Simulação Malha Viária

Eloísa Bazzanella e Maria Eduarda Buzana

1

Diagrama de Classes

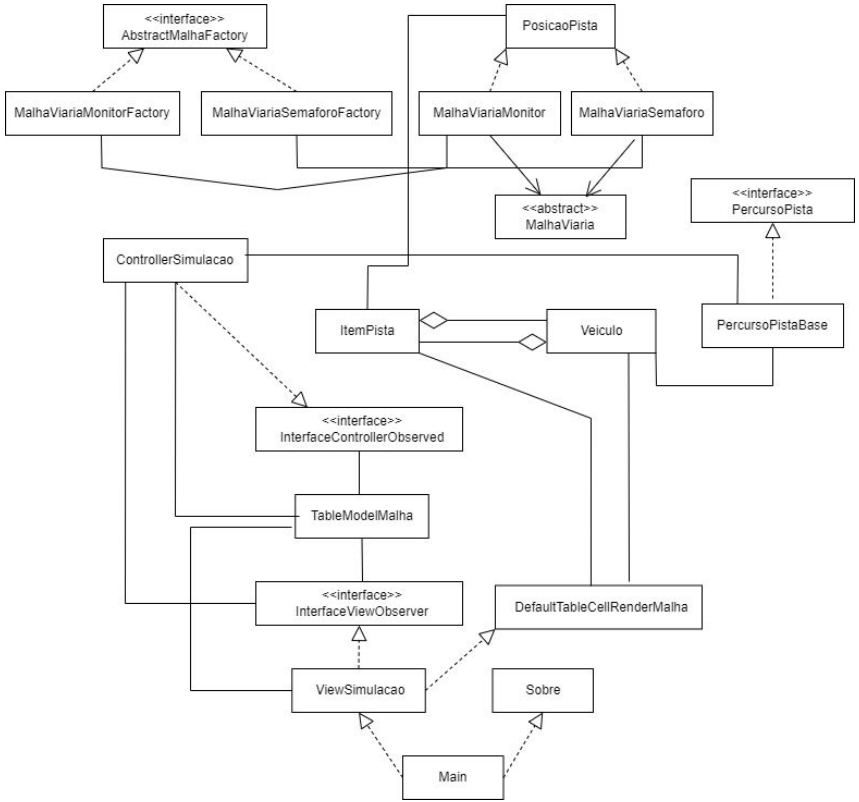
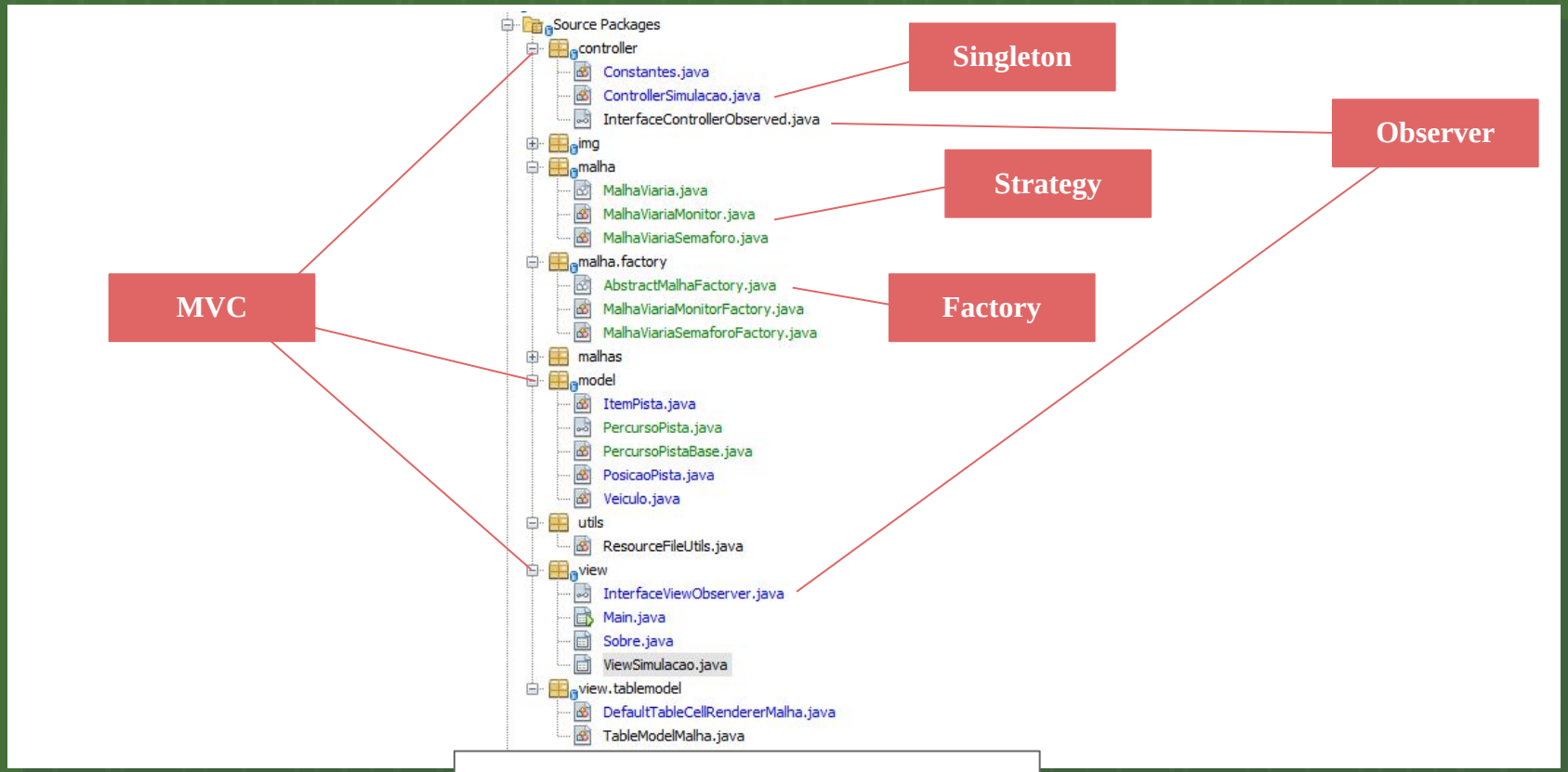


