Programación Concurrente y Distribuida

Ingeniería Informática
Facultad de Ciencias
Universidad de Cantabria.

Documento: Práctica 1

Sopa de Letras

Autores: Laura Barros
J.M. Drake
Fecha: 5-6 Octubre

Objetivo de la aplicación

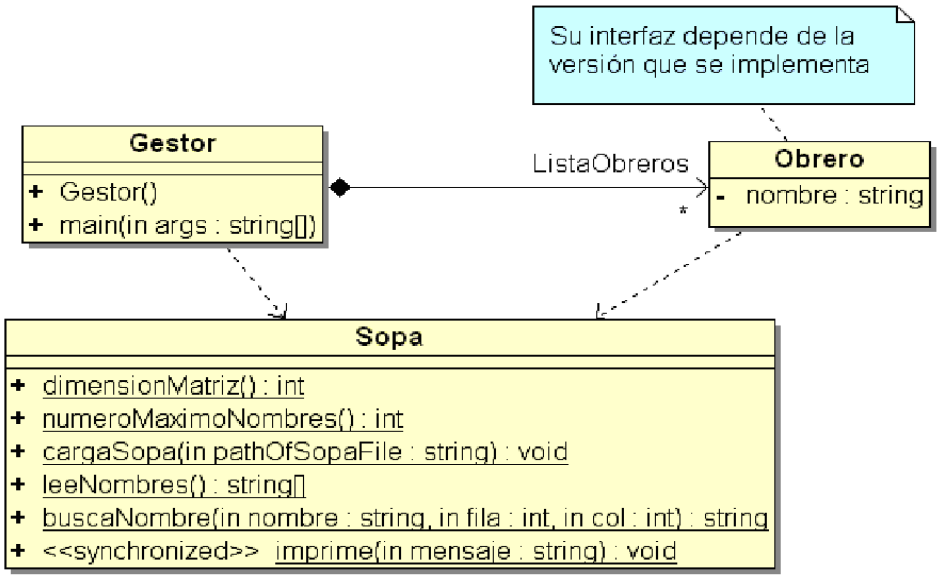
Mostrar las diferentes estrategias que puede seguir un gestor que tiene que ejecutar una tarea compleja que puede ser paralelizada en diferentes subtareas, para organizar su ejecución concurrente distribuyendo las subtareas entre un conjunto de objetos activos que denominamos obreros y que las ejecutan concurrentemente.

El objetivo del ejemplo es la búsqueda concurrente en una sopa de letras de un conjunto de nombres (JUAN, INES, CARMEN, JOSE, MARTA, CARLOS, FELIPE, ANA, PEPE, ROSA) que se encuentran en ella en diferentes posiciones y direcciones. El programa debe buscar la ocurrencia de esos nombres en cualquier dirección y los imprime en la consola indicando la posición de la primera letra y la dirección (LR, RL, UD, DU, DR, DL, UR, UL).

```
NCAHPFELIPESOMCPAPEOUJILANEVIO
VABASAEKJINESASANBPXACNAUJVANE
AAJOHAPFZAZAREKUFIATEAONIAJERA
TROSAOZAFENLCARMENPILRKAENAUIS
RPEUNAUJAPOFAERIASAADLLNPEEOCO
AUJNAAFOJSUEJAGALDAFPOAUINISOU
METUALDAOJILPDPALPMIUSULLAEOLA
EKACARLOSAUIACAOAJUANIRAENPAEG
OSJKANIIEUPLAURAIREBANOCFOTEBA
KALSSENIDSAREÑOARUALPASAEJUANS
NABAFKAGPEMMPIJAHJINESAUOSENFO
UJIAMEAAIEPDTAIFOLLADIAPLVAJHR
DAUNFOLANEHAGUEPILEFLSRNIRENZI
ELFAEPOIJUANEPSENINEIPLONEMRAC
OAYENSENPMAAPEERADFORAOMPSSCSO
RSOOAAUJAEEPHZALNAISUDSANAREEG
ANAUJAZAPAPANAFAUIUJUAINEUENIL
EHÑEJUANSAARENCARLOSDEPTAMPIAR
ASDEJUANPAATRAMEAAZAPARLNATAER
BONEINESGAAECARMENÑAPUAIPNDUAH
EFESIMANSENIADJUANFAJPAPAULATA
```

Especificación

En el diagrama de clases de la figura se muestra la arquitectura de la aplicación basada en tres clases.



