

Documentation interne

Projet de Programmation

L2D1



Illustration 1: Image de présentation

Projet L2D -Introduction à la bioinformatique

Les informations d'identification du document :

Référence du document :
Version du document : 2
Date du document : 01/05/2021
Auteurs : Mehdi Hamiche Manal Boutajar Adelin Bodnar

Les éléments de vérification du document :

Validé par :
Validé le :
Soumis le :
Type de diffusion :
Confidentialité :

*Les éléments
d'authentification :*

Maître d'ouvrage :	Chef de projet :
Date / Signature :	Date / Signature :

Projet L2D -Introduction à la bioinformatique

Sommaire

<u>1. Introduction</u>	5
1.1. Objectifs et méthodes	5
1.2. Documents de référence	6
<u>2. Guide de lecture</u>	7
2.1. Maîtrise d'œuvre	7
<i>2.1.1. Responsable</i>	<i>7</i>
<i>2.1.2. Personnel administratif</i>	<i>7</i>
<i>2.1.3. Personnel technique</i>	<i>7</i>
2.2. Maîtrise d'ouvrage	7
<i>2.2.1. Responsable</i>	<i>8</i>
<i>2.2.2. Personnel administratif</i>	<i>8</i>
<i>2.2.3. Personnel technique</i>	<i>8</i>
<u>3. Concepts de base</u>	9
<u>4. Description des modules</u>	11
<u>5. Description des classes</u>	20
<u>6. Annexes</u>	37

Projet L2D -Introduction à la bioinformatique

<u>7. Glossaire</u>	38
<u>8. Références</u>	39
<u>9. Index</u>	40

1. Introduction

Cette documentation facilite la maintenance de l'application par un tiers. Celle-ci peut prendre plusieurs formes en fonction du langage utilisé pour l'implantation dont javadoc pour java.

Toutefois, un document généré automatiquement à l'aide de ces outils est insuffisant.

Contenu :

- *Concept de base*
 - permet la compréhension du document
- *Description des modules (classes)*
 1. Rappel des objectifs du module
 2. Relations d'utilisation avec d'autres modules
 3. Lister les modules utilisés par ce module et ceux utilisant ce module
 4. Procédures externes
 5. Variables externes

1.1. Objectifs et méthodes

Lister toutes les classes de notre application pour mieux comprendre l'utilisation des classes les différents codes.

Modules

1. *Alignement global* - 2 séquences alignées via l'algorithme de Needleman - Wunsch

2. *Alignement multiple* - amélioration de l'alignement global (consensus) – alignement de trois séquences au moins
3. *Logo* - représentation graphique de l'alignement multiple
4. *Interface graphique* - relie les 3 modules et les présente

1.2. Documents de référence

Les documents du projet servant à l'élaboration du présent document :

- Le cahier des charges,
- Le cahier de recettes,
- La conception générale
 - ◆ Conception détaillée
 - ◆ Manuel d'installation
 - ◆ Manuel d'utilisation
 - ◆ Plan de tests

2. Guide de lecture

2.1. Maîtrise d'œuvre

La maîtrise d'œuvre présente l'équipe du développement chargé du bon suivi de la documentation interne et des besoins dont le maître d'ouvrage fait commande.

Elle représente l'équipe du développement :

- Adelin Bodnar
- Manal Boutajar
- Mehdi Hamiche

Cette équipe veillera au bon suivi de la documentation interne coordonnées avec la conception générale représentant les besoins des enseignants encadrants.

2.1.1. Responsable

Il est conseillé pour le responsable de la maîtrise d'œuvre de lire le document dans sa totalité afin de prendre conscience de l'ensemble des éléments.

2.1.2. Personnel administratif

Il est conseillé pour le personnel administratif de lire le concept de base et la description des modules.

2.1.3. Personnel technique

Il est conseillé pour le personnel technique de prendre en compte la partie sur le concept de base, ainsi que la description des modules.

2.2. Maîtrise d'ouvrage

Projet L2D - Introduction à la bioinformatique

La maîtrise d'ouvrage représente dans notre cas le client du projet, c'est-à-dire les personnes dont les besoins permettent la conception du projet.

La maîtrise d'ouvrage est assistée par l'équipe de la maîtrise d'œuvre et donc ce rôle sera assuré par les enseignants encadrants Dragutin Jastrebic et Koviljka Lukic Jastrebic.

2.2.1. Responsable

Il est conseillé pour le responsable de la maîtrise d'ouvrage de lire le document dans toute sa totalité afin de prendre conscience de l'ensemble des documents.

2.2.2. Personnel administratif

Il est conseillé pour le personnel administratif de lire le concept de base et la description des modules.

2.2.3. Personnel technique

Il est conseillé pour le personnel technique de prendre en compte la partie sur le concept de base, ainsi que la description des modules.

