Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Informática
Métodos de Programación

Estrategias de resolución de problemas: Búsqueda en Espacio de Estados

Alumno: Kevin Arévalo Fernández.

Profesor: Felipe Fuentes Bravo.

Contenido

1. Introducción

En este apartado del documento se explicará el objetivo del informe, así como también los conceptos básicos que debiera conocer el lector para la comprensión del mismo. Se abordará temas como la **Programación Orientada a Objetos**, el lenguaje de programación **Java** y una breve descripción de la estrategia de resolución de problemas denominada **Búsqueda en Estado de Estados**.

1.1 Programación Orientada a Objetos

Para comprender este concepto primero hay que preguntarse, desde el punto de vista informático, claramente, ¿Qué es la programación? Se puede responder a dicha interrogante de muchas maneras, siendo una de ellas la siguiente definición: "Programación es el proceso a través del cual un programa o aplicación informática es desarrollado... Los pasos que se abordan para crear el código fuente de un programa informático. De acuerdo con estos pasos, el código se escribe, se prueba y se perfecciona" (DefiniciónABC, 2017)

Ahora bien, la programación orientada a objetos es, entonces, una manera o estrategia de programación utilizada para resolver un problema específico a partir de la abstracción de la realidad y el contexto en el que se encuentra el problema. Consiste en clasificar a diferentes entes de la realidad que posean características en común en una especie de plantillas denominadas **Clases**, y luego a partir de estas plantillas es posible crear **objetos** relacionados a clases en particular. De esta manera el programador intenta crear objetos que representen a la realidad y que mediante su interacción sea posible la resolución del problema.

1.2 Lenguaje de Programación Java

Para programar, evidentemente se necesita de alguna manera dar instrucciones a la máquina (computadora) para que siga los pasos pensados por el programador para resolver el problema en el que está trabajando. Para que dicha acción sea posible, debe existir una **comunicación** entre humanos y máquinas, y esta comunicación es posible gracias a un lenguaje de programación, que al igual que el español o el inglés, posee reglas sintácticas, semánticas y gramaticales.

Existe una gran cantidad de lenguajes de programación, pero en esta oportunidad se trabajará con **Java**.

Java es un lenguaje de programación desarrollado originalmente por Sun Microsystems, pero que en la actualidad le pertenece a la empresa Oracle. Entre sus características importantes se encuentran que

permite al desarrollador trabajar con un enfoque orientado a objetos, y además de que puede correr en prácticamente cualquier máquina sin necesidad de realizar modificaciones en el código.

1.3 Búsqueda en Espacio de Estados

La búsqueda en Espacio de Estados consiste en una estrategia de resolución de aquellos problemas que se caracterizan por contar con una serie de acciones definidas que permiten llegar a solucionarlo, como por ejemplo un puzzle. Para abordar un problema con esta técnica es necesario identificar tres cosas:

1. Estado inicial: Como su nombre lo indica, es el estado a partir del cual se desea resolver el problema.
2. Transiciones o cambios de estados: La serie de acciones posibles que se pueden realizar para pasar de un estado a otro.
3. Estado final: Aquel estado o situación en donde el problema está solucionado.

El problema a resolver en esta oportunidad se abordará con este método.

1.4 Objetivo

El objetivo de este trabajo es resolver una sopa de letras utilizando la Búsqueda en Espacio de Estados. Esto mediante el lenguaje de programación Java y el Paradigma Orientado a Objetos.

2. Descripción del Problema

2.1 Entradas

Se considera una entrada como la de la figura

Se trata de un archivo de texto plano llamado **"Sopa.in"** que contiene la representación de una sopa de letras. Cada letra está separada por un espacio y las filas de la sopa están separadas por saltos de línea. Esta entrada debe poseer exclusivamente letras en minúscula.

La otra entrada se aprecia en la siguiente figura

Es otro archivo de texto plano llamado **“Palabras.in”**, que contiene las palabras que el usuario desee buscar en la sopa de letras, separadas por saltos de línea. Cada letra de cada palabra está escrita en minúscula.

2.2 Funcionalidad del programa

Ya se explicó la entrada al programa. Ahora se explicará la función que éste debe cumplir.

