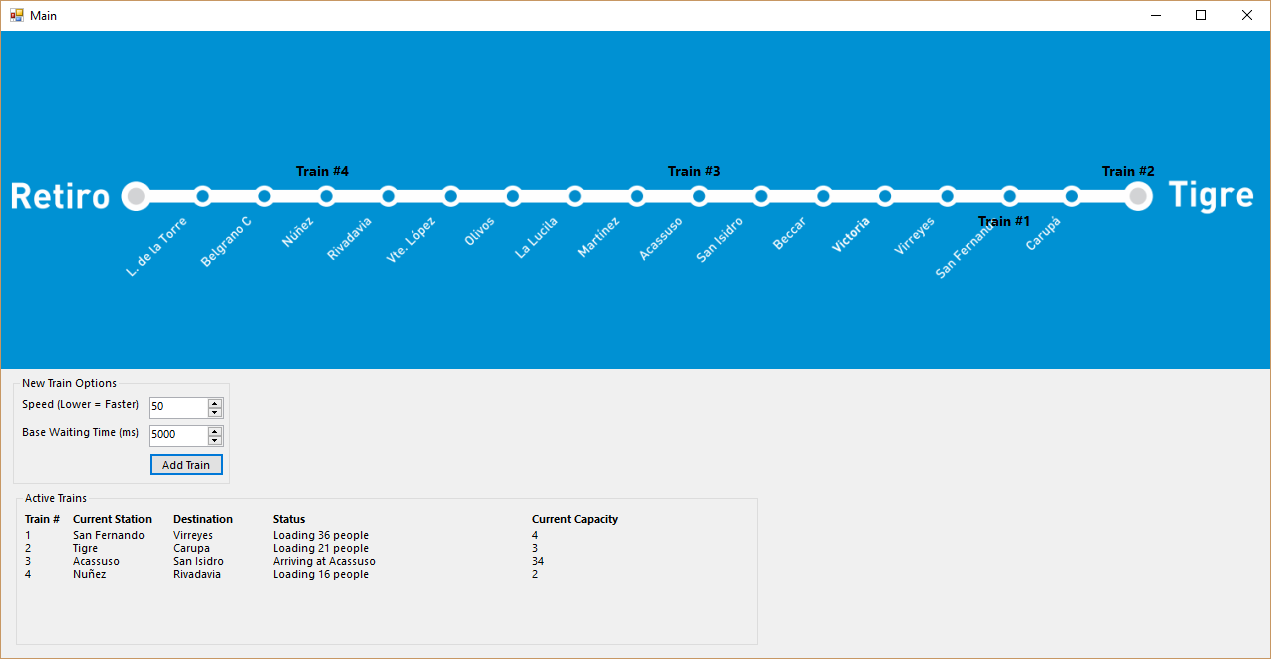
Victor Grycuk

Trabajo Practico de Multi-Threading

Práctica Profesional

# Descripción

Se realizó un simulador de línea de tren, en el cual se pueden agregar varios trenes corriendo en diferentes hilos sobre la misma línea.



# Funcionamiento

Se crearon 3 clases para implementar el funcionamiento: Central, Station, y Train.

Cuando un tren llega a una estación, descarga gente del tren, carga gente de la estación, valida si la estación siguiente está libre, y ahí hay dos alternativas.

Si la estación está libre, libera la estación actual y pasa a ocupar la siguiente.

Si la estación no está libre, se queda esperando a que el tren de la estación siguiente le de la señal de avanzar. Para este último casi también se tiene en cuenta un tiempo de *time out* en el que si no recibió señal de avanzar vuelve a validar si la estación siguiente está libre.

## Station



La clase Station varias propiedades, y una sola función para generar un numero al azar entre 0 y 50 para simular la cantidad de gente esperando en la estación.

FordwardTrack y BackwardTrack son para diferenciar los trenes que van en diferentes sentidos. Puede haber dos trenes en la misma estación siempre y cuando vayan en diferentes sentidos.

PreviousStation y NextStation son para encadenar las estaciones entre sí. Las excepciones son la estación inicial la cual no tendrá estación anterior, y la estación final que no tendrá estación siguiente.

MapLocation contiene las posiciones X e Y del mapa de estaciones. Es usado por el label del tren para actualizar un tren en el mapa cuando llega a la estación.

## Central

### Propiedades



Stations simplemente tiene todas las estaciones encadenadas entre sí.

En lugar de simplemente iniciar los trenes, son agregados a la cola de \_trainDeposit, un proceso valida si la estación inicial está liberada, y si lo está va quitando trenes del deposito y los agrega a la estación inicial. Similar a un modelo de productor-consumidor.

\_activeTrains es una lista de los trenes que se encuentran funcionando en la línea.

Locker es el objeto usado para sincronizar los threads usados para monitorear la estación inicial.

### Constructor



Dentro del constructor se crea la lista de estaciones, y luego se cicla para cada elemento para generar el encadenamiento de estaciones, agregando como estación anterior el elemento anterior, y al elemento anterior se agrega como estación siguiente el elemento actual.

La ubicación de la estación en el mapa semi automatizado, calculando la distancia ente cada estación cada 62 pixeles.

