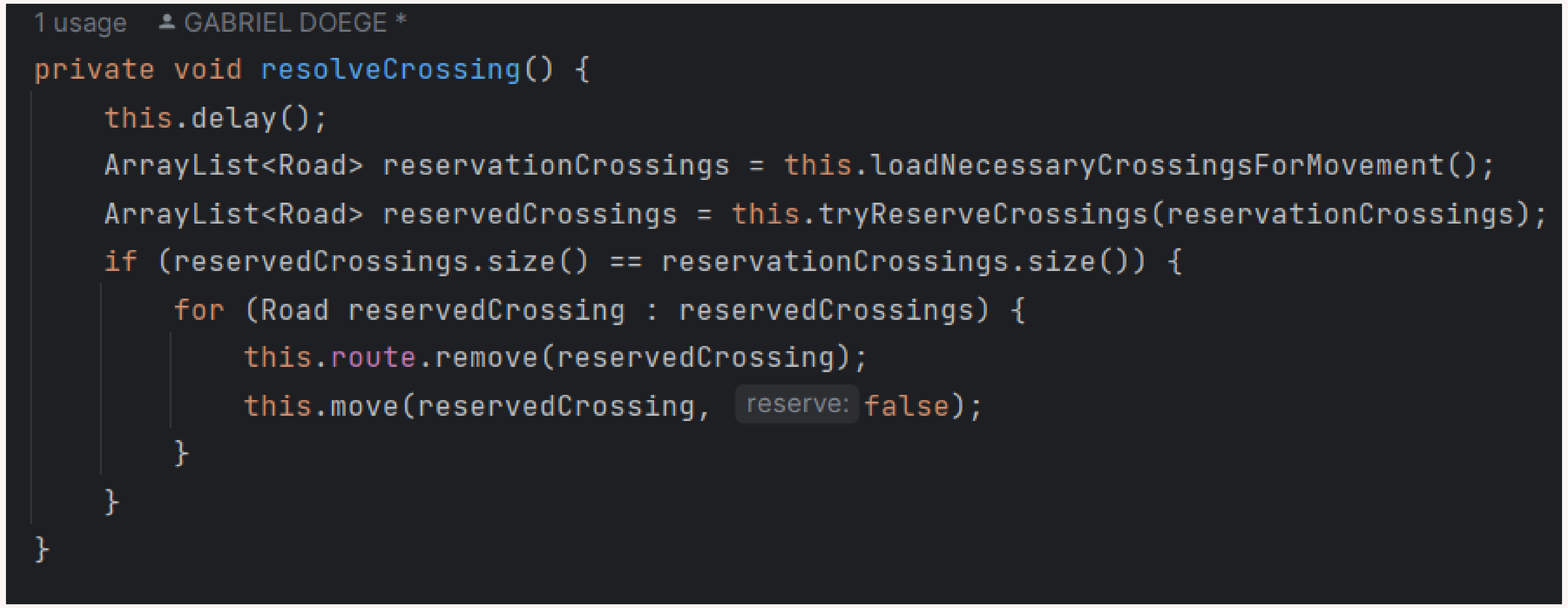
Veiculos na fila: 34

VEHICLE.JAVA



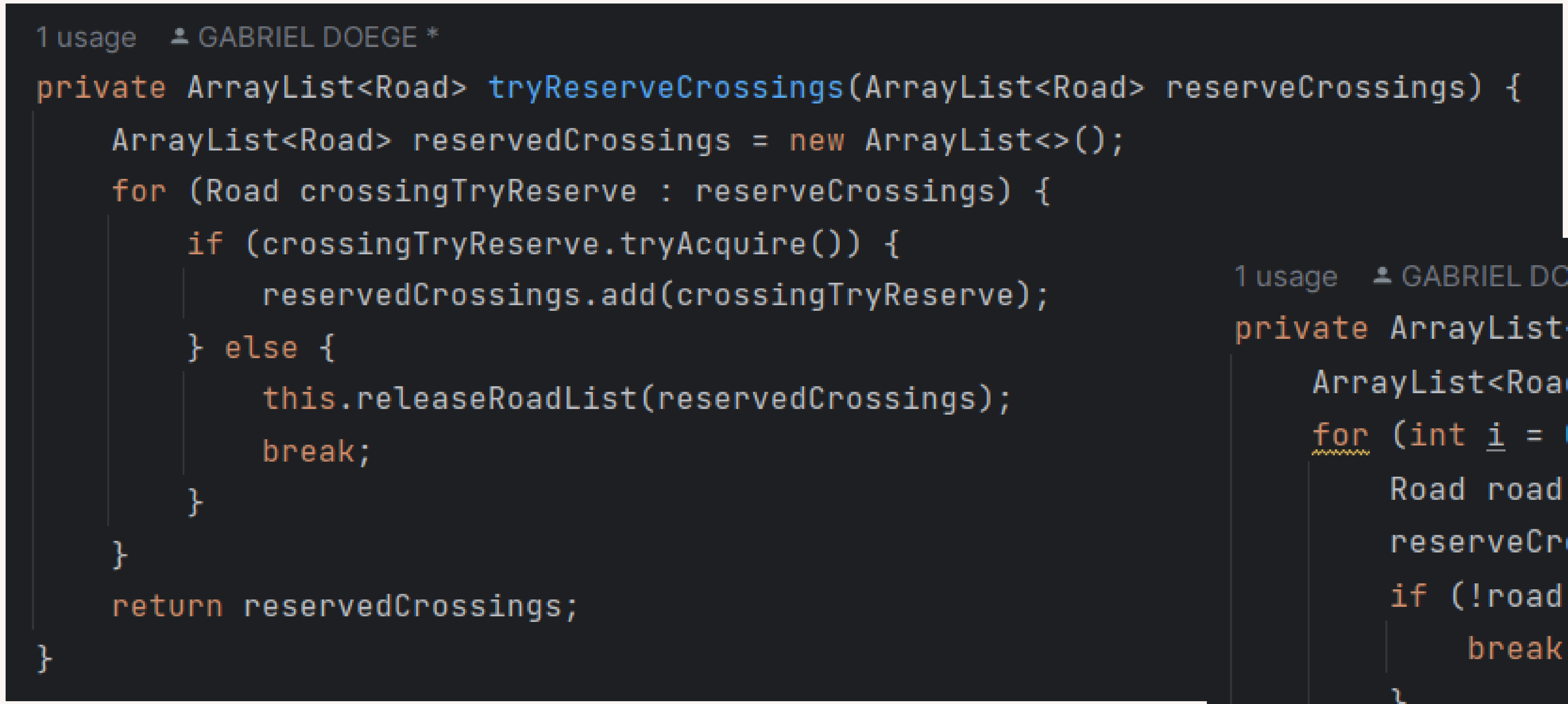
```
GABRIEL DOEGE *
@Override
public void run() {
    while (!this.ended) {
        while (!route.isEmpty()) {
            int nextRoadIndex = 0;
            if (route.get(nextRoadIndex).isCrossing()) {
                resolveCrossing();
            } else {
                Road road = this.route.remove(nextRoadIndex);
                this.move(road,  reserve: true);
            }
        }

        this.getActualRoad().removeVehicle();
        this.getActualRoad().release();
        this.controllerSimulation.removeCarOnMesh( vehicle: this);
        this.controllerSimulation.updateCell(this.getActualRoad());
        this.end();
    }
}
```