3. Concepts de base

Pour une meilleure compréhension du document, voici quelques définitions :

- **Bio-informatique** : Discipline permettant d'analyser de l'information biologique qu'il y a dans les séquences nucléotidiques et protéiques. La bio-informatique est à la frontière de la biologie, de l'informatique et des mathématiques (bases de données volumineuses, algorithmes performants pour traiter les informations stockées).
- **Différents modules** :
 - **Alignement global** : recouvre les séquences alignées (par exemple 2 séquences) sur l'ensemble de leur longueurs,
 - backtracking - algorithmes qui permettent de résoudre des problèmes algorithmiques (de satisfaction de contraintes). Ils permettent de tester systématiquement l'ensemble des affectations potentielles du problème.
 - **Alignement multiple** : aligner un ensemble de séquences homologues, comme des séquences protéiques ou nucléiques qui assure des fonctions similaires dans différentes espèces vivantes,
 - *séquences consensus* - séquence nucléotidique ou peptidique la plus fréquente à chaque position d'un alignement de séquences.
 - **Weblogo** : application web conçue pour rendre la génération de logos de séquence aussi simple que possible. Les logos de séquences sont une représentation graphique d'un alignement de séquences multiples d'acide aminé ou d'acide nucléique,
 - **Interface Graphique** : application permettant de relier les 3 modules et les présente de manière simple et efficace.

Projet L2D - Introduction à la bioinformatique

- **Algorithme Needleman-Wunsch** : effectue un alignement global maximal de deux chaînes de caractères. Utilisé en bio-informatique pour aligner des séquences de protéines ou de nucléotides.
- **Format des séquences** : Fasta
- **Fasta** : format de fichier texte dans lesquels les séquences sont affichées par une suite de lettres (séquences protéiques et nucléiques). Elle permet de stocker les séquences protéiques et nucléiques.
- **Programmation orienté objet** : met en œuvre différents objets (instances de classes). Chaque objet associe des données et des méthodes (fonctions) agissant exclusivement sur les données.

4. Description des modules

I. Interface graphique

Rappel des objectifs du module

Ce module permet la présentation de nos 3 modules (alignement global, alignement multiple, logo).

Relations d'utilisation avec d'autres modules

Permettre l'affichage avec l'alignement global, multiple et weblogo.

Lister les modules utilisés par ce module

- `import static Align_Global.ResAG.r3;`
- `import java.awt.Toolkit;`
- `import java.awt.event.ActionEvent;`
- `import java.awt.event.ActionListener;`
- `import java.awt.event.WindowEvent;`
- `import java.io.File;`
- `import java.util.ArrayList;`
- `import javax.swing.JFileChooser;`
- `import javax.swing.JTabbedPane`

Procédures externes

Affichage des modules logo, alignement global et multiple via l'interface graphique.

II. Alignement global

Rappel des objectifs du module

Ce module donne l'alignement de 2 séquences via l'algorithme de Needleman – Wunsch.

Relations d'utilisation avec d'autres modules

Relation avec l'interface graphique qui permet l'affichage de l'alignement, de la classe (.java) :

- **AlignementG**,
- **Chemin**,
- **Fasta1**, permet de lire un fichier de format fasta,
- **Main** qui permet l'exécution de l'alignement global (et multiple),
- **Sequence** permet d'avoir l'ID nature des séquences (PROTEINE ou ADN),
- **ADN** qui hérite de la classe Sequence,
- **PROTEINE** qui hérite de la classe Sequence,
- **ResAlignement**, affiche les résultats de l'alignement global,
- **Matrice**, pour les valeurs gap, match, mismatch,
- **Fleches**, permet la mise en mémoire du mouvement qui donne le meilleur score

Lister les modules utilisés par ce module

- *class AlignementG :*

Projet L2D - Introduction à la bioinformatique

- `import java.util.ArrayList;`
- **class Fasta1 :**
 - `import java.util.ArrayList ;`
 - `import java.util.Scanner ;`
 - `import java.io.File ;`
 - `import java.io.FileWriter ;`
 - `import java.io.FileNotFoundException`
- **class FrameAG :**
 - `import javax.swing.JFrame ;`
 - `import javax.swing.JoptionPane`

Procédures externes

Se relie à l'interface graphique avec **FramePrincipale** et toutes ses classes pour l'affichage.

Variables externes

Attributs privés pour :

- **class Chemin**
 - `Sequence seq_sup ;`
 - `Sequence seq_inf ;`
- **class Fasta1**

Projet L2D - Introduction à la bioinformatique

- String nomFichier ;
- ArrayList<Sequence> sequences ;
- **class Main**
 - StringBuffer seq1, seq2, seq3, seq4 ;
 - Alignement al1, al2 ;
 - Fasta f ;
- **class Sequence**
 - String id ;
 - StringBuffer seq ;
 - static String nature ;
- **class ADN extends Sequence**
 - String alphabet ;
- **class PROTEINE extends Sequence**
 - String alphabet ;
- **class Matrice**
 - static int match, mismtach, gap ;
 - StringBuffer seq_haut, seq_gauche ;
 - ArrayList<ArrayList<Integer>> tableau ;

- `ArrayList<ArrayList<Fleches>> tableauFleches ;`
- **class Fleches**
- `ArrayList<String> fleches ;`

III. Alignement multiple

Rappel des objectifs du module

Calcul un score de similarité entre toutes les paires de séquences par comparaison des séquences deux à deux; ensemble de scores d'alignement qui sont regroupés dans une matrice de similarités.

