Environnement de développement – L3 II
TP 1 : Premier (?) contact avec Eclipse

Mihaela Sighireanu
(`www.liafa.jussieu.fr/~sighirea/cours/edi/`)

L'objectif de ce TP est de vous familiariser avec les concepts de la plate-forme Eclipse vus en cours. Il vous permet également un premier contact avec le JDK (*Java Developement Kit*) que nous allons étudier plus en profondeur par la suite.

1 Lancer et configurer Eclipse

1. Lancer Eclipse à partir d'un terminal ou à partir du menu d'Applications. Un message d'erreur sur l'absence de navigateur peut apparaître ; il s'agit d'un problème de configuration, ignorez-le pour le moment ! Le premier lancement d'Eclipse est long car il cherche à travers le reseaux des parametres de lancement.
2. Choisir (pour le moment) l'emplacement de votre espace de travail à la racine de votre FS.
3. Entrer dans l'espace Tutorial est naviguer dans les manuels offerts.
4. Revenir à la page d'introduction et cliquer sur l'icone de l'espace de travail.
5. Dans le menu Help, choisir About Eclipse SDK. Visualiser les differents détails proposés (plug-ins, features, options).
6. Dans Help → Preferences, visualiser la version de JRE qui est par defaut. Changer cette version pour la version de Sun.

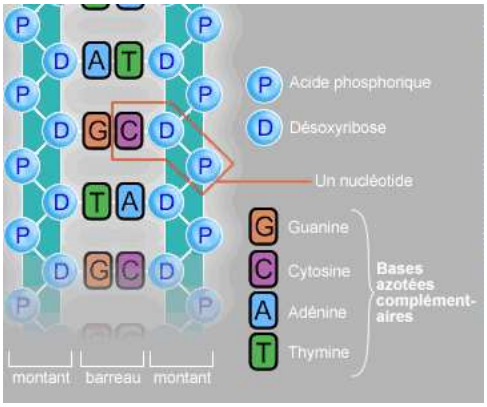
2 Programmation Java sous Eclipse

Pour travailler avec le JDK, il suffit de :

1. Créer un nouveau projet dont le type est "Java Project".
2. Assurer que la version de JRE utilisee est celle de Sun.
3. Définir un nouveau paquet.
4. Créer dans le paquet des sources Java soit en partant d'un fichier vide, soit en important des fichiers existants. Par exemple, pour le probleme ci-dessous, importer un par un les fichiers qui se trouvent dans le répertoire `/ens/sighirea/Java/tp_ADN/`.
5. Visualiser les fichiers importés dans l'éditeur et observer les differentes vues du plan de travail.

2.1 L'ADN

L'ADN[1], sigle de acide désoxyribonucléique, est une longue molécule que l'on retrouve dans tous les organismes. Sa célèbre structure en forme de double hélice a été découverte en 1953 par James Dewey Watson, Francis Crick et leurs collaborateurs.



Un polymère de bases désoxyribonucléiques est constitué de répétitions de nucléotides formées d'un groupe phosphate (P) lié à un sucre (le désoxyribose, D), et à une base azotée A, T, C ou G. Le squelette est formé de la répétition sucre - phosphate, ce qui change est la base.

Quatre bases ont été identifiées : l'adénine (A) et la guanine (G) font partie de la famille des purines. La thymine (T) et la cytosine (C) sont de la famille des pyrimidines. Elles sont complémentaires entre elles et uniquement associables l'une avec l'autre. Un "brin" d'ADN est formé de la répétition ordonnée de ces quatre bases.

Les deux brins antiparallèles d'ADN sont toujours étroitement reliés entre eux par des liaisons hydrogène (également appelées ponts hydrogène ou encore simplement liaisons H ou ponts H) formées entre les bases complémentaires A-T et G-C. Ces deux brins d'ADN sont dit complémentaires car les purines (Adénine et Guanine) d'un brin font toujours face à des pyrimidines de l'autre brin (Thymine et Cytosine). L'adénine est complémentaire à la thymine et la guanine est complémentaires à la cytosine. Deux liaisons hydrogène retiennent ensemble la paire A-T et trois retiennent la paire G-C.