Diagrama de Classes



Estrutura do Projeto

2

Traços de Execução

```
private void onClickCarregar(java.awt.event.ActionEvent evt) {
    int fileChooseResponsee = FileChooser.showSaveDialog(null);

    if (fileChooseResponsee == JFileChooser.APPROVE_OPTION) {
        File arquivoSelecionado = new File(FileChooser.getSelectedFile().getAbsolutePath());

        try {
            controllerSimulacao.carregaSimulacao(arquivoSelecionado);
            ViewSimulacao telaSimulacao = new ViewSimulacao(controllerSimulacao);
            telaSimulacao.setVisible(true);

        } catch (IOException ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Criação da Malha

```java
public void carregaSimulacao(File arquivo) throws IOException {
    this.arquivo = arquivo;

    this.criaMalha(false);
    this.inicializarMalha();
    notifyTableModel(new TableModelMalha(this));
}

public void criaMalha(boolean usaMonitor) throws FileNotFoundException, IOException {
    BufferedReader in = new BufferedReader(new FileReader(arquivo));
    this.linhas = Integer.parseInt(in.readLine());
    this.colunas = Integer.parseInt(in.readLine());

    this.factoryMalha(false);

    for (int linhaAtual = 0; linhaAtual < linhas; linhaAtual++) {
        String[] listaTipos = in.readLine().split("\t");
        for (int colunaAtual = 0; colunaAtual < colunas; colunaAtual++) {
            int tipoPista = Integer.parseInt(listaTipos[colunaAtual]);
            PosicaoPista posicaoPista = new PosicaoPista(linhaAtual, colunaAtual);
            malhaViaria.adicionarItemPista(posicaoPista, new ItemPista(tipoPista, posicaoPista
        }
    }
}
```

Criação da Malha

Define o método a ser utilizado

