## 65DSD - Trabalho 2 Malha Viária (Controle de Trânsito)

Lucas Martendal e Victor Beltramini

## Objetivo do Trabalho

- Simular uma Malha Viária de Trânsito, com carros, estradas e cruzamentos.
- Fazer com que os carros trafeguem pelas estradas da malha de maneira comportada.
  - Sem sair das estradas.
  - Sem bater ou "passar por cima" de outros carros.
  - Sem "cortar" ou bloquear cruzamentos.
- Simular carros com diferentes velocidades
- Dar autonomia aos carros para escolherem caminhos nos cruzamentos.
  - Aguardar caminho escolhido dos cruzamentos estar livre para passar (aguardar parado no semáforo).
  - Reservar caminho escolhido do cruzamento enquanto estiver passando, para não ser "cortado" ou bloqueado.
- Utilizar 2 métodos de controle dos cruzamentos/semáforos: Mutex e Monitor.

## Outros Objetivos do Trabalho

- Permitir o carregamento e a execução de vários mapeamentos de Malhas Viárias (diferentes topologias/layouts).
- Permitir a adição de vários carros à Malha, até um número máximo, pré-definido pelo usuário.
- Permitir a adição de carros à Malha em intervalos de tempo pré-definido pelo usuário.
- Implementar re-adição automática do carro à Malha, quando ele chega em um ponto de saída e sai dela.
- Implementar 2 botões de interrupção:
  - Botão para interrupção imediata de todos os carros na Malha.
  - Botão de interrupção de adição e re-adição de carros, permitindo que os atuais saiam naturalmente da malha.

### Mutex vs Monitor

#### Mutex:

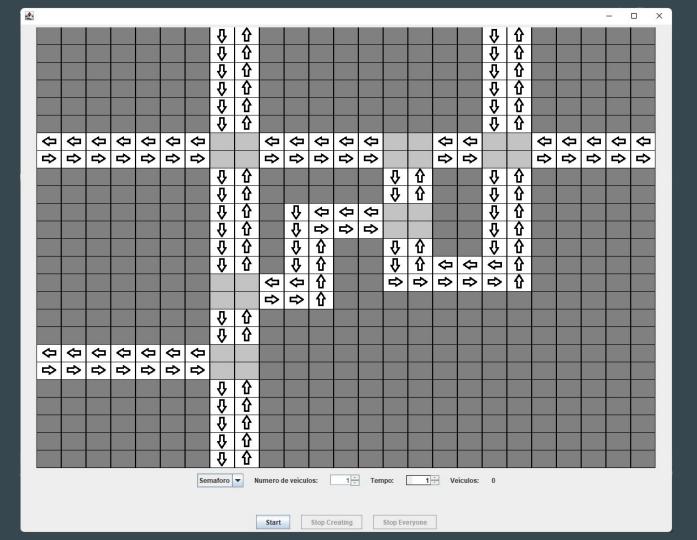
- Exclusão Mútua.
- Método de sincronização usado para controlar acesso a recursos compartilhados.
- Permite apenas uma Thread executar seção de código específica por vez.
- Utiliza a interface java.util.concurrent.locks.Lock do Java e implementa ReentrantLock.
- Usa as operações básicas lock() e unlock().

### Mutex vs Monitor

#### Monitor

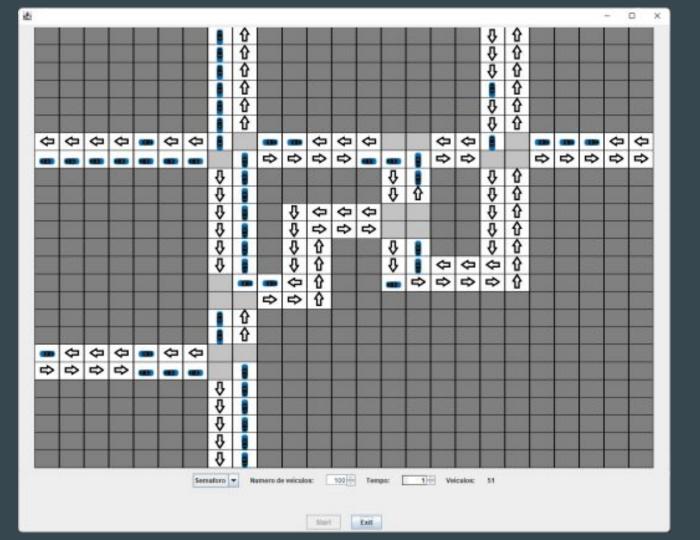
- Conceito de alto nível que combina Mutexes com variáveis condicionais.
- Em Java, cada objeto pode atuar como Monitor usando o lock intrínseco.
- Usa métodos como wait(), notify() e notifyAll().

