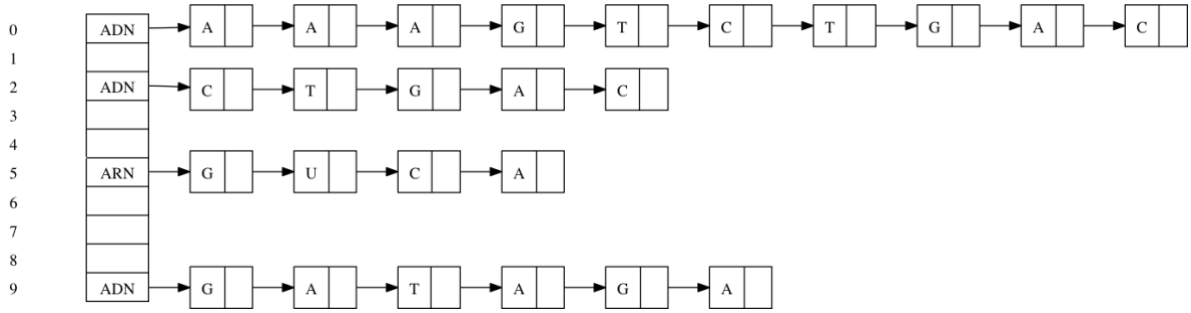
Laboratorio T2: Secuencias genéticas con listas

Una de las principales aplicaciones de la bioinformática es la representación de secuencias genéticas. Una secuencia genética es una sucesión de letras representando la estructura primaria de una molécula real o hipotética de ADN o ARN o banda, con la capacidad de transportar información. En el caso del ADN las posibles letras son A, C, G, y T, que simbolizan las cuatro subunidades de nucleótidos de una banda ADN (adenina, citosina, guanina, timina). En el caso del ARN, las secuencias están formadas por las letras A, C, G, y U, que simbolizan los nucleótidos (adenina, citosina, guanina, uracilo). Habitualmente las secuencias se presentan pegadas unas a las otras, sin espacios, como en la secuencia de ADN AAAGTCTGAC.

En esta práctica se van a desarrollar un programa `Gen.java` con las funciones básicas para manejar secuencias genéticas. Para ello hay que implementar la estructura de datos que aparece en la figura, que consiste en un array de secuencias genéticas (`Secuencia.java`).



Una secuencia genética consiste en un tipo (ADN o ARN) y una lista enlazada, donde cada elemento de la lista es un nucleótido.

El programa principal lee un fichero de entrada que contiene una serie de instrucciones para manipular las secuencias genéticas, que deben ser procesadas.

Entrega 1 (final de la sesión de laboratorio 60% de la nota)

El programa debe implementar las siguientes instrucciones, cada una con un conjunto de parámetros:

- **`insert`** `pos tipo secuencia`. Insertar en la posición `pos` una secuencia genética de `tipo` ADN o ARN con la `secuencia` de caracteres dada. Si ya hubiera una secuencia en la posición `pos`, la nueva secuencia reemplaza a la anterior. Si la `secuencia` a insertar no contiene ningún carácter, se insertará en esa posición una secuencia vacía.
- **`remove`** `pos`. Borrar la secuencia que se encuentra en la posición `pos` del array.
- **`print`**. Imprimir todas las secuencias genéticas indicando para cada una la posición que ocupan en el array y su tipo.
- **`print`** `pos`. Imprimir solo la secuencia que se encuentra en la posición `pos`.

El formato para imprimir es el siguiente <posición> <tipo> <secuencia>. Por ejemplo

```
0 ADN AAGTCTGAC
```

- **`findmotif`** `pos motivo`. Buscar en la secuencia de la posición `pos` todas las apariciones de la sub-secuencia `motivo`. Esta instrucción muestra por pantalla la lista con las posiciones de inicio de todas las apariciones de la sub-secuencia `motivo`. Los elementos de la lista están ordenados de forma ascendente. Si no encuentra el motivo en la secuencia no muestra nada.