Relations d'utilisation avec d'autres modules

Relation avec l'interface graphique qui permet l'affichage de l'alignement, de la classe :

- **Fasta1**, permet de lire un fichier de format fasta,
- **Main** qui permet l'exécution de l'alignement global (et mutiple),
- **Sequence** permet d'avoir l'ID nature des séquences (PROTEINE ou ADN),
- **ADN** qui hérite de la classe Sequence,
- **PROTEINE** qui hérite de la classe Sequence,
- **Matrice**, pour les valeurs gap, match, mismatch
- **FrameAM**, permet affichage dans l'interface graphique

Lister les modules utilisés par ce module

- **class Alignement_Multiple**

Projet L2D - Introduction à la bioinformatique

- `import java.util.ArrayList ;`
- **class Fasta1**
 - `import java.util.ArrayList ;`
 - `import java.util.Scanner ;`
 - `import java.io.File ;`
 - `import java.io.FileWriter ;`
 - `import java.io.FileNotFoundException ;`
- **class FrameAM**
 - `import javax.swing.JFrame ;`
 - `import javax.swing.JOptionPane ;`

Procédures externes

Se relie à l'interface graphique avec **FramePrincipale** et toutes ses classes pour l'affichage.

Variables externes

- **class Fasta1**
 - `String nomFichier ;`
 - `ArrayList<Sequence> sequences ;`
- **class Main**
 - `Alignement_Multiple alm, alm2 ;`
 - `Fasta f1, f2 ;`

- **class Sequence**
 - String id;
 - StringBuffer seq;
 - static String nature ;
- **class ADN extends Sequence**
 - String alphabet ;
- **class PROTEINE extends Sequence**
 - String alphabet ;

IV. Logo

Rappel des objectifs du module

Ce module permet la représentation graphique de l'alignement multiple.

Relations d'utilisation avec d'autres modules

Relation avec l'interface graphique et l'alignement multiple qui permet l'affichage du logo, et des différentes classes :

- **Weblogo**, permet de se connecter au site weblogo récupérer l'image du site et l'afficher avec Jframe,
- **TestWeblogo**, permet l'exécution des deux méthodes de la classe Weblogo

Lister les modules utilisés par ce module

- **class Weblogo**
 - import java.net.URL ;

Projet L2D - Introduction à la bioinformatique

- `import java.net.URLConnection ;`
- `import javax.imageio.ImageIO ;`
- `import javax.swing.ImageIcon ;`
- `import javax.swing.JFrame ;`
- `import javax.swing.JLabel ;`
- `import java.net.MalformedURLException ;`
- `import java.io.PrintStream ;`
- `import java.awt.BorderLayout ;`
- `import java.awt.image.BufferedImage ;`
- `import java.io.BufferedReader ;`
- `import java.io.ByteArrayInputStream ;`
- `import java.io.File ;`
- `import java.io.FileReader ;`
- `import java.io.IOException ;`
- `import java.io.InputStream ;`

Procédures externes

Se relie à l'interface graphique avec **FramePrincipale** et toutes ses classes pour l'affichage.

Variables externes

Attribut privés :

- *class Weblogo*
 - String url

5. Description des classes

package Projet_Final

1 - Classe (AlignementG.java)

Utilisation de la classe :

- import java.util.ArrayList ;

En attribut, liste les alignements avec ArrayList, séquences à aligner (seq_haut et seq_gauche) et Matrice NW.

Méthodes :

- public void **backtrack** - calcul des alignements
- public void **afficheChemins()** - affiche le chemin
- public String **cheminsToString()** - retourne toString()
- public void **afficheAlignement()** - affiche l'alignement
- public String **toString()** - retourne StringBuffer s toString()
- public void **afficheSup()** - affiche seq_SupToString()
- public StringBuffer **seq_SupToString()** - retourne this.getSeq_sup().getSeq()
- public void **afficheInf()** - affiche seq_InfToString()
- public StringBuffer **seq_InfToString()** - retourne this.getSeq_inf().getSeq()

Différentes méthodes Gets :

Projet L2D - Introduction à la bioinformatique

- public Chemin **getAlignement()** - retourne chemins.get(0)
- public Sequence **getSeq_sup()** - retourne this.getAlignement().getSeq_sup()
- public Sequence **getSeq_inf()** - retourne this.getAlignement().getSeq_inf()
- public int **getScore()** - retourne this.getAlignement().getScore()
- public ArrayList<Chemin> **getChemins()** - retourne chemins
- public Matrice **getMatrice()** - retourne matrice

2- Classe (Chemin) qui est dans la le fichier AlignementG.java

Attributs privés :

