75.42 Taller de Programación I

Informe: Ejercicio N° 3

"Word Mangling: alteración de palabras basada en reglas"

1. Introducción

El word mangling (o alteración de palabras) es una serie de reglas que definen cómo transformar una palabra en otra a partir de simples modificaciones. Es nuestra tarea implementar un módulo de word mangling para ser integrado a un password cracking.

El programa a construir será un aplicativo por consola que leerá desde un archivo una serie de reglas de mangling. Leerá además desde otro archivo, o bien desde la entrada estándar, un listado de palabras. Por cada palabra leída se le aplicará la serie de modificaciones establecidas por las reglas especificadas y las nuevas palabras resultantes serán escritas a un archivo de salida, o bien directamente a la salida estándar.

Detalles mas precisos de la problemática y de las condiciones preestablecidas se pueden encontrar en el enunciado del ejercicio[1].

2. Consideraciones de diseño

Para la correcta implementación de la solución fue necesario plantear y establecer cómo se debería comportar el sistema ante ciertas situaciones que no fueron especificadas en el enunciado del problema. A continuación se listan las contemplaciones instauradas:

- El archivo de salida, de existir previamente a la ejecución del programa, no será truncado para recibir nuevos resultados, sino que se realiza la inserción de los nuevos resultados al final del mismo;
- Al establecer como regla la instrucción *revert*, el usuario debe responsabilizarse de no aplicar una cantidad de reverts que excedan a lo posiblemente aplicable;
- El caractér *";"* de fin de instrucción no debe ir pegado al final de los argumentos de la misma, sino que debe dejarse un espacio intermedio.

3. Diseño

Para el diseño de la solución se acudió primeramente a la determinación de entidades que aparecen en la problemática. En un primer escenario aparecen las entidades que se encargarán de recibir y de transmitir las palabras que se procesen. De esta manera, se decidió crear dos jerarquías de clases: *Receptor* y *Transmisor*. Ambas son clases abstractas, quienes son heredadas por las clases que modelan los distintos modos de recepción y transmisión permitidos. Esta jerarquía se muestra en la *Figura 1*.

Por un lado tenemos las clases *RxEntradaEstandar* y *RxArchivo*, siendo estas las que conforman la recepción de palabras a través de la entrada estándar y a través de un archivo respectivamente.

Por otro lado se encuentran las clases *TxSalidaEstandar* y *TxArchivo*, las cuales conforman la transmisión o emisión de palabras hacia la salida estándar y hacia un archivo.

[1] Se ha evitado hacer un relevamiento de la totalidad de la información que nos fue conferida, de manera de poder mantener el foco del informe en la forma en que se ha encarado la solución del problema.

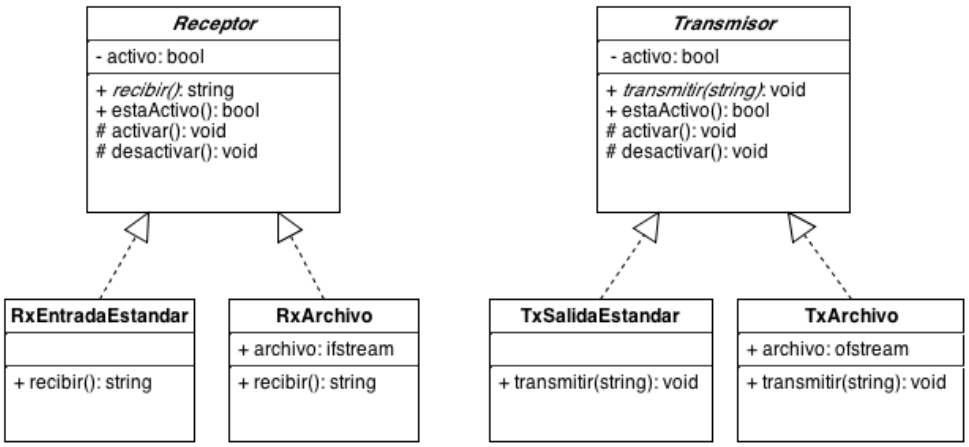

Figura 1: *Diagrama de clases de los receptores y transmisores.*

Siguiendo adelante con el modelado de entidades, nos encontramos ahora con posiblemente la mas significativa del sistema ya que con estas se representan las distintas reglas a aplicar sobre las palabras. La clase madre, *Regla* es una clase abstracta de quien heredan todas las demás clases que representan una regla. Todas deben implementar el método virtual *aplicar* de manera de que poliformicamente se apliquen las reglas a cada palabra. Notará el lector que dicho método no recibe un string sino que, en su lugar, recibe una lista de referencias de strings. Este peculiar detalle se explica más adelante en la *Sección 3.2*. En la *Figura 2* se muestra el diagrama de clases dicha jerarquía.

