Rapport - Sequence_matching - Test unitaire

LI YUANYUAN 19.11.2017

Projet original ＜ sequence_matching ＞ execute sur VisualStudio2012. Il y a trois parties dans ce projet: séquence ＆ paramètre, méthode de calculer, run ＆ afficher. Il faut ajouter la librairie ＜cppunit＞ dans le projet de ＜testlyy＞. Donc, pour chaque partie, je choisis quelques classes pour tester. Le nom de mon projet de test est ＜ testlyy ＞

Ⅰ. Séquence & Paramètre

1.1. Set paramètre directement

Pour ce projet，on peut calculer des paramètres avec des type: caractère, numérique.
< *OperatorDistance* > et < *Element* > sont des classe abstraites. Donc, je choisis 4 classes: < *Character* >，< *Numeric* >，< *CharacteristicVector* > *et* < *Sequence* > pour tester les méthodes de：get，set，add，remove，size.

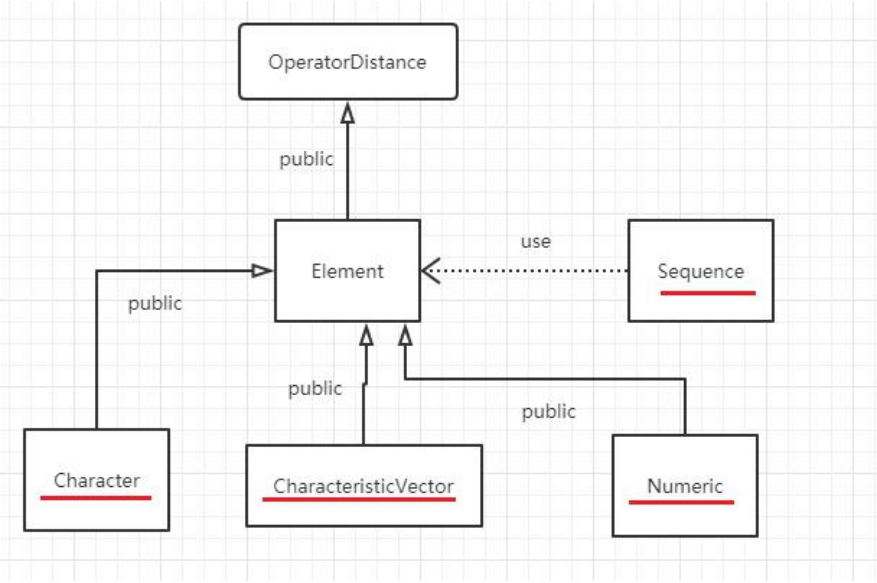

Diagramme de classe avec séquence-1

1.1.1. Les méthodes de test：

(1). *< Character >：*

char getValue()；
void setValue(char c)；
*float distance(Element *eOD1, Element *eOD2)；*
string toString()；
*Element * copy()；*

Ce sont des méthodes pour configurer l' élément de séquence (caractère)，nous avons besoin

d′ appeler ces méthodes plusieurs fois dans le projet.

(2). *< Numeric >：*

float getValue()；
void setValue(float value)；
*float distance(Element *eOD1, Element *eOD2)；*
string toString()；
*Element * copy()；*

Ce sont des méthodes pour configurer l' élément de séquence (numérique)， nous avons besoin d' appeler ces méthodes plusieurs fois dans le projet.

(3). *< CharacteristicVector >：*

float getAt(unsigned int id)；
void addValue(float elt)；
void removeValue(unsigned int index)；
void setValue(unsigned int index, float value)；
int getSize()；
*float distance(Element *eOD1, Element *eOD2)；*
string toString()；
*Element * copy()；*

Ce sont des méthodes pour configurer l' élément de séquence (liste de chiffre)， nous avons besoin d' appeler ces méthodes plusieurs fois dans le projet.

(4). *< Sequence >：*

*void addElement(Element * elt)；*
void removeElement(unsigned int index)；
*Element * getElement(unsigned int index)；*
int getSize()；

Cette classe est la base de ce projet, nous avons besoin d' appeler des méthodes - add，remove，pour modifier la séquence． Quand nous calculons, nous avons aussi besoin d' appeler getsize().

1． 2． Set paramètre par lire des fichiers

Ou peut aussi obtenir les paramètres par lire des fichier (XML & CSV).
< CSVParser > et *< XMLParser >* sont des classe abstraites. Donc, je choisis 2 classes:*< SequenceParser >, < ParamPaser > et < Paramtrage >* pour tester si les fichiers sont bien lues.