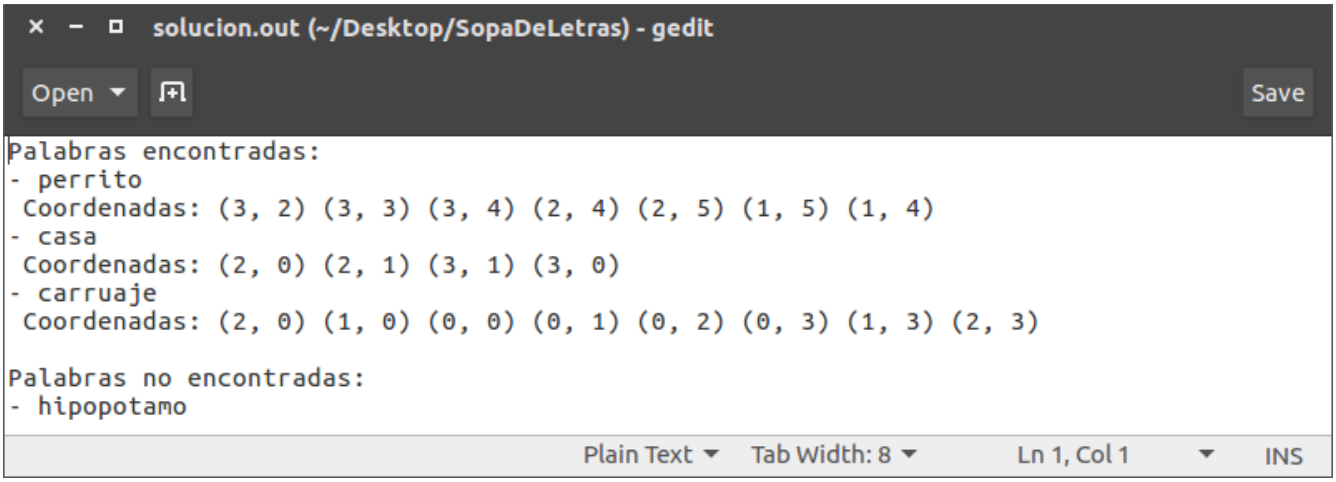
Como se dijo anteriormente la idea del programa es resolver la sopa de letras, buscando las palabras especificadas en la entrada. Sin embargo, esta sopa de letras es diferente a las comunes, ya que las palabras no necesariamente deben estar “en línea” dentro de la sopa. La forma en que se deben encontrar es simplemente que cada letra esté arriba o abajo, o a los lados de la anterior, sin repetirse la posición de ninguna letra de la palabra.

Ejemplo:

La palabra casa cumple con la regla explicada; no necesariamente está en una línea recta y sus letras son adyacentes a la anterior. Además ninguna letra es “re-utilizada” en su formación.

2.3 Salidas

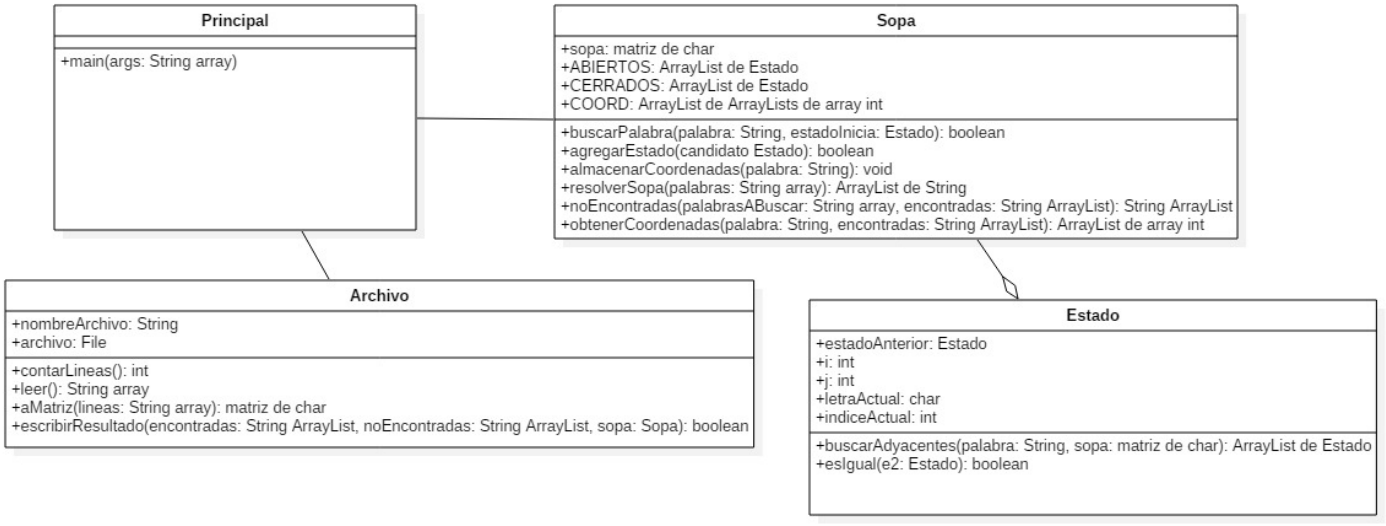
La salida del programa es un archivo de texto plano que especifica las palabras encontradas (y sus coordenadas) y las no encontradas de la sopa de letras. Para las entradas expuestas anteriormente, la salida correspondiente es la siguiente.