Finalmente, se inicializan dos procesos, el depositMonitor y el dispatcher. A continuación, se muestran en detalle.

### Métodos



depositMonitor valida si hay trenes en la cola de trenes a la espera de ser iniciados, y si la estación inicial está desocupada. De ser así da paso al siguiente proceso, si no se queda continuamente validando.



Dispatcher inicia a la espera de que DepositMonitor lo despierte para despachar trenes. Una vez iniciado, quita un tren de la cola de espera, le asigna un numero basado en la cantidad de trenes actuales, le asigna la estación inicial, lo agrega en la estación inicial, y lo inicia.

Finalmente despierta a DepositMonitor.



AddTrain es el método público usado por el formulario principal para agregar trenes. Simplemente agrega un tren a la cola de espera de trenes y despierta los procesos que validan la cola de trenes.



UpdateStatus actualiza los controles de la UI correspondientes al tren que lo llama. Estos controles son los que se muestran en la tabla en el form principal.



Similar a UpdateStatus, actualiza la posición del tren en el mapa.

## Train

### Propiedades



Number es el numero de tren asignado al momento de agregarlo a la línea de tren.

CurrentCapacity es la cantidad de personas actualmente en el tren, también es usado para calcular el *time out* cuando queda a la espera. A mayor cantidad de gente en el tren, menor es el tiempo del *time out*.

Speed es usado para calcular cuánto tiempo un tren va a tardar en cargar y descargar personas.

BaseWaitingTime es el time *out base*, junto con la cantidad de gente actual se calcula el tiempo que el tren queda a la espera antes de volver a validar si la siguiente estación está libre.

Status es una descripción del estado actual del tren.

Speed es usado para calcular cuánto tiempo un tren va a tardar en cargar y descargar personas.

ForwardDirection Indica en qué dirección se dirige el tren. Usado para saber en qué andén de la estación debe posicionarse.

Waiting indica si el tren está esperando luz verde para avanzar a la siguiente estación. Usados por otros trenes para saber si deben hacer pulse.

CurrentStation es la estación donde está actualmente posicionado el tren.

Los labels son usados para mostrar en la UI las propiedades del tren.

Locker se usa para sincronizar los threads al momento de validad las estaciones.

### Constructor



El constructor simplemente inicializa el objeto Random(), y setea la capacidad actual en cero.

### Métodos



Simplemente actualiza el mapa con la posición inicial, y llama a Process.



Loop de 3 métodos, de los cuales GoNextStation() contiene el código sincronizado.



Ambos métodos son muy parecidos, ya que utilizan cargan o descargan gente usando un numero al azar usando un tope, actualizan el estado, y utilizan Thread.Sleep() para simular el tiempo que las personas tardan en bajar y subir del tren.

El tiempo que demora en subir o bajar está ligado a la cantidad de personas. Mientras más personas bajen o suban del tren, más se va a demorar en salir de la estación.

Se agregó un +1 para evitar escenarios en que 0 personas bajan o suben del tren haciendo que se ejecute Thread.Sleep(0), el cual provoca un funcionamiento no deseado.



Llama al método de Central para reportar el estado actual del tren.



Calcula el tiempo en que un tren espera antes de volver a validad si la siguiente estación está liberada, en casos en que no haya recibido la luz verde por parte de otro tren. Mientras más gente haya en el tren, menos es el tiempo esperado antes de volver a validar.



Este método contiene el código sincronizado, usado principalmente para hacer el cambio de estación.

Inicialmente verifica si necesita cambiar de dirección y obtiene la siguiente estación a cuál avanzar.

Luego verifica si la estación siguiente está liberada, de no estarlo se queda a la espera a recibir el aviso de avanzar, o si se cumple el tiempo de espera, vuelve a validar si está libre.

De recibir luz verde de avanzar o la estación está liberada, avanza a la siguiente estación primero validando si hay algún tren en la estación anterior a la espera de recibir la luz verde, de haberlo hace PulseAll, para despertar todos los trenes en estado de esperar para que vuelvan a validar su estación siguiente.

Luego cambia de estación, y finalmente actualiza el estado en el mapa de la línea.



Cuando valida si debe cambiar de dirección lo hace en base a en qué dirección está yendo y si la estación actual tiene o no estación siguiente o anterior.

Básicamente indica si se llegó a alguna de las dos terminales.



Contiene la lógica de cambio de estación, validando en qué andén debería posicionarse dependiendo en cual de las dos terminales está.



Contiene la lógica para para validar si debería despertar algún thread que esté a la espera, valiando si la estación siguiente o anterior (dependiendo si es terminal y cual) tiene algún tren en espera.

## Main

### Propiedades



Solo tiene una propiedad, que es la instancia de la central.

### Métodos



Simplemente inicializa una nueva instancia de Central.



Genera una nueva instancia de Train, asignando las propiedades Speed y BaseWaitingTime según lo que se haya configurado en la UI. También inicializa los *labels*.

Luego agrega a la tabla de trenes activos cada label del tren. Finalmente agrega el tren a la central.



Los label de la posición actual de cada tren requieren una configuración diferente ya que para que sea transparentes es necesario asignarles la imagen de fondo como *parent*.

Adicionalmente, no son visibles hasta que el tren no es iniciado.