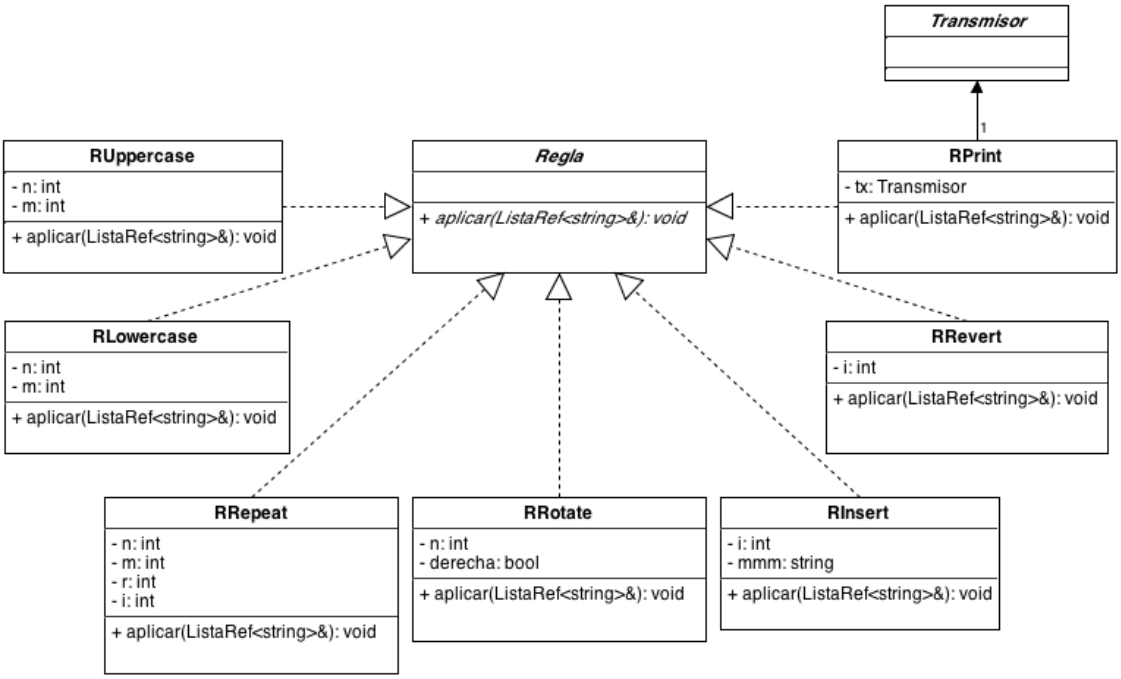

Figura 2: *Diagrama de clases de las reglas.*

3.1. Recepción y transmisión hacia la I/O Estándar

Una de las decisiónes de diseño tomadas fue permitir que el programa se comporte mínimamente como un stream. Con esto nos referimos a que, cuando la recepción se realiza a través de la entrada estándar, a medida que se ingresa una palabra, esta se procesa y es transmitida. En el caso de ser a través de la salida estándar la transmisión, los resultados son emitidos uno a uno a medida que se van ingresando palabras.

Con esto se evitó tener que cargar primero todas las palabras a procesar, desembocando inevitablemente en la espera de la generación de resultados por parte del usuario.

3.2. Principio básico de funcionamiento

El funcionamiento del sistema se centra principalmente en la clase *WordMangling*. Esta contiene como atributo una lista de objetos que son *Regla*.

Al darle la orden al objeto WordMangling de ejecutarse el proceso de alteración, se le envía un receptor que le proporcionará las palabras. De esta manera, palabra tras palabra es tratada, transformandola según las reglas contenidas en orden de aplicación en la lista de reglas.

