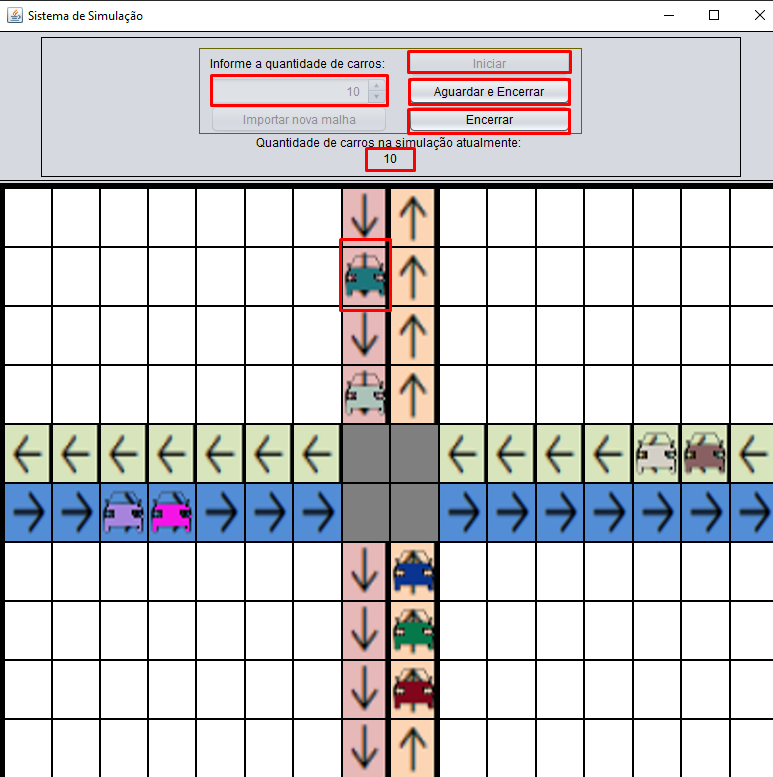


Configurações Iniciais

**Botão Caminho:** Classe configurações método de botaoAbrirListeners() vai simplesmente pegar o meu caminho absoluto e carregar ele no método de carregarArquivo() da classe de controle. Nessa classe de controle ele vai ser lido e mandado para a classe de LeituraMalha passando um texto que são os dados que estão dentro desse arquivo. Na classe LeituraMalha no método construtor basicamente vai passar as informações para o método de leituraArquivo() que vai ler as linhas e colunas da malha, e fazer a montagem dela passando as colunas e linhas, e por fim vai fazer ainda uma verificação na leitura desse arquivo. Ainda na classe de controle ele vai passar a leitura para a classe de controleArquivoMalha que vai realizar a obtenção da matriz em obterMatriz() na classe de LeituraMalha, assim esse método vai retornar a minha malhaMatriz que é um array bidimensional.

**Inserir Malha e Tipo de visualização (MONITOR E SEMAFOR):** Classe Configuracao método botaoCriarListeners() vai verificar qual o campo foi selecionado monitor ou semáforo. Por exemplo vou seguir a semáforo definirFabrica(“Semaforo”) no definirFabrica() que esta na classe Controle vai definir o tipo de fabrica. Por exemplo se seguir no SemaforoFabrica() vai para a classe de semáforo SemaforoFabrica() assim vai constuirCasa() passando dados como valor, colunas e linhas e criando uma nova instancia da classe de SemaforoCasas(), nessa classe e minha implementação do tipo semáforo que vai ter no seu construtor a passagem de um novo Semaphore, aceitando apenas 1 thread que é a quantidade que podem acessar simultaneamente o recurso. Quando ele fazer toda a definição de fabrica ele vai pegar a referencia de controle da malha e iniciar a malha no método de iniciarMalha() na classe ControleArquivoMalha assim iniciando as casas, extremidades e comandos. Basicamente se a telaPrincipal for diferente de nulo, ele vai redefinirMalhaObservadores() e remover eles.

**Cancelar:** Simplesmente vai sair da aplicação com o System.exit(0).



**Selecionar quantidade de carros:** Basicamente vai ter um spiner “qtdCarros” colendo as informações da quantidade de carros que vai entrar na simulação.

**Botão iniciar:**  Na classe Principal método botaoIniciarListeners() vai primeiramente desativar alguns botões e constante da quantidade de carros na simulação. Então basicamente a quantidade de carros vai ser passado na controle e iniciar a simulação passando essa quantidade. No método iniciarSimulacao() classe controle vai ser pegado essa quantidade de carros que foi passada e basicamente vai ser setado a simulação como true, e vai ter um for que vai percorrer essa quantidade de carros e adicionar ele no método de novoCarroMalha(), nesse método vai criar uma nova instancia de carro inserindo ele na quantidade de carros em espera e por fim adicionando ele na tabelaObservadorMalha, voltando para o método de iniciarSimulacao() ele vai criar um respawn “reaparecimento” que é uma thread addAutomatico da classe controle que é um executável para ativar a simulação percorrendo a quantidade de carros em espera e a quantidade de carros na malha, assim criando uma novoCarroMalha().

