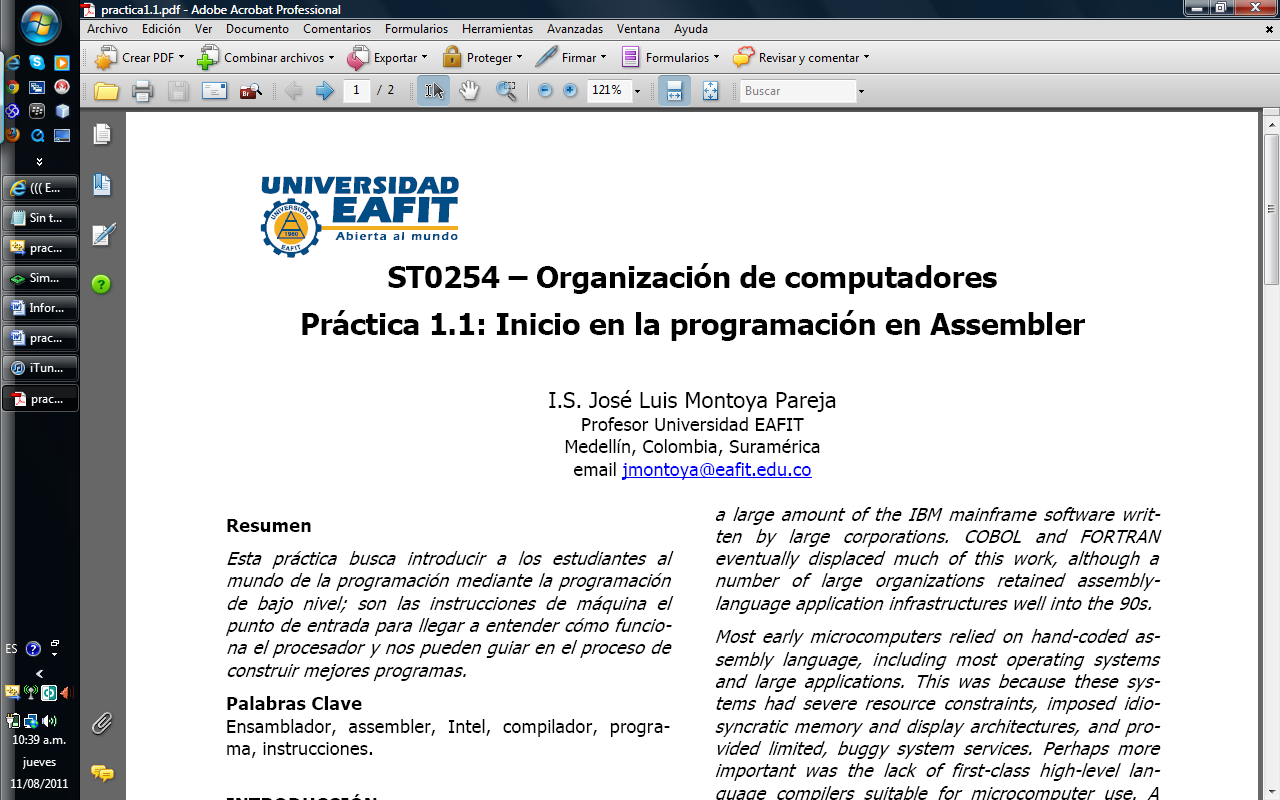
****

**ST0254 – Organización de computadores**

**Práctica 2: Final**

***Ernesto Quintero, Luisa Fda. Querubín, Jorman Bustos***

*Universidad EAFIT*

*Medellín, Colombia, Suramérica*

[equinte5@eafit.edu.co](mailto:equinte5@eafit.edu.co)

[lquerubi@eafit.edu.co](mailto:lquerubi@eafit.edu.co)

[jbustos@eafit.edu.co](mailto:jbustos@eafit.edu.co)

**Resumen**

**Tema 2: Karel AirCargo**

Karel AirCargo es una pequeña compañía de carga especializada en el transporte de beepers desde las dos fábricas hasta los 4 puntos de distribución de la ciudad Sur. La idea es hacer la programación en java para realizar el transporte de sirenas.

**Objetivos**

**1. Objetivo General**

Construir un programa Java con el framework de KarelJRobot que permita a Karel y siete robots mas, transportar las sirenas desde las fábricas en la ciudad Norte a los puntos de distribución en la ciudad Sur.

**2. Objetivos Específicos**

**a.** Crear 2 robots tipo Karel que transporten las sirenas desde las fábricas ubicadas en los puntos (20,3) y (20,7) y los lleven a la bodega del aeropuerto de la ciudad norte (16,8). Estos pueden llevar hasta 5 sirenas.