- Clase **Gestor**: Existe un único objeto manager en la aplicación que es activo a través del thread del procedimiento main() de la aplicación. El gestor carga del fichero cuyo path se recibe en su lanzamiento haciendo uso del método estático Sopa.CargaSopa(). De forma repetida, el gestor invoca el método estático Sopa.leeNombres() y en él se queda suspendido a la espera de que el operador introduzca la lista de nombres que quiere buscar en la sopa. El operador la introduce como una lista de nombres separados por espacios. Por ejemplo:

  Juan, Ines, Felipe, Carmen, Pepe, Juan, Luis ↵ ***return***

  El manager utilizando las diferentes estrategias que se irán proponiendo, delega en un conjunto de obreras la búsqueda de cada una de los nombres propuestas. Espera a que todas las obreras hayan finalizado, y luego vuelve a quedar a la espera de que el operador proponga desde el teclado una nueva lista de nombres. El manager cierra la aplicación cuando recibe del operador una lista sin ningún nombre.

  - ***Gestor()*** => Constructor del objeto Gestor. De acuerdo con la estrategia de implementación, instancia todos los elementos que se requieren para ejecutar la aplicación;
  - ***main(String[] args)*** => Método que lanza la aplicación. En args debe pasarse el nombre del fichero en el que se almacena la sopa de letras.

- Clase **Obrero:** Durante la ejecución de cada búsqueda, deberán instanciarse o estar instanciadas tantos objetos de la clase Obrero como nombres se vayan a buscar. Cada obrero participante recibe al comienzo de la búsqueda un nombre que debe buscar en la sopa de letras. Recorre la matriz y en cada posición haciendo uso del método estático *Sopa.busca()* comprueba si en ella se inicia el nombre buscado, y por cada uno que encuentra lo escribe sobre la impresora haciendo uso del procedimiento estático *Sopa.imprime()*, indicando su posición y la dirección en que se encuentra. Termina notificando al gestor de alguna manera que ha terminado su tarea. Los obreros ofrecen diferentes interfaces de acuerdo con la estrategia que se utilice.

- Clase **Sopa**: Constituye la infraestructura pasiva y estática en la que se encuentra almacenada la sopa de letras y a la que acuden los obreros para buscar los nombres en la sopa, y para imprimir los resultados encontrados. Contiene el algoritmo de búsqueda que es utilizado por los obreros para encontrar su nombre de consigna. Los procedimientos que ofrecen su interfaz son:

  - **static void cargaSopa(String pathOfSopaFile) =>** Es invocado por el manager para que lea la sopa de letras del fichero cuyo path se pasa como parámetro.
  - **static String[] leeNombre() =>** Espera que el operador introduzca una lista de nombres desde el teclado. Retorna un array de strings con un nombre en cada elemento. Si no se introduce ningún nobre retorna un string nulo.