```
public void factoryMalha(boolean usaMonitor) {
    if(usaMonitor) {
        AbstractMalhaFactory factory = new MalhaViariaMonitorFactory();
        malhaViaria = factory.criarMalha(linhas, colunas);
    } else {
        AbstractMalhaFactory factory = new MalhaViariaSemaforoFactory();
        malhaViaria = factory.criarMalha(linhas, colunas);
    }
}
```

Criação da Malha

```java
public class TableModelMalha extends AbstractTableModel {

    private InterfaceControllerObserved controller;

    public TableModelMalha(InterfaceControllerObserved controller) {
        this.controller = controller;
    }

    @Override
    public int getRowCount() {
        return this.controller.getMalhaRodoviaria().length;
    }

    @Override
    public int getColumnCount() {
        return this.controller.getMalhaRodoviaria()[0].length;
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        return this.controller.getMalhaRodoviaria()[rowIndex][columnIndex];
    }

}
```

```java
public class DefaultTableCellRendererMalha extends DefaultTableCellRenderer {
```

Renderização da JTable

```java
public void iniciarSimulacao(int quantidadeVeiculo, boolean usaSemaforo, boolean usaMonitor) throws
    notifyButtonChanged(true);

    if(usaMonitor) {
        this.malhaViaria = null;
        this.criaMalha(true);

        this.inicializarMalha();
        notifyTableModel(new TableModelMalha(this));
    }

    new Thread(() -> {
        while (start) {
            if (this.veiculos.size() < quantidadeVeiculo) {
                iniciarNovoVeiculo(malhaViaria.entradaAleatoria());
                notifyTableModelChanged();
            }
            sleepThread();
        }
    }).start();
}
```

Define quantidade veículos

Iniciar Simulação

```
private void definirEntradasSaidas() {
    for (int linhaAtual = 0; linhaAtual < linhas; linhaAtual++) {
        for (int colunaAtual = 0; colunaAtual < colunas; colunaAtual++) {
            ItemPista itemPistaAtual = malha[linhaAtual][colunaAtual];

            adicionarPistasAdjacentes(itemPistaAtual, new PosicaoPista(linhaAtual, colunaAtual));

            if (primeiraLinha(linhaAtual)) {
                definirEntradaSe(itemPistaAtual, ESTRADA_BAIXO);
                definirSaidaSe(itemPistaAtual, ESTRADA_CIMA);
            }

            if (ultimaLinha(linhaAtual)) {
                definirEntradaSe(itemPistaAtual, ESTRADA_CIMA);
                definirSaidaSe(itemPistaAtual, ESTRADA_BAIXO);
            }

            if (primeiraColuna(colunaAtual)) {
                definirEntradaSe(itemPistaAtual, ESTRADA_DIREITA);
                definirSaidaSe(itemPistaAtual, ESTRADA_ESQUERDA);
            }

            if (ultimaColuna(colunaAtual)) {
                definirEntradaSe(itemPistaAtual, ESTRADA_ESQUERDA);
                definirSaidaSe(itemPistaAtual, ESTRADA_DIREITA);
            }
        }
    }
```

Definir Entradas/Saídas

```java
private void definirEntradaSe(ItemPista itemPistaAtual, int direcaoPistaAtual) {
    if (itemPistaAtual.getTipo() == direcaoPistaAtual) {
        itemPistaAtual.setIsEntrada(true);
        adicionarNovaEntrada(itemPistaAtual);
    }
}

private void definirSaidaSe(ItemPista itemPistaAtual, int direcaoPistaAtual) {
    if (itemPistaAtual.getTipo() == direcaoPistaAtual) {
        itemPistaAtual.setIsSaida(true);
    }
}
```

Definir Entradas/Saídas