- Sequence seq_sup, seq_inf - les séquences à aligner et int score qui sera le score de l'alignement

Méthodes permettant la lecture :

- public Sequence getSeq_sup() - retourne seq_sup
- public Sequence getSeq_inf() - retourne seq_inf
- public int getScore() - retourne score

Méthodes permettant l'écriture :

- public void setScore(int s) – donne score = s
- public void setSeq_sup(Sequence s) – donne seq_sup = s
- public void setSeq_inf(Sequence s) – donne seq_inf = s

3 - Classe (Alignement_Multiple.java)

Projet L2D - Introduction à la bioinformatique

Calcul un score de similarité entre toutes les paires de séquences par comparaison des séquences deux à deux; ensemble de scores d'alignement qui sont regroupés dans une matrice de similarités.

Utilisation de la classe `import java.util.ArrayList ;`

Attributs privés :

- ◆ Fasta fasta - permet de lire les fichiers fasta ;
- ◆ `ArrayList<Sequence>` sequences ;
- ◆ `ArrayList<Alignement>` alignements - faire les alignements multiples

Méthodes :

- `public void calcul()` - permet l'alignement de séquences (consensus)
- `public void afficheSeqMult()` - affiche les séquences multiples
- `public String toString()` - retourne un `toString()`
- `public void exporterAlnfa()` - crée un fichier

4 - Classe (Fasta1.java)

Utilisation des classes :

- ◆ `import java.util.ArrayList ;`
- ◆ `import java.util.Scanner ;`
- ◆ `import java.io.File ;`
- ◆ `import java.io.FileWriter ;`

Projet L2D - Introduction à la bioinformatique

- ◆ `import java.io.FileNotFoundException ;`

Fichier lu par l'alignement global et multiple pour pouvoir faire les alignements de séquences protéiques et nucléiques avec en attribut privés :

- `String nomFichier` - nom de fichier ;
- `ArrayList<Sequence> sequences` – liste des séquences

Utilisation de Scanner permet de parcourir un flux et d'en extraire son contenu en utilisant des expressions régulières ;

File pour lire le fichier fournit des commodités pour l'écriture des fichiers caractères ;

Méthodes :

- `public String getNom()` - retourne `nomFichier`
- `public ArrayList<Sequence> getSequences()` - retourne `sequences`

Méthode static :

- `public static void creerAlnfa(String nom, String sequences)` - permet de créer le fichier `aln-fa` depuis un fichier `fasta` et en gérant les erreurs avec `Exception` (`import java.io.FileNotFoundException`) ;

5 - Classe (Main.java)

Permet l'exécution de l'alignement global et multiple ;

Attribut :

- ◆ `StringBuffer seq1, seq2` - permet de mettre manuellement les séquences pour l'alignement ;
- ◆ `Alignement al1` ;

Méthodes :

- `Matrice.setMatch()` - définit valeur match
- `Matrice.setMismatch()` - définit valeur mismatch
- `Matrice.setGap()` - définir les valeurs de gap
- `all.getMatrice().affiche()` - affiche matrice
- `all.getMatrice().afficheFleches()`
 - Vérifie que la classe matrice marche bien
- `all.afficheAlignement()` - affiche le résultat de l'alignement

Ensuite, la même chose pour les séquences de type protéine.

Pour les fichiers :

`Fasta f = new Fasta(">1\nACCTTG\n>2\nCTTG\n>3\nATGTTG")` - extrait les séquences

`ArrayList<Sequence> sequences = f.getSequences()` - permet d'obtenir les séquences du fichier fasta

`Sequence.afficheList(sequences)` - affiche les séquences extraites

Pour l'alignement multiple :

Appel de la classe `Alignement_Multiple` `alm = new Alignement_Multiple(f)` ;

Méthodes :

- `alm.calcul()` ;
- `alm.afficheSeqMult()` ;

Projet L2D - Introduction à la bioinformatique

- `Sequence.setNature()` - sélection de la nature des séquences (ADN ou PROTEINE) ;
- `Fasta f1=new Fasta("INSULINE.fasta",new File("INSULINE.fasta"))` - permet d'extraire les séquences
- `ArrayList<Sequence> sequences1 = f1.getSequences()` - permet d'obtenir les séquences du fichier fasta
- `Sequence.afficheList(sequences1)` - affiche les séquences extraites
- `Alignement_Multiple alm1 = new Alignement_Multiple(f1)` - pour deuxième séquences

Ensuite, la même chose que pour le premier alm :

- `alm1.calcul();`
- `alm1.afficheSeqMult();`
- `alm1.exporterAlnfa();`
- `Fasta f2=new Fasta("INSULINE.aln-fa",new File("INSULINE.aln-fa"))` - permet d'extraire les séquences
- `ArrayList<Sequence> sequences2 = f2.getSequences()` - permet d'obtenir les séquences du fichier fasta
- `Sequence.afficheList(sequences2)` - affiche les séquences extraites

6 - Classe (Matrice.java)

Donne les valeurs de gap, match et mismatch.