3. Desarrollo de la solución

Para trabajar en la solución se diseñó un algoritmo que trabaja con 4 clases, que son representadas en el diagrama de clases a continuación.

3.1 Diagrama de Clases



* El diagrama se encuentra adjunto en la carpeta de este trabajo, para una mejor apreciación.

3.2 Explicación de la Solución

Para abordar el problema se utilizó en primer lugar la división en subproblemas, cada uno de ellos se explica a continuación.

3.2.1 Recibir las entradas

Este problema se trata de leer los archivos de texto “Sopa.in” y “Palabras.in”. La sopa se almacena en una estructura de datos de una matriz de chars y las palabras a buscar se almacenan en un array de Strings. Para resolver el problema se crea un objeto de la clase **Archivo**, y mediante sus métodos **leer()** y **aMatriz()** se almacena la sopa de letras, mientras que para almacenar las palabras sólo es necesario el método **leer()**. Esta es una tarea relativamente sencilla y que no representa la complejidad del problema en sí, por lo cual no se explicará a detalle.

A considerar: Los archivos de las entradas deben estar en la misma carpeta del programa. El programa falla si las entradas no están en el formato especificado. La lista de palabras debe contener sólo las palabras sin espacios al final, sin repetirse, y sin ninguna línea vacía. La sopa de letras debe contener la misma cantidad de caracteres en todas sus filas y no debe contener líneas vacías. Un archivo completamente vacío no es aceptado para ninguna de las dos entradas.