Por ejemplo, si en la posición 0 tenemos la secuencia AAAAAGTGATAGA, la instrucción `findmotif 0 GA` muestra por pantalla la línea: `7 11`

- **`profile`** `long`. Calcular la matriz de perfil de las *secuencias de ADN* almacenadas en la estructura y mostrarla por pantalla. Solo se tendrán en cuenta aquellas secuencias que tengan

al menos longitud `long`. La matriz de perfil es una matriz de dimensiones 4x `long`. Cada fila está asociada con un nucleótido (fila 0 →A … fila 3→T), y cada celda almacena las veces que aparece el nucleótido en la posición.

Por ejemplo, si tenemos las secuencias de ADN AAAGT y CTAGA

La instrucción `profile 5` muestra por pantalla la siguiente matriz, donde la celda [0][i] nos dice cuantas secuencias tienen una A en la posición i, la celda [1][i] cuantas tienen una C en la posición i, etc.

```
1 1 2 0 1
1 0 0 0 0
0 0 0 2 0
0 1 0 0 1
```

Entrega 2 (entrega viernes 6 Nov. 40% de la nota)

Además de las instrucciones de la entrega 1, hay que extender el programa para implementar las siguientes instrucciones, cada una con un conjunto de parámetros:

- **`clip`** `pos start end`. Reemplazar la secuencia en la posición `pos` por la sub-secuencia que se obtiene con todas las letras desde la letra en la posición `start` hasta la letra en posición `end`. Si no se especifica `end`, se considera hasta el final de la secuencia.

Por ejemplo, si en la posición 0 se almacena la secuencia de ADN AAAAAGTGATAGA, la instrucción `clip 0 5 8` cambia el contenido de la posición 0 por la secuencia GTGA.

- **`copy`** `pos1 pos2`. Copia la secuencia de la posición `pos1` a la posición `pos2`.

- **`swap`** `pos1 start1 pos2 start2`. Intercambia la cola de la secuencia en `pos2` desde la letra en la posición `start2`, hasta el final; por la cola de la secuencia en la posición `pos1` desde la letra `start1` hasta el final.

Por ejemplo, si tenemos las siguientes secuencias:

0 ADN AAAAAGTGATAGA

1 ADN CTGAC

La instrucción swap 0 4 1 3 modifica las secuencias del siguiente modo (recuerda que las posiciones empiezan en 0):

0 ADN AAAAA**AC**

1 ADN CTG**AGTGATAGA**

- **`transcribe`** `pos`. Convierte la secuencia de ADN de la posición `pos` en otra de ARN. Para realizar esta operación hay que cambiar todas las T por U, y después cambiar todas las letras por su complementarias (es decir las A se transforman en U y viceversa, las C se transforman en G y viceversa). Una vez terminada la operación se invierte la secuencia.

Por ejemplo, para la secuencia de ADN TGAC, primero al cambiar T por U se transforma en UGAC, al complementar las letras se transforma en ACUG y finalmente se invierte la secuencia, dando como resultado GUCA.

Anexo

- Para simplificar la implementación, no hace falta comprobar que la secuencia de entrada es correcta, es decir, no hace falta comprobar que los nucleótidos son los correctos si la secuencia es ADN o ARN.

- Si cualquier instrucción tuviera algún error, (por ejemplo, si *start* > *end*, o si hubiera un número negativo, o si no se encuentra la sub-secuencia), entonces esa instrucción no debe ejecutarse.

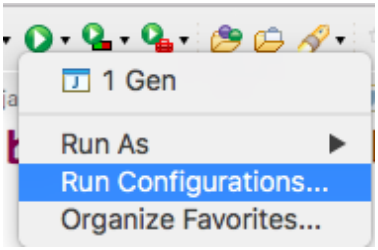
- No pueden utilizarse ninguna de las clases de estructuras de datos predefinidas en el paquete `java.util`, (`ArrayList, LinkedList, SortedList, HashSet`, etc…) y tampoco

puede usarse la clase `String` para almacenar las secuencias genéticas, tiene que tratarse como una lista enlazada de nodos `Nucleotido`. Sí se pueden usar las implementaciones de la interfaz `List` vistas en clase.