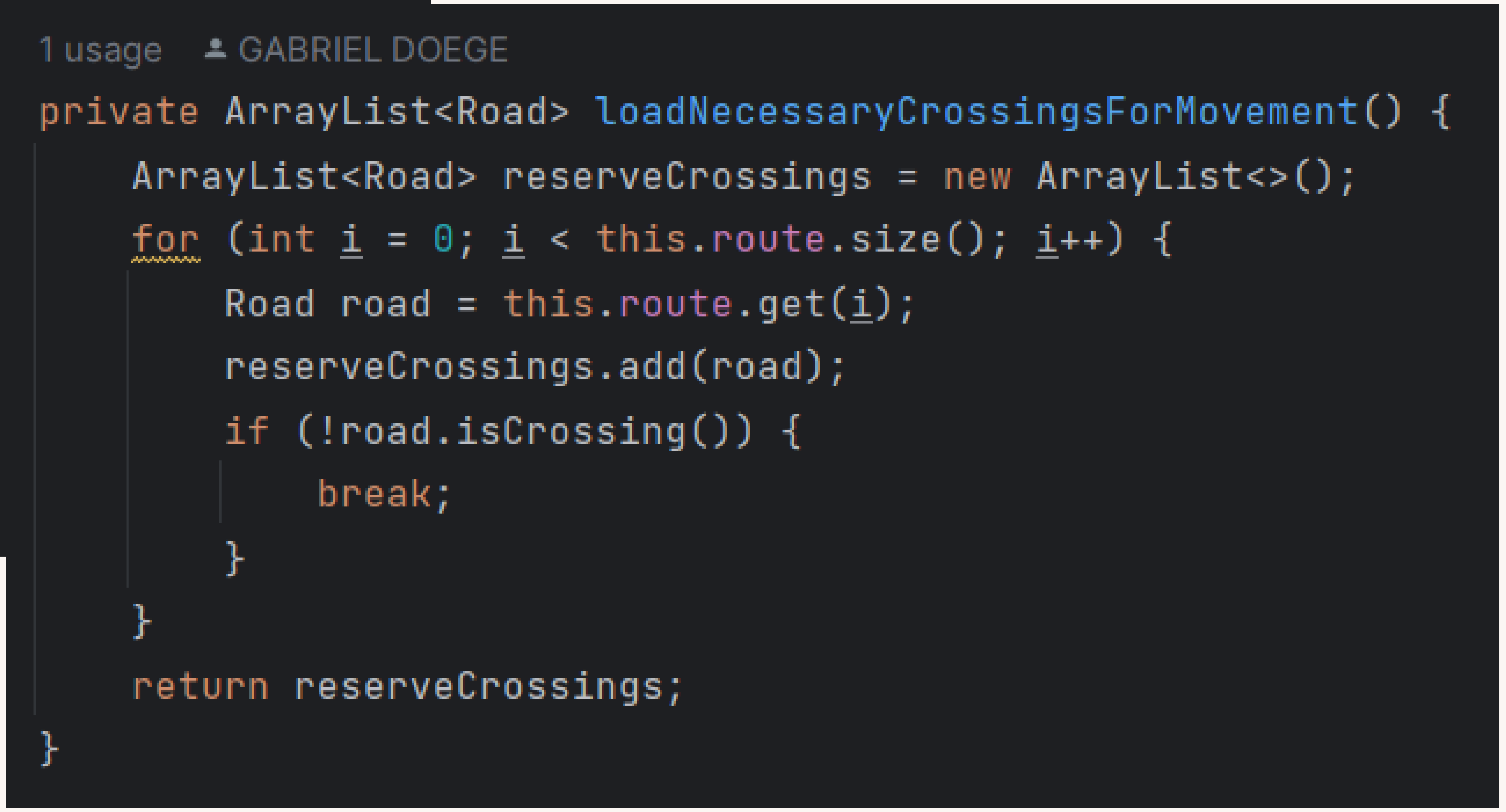


```
1 usage   GABRIEL DOEGE *
private void resolveCrossing() {
    this.delay();
    ArrayList<Road> reservationCrossings = this.loadNecessaryCrossingsForMovement();
    ArrayList<Road> reservedCrossings = this.tryReserveCrossings(reservationCrossings);
    if (reservedCrossings.size() == reservationCrossings.size()) {
        for (Road reservedCrossing : reservedCrossings) {
            this.route.remove(reservedCrossing);
            this.move(reservedCrossing,  reserve: false);
        }
    }
}
```