3.2.2 Solucionar la sopa de letras

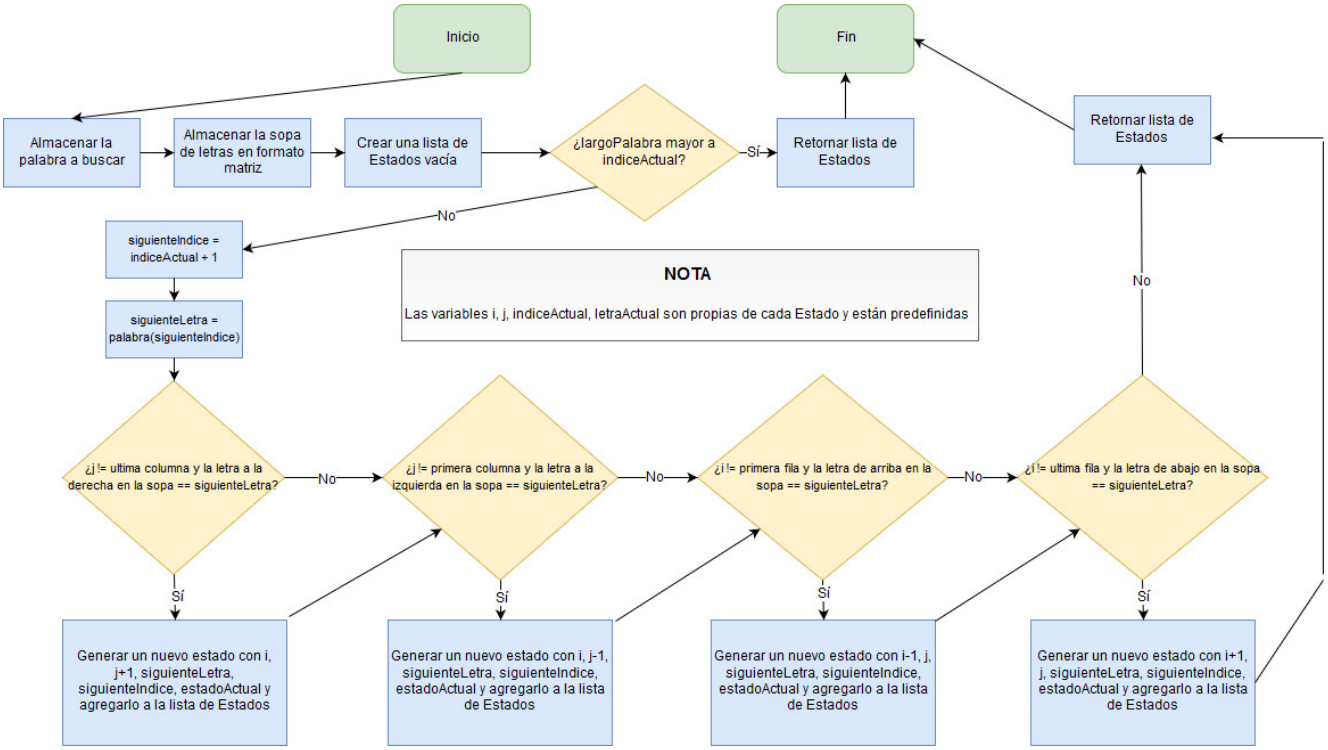
Se trata del problema principal, y el más complejo. Para solucionarlo se recurre a la Búsqueda en Espacio de Estados en anchura, por eso es que se crea una clase **Estado** para abstraer cada estado del problema y así poder llegar desde un estado inicial hasta uno final.

La clase Estado posee como atributos un char, sus coordenadas i,j dentro de la sopa y el índice que ocupa dicha letra dentro de la palabra a buscar. Además posee el atributo estadoAnterior.

- Estado Inicial: Primera letra de la palabra a buscar, sus coordenadas dentro de la matriz sopa e índice 0. No posee estado anterior.
- Transiciones: Moverse arriba, abajo o a los lados, siempre y cuando la letra en esa posición sea la siguiente en la palabra. Se aumenta el índice en una unidad.
- Estado Final: La letra es igual a la última letra de la palabra a buscar y además el índice corresponde al último posible de la palabra (para evitar llegar a un estado final antes de tiempo en palabras con letras repetidas).

El método encargado de generar estados se denomina **buscarAdyacentes** y recibe como parámetros la matriz sopa y la palabra que se está buscando. Básicamente genera todos los estados posibles a partir del estado actual, imponiendo las condiciones descritas y devuelve un ArrayList de Estados. Su diagrama de flujo es el siguiente.