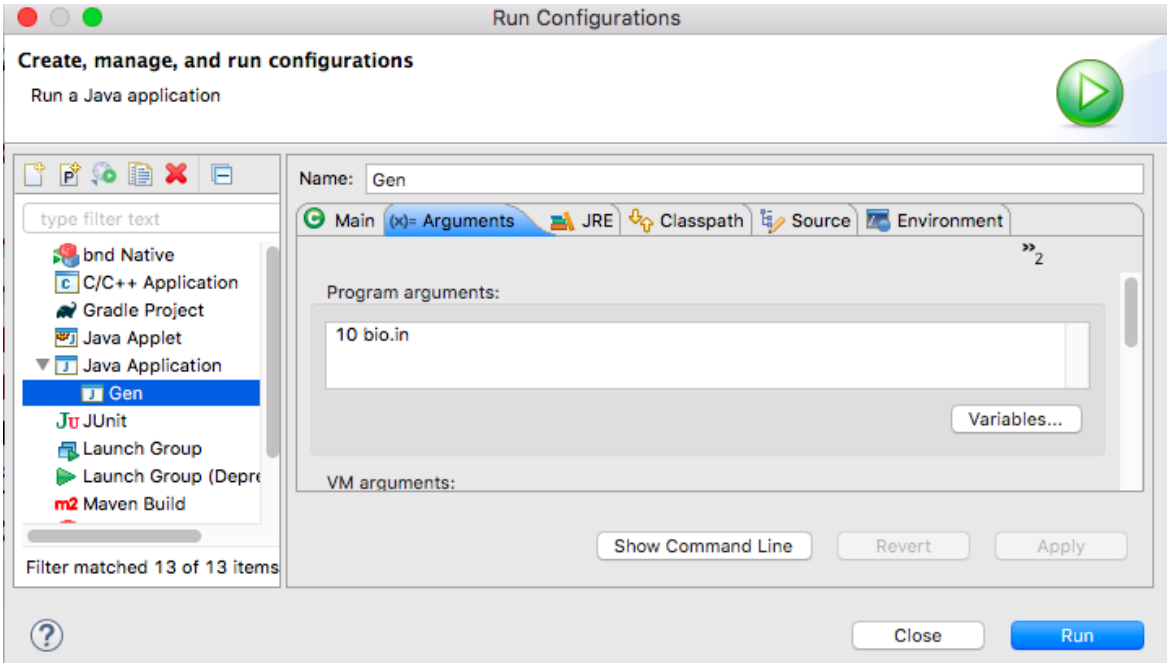
- Al ejecutar el programa principal hay que pasar como argumento primero el tamaño del array, y segundo el nombre del fichero de instrucciones. Si hacemos la llamada por línea de comandos utilizaremos.

```
java Gen 10 bio.in
```

Para pasar argumentos de entrada en Eclipse, pulsar en la flecha que aparece en el botón Run, y seleccionar "Run Configurations":



En la pestaña Arguments, incluir en el campo "Program arguments" los argumentos en el orden correspondiente separados por espacios:



ENTREGA DE LA PRACTICA

Para que funcione el evaluador automático hay que realizar la entrega del siguiente modo:

- Todas las clases deben estar en el paquete por defecto (defautl package), NO se puede utilizar cualquier otro paquete.

- **Se comprimirán en un fichero Gen.zip** los ficheros que encontraréis en la carpeta src del proyecto (ficheros .java). **El zip NO debe contener la carpeta, SOLO los ficheros.**

- La salida a mostrar por el programa tiene que ser por la consola mediante System.out.println, no escribiendo en un fichero.

- Se proporcionan ficheros de entrada (entrega1.in y entrega2.in) para realizar pruebas locales. La salida mostrada por pantalla debe seguir el formato que se observa en el fichero de salida correspondiente (entrega1.out y entrega2.out) para que la evaluación automática de por válido el resultado.