```java
public void iniciarNovoVeiculo(ItemPista itemPistaAtual) {
    Veiculo veiculo = new Veiculo(malhaViaria);
    this.addVeiculo(veiculo);
    veiculo.setPistaAtual(itemPistaAtual);
    new Thread(veiculo::start).start();
}
```

Iniciar um Veículo

```java
public synchronized void start() {
    while (ControllerSimulacao.getInstance().isRunning()) {
        try {
            if (this.pistaAtual.isSaida()) {
                this.pistaAtual.setOcupada(false);
                this.pistaAtual.setVeiculo(null);

                removeVeiculo();

                return;
            }

            if (this.percurso == null || !this.percurso.hasNext()) {
                this.percurso = this.getPercurso().iterator();
            }

            ItemPista proximaPista = this.percurso.next();

            this.pistaAnterior = this.pistaAtual;
            this.pistaAnterior.setVeiculo(null);
            this.pistaAnterior.setOcupada(false);

            this.pistaAtual = proximaPista;

            malhaViaria.adicionarVeiculo(pistaAtual.getPosicaoPista(), this);

            ControllerSimulacao.getInstance().notifyTableModelChanged();

            Thread.sleep(this.getVelocidade());
```

```java
private int velocidadeAleatoria() {
    Random random = new Random();
    int acrescimo = random.nextInt(250);
    return 250 + acrescimo;
}
```

Velocidade Aleatória

Se encontra saída

Executa Percurso

Memorizem isso!!!!

Iniciar um Veículo

```java
@Override
public List<ItemPista> getPercurso(ItemPista pistaAtual, ItemPista pistaAnterior) {
    List<ItemPista> percurso = new ArrayList<>();
    ItemPista proximaPista = null;

    do {
        proximaPista = this.getProximaPista(pistaAtual, pistaAnterior);
    } while (proximaPista.isOcupada() || !proximaPista.isTransitavel());

    proximaPista.setOcupada(true);

    percurso.add(proximaPista);

    if(proximaPista.isCruzamento()) {
        percurso.addAll(this.getPercurso(proximaPista, pistaAtual));
    }

    return percurso;
}
```

Reserva Caminho

Montagem do Percurso

```
public ItemPista getProximaPista(ItemPista pistaAtual, ItemPista pistaAnterior) {
    ItemPista pista = null;

    switch (pistaAtual.getTipo()) {
        case 1: {
            if(pistaAtual.getPistaDireita().getTipo() == pistaAtual.getTipo()){
                Random random = new Random();
                int opcao = random.nextInt(5);

                if (opcao == 1) {
                    pista = pistaAtual.getPistaDiagonalCimaDireita();
                } else {
                    pista = pistaAtual.getPistaCima();
                }
            }
            else if(pistaAtual.getPistaEsquerda().getTipo() == pistaAtual.getTipo()) {
                Random random = new Random();
                int opcao = random.nextInt(5);

                if (opcao == 1) {
                    pista = pistaAtual.getPistaDiagonalCimaEsquerda();
                } else {
                    pista = pistaAtual.getPistaCima();
                }
            }
            else {
                pista = pistaAtual.getPistaCima();
            }
            break;
        }
```

Troca de Pista em Pista Dupla

Montagem do Percurso

```
        case 12: {
            if (pistaAnterior.getTipo() == 10 || pistaAnterior.getTipo() == 12) {
                pista = pistaAtual.getPistaEsquerda();
            } else {
                Random random = new Random();
                int opcao = random.nextInt(2);

                if (opcao == 1) {
                    pista = pistaAtual.getPistaBaixo();
                } else {
                    pista = pistaAtual.getPistaEsquerda();
                }
            }
            break;
        }
    }
```

Cruzamento

Montagem do Percurso

```java
public MalhaViariaMonitor(int linhas, int colunas) {
    super(linhas, colunas);
}