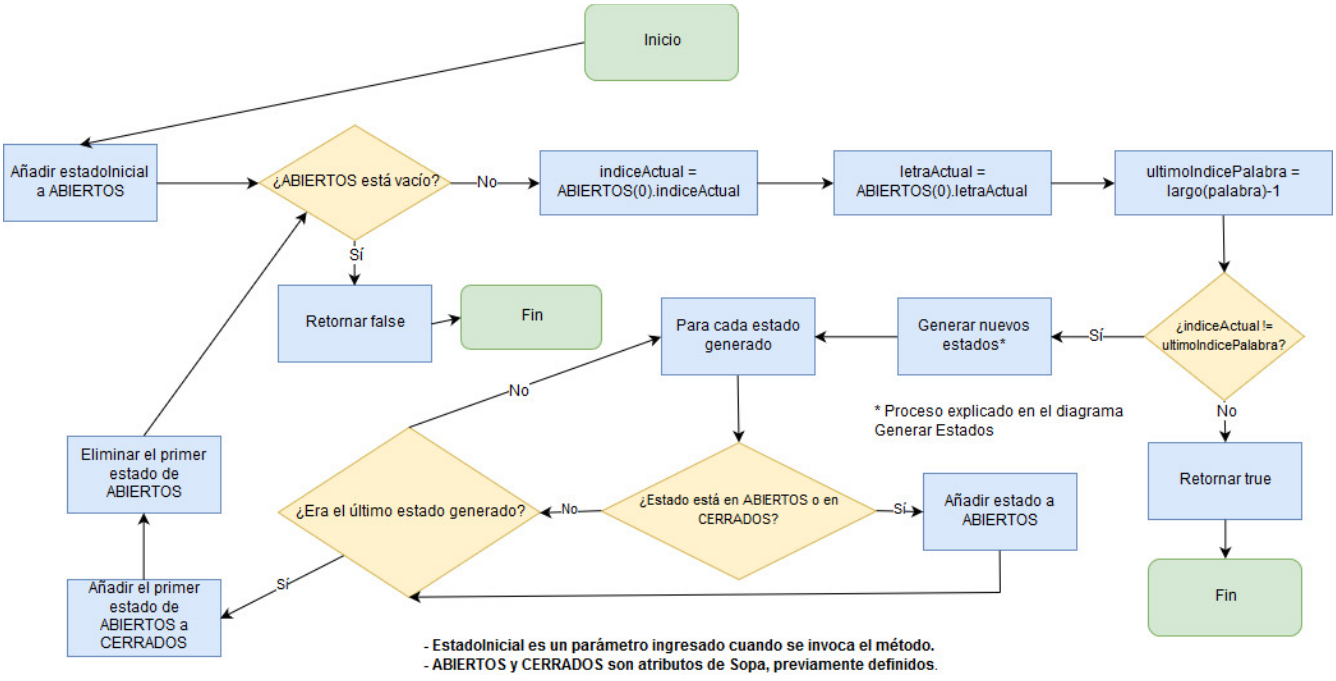
3.2.2.1 Diagrama de flujo Buscar Adyacentes (generar estados)



Para buscar una palabra en la sopa, primero se crea la clase **Sopa**, que posee como atributos dos ArrayList de Estados, denominados ABIERTOS y CERRADOS. Además de una matriz de chars que contiene todos los caracteres de la sopa. Por último un tercer ArrayList, pero que posee ArrayLists de int[], denominado COORD, que se utiliza para almacenar las coordenadas de cada palabra encontrada.