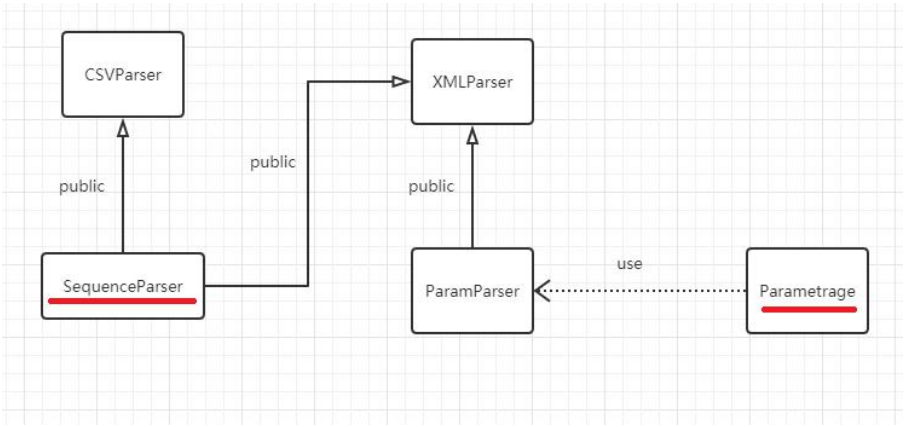

Diagramme de classe avec séquence-2

1.2.1. Les méthodes de test：

(1). *< SequenceParser >：*

*void * readXML(void * doc);*

*void * readCSV(string docname);*

Nous avons besoin de lire des fichiers pour configurer les paramètres. Donc, c' est très important de lire des fichiers correctement et de configurer séquences avec les contenu de fichier.

(2). *< paramtrage >：*

void setS1Weight(unsigned int index, float value);

float getS1Weight(unsigned int index);

void setS2Weight(unsigned int index, float value);

float getS2Weight(unsigned int index);

void setMatrixWeight(unsigned int index1,unsigned int index2, float value);

float getMatrixWeight(unsigned int i1, unsigned int i2);

void setS1Size(unsigned int value);

int getS1Size();

void setS2Size(unsigned int value);

int getS2Size();

Si nécessaire, nous avons besoin de configurer des poids de distance.

(3). < ParamParser >:

void * readXML(void * doc);

Si nécessaire,nous avons besoin de configurer des poids de distance par lire des fichiers （.XML）. Mais, il y a des problèmes pendant lire fichier，il y a une exception:< rapidxml::parse_error >.

Ⅱ．Méthode - Calculer

Quand on obtention des séquences，on peut calculer．Il y a quatre méthodes que on peut choisir :Levensthein, Dynamic Time Warping, Minimum Variance Matching, Longest Common Sequence . < *Correpondance* > est la classe abstraite. Donc, je choisis 4 classes:< *LevenstheinCorrepondance* >, < *DTWCorrespondance* >, <*MVMCorrespondance* > et <

LCSCorrespondance > pour tester si les résultats de ces méthodes sont corrects.

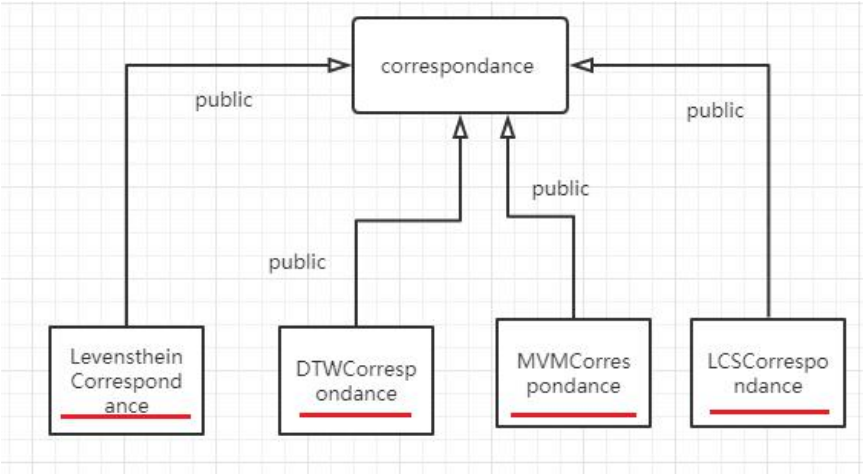

Diagramme de classe avec méthode

3.1. Les méthodes de test：

(1). < *LevenstheinCorrepondance* >：

*ResultatCorrespondance * correspondre(model::Sequence * s1, model::Sequence * s2);*

C' est la méthode pour calculer la distance. Nous avons besoin d' appeler cette méthode dans la méthode run().

(2). < *DTWCorrespondance* >：

*ResultatCorrespondance * correspondre(model::Sequence * s1, model::Sequence * s2);*

C' est la méthode pour calculer la distance. Nous avons besoin d' appeler cette méthode dans la méthode run().

(3). <*MVMCorrespondance* >：

*ResultatCorrespondance * correspondre(model::Sequence * s1, model::Sequence * s2);*

C' est la méthode pour calculer la distance. Nous avons besoin d' appeler cette méthode dans la méthode run().

