Solución Ejercicio Trenes

Ejercicio 1

Sólo se admite 1 tren dentro en cada momento. Los demás, esperan.

Variables de estado:

boolean ocupado = false

TRUE si hay un tren dentro

```
public class MonitorTunel_1
        extends Monitor {
    private final Tramo tramo1;
    private final Tramo tramo2;

    private boolean ocupado = false;

    public MonitorTunel_1(Tramo tramo1, Tramo tramo2) {
        this.tramo1 = tramo1;
        this.tramo2 = tramo2;
    }
```

```
    public synchronized void entro(
        Tren tren, Tramo tramo, Enlace entrada) {
        while (ocupado)
            waiting();
        ocupado = true;
    }
```

```
    public synchronized void salgo(
        Tren tren, Tramo tramo, Enlace salida) {
        ocupado = false;
        notifyAll();
    }
```

```
    private void waiting() {
        try {
            wait();
        } catch (InterruptedException ignored) {
        }
    }
}
```

Ejercicio 2

Sólo puede haber un tren dentro; pero se hace un reparto equitativo de quién puede entrar alternando la entrada por uno u otro tramo.

Variables de estado:

int esperando1 = 0;

Lleva la cuenta de cuántos trenes hay esperando entrar por el tramo 1.

int esperando2 = 0;

Lleva la cuenta de cuántos trenes hay esperando entrar por el tramo 2.

boolean ocupado = false;

TRUE si hay un tren dentro.

int ultimaEntrada = 1;

Memoriza si la última entrada fue por el tramo 1 o por el tramo 2.

```
public class MonitorTunel_2
        extends Monitor {
    private final Tramo tramo1;
    private final Tramo tramo2;

    private int esperando1 = 0;
    private int esperando2 = 0;
    private boolean ocupado = false;
    private int ultimaEntrada = 1;

    public MonitorTunel_2(Tramo tramo1, Tramo tramo2) {
        this.tramo1 = tramo1;
        this.tramo2 = tramo2;
    }
```

```
    public synchronized void entro(
        Tren tren, Tramo tramo, Enlace entrada) {
        if (tramo.equals(tramo1)) {
            esperando1++;
            while (ocupado
                || ultimaEntrada == 1 && esperando2 > 0)
                waiting();
            esperando1--;
            ocupado = true;
            ultimaEntrada = 1;
        }
        if (tramo.equals(tramo2)) {
            esperando2++;
            while (ocupado
                || ultimaEntrada == 2 && esperando1 > 0)
                waiting();
            esperando2--;
            ocupado = true;
            ultimaEntrada = 2;
        }
    }
```

```
    public synchronized void salgo(
        Tren tren, Tramo tramo, Enlace salida) {
        ocupado = false;
        notifyAll();
    }
```

```
    private void waiting() {
        try {
            wait();
        } catch (InterruptedException ignored) {
        }
    }
}
```