Contenido

[Introducción 1](#_Toc527286004)

[Antecedentes 1](#_Toc527286005)

[Entorno de Desarrollo 1](#_Toc527286006)

[Enunciado Practico 1](#_Toc527286007)

[Explicación del problema a resolver 1](#_Toc527286008)

[Solución 2](#_Toc527286009)

[Descripción y justificación de la solución 2](#_Toc527286010)

[Código fuente de la aplicación e interfaces 2](#_Toc527286011)

[Explicación 3](#_Toc527286012)

[Algoritmos 3](#_Toc527286013)

[Clases 3](#_Toc527286014)

[Estructuras de datos 4](#_Toc527286015)

[Listado de pruebas 4](#_Toc527286016)

[Entrada 4](#_Toc527286017)

[Ejecución 4](#_Toc527286018)

[Salida 4](#_Toc527286019)

[Conclusiones 4](#_Toc527286020)

# Introducción

## Antecedentes

El programa ha sido diseñado en un ordenador con sistema operativo Windows 7 con 8Gb de memoria RAM, se ha necesitado la creación de varias carpetas y ficheros

## Entorno de Desarrollo

El entorno de Desarrollo ha sido java con eclipse.

# Enunciado Practico

## Explicación del problema a resolver

* Implementar una aplicación que simule una línea de metro.
* La línea consta de N estaciones y un máximo de N-1 Trenes.
* En cada estación existe un semáforo que inicialmente se encuentra en verde y se pone en rojo cuando el tren sale de la estación.
* Cuando un tren sale de la estación siguiente pone en verde el semáforo de la estación anterior.

# Solución

## Descripción y justificación de la solución

Se ha tenido que crear una clase Tren que es la encargada de saber la distancia que tienen recorridos los trenes y de llamar a las estaciones.

Se ha tenido que crear una clase Estaciones que es la que se encarga de controlar los semáforos y su funcionamiento

Se ha tenido que crear una clase llamada Semáforos para albergar todos los semáforos de las estaciones

Se ha tenido que crear la clase Línea que es la encargada de ejecutar todas las otras clases

## Código fuente de la aplicación e interfaces

**package** Trenes;

**import** javax.swing.JFrame;

**public** **class** Linea **extends** JFrame {

**private** **static** Tren *a\_tren*[];

**private** **static** **final** **int** ***NUM\_TREN***=3;

**private** **static** **final** **int** ***NUM\_ESTACIONES***=4;

**public** **static** **void** main (String[] args) {

*a\_tren*=**new** Tren[***NUM\_TREN***];

System.***out***.println("Lanzo los trenes");

**for** (**int** i=0;i<***NUM\_TREN***;i++) {

*a\_tren*[i]=**new** Tren(i+1,***NUM\_ESTACIONES***);

*a\_tren*[i].start();

}

}

}

**package** Trenes;

**import** java.util.concurrent.Semaphore;

**public** **class** Semaforos {

**public** **static** Semaphore *estacion1*=**new** Semaphore(1);

**public** **static** Semaphore *estacion2*=**new** Semaphore(1);

**public** **static** Semaphore *estacion3*=**new** Semaphore(1);

**public** **static** Semaphore *estacion4*=**new** Semaphore(1);

}

**package** Trenes;

**import** java.util.Random;

**public** **class** Tren **extends** Thread {

**private** **int** numEstaciones;

Tren(**int** i, **int** numEstaciones) {

**this**.setName("tren" + i);

**this**.numEstaciones=numEstaciones;

}

**public** **void** run() {

**int** vuelta = 0;

**while** (vuelta <= 10) {// **numero de vueltas**

**try** {

Estacion.*estacion*(getName());

} **catch** (InterruptedException e) {

e.printStackTrace();

}

vuelta = vuelta + 1;

System.***out***.println("soy " + getName() + " he terminado la vueta " + vuelta);

}

}

**public** **static** **void** recorrido(**int** disARecorrer, String nombre) {

**int** dis = 0;

**int** disT = 0;

Random ran = **new** Random();

**do** {

dis = ran.nextInt(70);

disT = dis + disT;

System.out.println("Soy " + nombre + " me quedan " + (disARecorrer - disT) + " metros");

**try** {

Tren.*sleep*(300 - dis);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

} **while** (disT <= disARecorrer - 70);

disT = 0;

dis = 0;

}

}

**package** Trenes;

**import** **static** Trenes.Semaforos.*estacion1*;

**import** **static** Trenes.Semaforos.*estacion2*;

**import** **static** Trenes.Semaforos.*estacion3*;

**import** **static** Trenes.Semaforos.*estacion4*;

**import** java.util.concurrent.Semaphore;

**public** **class** Estacion {

**final** **static** Semaphore ***semaforos***[] = { *estacion1*, *estacion2*, *estacion3*, *estacion4* };

**final** **static** **int**[] ***dis*** = { 100, 150, 50, 200 };

**public** **static** **void** estacion(String nombre) **throws** InterruptedException {

**int** numEstaciones = 4;

**for** (**int** i = 0; i <= numEstaciones - 1; i++) {

Tren.*sleep*(1000);

***semaforos***[i].acquire();

System.out.println("Soy el " + nombre + " estoy cogiendo pasajeros en la parada 1");

**if** (i < 1) {

***semaforos***[3].release();

} **else** {

***semaforos***[i - 1].release();

}

System.***out***.println("Soy el " + nombre + " Saliendo " +(i+1));

Tren.*recorrido*(***dis***[i], nombre);

}

}

}

# Explicación

## Algoritmos

La utilización de varios for es debida a la necesidad de repetir el mismo código un número repetido de veces concretas, mientras que la utilización del while es debido a que queremos que de un numero de vueltas completa a la linea

## Clases

La clase principal llamada línea que se encarga de lanzar los hilos 3 ya que hay 4 estaciones

La clase Semáforos que se la que contiene los cuatro semáforos

La clase tren que se encarga de crear los hilos y de controlar el número de vueltas

La clase estación que se encarga de controlar los semáforos

## Estructuras de datos

La estructura de datos son ints que controlan las vueltas, la distancia y el número de trenes

# Listado de pruebas

## Entrada

No hay datos de entrada que analizar lo único que se ha podido analizar

## Ejecución

Las pruebas de ejecución han sido mandar más trenes que paradas y lo que sucede es que al terminar una vuelta se cuela otro tren sin y el anterior se queda en espera

## Salida

Los únicos datos de salida son la creación de los trenes, los trenes recogiendo pasajeros, la distancia recorre el tren, y el tren saliendo de la línea

# Conclusiones

Un trabajo muy curioso sobre semáforos al cual no le he conseguido poner entorno ya que no ha habido forma de mover los trenes.