Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Laboratorio de Algoritmos y Estructuras III
Septiembre-Diciembre 2017

Orden Topológico

1. Descripción de la actividad

Es común en el proceso de desarrollo de software dividir un proyecto diferentes archivos, en donde el código de unos archivos llaman a clases y métodos implementados en otros archivos, creandose así dependencias entre los archivos. Estas dependencias deben tomarse en cuenta a la hora de compilar el proyecto, ya que inducen un orden de compilación. Por ejemplo, usted en su proyecto 1 implemento la clase Digrafo.java, que depende de la clase Grafo.java, que a su vez dependen de Lado.java, y Vertice.java, de modo que para compilar el proyecto debía compilar primero Vertice.java y Lado.java, luego Grafo.java y de último Digrafo.java.

Se desea que usted cree una aplicación que dada las dependencias entre un conjunto de archivos java, determine un orden lineal en que deben compilarse estos archivos, respetendo la precedencia natural inducida por las dependencias entre ellos.

2. Entrada de datos

El programa se debe poder ejecutar desde la consola con el siguiente comando:

```
>java Orden <instancia>
```

Donde instancia es el nombre del archivo que contiene los datos de los nombres de los archivos java y las dependencias entre ellos. El formato del archivo es de la siguiente forma: por cada línea se tiene las dependencias de un archivo escrito de la forma $< nombreArchivo >:< dependencias >$, donde $< dependencias >$ es una lista de nombres de archivos (separados por un espacio) y $< nombreArchivo >$, es el nombre de un archivo cuya lista de archivos del que depende es $< dependencias >$.

$$\texttt{nombreArchivo\_1} : \texttt{nombreDependencia\_1}_1 \ldots \texttt{nombreDependencia\_1}_{n_1}$$
$$\vdots$$
$$\texttt{nombreArchivo\_m} : \texttt{nombreDependencia\_m}_1 \ldots \texttt{nombreDependencia\_m}_{n_m}$$

3. Salida de los Datos

Se debe imprimir la secuencia de los nombres de todos los archivos en un orden en que pueden ser compilados respetando la precedencia inducida por las dependencias. Los nombres de los archivos en la secuencia ordenada, deben estar separados por un sólo espacio.

4. Ejemplo

Dado el siguiente archivo de entrada dependencias.txt:

```
Digrafo.java: Grafo.java Vertice.java Arco.java Lado.java
Grafo.java: Vertice.java Lado.java
Arco.java: Lado.java Vertice.java
```

Un resultado posible y correcto al correr la aplicación puede ser el siguiente:

```
> java Orden dependencias.txt
> Vertice.java Lado.java Grafo.java Arco.java Digrafo.java
```

5. Pruebas

Adicionalmente se hace entrega del paquete JUnit3.8.1 (el archivo junit-3.8.1.jar) para que usted experimente y pueda realizar pruebas a su software de forma rápida. En el archivo TestOrden.java se encuentran escritos 20 casos de prueba en el formato del framework JUnit3.8.1, que permite que usted pueda ejecutar todas las pruebas y ver los resultados de forma automática. Usted debe asegurarse antes de entregar su laboratorio, que su programa pasa estas 20 pruebas.

Para compilar el archivo TestOrden.java escriba el siguiente comando

```
> javac -cp <ruta de junit-3.8.1.jar>:. TestOrden.java
```

Para desplegar la interfaz gráfica de JUnit escriba el siguiente comando

```
> java -cp <ruta de junit-3.8.1.jar>:. junit.swingui.TestRunner TestOrden &
```

Mantenga la ventana emergente abierta durante el desarrollo y ejecute las pruebas las veces que desee haciendo click en el botón *Run*. Usted puede ver cuales fueron las pruebas que paso el software y cuales no, en la pestaña *Test Hierarchy*.