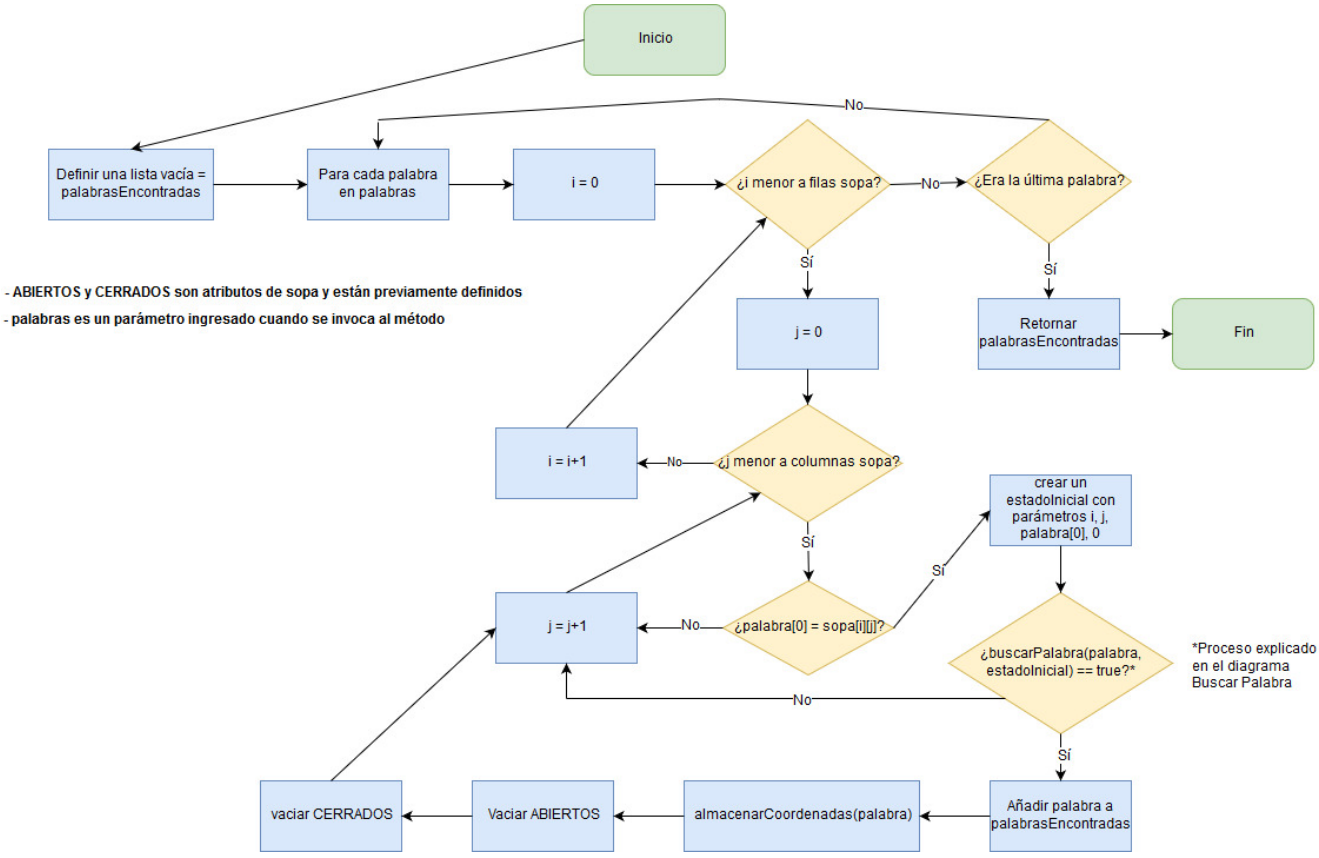
Para buscar una palabra se recurre al método **buscarPalabra** que recibe como parámetros la palabra a buscar y un estado inicial. Devuelve un booleano de acuerdo al resultado de la búsqueda. Su proceso es explicado en su diagrama de flujo a continuación.

3.2.2.2 Diagrama de flujo Buscar Palabra



Finalmente se repite el proceso para cada palabra a buscar. El método encargado es denominado **resolverSopa** y recibe como parámetro la lista de palabras a buscar. Además de resolver la sopa (saber qué palabras se encuentran y cuáles no), también almacena las coordenadas de cada palabra encontrada en el atributo COORD.