Projet L2D - Introduction à la bioinformatique

Attribut privés :

- ◆ static int match, mismatch, gap - les paramètres de calcul
- ◆ StringBuffer seq_haut, seq_gauche - les séquences à aligner
- ◆ ArrayList<ArrayList<Integer>> tableau - Tableau valeurs et séquences
- ◆ ArrayList<ArrayList<Fleches>> tableauFleches - Tableau de mise en mémoire mouvements

Méthodes :

- public void **init()** - permet le remplissage des séquences et valeurs gap
- public void **calcul()** - permet le remplissage de score et mouvements

Méthodes Gets :

- public StringBuffer **getHaut()** - retourne seq_haut
- public StringBuffer **getGauche()** - retourne seq_gauche
- public ArrayList<ArrayList<Integer>> **getTableau()** - retourne tableau
- public ArrayList<ArrayList<Fleches>> **getFleches()** - retourne tableauFleches

Méthodes qui affiche la matrice :

- public void **affiche()** - affiche tableauToString()
- public String **tableauToString()** - retourne un toString()

Méthodes qui affiche les flèches :

- public void **afficheFleches()** - affiche flechesToString()

Projet L2D - Introduction à la bioinformatique

- public String **flechesToString()** - retourne s.toString()

Méthodes Set Match, Mismatch, Gap :

- public static void **setMatch(int val)** - match = val
- public static void **setMismatch(int val)** - mismatch = val
- public static void **setGap(int val)** - gap = val

7 - Classe (Fleches) qui est dans le fichier Matrice.java

Permet la mise en mémoire du mouvement qui donne le meilleur score.

Attribut privés :

- ◆ ArrayList<String> fleches

Méthodes :

- public ArrayList<String> **getFleches()** - retourne fleches et donne l'accès au mouvement

8 - Classe (Sequence.java)

Permet d'avoir l'id nature des séquences.

Attribut privés :

- ◆ String id
- ◆ StringBuffer seq
- ◆ static String nature

Méthodes Gets et Sets :

Projet L2D - Introduction à la bioinformatique

- public String **getId()** - retourne id
- public StringBuffer **getSeq()** - retourne seq
- public void **setSeq(StringBuffer seq)** – retourne this.seq=seq
- public static String **getNature()** - retourne nature
- public static void **setNature(String nature)** - Sequence.nature=nature

Méthodes permettant l’affichage :

- public String **toString()** - retourne id+"\n"+seq
- public void **affiche()** - affiche this.toString()
- public static String **listToString(ArrayList<Sequence> sequences)** {

```
StringBuffer s = new StringBuffer();

for(int i =0; i<sequences.size();i++) { //

    s.append(sequences.get(i).toString()+"\n");

}

return s.toString();

}
```

- public static void **afficheList(ArrayList<Sequence> sequences)** – affiche Sequence.listToString(sequences)

7 - Classe (ADN) extends Sequence

Attribut privés :

Projet L2D - Introduction à la bioinformatique

- ◆ String alphabet - pour l'ADN

Méthode :

- public boolean **estValide()** - permet de savoir si la séquence nucléique est valide ou pas

Méthodes Gets et Sets :

- public String **getId()** - retourne super.getId()
- public StringBuffer **getSeq()** - retourne super.getSeq()
- public void **setSeq(StringBuffer seq)** – retourne super.setSeq(seq)

8 - Classe (Proteine) extends Sequence

Attribut privés :

- ◆ String alphabet - pour mettre la séquence de protéine

Méthode :

- public boolean **estValide()** - permet de savoir si la séquence nucléique est valide ou pas

Méthodes Gets et Sets :

- public String **getId()** - retourne super.getId()
- public StringBuffer **getSeq()** - retourne super.getSeq()
- public void **setSeq(StringBuffer seq)** – retourne super.setSeq(seq)

9 - Classe (Weblogo) qui est dans le fichier TestWeblogo.java

Utilisation des classes :

- `import java.net.URL ;`
- `import java.net.URLConnection ;`
- `import javax.imageio.ImageIO ;`
- `import javax.swing.ImageIcon ;`
- `import javax.swing.JFrame ;`
- `import javax.swing.JLabel ;`
- `import java.net.MalformedURLException ;`
- `import java.io.PrintStream ;`
- `import java.awt.BorderLayout ;`
- `import java.awt.image.BufferedImage ;`
- `import java.io.BufferedReader ;`
- `import java.io.ByteArrayInputStream ;`
- `import java.io.File ;`
- `import java.io.FileReader ;`
- `import java.io.IOException ;`
- `import java.io.InputStream ;`

Attribut privés :

Projet L2D - Introduction à la bioinformatique

- String url – adresse du site Weblogo3

Méthodes :