**Botão Aguardar e Encerrar:** Na classe Principal método botaopararListeners() basicamente vai desativar os botões de encerrar e o seu próprio, e acessar o controle da método stopReaparecimento() que basicamente vai encerrar o reaparecimento da variável ativadaSimulacao para false.

**Botão Encerrar:** Na classe Principal método botaoEncerrarListeners() vai basicamente encerrar na simulação desabilitando o botão encerrar. Entrando no método encerrarSimulacao() que esta na classe de controle vai basicamente dar um System.exit(0) e sair do programa.

**Quantidade de carros na simulação:**  Está na classe Principal basicamente vai setar na labelQtdAtualSimulacaoCarros uma variável que é a quantidade de carros que estão na simulação atualmente. No método notificarQtdCarros() vai passar na label labelQtdAtualSimulacaoCarros a quantidade de numCarro “numero de carros”.

**Classes e Métodos importantes**

- Controle

* carregarArquivo()

Basicamente responsável por fazer a leitura do arquivo da matriz e passar para a classe controleArquivomalha as especificações dessa malha como as linhas, colunas.

* iniciarSimulacao()

Basicamente responsável por fazer a inicialização da simulação recebendo um quantidade de carros que foi informado e também definindo a simulação como verdadeira para começar o preenchimento. Então vai percorrer a quantidade de carros e adicionando eles na lista de espera. E basicamente vai adicionar eles no reaparecimento no addAutomatico que faz a ativação definitiva da simulação.

* addAutomatico()

Responsável por realizar o executável da inicialização da simulação. Então ele vai fazer a ativação da simulação e realizar um for para somar a quantidade de carros em espera com a quantidade de carros na malha atualmente, assim tem que ser menor que a quantidade de carros que foi passado para iniciar a simulação. Com isso vai ser adicionado um novoCarroMalha() colocando ele em lista de espera e inserindo ele no observador.

- ControleArquivoMalha

* iniciarCasas ()

Basicamente vai pegar a fabrica que foi definida e realizar a quantidade de linhas e colunas que tem no arquivo, e fazer assim a inicialização das casas.

* definirExtremidadeCasa()

Basicamente responsável por realizar a definição das casa colunas e linhas, então basicamente se houver valor na casa significa que a mesma faz parte da malha. Então ele vai adicionando o reaparecimento da casa

* definirComandos()

Basicamente responsável por realizar os comandos de movimentação, então ele vai adicionar os comandos dentro da casa. Vai percorrer a estrutura para encontrar as possíveis rotas de cada casa.

* verificarCruzamento()

Basicamente vai verificar os valores da casa, se os valores forem diferentes de (1,2,3,4) quer dizer que é cruzamento.

* definirCaminhoCruz()

Vai definir o caminho dos cruzamento, então basicamente vai capturar os possíveis movimentos das entradas. Então ele vai verificar as casas validas e fazer a movimentação.

* verificarCasaValida()

Basicamente se o valor da casa for diferente de 0 que dizer que é valida.

- TabelaMalha

* TabelaMalha()

Responsável por inicializar a minha tabela passando o JPanel assim ele vai passar a controle para construir as especificações da tabela, adicionando os observadores.

* iniciarBuffert()

Responsável por fazer a inicialização das colunas e linhas, pegando o tamanho. Vai realizar a inicialização das propriedades e inicializar as imagens. Fazendo isso para repintar e revalidar.

* inicializarImagens()

Basicamente realizar as configurações de louty da tabela.

* inserirCarro()

Responsável por inserir o carro na tabela.

* movimentarCarro()

Fazendo a movimentação desse carro na tabela, definindo ele em um coluna e na linha.

* printar()

Responsável por realizar o printar a tabela.

- LeituraMalha

* obterMatriz ()

Basicamente vai verificar o que tem dentro da matriz assim para retornar a matriz para a classe de controle.

* leituraArquivo()

Vai realizar a leitura da quantidade de colunas e linhas assim criando ela, e vai verificar erros no arquivo.

- MapaImagens

* inicializacao ()

Vai realizar a inicialização das imagens

* obterImagem()

Dependendo a opção vai retornar a referencia da imagem

* getImagem()

vai retorna as referencia de inicialização das minhas imagens

* substituirCor()

Realizar a substituição das cores

* gerarCorAleatoriamente()

realizado pela classe Carro para gerar uma cor aleatória para o carro