## Interface do Simulador



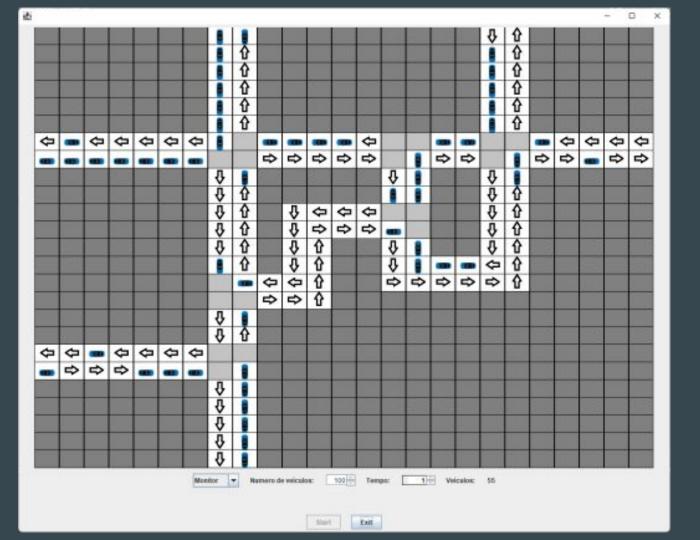
## Interface do Simulador

(Executando Semáforo/ Mutex)

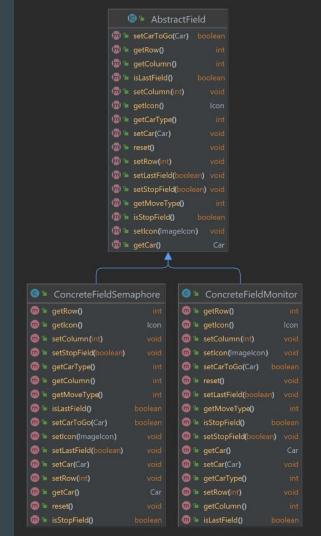


## Interface do Simulador

(Executando Monitor)



# Diagrama de Classes



|   | <b>™</b> Handler   |                         |  |
|---|--|-------------------------|--|
|   | updateFields()   |                         |  |
|   | changeSimulationType(String)   |                         |  |
|   | stop()   |                         |  |
|   | istart()   |                         |  |
|   | attach (Observer)  |                         |  |
|   | getMesh()  | Mesh                    |  |
|   | renderField(int, int)  |                         |  |
|   | <u> </u>   |                         |  |
| MeshHandler                                 |  |                         |  |
| (m) %                                       | start()  |                         |  |
| <b>⊕</b> %                                  | getFirstCell()   | nteger[]                |  |
| <b>1 1 1 1</b>                              | updateCarCount(Car)  |                         |  |
| <b>1</b>                                    | detach (Observer)  |                         |  |
| @ getCellAtPosition(int, int) AbstractField |  |                         |  |
| <b>⊚</b> •                                  | notifyExitButton(boolean)  |                         |  |
| (m) %                                       | addCars(Car)   |                         |  |
| (m) °L                                      | changeSimulationType(String)   |                         |  |
| (m) %                                       | getCars()  |                         |  |
| (m) The                                     | notifyCounter()  |                         |  |
| (m) Te                                      | attach (Observer)  |                         |  |
| <b>6</b>                                    | getSimulationType()  |                         |  |
| <b>1</b>                                    | stop()   |                         |  |
| (in   | setStopped(boolean)  |                         |  |
| <b>₽</b>                                    | getInstance() Mesh   | Handle                  |  |
| (m) (m)                                     | updateFields()   |                         |  |
| (m) %                                       | notifyStartButton(boolean)   |                         |  |
| <b>6</b>                                    | getMesh()  | Mesh                    |  |
| <b>6</b>                                    | notifyStopButton(boolean)  |                         |  |
| <b>@</b> •                                  | isStopped()  |                         |  |
| (m) %                                       | renderField(int, int)  |                         |  |
| 8 8 8                                       | notifyStartButton(boolean) getMesh() notifyStopButton(boolean) isStopped() | Mesh<br>void<br>boolear |  |

| © ⁰ Car    |                                |               |  |
|------------|--------------------------------|---------------|--|
| <b>@</b> • | updateFront()                  |               |  |
| <b>6</b>   | setColumn(int)                 |               |  |
| <b>6</b>   | getRow()                       |               |  |
| <b>@</b> = | getColumn()                    |               |  |
| <b>@</b> • | setOutOfRoad(boolean)          |               |  |
| <b>6</b>   | setField(AbstractField)        |               |  |
| <b>6</b>   | getSpeed()                     |               |  |
| <b>6</b>   | setSpeed()                     |               |  |
| <b>@</b> • | run()                          |               |  |
| <b>@</b> • | getField()                     | AbstractField |  |
| <b>@</b> • | setFirstPosition(Integer, Inte |               |  |
| @ <u>*</u> | getNextField()                 |               |  |
| <b>@</b> • | setNextField(AbstractField)    |               |  |
| <b>1</b>   | setRow(int)                    |               |  |

# Diagrama de Classes