- public void **getWbAdressAndImage()** - utilisation de try catch (MalformedURLException e1) , (IOException e2), (Exception e3) - permet d'abord de lire le fichier en string pour les paramètres avec Utilisation de File, FileReader, BufferedReader et StringBuffer et String line qui va lire ligne par ligne le fichier, ajout du String encodedSequence qui va permettre l'encodage de la séquence grâce à la classe URLEncoder (URL - résultat qui permet l'affichage de l'image)
 - Utilisation de la classe URL pour la connexion au site et URLConnection pour ouvrir la connexion,
 - Utilisation de la classe Prinstream qui permet de régler les paramètres du site,
 - Utilisation de la classe InputStream sur co et transformation en byte[] pour pouvoir afficher l'image facilement.
- public void **getJFrame()**
 - Création de JFrame pour afficher le logo avec ensuite un String dans lequel il y a le lien ou se situe le fichier pour pouvoir l'afficher,
 - Création de JLabel avec un alignement à gauche et enfin ajout des deux JLabel à JFrame pour l'affichage.

10 - Classe TestWeblogo

Méthode main qui fait appel aux deux méthodes de Weblogo à savoir

- weblogo.**getWbAdressAndImage()** ;

■ `weblogo.getJFrame()` ;

11 - Classe (FramePrincipale.java) extends javax.swing.JFrame

Utilisation des classes :

On utilise dans les classes de l'interface les composants graphiques suivants :

JFrame : elle permet de créer et de gérer des objets de type cadre ou fenêtre.

Elle peut contenir d'autres composants d'interface tels que des boutons, des zones de saisie, ...ect.

JTextField : permettent de définir, sélectionner et manipuler du texte dans un champ de texte dynamique.

JPanel : une boîte dans laquelle on peut placer des composants de l'interface graphique.

JButton : est utilisée pour créer un bouton qui entraîne une action lorsqu'il est cliqué.

JRadioButton : Le bouton radio est utilisé pour sélectionner une option parmi plusieurs options.

JComboBox : affiche une liste et l'utilisateur peut sélectionner une option dans cette liste spécifiée.

JLabel : une zone pour afficher une chaîne courte ou une image ou les deux.

Méthodes :

■ `public void actionPerformed(java.awt.event.ActionEvent evt)` :

l'interface `ActionListener` envoie des événements à la méthode `actionPerformed()`.

■ private void **parcourirActionPerformed(java.awt.event.ActionEvent evt)** :

Applique des évènements sur le bouton parcourir afin de récupérer un fichier et faire l'alignement sur les séquences contenue dans ces fichiers fasta en faisant appel aux classes et méthodes de l'alignement.

■ private void **mismatchActionPerformed(java.awt.event.ActionEvent evt)** :

prend la valeur saisie dans le champs mismatch et puis convertir cette valeur en int en utilisant la méthode parseInt().

■ private void **matchActionPerformed(java.awt.event.ActionEvent evt)** :

prend la valeur saisie dans le champs match et puis convertir cette valeur en int en utilisant la méthode parseInt().

■ private void **gapActionPerformed(java.awt.event.ActionEvent evt)**

prend la valeur saisie dans le champs gap et puis convertir cette valeur en int en utilisant la méthode parseInt().

■ private void **jComboBox1ActionPerformed(java.awt.event.ActionEvent evt)** :

fait choisir la nature des séquences à saisir (adn ou protéine)

■ private void **btalignerActionPerformed(java.awt.event.ActionEvent evt)** :

fait les opérations d'alignement global sur les séquences saisies.

■ private void **textField2ActionPerformed(java.awt.event.ActionEvent evt)**

fait les opérations d'alignement global sur les séquences saisies ou parcouru.

■ `private void fichierActionPerformed(java.awt.event.ActionEvent evt) :`

permet de parcourir le fichier .fasta afin de réaliser l'alignement.

// Variables declaration à ne pas modifier

- ◆ `private javax.swing.JButton btnalign2 ;`
- ◆ `private javax.swing.JButton btnaligner ;`
- ◆ `private javax.swing.JButton btnaligner1 ;`
- ◆ `private javax.swing.JTextField fichier ;`
- ◆ `public static javax.swing.JTextField gap ;`
- ◆ `public static javax.swing.JComboBox<String> jComboBox1 ;`
- ◆ `private javax.swing.JLabel jLabel1 ;`
- ◆ `private javax.swing.JLabel jLabel2 ;`
- ◆ `private javax.swing.JLabel jLabel3 ;`
- ◆ `private javax.swing.JLabel jLabel4 ;`
- ◆ `private javax.swing.JLabel jLabel5 ;`
- ◆ `private javax.swing.JLabel jLabel7 ;`
- ◆ `private javax.swing.JLabel jLabel9 ;`
- ◆ `private javax.swing.JPanel jPanel1 ;`
- ◆ `private javax.swing.JScrollPane jScrollPane2 ;`

- ◆ `private javax.swing.JTabbedPane jTabbedPane1 ;`
- ◆ `public static javax.swing.JTextField match ;`
- ◆ `public static javax.swing.JTextField mismatch ;`
- ◆ `public static javax.swing.JButton parcourir ;`
- ◆ `public static javax.swing.JTextArea textArea ;`
- ◆ `public static javax.swing.JTextField textField1 ;`
- ◆ `public static javax.swing.JTextField textField2 ;`

12 – Classe (**ResAlignement.java**) extends **javax.swing.JFrame**

La fenêtre **ResAlignement** affiche les résultats de l'alignement global ou multiple des séquences saisies, en l'affichant sur un `textArea` qu'on a déclaré `public static` pour y pouvoir accéder par les autres classes.