VEHICLE.JAVA

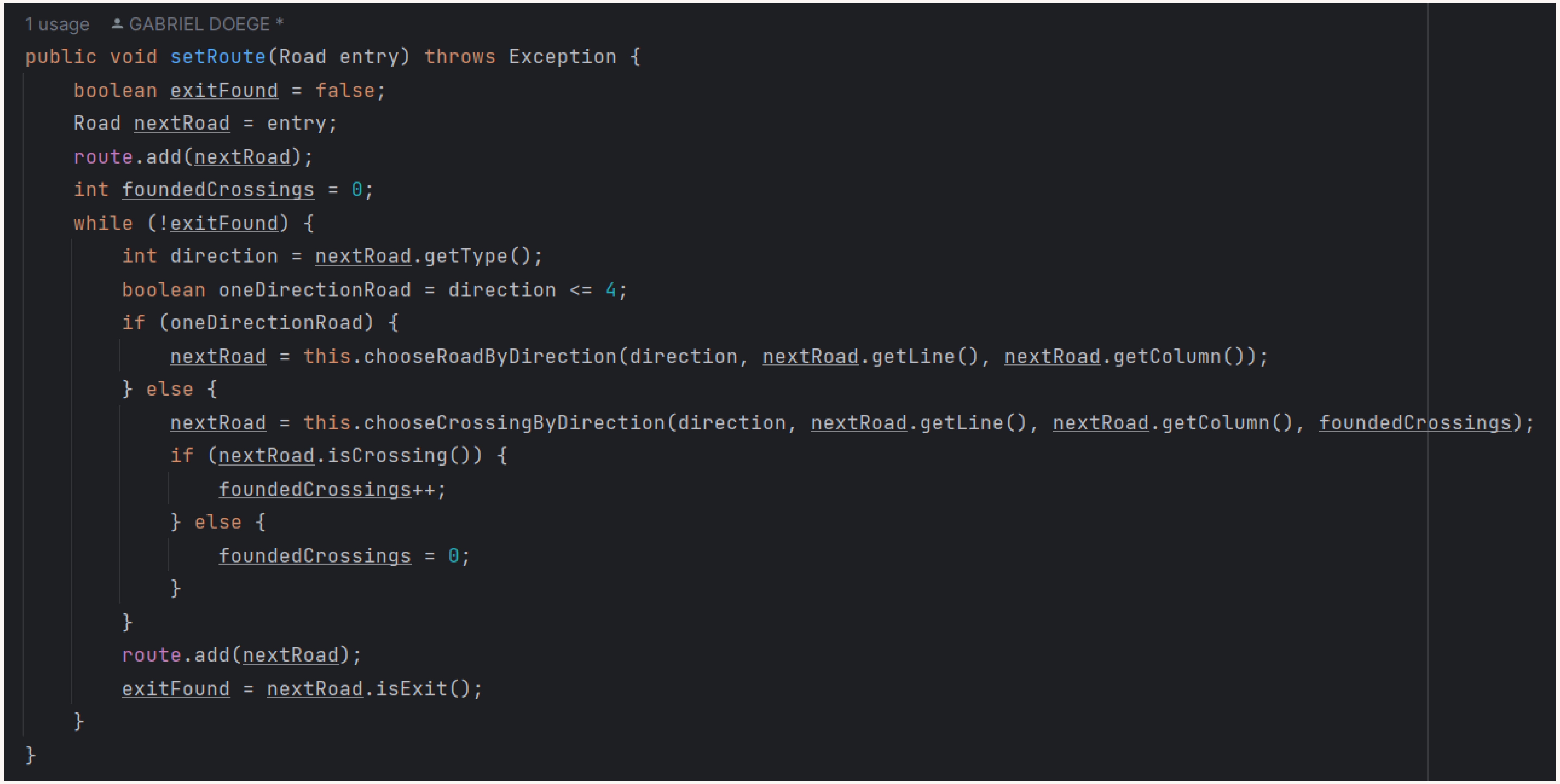


```
1 usage   GABRIEL DOEGE *
private ArrayList<Road> tryReserveCrossings(ArrayList<Road> reserveCrossings) {
    ArrayList<Road> reservedCrossings = new ArrayList<>();
    for (Road crossingTryReserve : reserveCrossings) {
        if (crossingTryReserve.tryAcquire()) {
            reservedCrossings.add(crossingTryReserve);
        } else {
            this.releaseRoadList(reservedCrossings);
            break;
        }
    }
    return reservedCrossings;
}
```



