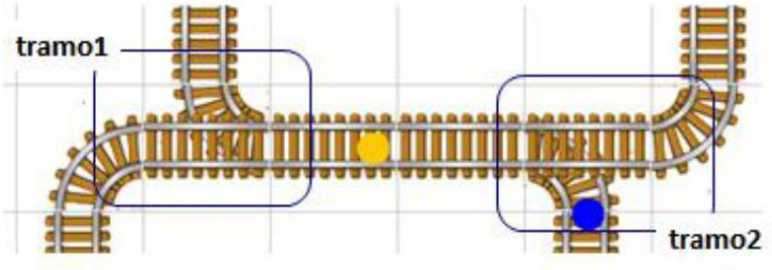
Trenes

En abstracto, tenemos una zona de vía compartida por varios trenes y queremos controlar la entrada de ternes por uno y otro lado




```
public class MonitorTunel
        extends Monitor {
    private final Tramo tramo1;
    private final Tramo tramo2;

    public MonitorTunel(Tramo tramo1, Tramo tramo2) {
        this.tramo1 = tramo1;
        this.tramo2 = tramo2;
    }

    // la thread se detiene hasta que pueda entrar
    public void entro(Tren tren, Tramo tramo, Enlace entrada) {
    }

    // la thread sale
    public void salgo(Tren tren, Tramo tramo, Enlace salida) {
    }

}
```

Se proponen 4 ejercicios de control del tramo compartido:

Ejercicio 1

El monitor debe controlar que en el tramo de vía compartido nunca hay más de 1 tren en cada momento. Es decir, o está vacío o está ocupado con 1 tren.

Cuando un tren quiere entrar y hay otro tren dentro, queda esperando hasta que la vía esté libre.

Cuando un tren sale, se lo notifica a los trenes que estén esperando.

Ejercicio 2

Al igual que el ejercicio 1, no puede haber más de un tren dentro. Pero además, cuando un tren sale, tiene preferencia para entrar un tren que esté esperando en dirección opuesta.

Se trata de hacer que el recurso se comparta de forma equitativa (fair).

Ejercicio 3

Se permite que haya varios trenes circulando en la misma dirección por el tramo compartido.

Se trata de optimizar el uso de un recurso compartido.

Ejercicio 4

Al igual que el ejercicio 3, puede haber varios trenes circulando en el mismo sentido. Pero además, cuando un tren sale, tiene preferencia para entrar un tren que esté esperando en dirección opuesta. Es decir, que un nuevo tren sólo entra si no hay nadie esperando en sentido contrario y si no hay nadie circulando en sentido contrario.

Se trata de hacer que el recurso se comparta de forma equitativa (fair).