Cada resultado de una transformación es insertada en una lista de referencias auxiliar. Esta lista es creada por el mismo proceso de WordMangling y es pasada a cada regla (a través del método *aplicar*) debido a que estas últimas deben de poder aplicarse sobre cualquiera de las transformaciones previamente realizadas.

4. Esquema del diseño

A continuación, en la *Figura 3*, se ilustra el diagrama de clases principal que representa la relación entre las entidades que participan del sistema.

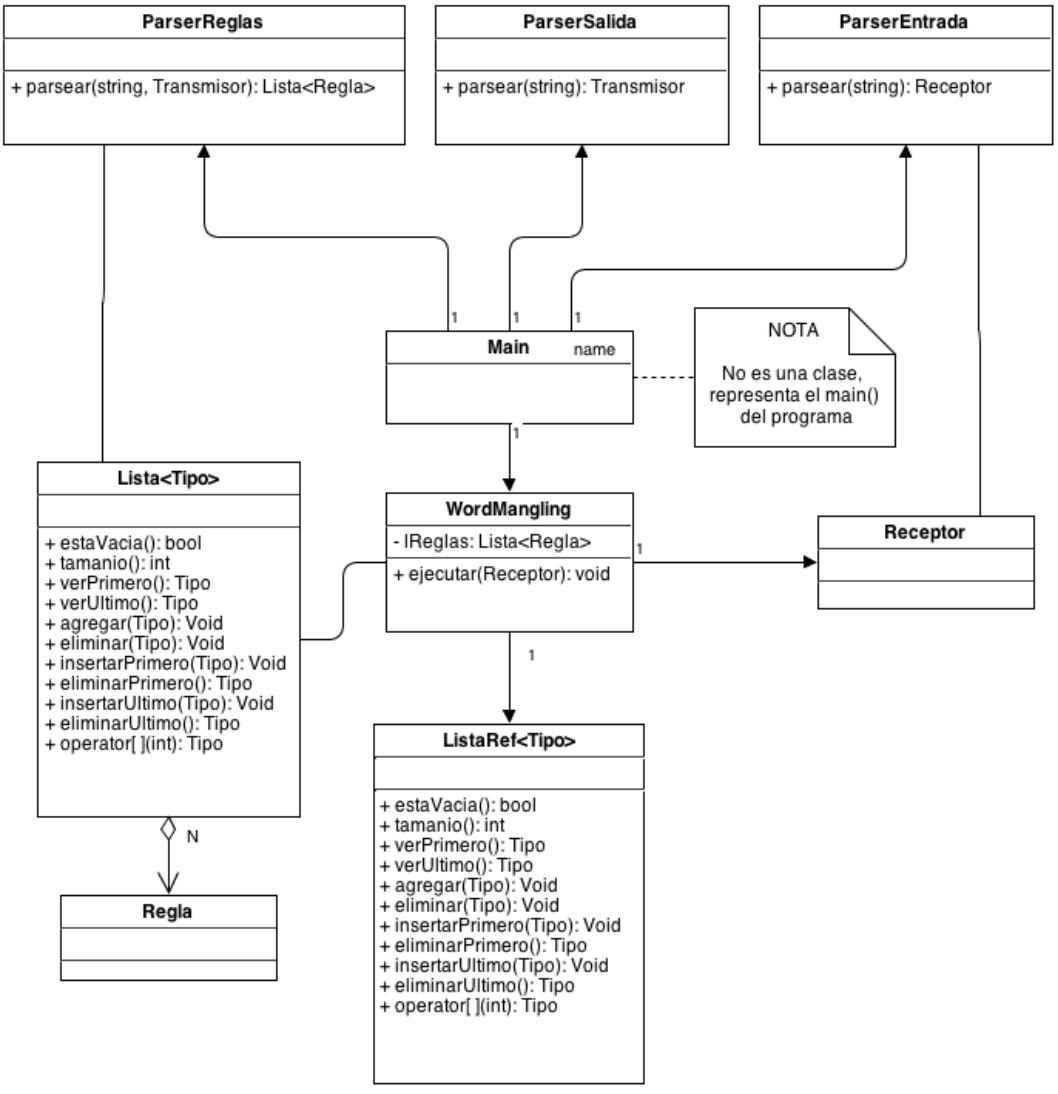

Figura 3: *Diagrama de clases general.*