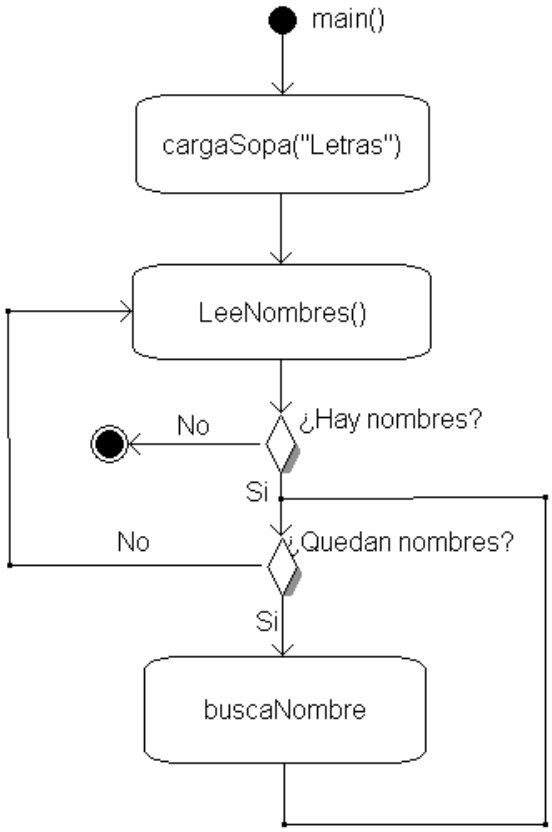
  - **static void String buscaNombre(String nombre, int fila, int col)** => Procedimiento reentrante que es invocado concurrentemente por cada obrero

para buscar su nombre. Busca todas las ocurrencias del nombre que se inicie en la fila y columna que se pasan como parámetro. Retorna un mensaje que contiene el nombre, la fila, la columna, y la dirección o direcciones en que se ha encontrado.

- **synchronized static void imprime(String mensaje)** => Imprime en la consola el texto que se pasa como parámetro mensaje. Es un método synchronaized, por lo que si se invoca concurrentemente, se secuencializa su ejecución.

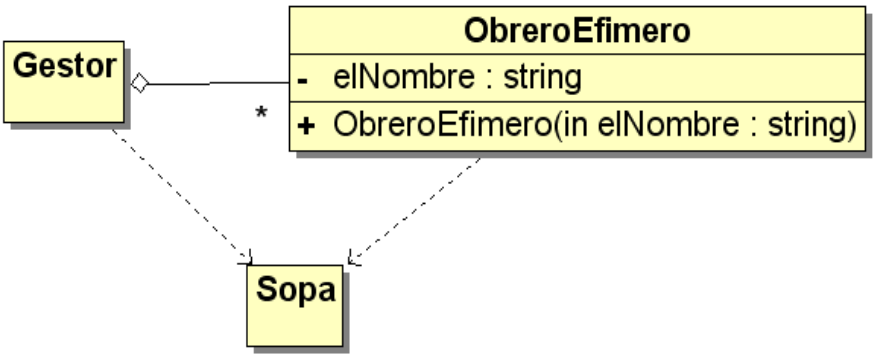
Diseño secuencial.

El gestor realiza el proceso completo buscando en el propio thread del método principal main() los nombres que introduce el operador.



Diseño concurrente.

El gestor una vez que ha recibido la lista de nombres que deben ser buscados, crea **dinámicamente un nuevo obrero** por cada nombre, y le pasa el nombre que debe buscar a través del constructor. Los **obreros son efímeros**, y se destruyen tras haber realizado la búsqueda y su impresión. El **gestor queda suspendido** a la espera de que cada uno de los obreros haya terminado y luego vuelve a requerir al operador una nueva lista de nombres o la orden de finalizar.



Actividades a realizar por el alumno (Miércoles 5 y Jueves 6 de Octubre):

- Desarrollar el código Java de la implementación concurrente que se propone. Se utilizará como código base la implementación secuencial que se encuentra en ***SopaLetras_Práctica1_Secuencial.zip***

- **Diagrama de actividad:** realizar el diagrama de actividad en UML (utilizando la herramienta BOUML) que muestre los nuevos flujos de control (threads) que aparecen en esta versión así como las actividades que realizan.

- **Diagrama de secuencias**: realizar un diagrama de secuencias que muestre los mensajes que se intercambian entre los distintos objetos que intervienen en la versión concurrente de la aplicación.