@Override
public synchronized void adicionarVeiculo(PosicaoPista posicaoPista, Veiculo veiculo) {
    int linha = posicaoPista.getLinha();
    int coluna = posicaoPista.getColuna();
    malha[linha][coluna].setVeiculo(veiculo);
}

@Override
public synchronized void removerVeiculo(PosicaoPista posicaoPista, Veiculo veiculo) {
    int linha = posicaoPista.getLinha();
    int coluna = posicaoPista.getColuna();
    malha[linha][coluna].setVeiculo(null);
    notificarController(veiculo);
}
```

Deslocamento do Veículo nas Pistas com Monitor

Montagem do Percurso

```java
private final Semaphore mutex;

public MalhaViariaSemaforo(int linhas, int colunas) {
    super(linhas, colunas);

    mutex = new Semaphore(1);
}

@Override
public void adicionarVeiculo(PosicaoPista posicaoPista, Veiculo veiculo) {
    try {
        mutex.acquire();
        int linha = posicaoPista.getLinha();
        int coluna = posicaoPista.getColuna();
        malha[linha][coluna].setVeiculo(veiculo);
    } catch (InterruptedException e) {
        e.printStackTrace();
    } finally {
        mutex.release();
    }
}
```

Deslocamento do Veículo nas Pistas com Semáforo

Montagem do Percurso

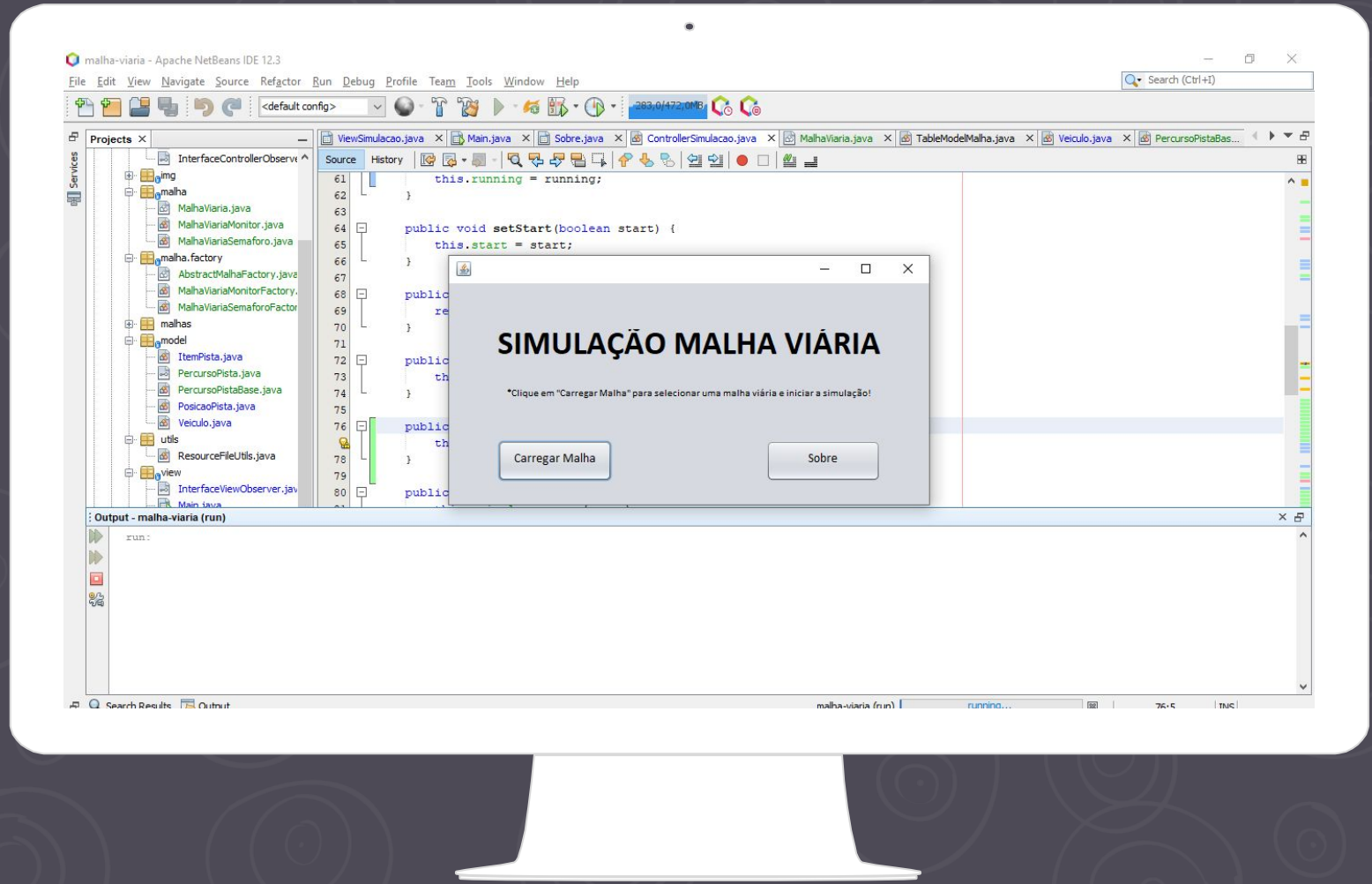
```
private void btFinalizarDefActionPerformed(java.awt.event.ActionEvent evt) {
    this.controllerSimulacao.setStart(false);
    this.controllerSimulacao.setRunning(false);
    this.controllerSimulacao.removeTodosVeiculos();
    this.controllerSimulacao.inicializarMalha();
    this.controllerSimulacao.notifyTableModelChanged();
    this.atualizaButton(false);
```

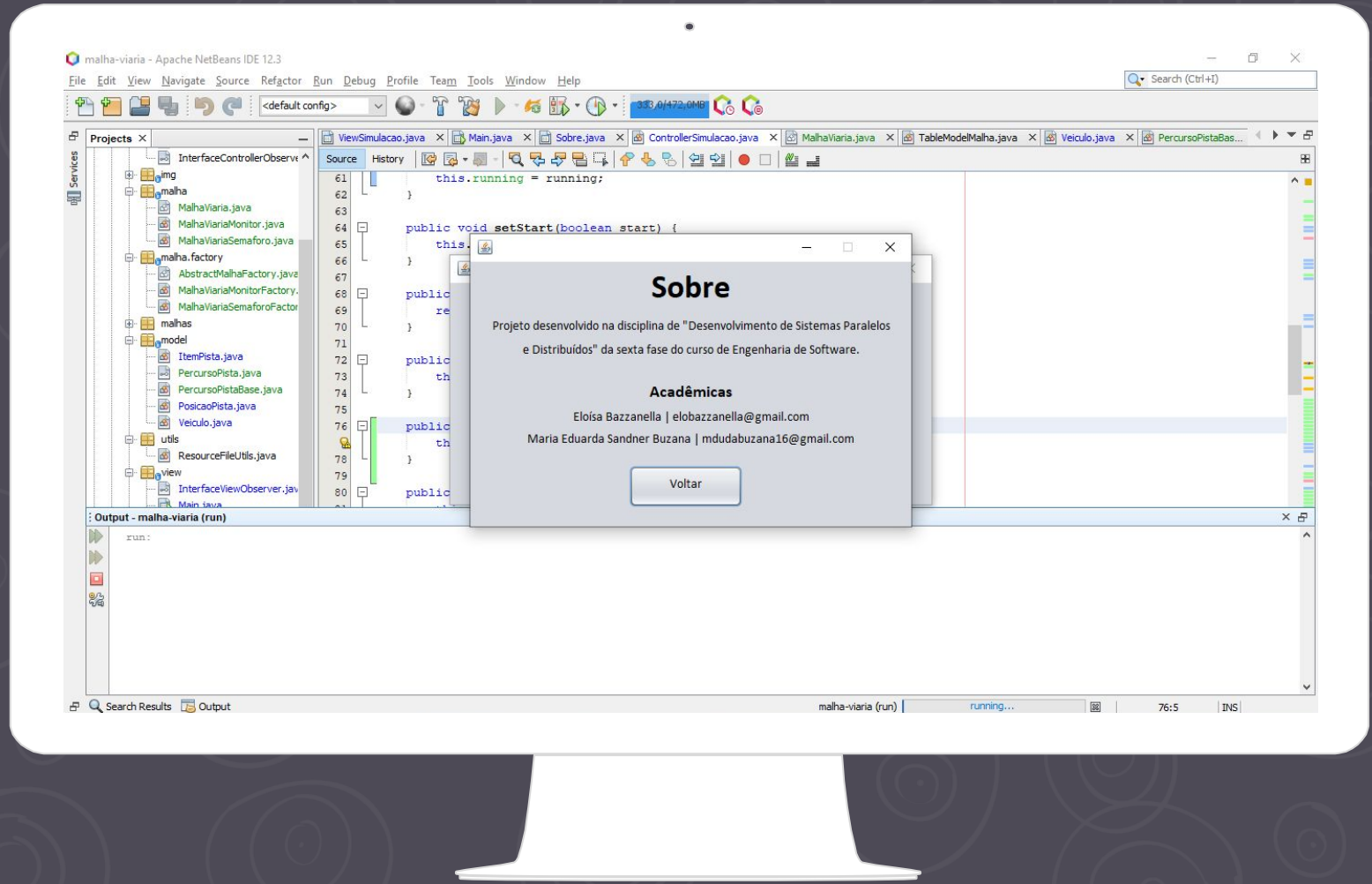
Limpa tudo e deixa pronto para executar de novo

Finalizar Simulação

3

Resultado


malha-viaria - Apache NetBeans IDE 12.3
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Services
ViewSimulacao.java
Main.java
Sobre.java
ControllerSimulacao.java
MalhaViaria.java
TableModelMalha.java
Veiculo.java
PercursoPistaBas...
Source
History
InterfaceControllerObserv
img
malha
MalhaViaria.java
MalhaViariaMonitor.java
MalhaViariaSemaforo.java
malha.factory
AbstractMalhaFactory.java
MalhaViariaMonitorFactory.
MalhaViariaSemaforoFactor
malhas
model
ItemPista.java