3.2.2.1 Diagrama de flujo Resolver Sopa



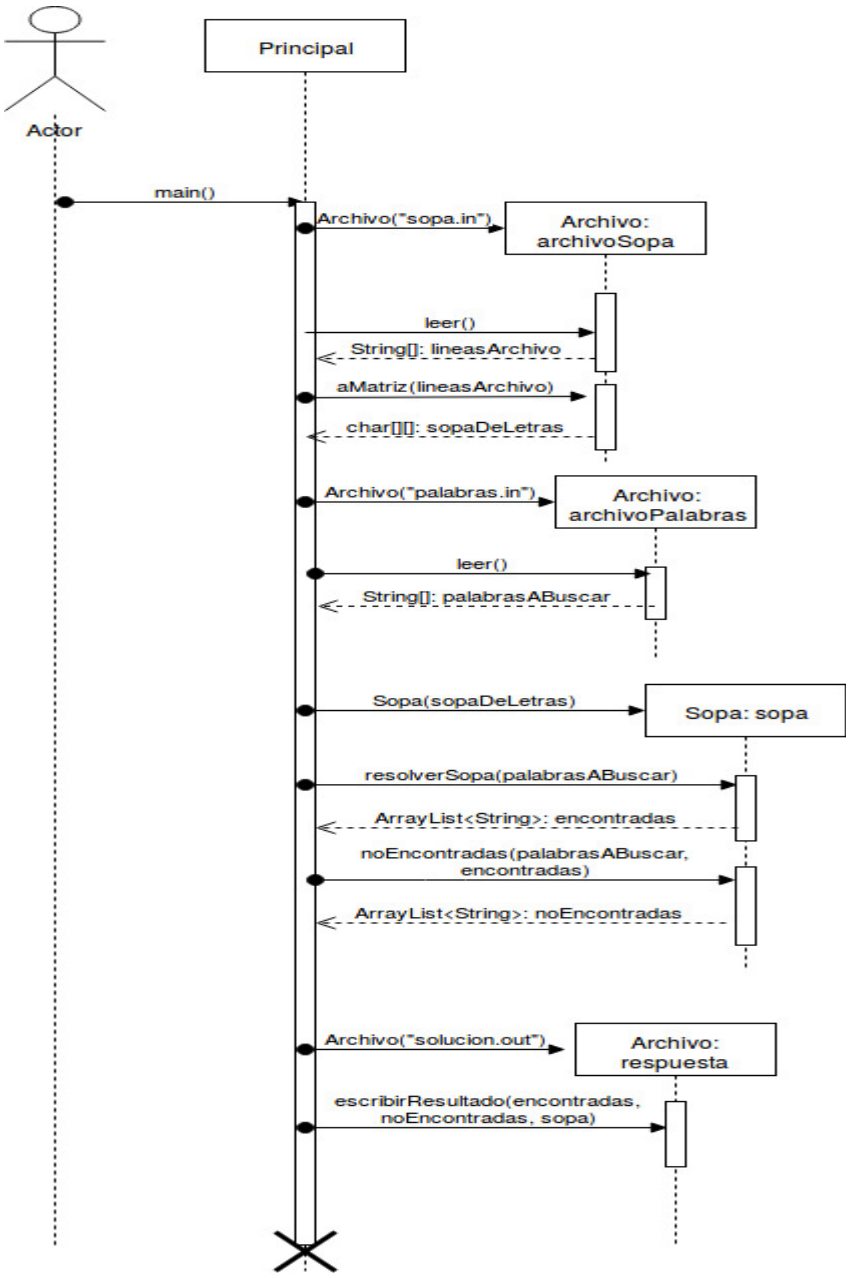
3.2.3 Generar salidas

El tercer problema también lo resuelve la clase **Archivo**, esta vez con su método **escribirResultado**, que recibe como parámetros las palabras encontradas, las no encontradas y un objeto de tipo sopa, que es la que se resolvió previamente. Este problema es trivial y no demasiado complejo, por lo tanto no se explicará en detalle. Si bien el enunciado del problema sugiere que el programa genere una única salida (el archivo de texto), también se despliega el resultado a través de consola, para hacer la interacción con el usuario un poco más amigable.

A considerar: El archivo generado por el programa siempre tendrá el nombre “Solucion.out” y se encontrará en la misma carpeta que contiene al programa. Si el programa se ejecuta por más de una vez, la respuesta será escrita en el mismo archivo previamente generado (con la solución anterior) pero no lo sobreescribe, por lo tanto ahora el archivo contendrá la respuesta actual y las anteriores. Si se desea solamente la solución actual, se debe eliminar previamente (si existiera) el archivo “Solucion.out”.

3.3 Diagrama de secuencia

Finalmente, a modo de resumen, el siguiente diagrama ilustra la secuencia del programa.



4. Conclusiones

La búsqueda en Espacio de Estados resultó ser una técnica eficiente en la resolución del problema planteado. La complejidad de este método se resume a analizar y abstraer el problema para lograr identificar correctamente la estructura de un Estado, sus posibles transiciones y definir un Estado Final, pero una vez superada esta barrera de análisis, no es muy complejo llevar la solución a código.

Las mejoras que se podrían implementar en el programa son una mayor flexibilidad con el formato de las entradas; es decir, podría ampliarse la capacidad de lectura del programa y aceptar más tipos de formatos. Además podría ser más flexible en la lectura en sí, por ejemplo si el usuario ingresara una lista de palabras separadas por comas, que el programa sea capaz de reconocerlo y ordenarlo adecuadamente para adaptarlo al formato requerido para la solución. Otra mejora podría ser dar la posibilidad al usuario de elegir cualquier archivo de texto, almacenado en cualquier lugar del disco duro (sin imponer la restricción de que el nombre de archivo debe ser “Sopa.in”) y también escoger el nombre y ruta del archivo de salida.

A pesar de las posibles mejoras que se le podrían aplicar al programa, éste cumple con proporcionar una solución al problema de resolver una sopa de letras, por lo tanto el objetivo del trabajo se ha logrado exitosamente.