**b.** Crear 2 robots tipo Karel que transporten hasta 20 sirenas entre los aeropuertos de las ciudades Norte (16,8) y Sur (8,4).

**c.** Crear 4 robots que lleven las sirenas desde la bodega del aeropuerto de la ciudad Sur (8,4) a los centros de distribución ubicados en los puntos (1,2), (1,5), (1,7) y (1,10).

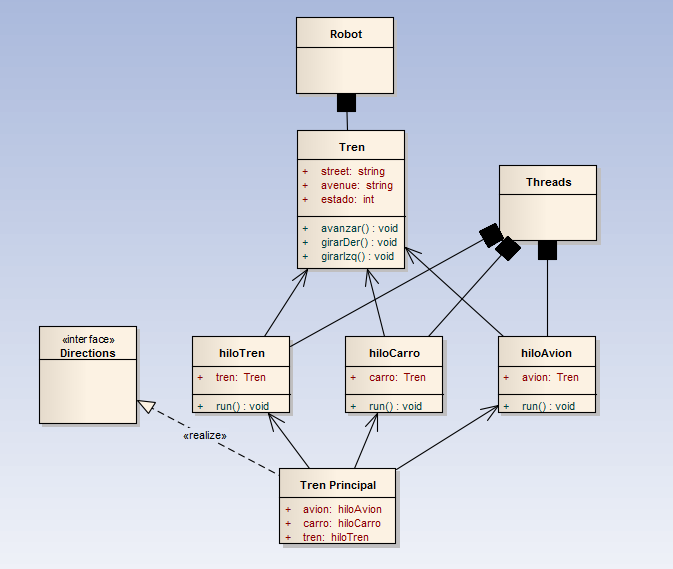
**Especificaciones del Programa**

* Cada robot es independiente uno del otro, es decir, no deben intercambiar o compartir información.
* En cada bodega solo puede haber un robot avión al tiempo, y los dos deben despegar al mismo tiempo.
* Como norma, los robots avión deben esperar hasta 1 minuto para que le lleven las sirenas

**Desarrollo**

Para poder dar solución a este problema tuvimos en cuenta los siguientes aspectos:

* Para poder orientarnos a través del mundo construimos una matriz que lo representara, llena de 0 y 1, 1 para las paredes que se encuentren presentes (las del borde y las que delimitan las zonas por donde pueden moverse los robots) y con el doble de tamaño, ya que también consideramos las posiciones intermedias entre los puntos dados por el mundo, ya que es allí donde se van a encontrar las paredes que delimitan las distintas zonas.
* Creamos 3 clases para los diferentes tipos de robot, una para los de la ciudad norte, otra para los del aeropuerto y otra para los de la ciudad sur. En cada uno de ellos se implementará un hilo que permitirá realizar las funciones correspondientes a cada una de las clases. Dicho hilo se ejecutará desde la clase principal que se encargará de crear 2 hilos para la ciudad norte, 2 hilos para el aeropuerto y 4 hilos para la ciudad sur, con las correspondientes coordenadas desde donde arranca cada uno de los robots.



* La implementación de cada una de las clases que contienen los hilos será de la siguiente manera: se tiene un método bfs que calcula la mínima distancia entre dos puntos cualquiera (entre el punto de partida y el lugar donde tiene que recoger las sirenas, entre el punto de recolección de sirenas y el punto de descargue y entre el punto de descargue y de recolección de las sirenas), un método que permite al robot desplazarse siguiendo esta ruta, según sea moverse a la derecha, a la izquierda, arriba o abajo. Otro aspecto muy importante es que utilizamos una variable llamada estado, que nos permite saber en qué estado se encuentra el robot. Definimos dos estados, el estado = 0 que es aquel que me permite desplazarme desde un punto cualquiera hasta el lugar donde tengo que cargar los beeper y una vez los cargue tomaré el estado = 1; y un estado = 1 que es cuando el robot está llevando beepers hasta el punto de descarga, una vez termine de descargar la variable estado tomará el valor 0 y procederá a actuar conforme a esta variable.

Bolsa= llena

Bolsa = vacia

**Funcionamiento de la Aplicación**

Para poder ver funcionando la aplicación procedemos de la siguiente manera:

Desde consola compilamos las diferentes clases con el siguiente comando:

*javac -d . -cp .;KarelJRobot.jar Tren.java*

Para ejecutar procedemos de la siguiente manera:

*java -cp .;KarelJRobot.jar TrenPrincipal*

Es importante señalar que para poder correr la aplicación en la misma carpeta donde estén guardadas las clases, debe estar también karelJRobot.jar

**Referencias**

<http://www.cs.tufts.edu/comp/10F/JKarel.htm>

<http://www.slideshare.net/czelada/hilos-en-java>

<http://linkcode.wordpress.com/2008/05/27/listas-en-java-arraylist/>