PercursoPista.java
PercursoPistaBase.java
PosicaoPista.java
Veiculo.java
utils
ResourceFileUtils.java
view
InterfaceViewObserver.jav
Main.java
this.running = running;
public void setStart(boolean start) {
this.start = start;
SIMULAÇÃO MALHA VIÁRIA
*Clique em "Carregar Malha" para selecionar uma malha viária e iniciar a simulação!
Carregar Malha
Sobre
Output - malha-viaria (run)
run:
Search Results
Output


malha-viaria - Apache NetBeans IDE 12.3
Sobre
Projeto desenvolvido na disciplina de "Desenvolvimento de Sistemas Paralelos e Distribuídos" da sexta fase do curso de Engenharia de Software.
Acadêmicas
Eloísa Bazzanella | elobazzanella@gmail.com
Maria Eduarda Sandner Buzana | mdudabuzana16@gmail.com
Voltar


Malha Viária
QUANTIDADE DE VEÍCULOS:
Executar utilizando Semáforo
Executar utilizando Monitor
Iniciar
Finalizar
Finalizar Definitivo


Malha Viária
QUANTIDADE DE VEÍCULOS:
30
Executar utilizando Semáforo
Executar utilizando Monitor
Iniciar
Finalizar
Finalizar Definitivo

3

Prática

Obrigada!

Alguma dúvida/sugestão?

Simulação Malha Viária

Eloísa Bazzanella e Maria Eduarda Buzana