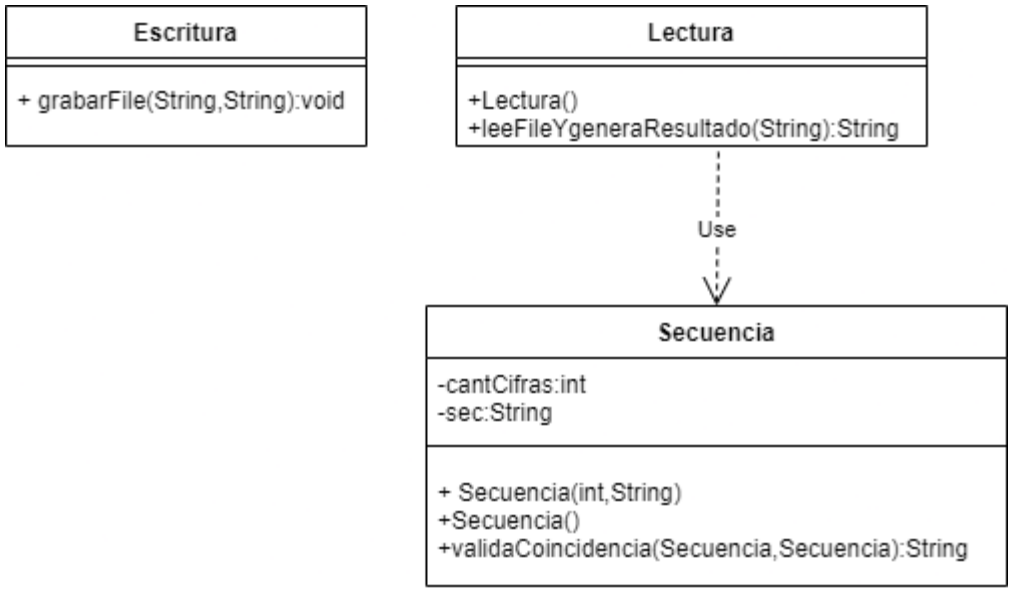
PTO 2B: Ejercicio Tipo OIA

1)



Responsabilidades Clases

Clase Escritura:

Esta clase sólo tiene un método: grabarFile(String, String), recibe dos cadenas de string, una con la ruta del archivo y otra con una cadena a grabar en un file en la ruta y nombre indicada en la primera cadena.

Clase Lectura:

Esta clase no tiene atributos, posee dos métodos:

Lectura(), que es un constructor sin parámetros y por defecto, que utiliza para instanciar la clase lectura y poder utilizar los métodos implementados en ella.

leerFileYgeneraResultado(String) que recibe la ruta del archivo a leer, este método realiza la lectura de los datos que se encuentra dentro de, archivo, a la vez se instancia en la implementación la clase secuencia dos objetos de ese tipo de clase que van a ser las secuencias propiamente dichas y su cantidad de cifras, luego utiliza un método definido en la clase secuencia(explicado en su responsabilidad), que devuelve una cadena con la salida del método.

Clase Secuencia:

Esta clase posee principalmente dos atributos:

cantCifras de tipo int que son la cantidad de cifras de la secuencia y la cadena de tipo string que es la secuencia propiamente dicha.

Tiene un constructor parametrizado Secuencia(int, String) donde recibe la cantCifras y la secuencia, la primera como int y la segunda como String.

Tiene un constructor sin parametros que se utiliza para instanciar y utilizar los métodos en esa clase.

Además tiene el método validaCoincidencia(Secuencia,Secuencia), que recibe dos objetos del tipo secuencia, en ella se encarga de validar si se encuentra alguna combinación de la segunda secuencia dentro de la primera, este método devuelve un String en el cual se muestra si hubo o no coincidencia: “SI” O “NO” concatenado por la cantidad de combinaciones de la segunda dentro de la primera y las posiciones donde van apareciendo esas combinaciones.

2)

Caso 1 : Misma longitud

Verifica que al tener una entrada N y una M con la misma longitud, devuelva una cadena por si o por no y cuantas veces se repite las combinaciones de la primera en la segunda y las posiciones de esas combinaciones, en caso de que no encuentre combinaciones devuelve un solamente un “NO”.

MismaLong.in	SalidaMismaLong.out
3 456 3 645	SI 1 1

Caso 2 : Números repetidos

Verifica que al tener una entrada N y una M con números repetidos, devuelva una cadena por si o por no y cuantas veces se repite las combinaciones de la primera en la segunda y las posiciones de esas combinaciones.

repetidos.in	SalidaRepetido.out
7 1112111 3 111	SI 2 1 5

Caso 3 : Todas las combinaciones

Verifica que al tener una entrada N y una M en la cual en la entrada N esta compuesta por todas las combinaciones posibles de la entrada M, devuelve una cadena por si o por no y cuantas veces se repite las combinaciones de la primera en la segunda y las posiciones de esas combinaciones.

NumConTodasCombinaciones.in	salidaConTodasCombinaciones.out
18 123213231321132312 3 123	SI 10 1 3 4 5 7 9 10 13 15 16

3)

Tenemos un while que va:

While(N>=M)

Sabiendo que N es la cantidad de cifras de la primer secuencia y M la cantidad de la segunda

Por enunciado sabemos que M<=N

Como nuestro algoritmo depende de la entrada N, la complejidad computacional quedara o(N)

Porque dependiendo del valor N se ejecutará tantas veces hasta que deje de cumplirse la condición previamente mencionada.

RTA: O(N)

4)

Para la resolución principal del enunciado se utilizo el método validarCoincidencias que recibe dos objetos del tipo secuencia la cantidad de cifras y la secuencia propiamente diche, devuelve un string, en caso de se encuentre una combinación posible de la segunda secuencia dentro de la primera, es decir si por ejemplo tengo 123 y en secuencia aparece 321 esta secuencia es valida para decir que tengo al menos una, la cadena ya sabemos que va tener un “si” y luego cuento cuantas veces aparece y lo agrego a la cadena que vamos a devolver y posteriormente guardarla en un file, primero se recorre la cadena 1 M veces(cantidad de la segunda cadena) buscando combinanciones, luego las que vaya encontrando almacena en un arrayList de integer las posiciones donde empieza esa combinación finalmente cuando se termine este procedimiento y fuimos usando un contador para saber las coincidencias, verifico que este contador no sea 0, en caso de serlo la cadena tendrá un “NO”, caso contrario tendrá un “si” con las cantidad de combinaciones encontradas y una línea con las posiciones donde aparecieron dichas combinaciones.

5)

Entrada.in

```
23
12341324213423143214312
3
123
```

Salida.out

```
SI 6
1 5 9 13 17 21
```