Fusion La température de fusion des acides nucléiques comme l'ADN dépend de la quantité de liaisons hydrogène présentes. Plus il y a de liaisons hydrogène dans une molécule d'ADN, plus l'énergie de liaison est élevée et plus sa température de fusion sera élevée.

Ainsi une molécule double brin composée uniquement de C-G (3 liens H) nécessitera plus d'énergie pour être ouverte qu'un ADN de même taille composé

[1] Les informations qui suivent ont été prises de l'encyclopédie libre Wikipédia.

de A-T (2 liens H). Ceci explique pourquoi la température de fusion de l'ADN varie en fonction de deux facteurs principaux :

- sa taille (exprimée en nombre de bases, généralement en kilobase kb ou mégabase Mb ...)
- son rapport (A+T)/(C+G) qui donne un indice des proportions de paires A-T versus C-G.

Information structurelle Chaque molécule d'ADN est caractérisée par les motifs crées par les paires A-T et C-G qui la composent. Il est donc important de savoir si un certain motif se trouve dans une molécule d'ADN afin d'indentifier les organismes dont elle fait partie.

2.2 Modélisation en Java

Afin de simplifier l'implémentation, on s'interesse à des molécules d'ADN de longueur fixée. (Cette contrainte peut être facilement enlevée si on utilise les collections de Java, qui seront vues prochainement en cours.)

Exercice 1 : Pour s'échauffer, commencer par modéliser les quatres bases A,T,C,G. Utiliser pour cela des classes abstraites et/ou l'héritage. Pour chaque base il faut pouvoir connaître son type et avec qui elle est complémentaire. Définir une méthode toString qui affiche en ASCII chaque base.

Exercice 2 : Le pont de l'ADN est le suivant objet à modéliser dans une classe PontADN. Il faut pouvoir connaître ses composantes gauche et droite (attention au passage des paramètres), son type (A-T ou C-G) et le nombre de liaisons hydrogène impliquées. N'oublier pas de définir la méthode toString.

Exercice 3 : Enfin, modéliser une molécule d'ADN de longueur fixée (constante) par la classe MoleculeADN. Ecrire les méthodes nécessaires pour :

- renvoyer une suite de ponts à partir d'une position donnée, sur une longueur donnée ;
- renvoyer la position de la première apparition d'un pont donné en paramètre ;
- afficher sous forme ASCII et applatie(comme dans la figure), sur la sortie standard, un fragment de la molécule dont les dimensions (point de départ et longueur) sont données en paramètres.

Exercice 4 : Ajouter à la classe MoleculeADN une méthode fusionRate qui calcule le rapport (A+T)/(C+G) pour un fragment de la molécule donné par le début du fragment et sa longueur.

Exercice 5 : Toujours pour la classe MoleculeADN, écrire une méthode searchPattern qui renvoie la position du premier pont hydrogène de la molécule qui fait partie du motif passé en paramètre de la méthode.

Exercice 6 : A ce point, on s'interesse à optimiser notre implémentation afin d'occuper moins de mémoire vive avec un objet ADN. (Il faut obtenir au moins un facteur de 1/8.)

- Tester quelle est la longueur maximale de la molécule d'ADN qu'on peut utiliser pour un objet sans modifier la mémoire disponible de l'interpréteur de Java.
- Comment peut-on minimiser la taille de la mémoire utilisée pour représenter une base ? Mais un pont ? Que ceci implique sur la structure du programme que vous avez écrit ?
- Quel est le facteur d'optimisation que vous obtenez ?

Correction 1 : (Les bases)
Voir fichiers :

```
/ens/sighirea/Java/ADN/BaseEnum.java
/ens/sighirea/Java/ADN/BaseHerit.java
/ens/sighirea/Java/ADN/baseHeritInt.java
/ens/sighirea/Java/ADN/BaseInt.java
/ens/sighirea/Java/ADN/BaseAbstr.java
```