(4). < *LCSCorrespondance* >：

*ResultatCorrespondance * correspondre(model::Sequence * s1, model::Sequence * s2);*

C' est la méthode pour calculer la distance. Nous avons besoin d' appeler cette méthode dans la méthode run().

Ⅲ. Run & Afficher

Pour run et afficher les résultats, il y a < *CommandLineApplicatio* > (run) et < *CommandeLineOutput* > (afficher). Je choisis < *CommandLineApplicatio* > et < *CommandeLineOutput* > pour tester si les commandes sont bien passées .

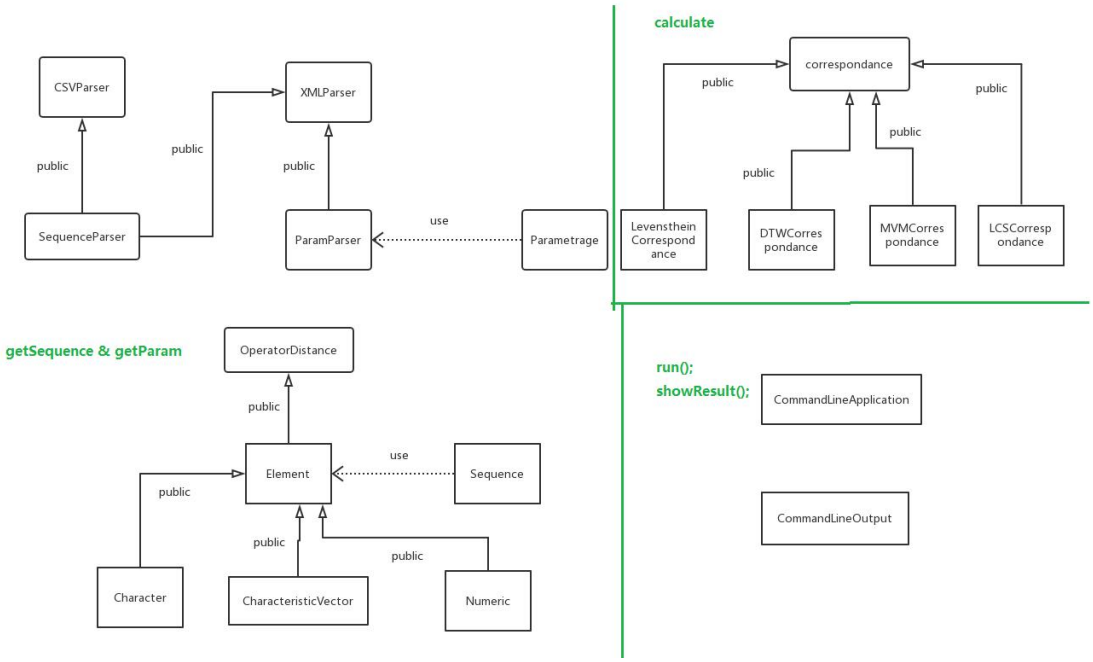

Diagramme de classe de projet

3． 1. Les méthodes de test：

(1). *< CommandLineApplication >*：

void CommandLineApplication::checkParams()；

D' abord, c' est très important de passer et analyser les commandes. Donc, je choisis de tester cette méthode. Pour tester cette classe，j′ ai crée une classe *＜TestCopieCommandeLineApp＞* qui hérite la classe *< CommandLineApplication >*. Dans la classe *＜TestCopieCommandeLineApp ＞*，on peut obtenir les attributs privés de la classe *< CommandLineApplication >*. Donc，on peut déterminer si ces attributs de la classe *< CommandLineApplication >* sont bien configurées.

(2). *< CommandeLineOutput >*:

string CommandLineOutput::format();

Cette méthode est pour bien afficher le résultat. Donc， nous avons besoin de tester cette méthode. Mais，il y a des méthodes privées de la classe *< CommandeLineOutput >* qui sont très importantes. Il a besoin de tester dans la future.

Ⅳ. Conclusion

Quand je teste la classe *< ParamPaser >*, il y a une problème sûr la méthode ： void *inout::ParamParser::readXML(void *doc). Dans la première fois，il peut lire le fichier，mais dans la deuxième fois ，il ne peut pas lire le fichier. Mais，je ne change rien dans le fichier.

Je teste ces 13 classes par ordre ci－dessous：

< Character >，< Numeric >，< CharacteristicVector >，< Sequence >，*< SequenceParser >*, *< ParamPaser >*, *< Paramtrage >*， *< LevenstheinCorrepondance >*, *< DTWCorrespondance >*, *<MVMCorrespondance >*， *< LCSCorrespondance >*， *< CommandLineApplicatio >* et *< CommandeLineOutput >*.