On a utilisé `JFrame.EXIT_ON_CLOSE` dans toute les classes de pour pouvoir libérer l'espace mémoire l'interface lors de la fermeture de l'application.

Cette fenêtre contient aussi un `JButton` 'retour' pour retourner à la page principale.

On a aussi passé comme paramètre à la méthode `setTitle()` de la fenêtre une chaîne de caractère qui apparaîtra comme titre de la fenêtre selon l'alignement saisie(global ou multiple).

13 – Classe (**FrameAG.java**) extends **javax.swing.JFrame**

Cette fenêtre contient les champs de saisies **`JTextField`** des valeurs `gap`, `match`, `mismatch` et la liste déroulante **`JComboBox`** pour choisir la nature des séquences à aligner, avec les deux champs de saisie des séquences à aligner, et puis un **`JButton`**

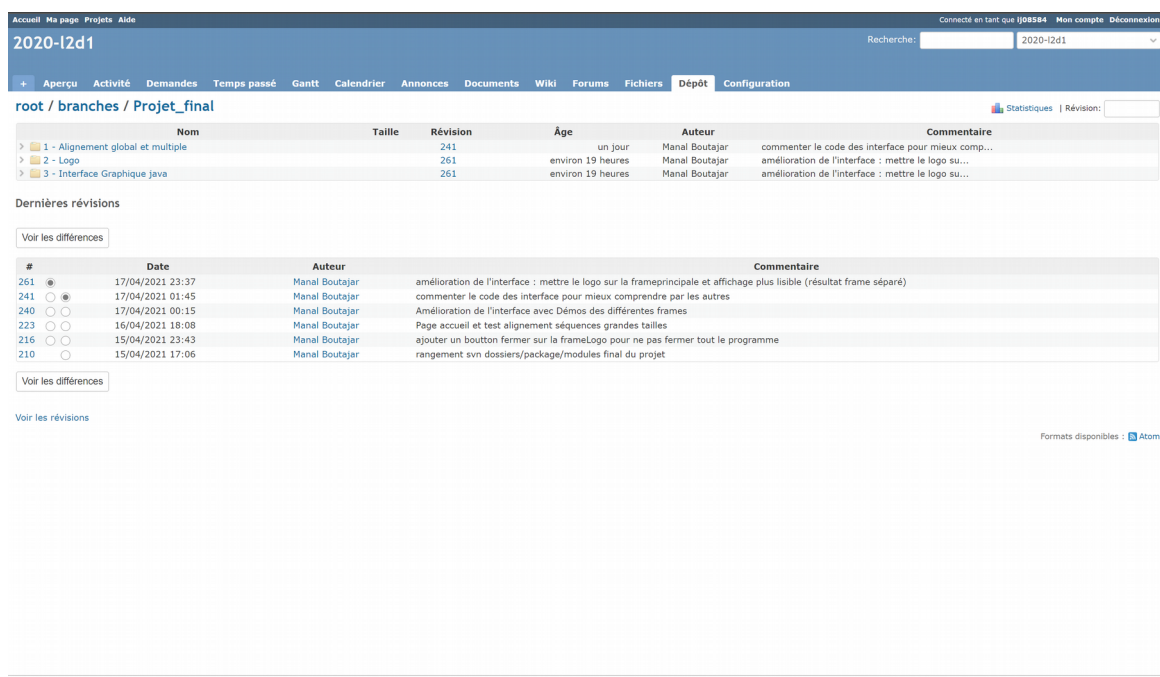
aligner pour déclencher les méthode de la classe **AlignementG** et réaliser les calculs sur la matrice et donc afficher le résultat avec le tableau et le score de l'alignement sur la fenêtre **ResAlignement**.

14 – Classe (FrameAM.java) extends javax.swing.JFrame

Cette fenêtre contient de même les champs de saisies **TextField** des valeurs gap, match, mismatch et la liste déroulante **jcombobox** pour choisir la nature des séquences à aligner, avec le champs de saisie manuelle des séquences en format .fasta ou un **JFileChooser** pour parcourir un fichier .fasta, et puis un **JBUTTON** aligner pour déclencher les méthode de la classe **Alignement_Multiple** afficher le résultat l'alignement sur la fenêtre **ResAlignement** avec son logo.

