# Trens com *Threads*

- Descrição
  - ✓ Atualizar o seu trabalho anterior de modo que o código de movimentação do trem seja implementado dentro uma *thread*
    - Todo o resto do código, a princípio, deve permanecer igual
    - A não ser que você queira melhorar algo no seu código anterior
  - ✓ Assim, você terá 2 *threads* em seu trabalho, uma para cada trem
  - ✓ Em algum ponto do seu código, possivelmente mas não necessariamente no método *main*, você irá dar o *start* nas *threads*
  - ✓ Veja um esboço de código para as 2 orientações de trilhos nos próximo *slides*

# Trens com *Threads*

- Exemplo de código para a orientação horizontal

```
class TremDeCima extends Thread {
  public void run() {
    // no run vem toda a sua logica de movimentacao do trem (de cima)
  }//fim do metodo run
}//fim da classe TremDeCima

class TremDeBaixo extends Thread {
  public void run() {
    // no run vem toda a sua logica de movimentacao do trem (de baixo)
  }//fim do metodo run
}//fim da classe TremDeBaixo

public class Principal {
    static public void main(String s[]) {

    //em algum ponto (inicial) do seu codigo voce ira instaciar
      e startar as threads
    TremDeCima Tc = new TremDeCima();
    TremDeBaixo Tb = new TremDeBaixo();
    Tc.start();
    Tb.start();

  }//fim do metodo main
}//fim da classe principal
```

# Trens com *Threads*

- Exemplo de código para a orientação vertical

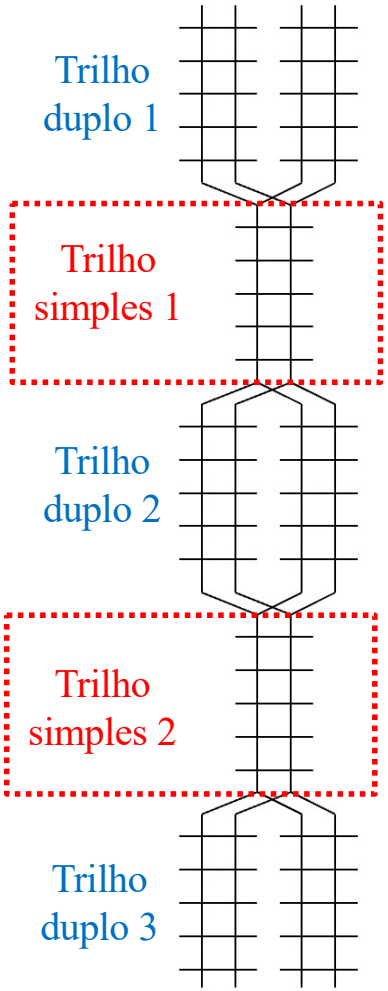
```
class TremDaEsquerda extends Thread {
  public void run() {
    //no run vem toda a sua logica de movimentacao do trem (da esquerda)
  }//fim do metodo run
}//fim da classe TremDaEsquerda

class TremDaDireita extends Thread {
  public void run() {
    //no run vem toda a sua logica de movimentacao do trem (da direita)
  }//fim do metodo run
}//fim da classe TremDaDireita

public class Principal {
    static public void main(String s[]) {

    //em algum ponto (inicial) do seu codigo você ira instaciar
      e startar as threads
    TremDaEsquerda Te = new TremDaEsquerda();
    TremDaDireita Td = new TremDaDireita();
    Te.start();
    Td.start();

  }//fim do metodo main
}//fim da classe principal
```

# Trens com *Threads* (quase) sem colisão

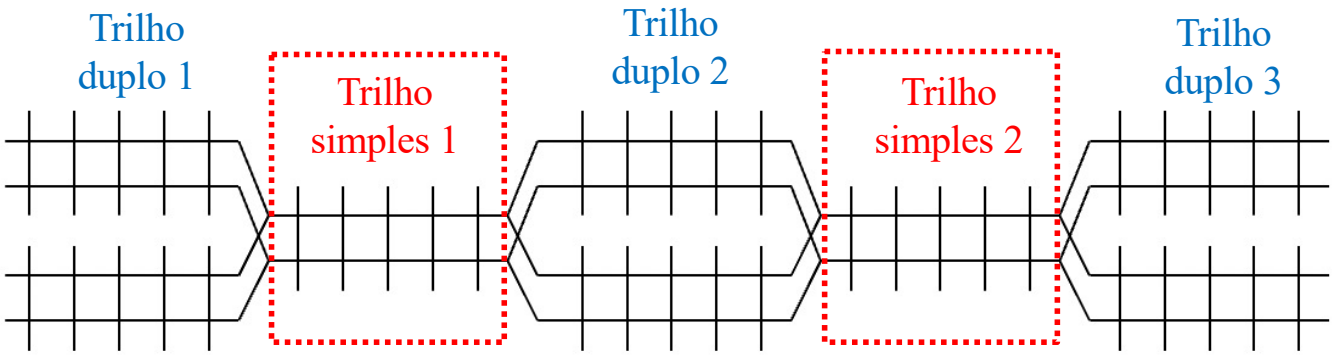
- Descrição
  - ✓ Após **atualizar** o trabalho para utilização de *threads*, **implemente** a exclusão mútua conforme orientações a seguir
  - ✓ No acesso ao recurso compartilhado (no caso o trilho compartilhado) acrescentar código de programação para evitar os efeitos das condições de corrida
  - ✓ Três algoritmos devem ser implementados e a escolha de qual estará sendo utilizado no momento será feita pelo usuário via GUI
    - Variável de travamento
    - Estrita alternância
    - Solução de Peterson
  - ✓ Perceba que durante a execução do programa, quando o usuário escolher uma nova opção/algoritmo os trens devem recomeçar na posição original/inicial
    - Para isto, basta chamar sua sub-rotina de *reset*

# GUI (*Graphical User Interface*)

**Orientação vertical**




**Orientação horizontal**




- Agora vamos nos preocupar e começar a resolver o problema da "colisão" dos trens no trecho compartilhado (trilho simples)
- Utilizando os algoritmo explicados em aula
- Todas as demais demais características do projeto permanecem iguais
  - Comportamento ao atingir o fim do percurso
  - Mecanismo para variar a velocidade dos trens individualmente
  - Botão *reset* para reiniciar a simulação