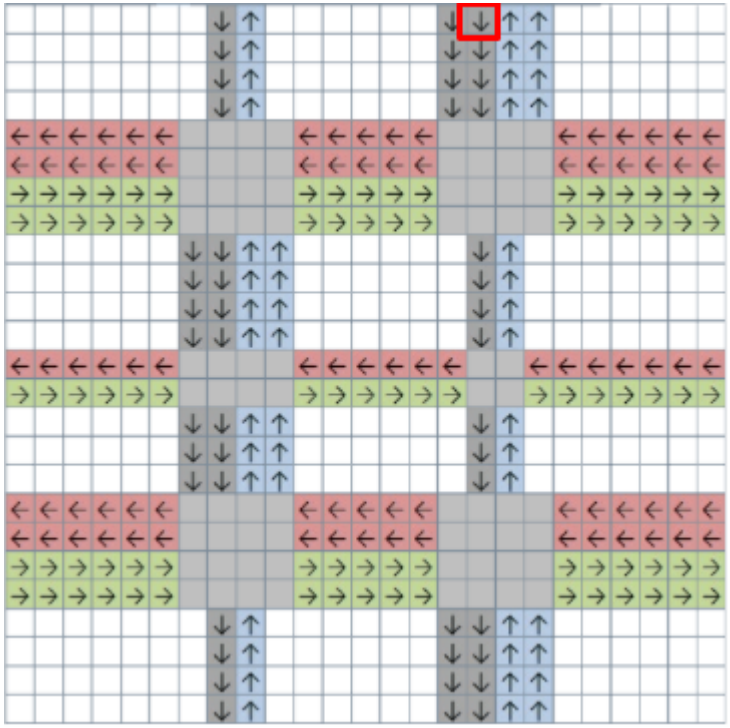
Supondo que o carro foi criado na parte superior da malha (aleatoriamente), em uma ViaBaixo (Imagem a seguir). A partir do carro:



```
andar() ->
    proximasVias = via.getProximasVias() //dependendo da via atual (neste exemplo é a via indicada na imagem acima)
```

```
    getProximasVias() ->
        direcoes = new ArrayList<>(); //Array de direções possíveis para a ViaBaixo
        direcoes.add("baixo"); // Pode ir para baixo
        direcoes.add("diagonal-baixo-esquerda"); // Pode ultrapassar pela esquerda
        direcoes.add("diagonal-baixo-direita"); // Pode ultrapassar pela direita

        return super.factoryProximasVias(direcoes); //Manda para o Method Factory gerar as Vias a partir das direções
```

```
        factoryProximasVias() ->
            proximas = new ArrayList<>(); //Array para verificar se há próximas vias para seguir
            ultrapassagens = new ArrayList<>(); //Array para verificar se há próximas vias de ultrapassagem para seguir

            for (String direcao : direcoes) { //Para cada direção indicada no próximas Vias
```

```
    switch (direcao) {} //Olha-se para qual os lados indicados (separa-se as diagonais para a ultrapassagem)

    if (via != null && !via.temVeiculo()) { //Se a via existir e NÃO ter nenhum veículo nela no momento, indicamos ela como uma possível próxima via
        proximas.add(via);
    }

    if (ultrapassagem != null && this.getClass() == ultrapassagem.getClass() && !ultrapassagem.temVeiculo()) { //Se a via de ultrapassagem existir e NÃO ter nenhum veículo nela no momento e o tipo da via for o mesmo que o tipo atual( para evitar ultrapassagens em vias de sentido contrário), indicamos ela como uma possível próxima via.
        ultrapassagens.add(ultrapassagem);
    }
}
return proximas.isEmpty() ? ultrapassagens : proximas; //se a lista de próximas vias estiver vazia significa que não tem caminho livre à frente, então é retornado a lista de ultrapassagem, caso contrário retorna-se as próximas vias (o carro fica com comportamento de ultrapassagem somente se a via à frente dele estiver ocupada)
```

```
if (!proximasVias.isEmpty()) ? //Tem vias livres para ocupar?
    proximaVia = proximasVias.get(Math.random() * proximasVias.size()) //pega aleatoriamente um índice da lista de próximas vias possíveis
    if (proximaVia instanceof Cruzamento) ? //verifica se a próxima via faz parte de um cruzamento
    (não é o caso do primeiro passo do carro, mas estaremos também explicado aqui, pois depois de algumas vias ele chegará no cruzamento)
        caminho = proximaVia.gerarCaminhoSaidaDeCruzamento()
```

```
        gerarCaminhoSaidaDeCruzamento() ->
        caminhosASeguir = calculaSaidasCruzamento(); //Gera todas as possibilidades de saída antes de entrar no cruzamento

        caminhoASeguir = caminhosASeguir.get(Math.random() * caminhosASeguir.size()); //Escolhe uma saída aleatória da lista gerada
        return caminhoASeguir;
```

```
    reservarCruzamento(caminho) // método para reservar as vias a serem percorridas no cruzamento (ponto crítico)
```

```
        reservarCruzamento() ->
            boolean tentativa;
            List<Via> reservadas = new ArrayList();
```

```
                    do {
                        tentativa = true;
                        for (Via viaCaminho : caminho) { // para cada via no caminho escolhido
                            tentativa = viaCaminho.tentaOcupar(); // tenta-se ocupa-la (no fundo usa-se o tryacquire com tempo aleatório)
                                                                  tryAcquire((int)(Math.random() * 300 + 100), TimeUnit.MILLISECONDS);
                            if (!tentativa) { // se não conseguir
                                for (Via reservada : reservadas) { // desocupe todas as vias já reservadas
                                    reservada.desocupar();
                                }
                                break;
                            } else {
                                reservadas.add(viaCaminho); // se conseguiu reservar, indique que já foi reservada
                            }
                        }
                    } while (!tentativa); // enquanto a tentativa de reservar todas as vias do cruzamento for falsa
                    return tentativa;
```

```
        desocupar() //desocupa a via atual
        for (Via viaCaminho : caminho) // Para cada via que já foi reservada
                via.setVeiculo(null) // Tiramos nosso veículo da via
                via = viaCaminho // colocamos a nova via do veículo como a próxima via da execução
                via.setVeiculo(this) // definimos o novo veículo da via como o carro que está executando a Thread
                viaCaminho.desocupar() //desocupamos as vias que já percorremos a fim de que outros carros já possam passar
else
// ## caso falso (Não é um cruzamento)
        irParaVia(proximaVia) ->
                proximaVia.ocupar(); //ocupamos o semáforo ou monitor da próxima via, ou seja, se tiver algum carro nela, o carro atual irá
                aguardar sair
                via.desocupar(); //desocupamos o semáforo ou monitor da via atual, ou seja, liberamos para outros carros obtê-la
                via.setVeiculo(null) // Tiramos nosso veículo da via
                via = viaCaminho // colocamos a nova via do veículo como a próxima via da execução
                via.setVeiculo(this) // definimos o novo veículo da via como o carro que está executando a Thread

… faça até (!via.isBordaSaida()); // o veículo executa até achar uma via que é uma das bordas da malha, ou seja, finaliza a execução
```