6. Annexes

- Forge – partie Projet_final



The screenshot shows the Forge project page for '2020-l2d1'. The page includes a navigation bar with links like 'Accueil', 'Ma page', 'Projets', and 'Aide'. Below the navigation bar, there is a search bar and a dropdown menu for the project name. The main content area displays the project structure under 'root / branches / Projet_final'. It lists three files: '1 - Aligement global et multiple', '2 - Logo', and '3 - Interface Graphique java'. Each file has a 'Taille' (Size), 'Révision' (Revision), 'Âge' (Age), 'Auteur' (Author), and 'Commentaire' (Comment). The 'Révision' column shows the revision number, and the 'Âge' column shows the time since the last revision. The 'Auteur' column shows the name of the author, and the 'Commentaire' column shows the commit message. Below the file list, there is a section for 'Dernières révisions' (Latest revisions) which shows a table of revisions with columns for '#', 'Date', 'Auteur', and 'Commentaire'. The table lists revisions 261, 241, 240, 222, 216, and 210, all by 'Manal Boutajar'. The 'Commentaire' column shows the commit messages for each revision. At the bottom of the page, there is a footer that says 'Powered by Redmine © 2006-2020 Jean-Philippe Lano'.

Illustration 2: Forge - partie où se trouve le projet

7. Glossaire

- ★ **BIO-INFORMATIQUE** : discipline permettant d'analyser de l'information biologique qu'il y a dans les séquences nucléotidiques et protéiques
- ★ **PROGRAMMATION ORIENTÉE OBJET** : met en œuvre différents objets (instances de classes). Chaque objet associe des données et des méthodes (fonctions) agissant exclusivement sur les données
- ★ **ALGORITHME NEEDLEMAN - WUNSCH** : effectue un alignement global maximal de deux chaînes de caractères

- ★ **ALIGNEMENT GLOBAL** : recouvre les séquences alignées (par exemple 2 séquences) sur l'ensemble de leur longueur
- ★ **ALIGNEMENT MULTIPLE** : aligner un ensemble de séquences homologues, comme des séquences protéiques ou nucléiques qui assure des fonctions similaires dans différentes espèces vivantes
- ★ **WEBLOGO** : application web conçue pour rendre la génération de logos de séquence aussi simple que possible
- ★ **INTERFACE GRAPHIQUE** : relie les 3 modules et les présente de manière simple et efficace

- ★ **GAP, MATCH, MISMATCH** : valeurs pour alignement global
- ★ **FASTA** : format de fichier texte dans lesquels les séquences sont affichées par une suite de lettres

8. Références

[Alignement de séquences](#)

[Alignement global](#)

[Alignement multiple](#)

[JAVA - Main \(site 1\)](#)

[JAVA - Main \(site 2\)](#)

[Chaînes de caractères](#)

[Tableau](#)

[Needleman et Wunsch – Java](#)

[Needleman et Wunsch – Python](#)

[Interface graphique en Java](#)

[Interface graphique avec SWING](#)

9. Index

Index des figures

<i>Illustration 1: Image de présentation</i>	<i>1</i>
<i>Illustration 2: Forge - partie où se trouve le projet</i>	<i>37</i>

Index lexical

alignement global	6, 10 sv, 15, 23, 33, 35, 38
Alignement global	5, 9, 12, 39
ALIGNEMENT GLOBAL	38
alignement multiple	6, 11, 17, 24
Alignement multiple	6, 9, 15, 39
ALIGNEMENT MULTIPLE	38
attribut	20, 23
Attribut	13, 18, 21 sv, 26 sv
consensus	6, 9, 22
fasta	12, 15, 22 sv, 33, 36
Fasta	10, 12 sv, 22, 24 sv
FASTA	38
gap	12, 14 sv, 24 sv, 33 sv

Projet L2D - Introduction à la bioinformatique

Gap	24, 27
GAP	38
gap,	36
interface graphique	11 sv, 15 sv, 32
Interface graphique	6, 11, 39
Interface Graphique	9
INTERFACE GRAPHIQUE	38
javadoc	5
jButton	35
JButton	32, 34 sv
jcombobox	36
jComboBox	33 sv
JCombobox	35
JComboBox	32, 34
Jframe	16 sv, 30 sv
JFrame	13, 31 sv, 35
jLabel	34
Jlabel	18, 30 sv

Projet L2D - Introduction à la bioinformatique

JLabel	32, 34
jPanel	34
JPanel	32, 34
JRadioButton	32
jTextField	35 sv
JTextField	32, 34 sv
l'application	5, 35
logo	9, 11, 17 sv, 29 sv, 36, 38
Logo	6, 17
LOGO	38
match	12, 14 sv, 24 sv, 33 sv
Match	24, 27
MATCH	38
match,	36
matrice	15
méthodes	5, 10, 17, 20, 31, 33, 38
Méthodes	20 sv, 26 sv, 31 sv
mismatch	12, 15, 24 sv, 27, 33 sv

Projet L2D - Introduction à la bioinformatique

Mismatch	24, 27
MISMATCH	38
ResAlignement	12, 35 sv
weblogo	11, 17, 31
Weblogo	9, 17 sv, 29 sv
WEBLOGO	38