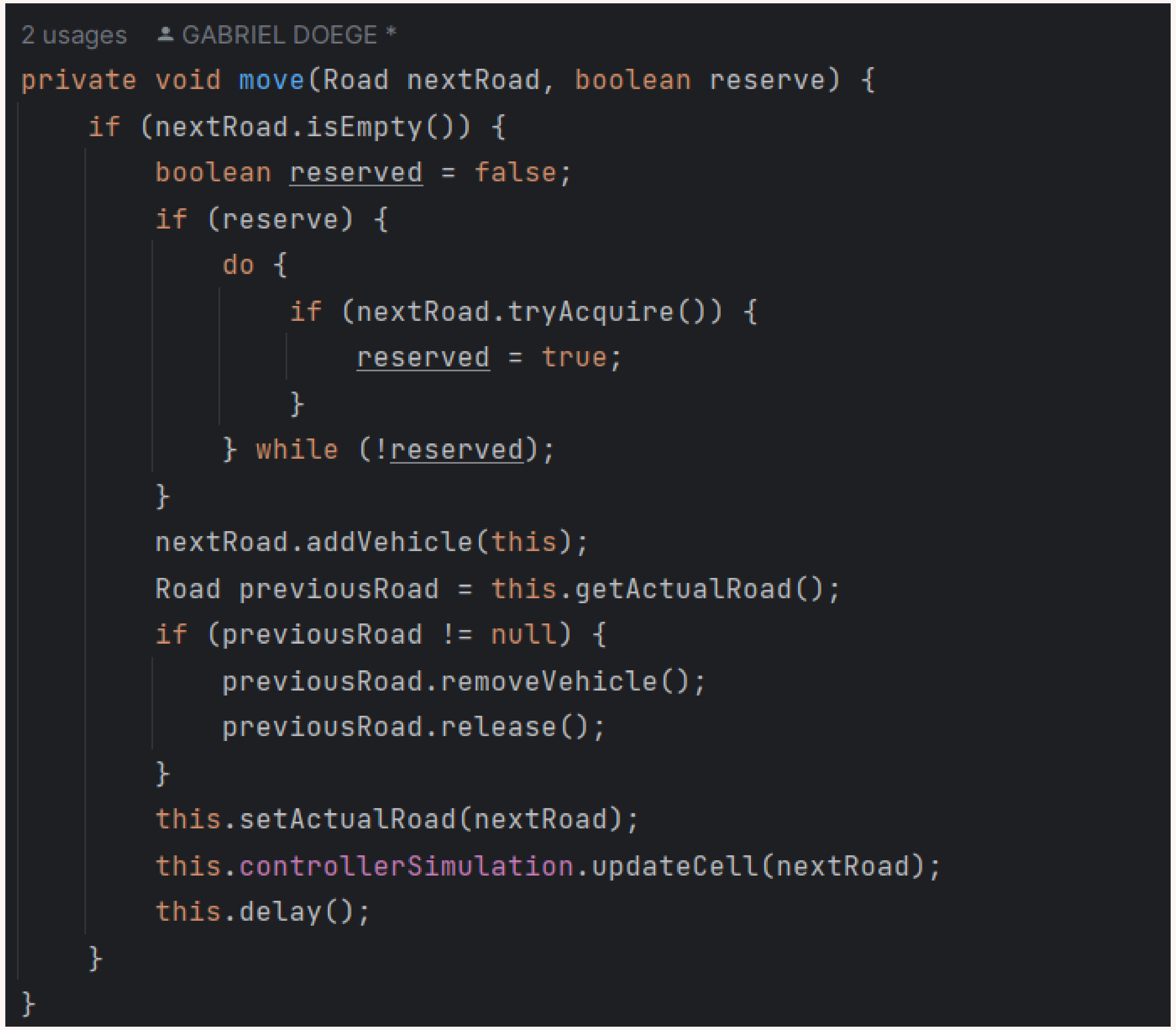
```
1 usage   GABRIEL DOEGE
private ArrayList<Road> loadNecessaryCrossingsForMovement() {
    ArrayList<Road> reserveCrossings = new ArrayList<>();
    for (int i = 0; i < this.route.size(); i++) {
        Road road = this.route.get(i);
        reserveCrossings.add(road);
        if (!road.isCrossing()) {
            break;
        }
    }
    return reserveCrossings;
}
```

VEHICLE.JAVA



```
1 usage   GABRIEL DOEGE *
public void setRoute(Road entry) throws Exception {
    boolean exitFound = false;
    Road nextRoad = entry;
    route.add(nextRoad);
    int foundedCrossings = 0;
    while (!exitFound) {
        int direction = nextRoad.getType();
        boolean oneDirectionRoad = direction <= 4;
        if (oneDirectionRoad) {
            nextRoad = this.chooseRoadByDirection(direction, nextRoad.getLine(), nextRoad.getColumn());
        } else {
            nextRoad = this.chooseCrossingByDirection(direction, nextRoad.getLine(), nextRoad.getColumn(), foundedCrossings);
            if (nextRoad.isCrossing()) {
                foundedCrossings++;
            } else {
                foundedCrossings = 0;
            }
        }
        route.add(nextRoad);
        exitFound = nextRoad.isExit();
    }
}
```

VEHICLE.JAVA


2 usages GABRIEL DOEGE *

```
private void move(Road nextRoad, boolean reserve) {
    if (nextRoad.isEmpty()) {
        boolean reserved = false;
        if (reserve) {
            do {
                if (nextRoad.tryAcquire()) {
                    reserved = true;
                }
            } while (!reserved);
        }
        nextRoad.addVehicle(this);
        Road previousRoad = this.getActualRoad();
        if (previousRoad != null) {
            previousRoad.removeVehicle();
            previousRoad.release();
        }
        this.setActualRoad(nextRoad);
        this.controllerSimulation.updateCell(nextRoad);
        this.delay();
    }
}
```

Muito Obrigado!